Marlboro College Plan of Concentration

Dylan Murphy-Mancini
Computer Science & Computer Security
May 2, 2017

Faculty Sponsor	Jim Mahoney	Professor of Computer Science
Secondary Reader	Matt Ollis	Professor of Mathematics
Outside Examiner	Ian Kozak	Green River Data Analysis

To demonstrate competency in computer science and computer security, this Plan is divided into the following weights:

Component	%
Development of and paper about a security scanner	50%
Systems programming: Modifying a running process	30%
Examinations on programming languages, and algorithms	20%

Contents

Acknowledgements

I would like to give a special thanks to my professor, faculty sponsor, and role model, Jim Mahoney, professor of computer science at Marlboro College. I could not have done this without you. Thank you for the hours after class, the late night games of email tag, and unwavering support through the process of Plan.

Forward

The question I get the most is "why security". I can distinctly recall the exact moment at which I knew I wanted to pursue it as a degree. It is an anecdote that begins in the middle of my second semester at Marlboro College. I was taking a class on microcontrollers. This lab based class centered around working with Arduinos– using them to explore circuits, code, switches, lights, and all that. It was an introductory level class, but for me became the embedded systems programming primer I was craving. During the last few weeks of the semester, each of us had to create a final project. People decided to work on everything from electronic thermometers to wirelessly controlled robots. There was even a student who made a fully functional theremin! On the final class we had to present our project to the class. I wasn't anywhere to be found. I showed up to my professor's office long after that class ended. The hours I poured into my project must have shown in my appearance because he took pity on me and granted permission to present my project for credit. I apologized profusely explaining my project took longer than I ever anticipated. I put my backpack on his desk and pulled out my Arduino– and there it was, just my Arduino. Nothing more than a bare Arduino. No wires connected to a breadboard, no LED's, no motors, no wireless modules, or theremin to be seen. "Is this it?", he said. I urged him, to plug it in to his computer. He didn't look amused but humored my request attaching the Arduino by USB. I'm paraphrasing but the dialogue that followed was essentially:

Professor: "What now?"

ME: "This is it. List your USB devices- type 'lsusb'"

Professor: "Okay... see what exactly?"

ME: "It's no longer a serial communication device. It's a MIDI device!"

Professor: "This is the most unimpressive impressive thing."

I had flashed the ROM of the Arduino with a program acting as a USB

controller and subsequently implemented the HID specification to echo as a native MIDI device. It was during this presentation I came to realize a couple of things: One, I truly love the communication protocols, their specifications, and their implementation. Two, there is absolutely no glory in loving them. From that moment on I drowned myself in low level languages like C, and started recreating standard GNU utility programs. It was a short hop onto security from there. I began on to do things like implement ping via ICMP, ARP spoofing, DNS redirection, and crack WEP. Studying security became this way to take the protocols I found fascinating and turn them into sport.

Modifying a Process on Linux

Introduction

There are many legitimate and nefarious reasons to interrupt the normal execution flow of a program. Some of the good reasons include: debugging, memory diagnostics or forensics, and sand-boxing for analysis. Reasons also include clandestine motives– hacking to bypass features or modify the behavior of a program for profit. On operating systems like Windows or OSX there exists a plethora of material to explain methods used to accomplish these goals. This is due to the fact these operating systems make up the majority of computer users. Linux is gaining popularity as a proprietary operating system replacement. As of March, 2017, 5.9% of desktops are running Linux as the main operating system– up 2.3% from 2009.[1] In my time studying operating system implementation, malware analysis, and reverse engineering, I found there to be a lack of material, especially in concern to proof-of-concept code (POC) for Linux on modifying a running process. This paper aims to document methods I've researched for Linux and provide POC implementation for future security students.

DLL Injection

A dynamic-link library (DLL) is a shared library loaded at run time by programs. DLL injection is a technique used for running code within the address space of another process by forcing it to load a DLL.[2] On Linux there are a number of environment variables that control the loading process, one of which is LD_PRELOAD. This environment variable can be set to specify a shared object that will be loaded before all others, effectively overriding any of the same symbols in other libraries.[3] For example, consider the following target program random.c (see Code Appendix for the source code). This program simply prints a random number in a never-ending loop using the rand() function from the standard c library (libc)– which by default is linked dynamically.

```
$ gcc random.c -o random
$ ./random
83
77
15
...
```

Listing 1: Compiling and running random.c with GCC and on Ubuntu

To implement DLL injection we first need to create a shared library that will take precedence. The file unrandom.c contains just a definition for the rand() function.

```
int rand(){
    return 42;
}
```

Listing 2: unrandom.c

Compile it as a shared library, which will produce a shared object (.so). When LD_PRELOAD is set to use this new library my very predictable rand() definition is used.

```
$ gcc -shared -fPIC unrandom.c -o unrandom.so
$ LD_PRELOAD=$PWD/unrandom.so ./random
42
42
42
...
```

Furthermore you can export LD_PRELOAD so you don't have to specify the new library each time you run the target program.

```
$ export LD_PRELOAD=$PWD/unrandom.so
$ ./random
42
42
42
...
```

Our new definition of the rand() function is executing code in the process's address space. To further exemplify this I have created an example intercepting, puts(), which is used within the prtinf(). The target program printfecho.c returns whatever string you pass into it (see Code Appendix for the source code of rintfecho.c). For example, passing in the string "Hello, World!" returns the string back:

```
$ gcc printfecho.c -o printfecho
```

```
$ ./printfecho "Hello, World!"
Hello, World!
```

The new shared library code in inspectprintf.c utilizes the dlsym library for working with libraries and symbols.

```
#define _GNU_SOURCE
#include <dlfcn.h>
#include <stdio.h>
#include <string.h>

static int (*real_puts)(const char* str) = NULL;

int puts(const char* str){

    /* printing out the number of characters */
    printf("puts:chars#:%lu\n", strlen(str));

    /* resolve the real puts function from glibc
     * and pass the arguments.
     */
    real_puts = dlsym(RTLD_NEXT, "puts");
    real_puts(str);
}
```

Listing 3: inspectprintf.c

The RTLD_NEXT flag specifies the next symbol after this one that shares name. The new puts definition essentially wraps the libc version. It intercepts the system call, executes code the program is unaware of (prints a character count of the string passed in), and then passes the arguments onto the real puts() function in libc.

```
$ gcc -shared -fPIC inspectprintf.c -o inspectprintf.so -ldl
$ LD_PRELOAD=$PWD/inspectprintf.so ./printfecho "Hello, World!"
puts:chars#:13
Hello, World!
```

GDB

One of the limitations of DLL injection is that the method can only override dynamically-linked functions. What would be the entry point if the target is statically compiled? Modifying the execution of such a program can be accomplished with the use of a debugger. Probably the most well-known method to hit pause on a program's execution and meddled around on Linux is through the use of the GNU Debugger (GDB). GDB's intended purpose is to run other

programs, allowing the user to exercise control over them, and examine variables when problems arise. GDB allows you to see what is going on 'inside' another program while it executes. At Marlboro, we use it heavily when learning C and Assembly. GDB can do four main kinds of things:[4]

- Start your program, specifying anything that might affect its behavior.
- Make your program stop on specified conditions.
- Examine what has happened, when your program has stopped.
- Change things in your program, so you can experiment with correcting the effects of one bug and go on to learn about another.

To show GDB modifying a process in action I have created a target program flag.c' (see Code Appendix for the source code). It is a minimal program consisting of a single conditional. An integer 'flag' is set to false at the start, and without anyway to change that the conditional it always evaluates to false.

```
$ gcc flag.c -o flag
$ ./flag
Flag is false
```

The program behaves as expected. Although this is a simple conditional on an integer, it is no different from the logic used to implement trial lock features in freeware. To begin changing the expected behavior of the flag program, I first load the binary into GDB as if to begin a the normal debugging process (The q flag is just to suppress GDB's welcome message).

```
$ gdb -q flag
Reading symbols from flag...(no debugging symbols found)...done.
(gdb) b main
Breakpoint 1 at 0x6a4
(gdb) r
Starting program: /home/us3r/Desktop/testdir/flag

Breakpoint 1, 0x00005555555546a4 in main ()
(gdb) disassemble
Dump of assembler code for function main:
   0x00005555555546a0 <+0>: push   %rbp
   0x00005555555546a1 <+1>: mov    %rsp,%rbp
=> 0x00005555555546a4 <+4>: sub    $0x20,%rsp
   0x00005555555546a8 <+8>: mov    %edi,-0x14(%rbp)
   0x00005555555546ab <+11>:  mov    %rsi,-0x20(%rbp)
   0x00005555555546af <+15>:  movl   $0x0,-0x4(%rbp)
   0x00005555555546b6 <+22>:  cmpl   $0x0,-0x4(%rbp)
   0x00005555555546ba <+26>:  je     0x5555555546ca <main+42>
   0x00005555555546bc <+28>:  lea    0xa1(%rip),%rdi        # 0
    x555555554764
   0x00005555555546c3 <+35>:  callq  0x555555554560
```

```
   0x00005555555546c8 <+40>:  jmp    0x5555555546d6 <main+54>
   0x00005555555546ca <+42>:  lea    0xa0(%rip),%rdi        # 0
    x555555554771
   0x00005555555546d1 <+49>:  callq  0x555555554560
   0x00005555555546d6 <+54>:  mov    $0x0,%eax
   0x00005555555546db <+59>:  leaveq
   0x00005555555546dc <+60>:  retq
End of assembler dump.
(gdb) x/4xb 0x00005555555546ba
0x5555555546ba <main+26>: 0x74  0x0e  0x48  0x8d
```

So far I've just set a breakpoint on the main method. Inspecting the assembly shows the conditional is on lines main+15, main+22 and main+26 (if (flag)). Address 0x00005555555546ba is shown in bytes because it is what I want to modify. The je operand at that address means jump equal or jump zero, testing the zero flag. The zero flag is used to check the result of an arithmetic operation– the cmpl at main+22. The hex code of je is 0x74. By changing it to 0x75 the je would become jne– jump if not equal.[5] This can be modification can be accomplished using GDB's set command.

```
(gdb) set *(char*)0x00005555555546ba=0x75
(gdb) disassemble
Dump of assembler code for function main:
   0x00005555555546a0 <+0>: push   %rbp
   0x00005555555546a1 <+1>: mov    %rsp,%rbp
=> 0x00005555555546a4 <+4>: sub    $0x20,%rsp
   0x00005555555546a8 <+8>: mov    %edi,-0x14(%rbp)
   0x00005555555546ab <+11>:  mov    %rsi,-0x20(%rbp)
   0x00005555555546af <+15>:  movl   $0x0,-0x4(%rbp)
   0x00005555555546b6 <+22>:  cmpl   $0x0,-0x4(%rbp)
   0x00005555555546ba <+26>:  jne    0x5555555546ca <main+42>
   0x00005555555546bc <+28>:  lea    0xa1(%rip),%rdi        # 0
    x555555554764
   0x00005555555546c3 <+35>:  callq  0x555555554560
   0x00005555555546c8 <+40>:  jmp    0x5555555546d6 <main+54>
   0x00005555555546ca <+42>:  lea    0xa0(%rip),%rdi        # 0
    x555555554771
   0x00005555555546d1 <+49>:  callq  0x555555554560
   0x00005555555546d6 <+54>:  mov    $0x0,%eax
   0x00005555555546db <+59>:  leaveq
   0x00005555555546dc <+60>:  retq
End of assembler dump.
(gdb) x/4xb 0x00005555555546ba
0x5555555546ba <main+26>: 0x75  0x0e  0x48  0x8d
(gdb) c
Continuing.
Flag is true
[Inferior 1 (process 21803) exited normally]
```

Continuing execution after my modification confirms the flag now evaluates as true. Even though you can attach to an already running process with 'GDB -p PID' (where PID is the process id of the process you want to attach to),

I wanted to create more permanent program modifications– ones that didn't require the need to be wrapped by GDB. This curiosity led me to look into how GDB accomplishes writing to memory with the set command.

PTRACE

At the heart of GDB (and almost any Linux debugging tool) lies the system call PTRACE. Operating systems offer services through a standard mechanism called system calls. They provide a standard application program interface for accessing the underlying hardware and low-level services, such as the file systems. When a process wants to invoke a system call, it puts the arguments to the system call in registers and invokes soft interrupt 0x80. This soft interrupt is like a gate to the kernel mode, and the kernel will then execute the system call after examining the arguments. [6]

```
$ man ptrace
PTRACE(2)                  Linux Programmer's Manual
      PTRACE(2)

NAME
       ptrace - process trace

SYNOPSIS
       #include <sys/ptrace.h>

       long ptrace(enum __ptrace_request request, pid_t pid,
                   void *addr, void *data);

DESCRIPTION
       The ptrace() system call provides a means by which one
    process (the "tracer") may observe and control the execution of
     another process (the "tracee"), and examine and change the
    tracee's memory and registers. It is primarily used to
    implement breakpoint debugging and system call
       tracing.

       A tracee first needs to be attached to the tracer.
    Attachment and sub-sequent commands are per thread: in a
    multithreaded process, every thread can be individually
    attached to a (potentially different) tracer, or left not
    attached and thus not debugged. Therefore, "tracee"
    always means "(one) thread", never "a (possibly multithreaded)
```

In short, the man page of PTRACE says it is possible to code a complete debugger using this system call alone. As POC I am going to use it to intercept getuid(). Although you can use PTRACE to attach to a running process it is faster to develop by calling fork(), using the parent to debug the child. The parent process is alerted every time something happens in the child. For

my example I will be requesting notification of on every system call. The first argument to PTRACE determines the service requested:[7]

PTRACE_TRACEME: Indicates that this process is to be traced by its parent.[14]

PTRACE_SYSCALL: Stops the tracee for inspection at the entry to or exit from a system call.[14]

PTRACE_GETREGS: Copy the tracee's general-purpose or floating-point registers, respectively, to the address data in the tracer.[14]

PTRACE_SETREGS: Modify the tracee's general-purpose or floating-point registers, respectively, from the address data in the tracer.[14]

The target program for my PTRACE example simply prints out the current user ID (UID). See the Code Appendix for the source code of printfecho.c. Running target.c without any modification:

```
$ gcc target.c -o target
$ ./target
Target program
user id: 1000
```

My PTRACE hack named tracer.c forks the target as a child.

```
#include <sys/ptrace.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <sys/syscall.h>
#include <sys/reg.h>
#include <sys/user.h>
#include <unistd.h>
#include <string.h>
#include <errno.h>

#define TARGET "/home/us3r/Desktop/plan/target"
#define NEW_UID 1337

int main(int argc,char** argv) {
  int status = 0;
  int syscall_n = 0;
  int entering = 1;
  struct user_regs_struct regs;
  int pid = fork();

  if (!pid) {
    ptrace( PTRACE_TRACEME, 0, 0, 0 );
    execv( TARGET, argv );
  } else {
    wait( &status );
```

```
    while (1) {
      ptrace( PTRACE_SYSCALL, pid, 0, 0 );

      wait( &status );

      if ( WIFEXITED( status ) ) break;

      ptrace( PTRACE_GETREGS, pid, NULL, &regs );
      syscall_n = regs.orig_rax;
      if ( syscall_n == SYS_getuid ) {
        if ( entering ) {
          entering = 0;
        }
        else {
          ptrace( PTRACE_GETREGS, pid, 0, &regs );
          regs.rax = NEW_UID;
          ptrace( PTRACE_SETREGS, pid, 0, &regs );
          entering = 1;
        }
      }
    }
  }

  return 0;
}
```

Listing 4: tracer.c

Running tracer in turn runs target, intercepts the getuid() system call and returns 1337 as the UID. Of course this method of overriding a system call could be used for debugging purposes on any system call or function.

```
$ gcc tracer.c -o tracer
$ ./tracer
Target program
user id: 1337
```

To mitigate this method of PTRACE interception you could easily detect if your process is being traced by PTRACE by calling PTRACE yourself. For example, I've written a demonstration program entitled ptest.c where PTRACE is called with PTRACE_TRACEME– this would return an error if the process is already being traced.

```
#include <stdio.h>
#include <sys/ptrace.h>

int main(){
    if (ptrace(PTRACE_TRACEME, 0, 1, 0) == -1) {
        printf("Don't trace me!!\n");
        return 1;
    }
```

```
    printf("Normal Execution... no tracing going on here.\n");
    return 0;
}
```

Listing 5: ptest.c

Running the program un-traced returns the message "Normal Execution":

```
$ gcc ptest.c -o ptest
$ ./ptest
Normal Execution... no tracing going on here.
```

However, the program does not like being traced by a debugger like GDB:

```
$ gdb -q ptest
Reading symbols from ptest...(no debugging symbols found)...done.
(gdb) r
Starting program: /home/us3r/Desktop/testdir/ptest
Don't trace me!!
[Inferior 1 (process 25656) exited with code 01]
(gdb)
```

An error is returned as the process can not call ptrace while it is being traced by GDB. This is a cat and mouse game– you could attach a debugger after the program does its trace check, and programs can implement further methods of trace detection but I hope this conveys how this game is essentially played.

Adding a System Call

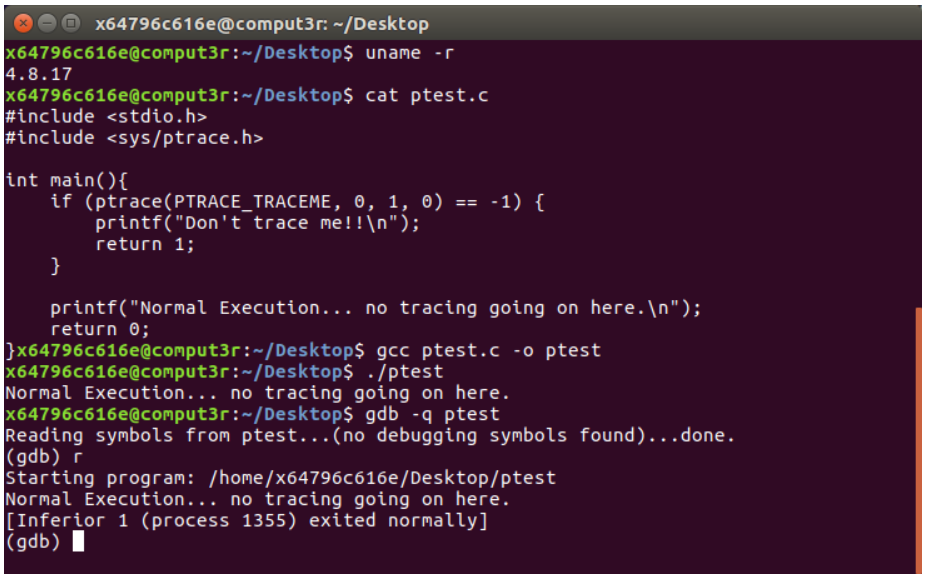
System calls can be viewed as entry points into the kernel through which programs request services from the kernel. After learning PTRACE can be detected, and myself trying to reverse programs with PTRACE detection, I wondered if it were possible to add a separate system call to the kernel– one that would give me the features of ptrace(), but not be ptrace(), and thus not be detected. This is probably the most backwards way of going about mitigating detection but this essay would not be complete with out mentioning the answer is yes– you can add system calls to a Linux kernel and nothing is stopping you from implementing a second version of PTRACE.[17]

Adding a system call or modifying anything about a Linux distribution can be accomplished by downloading the kernel's source, recompiling, and installing it into your system. Rather than detail all the files that need to be modified or walk through the system dependent compilation process, I've decided to highlight what to change in order to bypass ptest. The source code of ptrace and

other system calls are in the /kernel directory of the kernel source code. In ptrace.c on lines 401-430, we can see the definition for ptrace_traceme– the function that calculates a return value when calling PTRACE with PTRACE_TRACEME. On line 427 you can see my addition: ret = 0. This is how I've set the return value to always be true. With this modification, traceme will always return true instead of returning an error value when calling PTRACE within a traced process. /linux-4.8.0/kernel/ptrace.c lines 401 - 430

```
401 /**
402  * ptrace_traceme  --  helper for PTRACE_TRACEME
403  *
404  * Performs checks and sets PT_PTRACED.
405  * Should be used by all ptrace implementations for PTRACE_TRACEME.
406  */
407 static int ptrace_traceme(void)
408 {
409   int ret = -EPERM;
410
411   write_lock_irq(&tasklist_lock);
412   /* Are we already being traced? */
413   if (!current->ptrace) {
414     ret = security_ptrace_traceme(current->parent);
415     /*
416      * Check PF_EXITING to ensure ->real_parent has not passed
417      * exit_ptrace(). Otherwise we don't report the error but
418      * pretend ->real_parent untraces us right after return.
419      */
420     if (!ret && !(current->real_parent->flags & PF_EXITING)) {
421       current->ptrace = PT_PTRACED;
422       __ptrace_link(current, current->real_parent);
423     }
424   }
425   write_unlock_irq(&tasklist_lock);
426
427   ret = 0;
428
429   return ret;
430 }
```

After recompiling and installing the new kernel, the results of the same ptest.c as before are modified!



Linux Kernel Modules

The process of downloading kernel source and compiling for hours is a slow and painful way to develop a modification for the kernel or access kernel space. Rather than rebuild and reboot the kernel every time we want new functionality Linux supports kernel modules (LKM) which can be loaded and unloaded into the kernel upon demand. They extend the functionality of the kernel without the need to reboot the system. For example, one type of module is the device driver, which allows the kernel to access hardware connected to the system.[18]

Using LKM's speed up development time and I am able to display their rein over kernel space with more complexity in this section. Kernel code cannot access the libraries of code written for the Linux user space (like libc) as the kernel module lives and runs in only kernel space. The interface between kernel space and user space is clearly defined and controlled. We have a printk() function that can output information which can be viewed from within user space. For this POC I implemented a "Hello, World" LKM that has functionality comparable to reading /proc/PID/maps. The source code for lkm.c is in the Code Appendix. One note about compiling and running this would be to change the absolute path and PID of the target program. The target program here, loop.c, just runs continuously (the source code of which is also in the Code Appendix). For example:

```
$ gcc loop.c -o loop
```

```
$ ./loop
I'm a loop!
I'm a loop!
I'm a loop!
I'm a loop!
```

When the module is loaded into the kernel, its init method is called which displays the virtual memory addresses of the target program's code.

```
$ make && sudo insmod ./hello.ko && sudo rmmod hello && dmesg |
    tail -8
make -C /lib/modules/4.8.0-41-generic/build M=/home/us3r/Desktop/
    lkm modules
make[1]: Entering directory '/usr/src/linux-headers-4.8.0-41-
    generic'
  CC [M]  /home/us3r/Desktop/lkm/hello.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/us3r/Desktop/lkm/hello.mod.o
  LD [M]  /home/us3r/Desktop/lkm/hello.ko
make[1]: Leaving directory '/usr/src/linux-headers-4.8.0-41-generic
    '
[30973.299308] Hello, world init
[30973.299314]
               This mm_struct has 18 vmas.
[30973.299318]
               Code  Segment start = 0x556938442000, end = 0
    x5569384428cc
               Data  Segment start = 0x556938642dd0, end = 0
    x556938643010
               Stack Segment start = 0x7ffe16957c40
[30973.314581] Hello, world cleanup
```

Furthermore we have read and write access to this program's memory via the mm_struct which contains pointers to those addresses. In user land you could not access the code or data sections of a process for writing.

Writing and Reading Process Memory

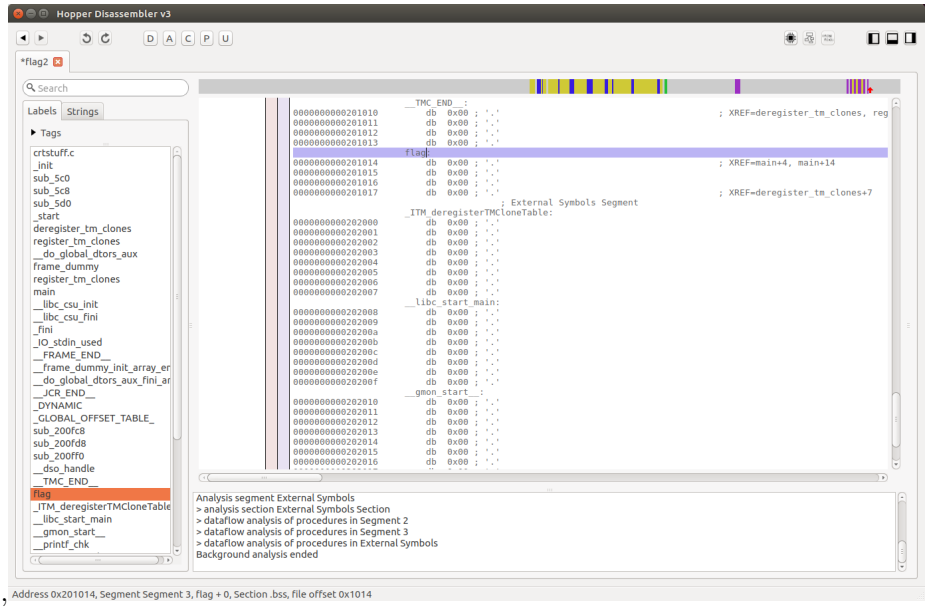
Actually modifying the in memory instructions of a process via an LKM is reinventing the wheel. There are actually two relatively unknown system calls I've come across in my research that can do the job. They are process_vm_writev and process_vm_readv– similar in fashion to Window's WriteProcessMemory and ReadProcessMemory. The man page states they are intended to write and read to a process's memory for fast inter-process communication. The bible of Linux malware analysis, "The Art of Memory Forensics", mentions how little these system calls are seen:

> "Since the 3.2 kernel version, Linux has provided two functions, process_vm_readv and process_vm_writev, that allow for reading and writing of a foreign process' memory without the use of ptrace. Although it means that the use of PTRACE_PEEKTEXT and PTRACE_POKETEXT could be avoided, all the other ptrace functionality in this section is still required. Furthermore, because these system calls are relatively new at the time of writing, we have yet to see any malware in the wild use them."[19]

With that being said I would like to provide the POC for these mysterious system calls. The target program flag2.c is similar to the first flag program in the GDB example, however it simply continues printing the flag's value in a never-ending loop (see the code appendix for the source code of flag2.c). Running flag2:

```
$ gcc flag2.c -Og -o flag2
$ ./flag2
Flag = 0
Flag = 0
...
```

In inspecting flag2's binary we can observe the memory offset (0x201014) of the initialized integer that is the actual flag. Below the image depicts me using Hopper Disassembler in lieu of GDB to determine this offset.



Using this information we can write a program that determines the base address of the target, uses the offset to calculate the flag's address, and writes to processes memory to change it. In my capture the flag hack, ctf.c, you can see me parsing the /proc/PID/maps file to determine the base address in memory.

```
#define _GNU_SOURCE
#include <stdio.h>
#include <stdlib.h>
#include <sys/uio.h>

#define FLAG_OFFSET 0x201014

int main(int argc, char const *argv[]){
  pid_t pid;
  struct iovec local[1];
  struct iovec remote[1];
  ssize_t nread;

  pid = atoi(argv[1]);


  char fname[255];
    FILE *f;

    sprintf(fname, "/proc/%i/maps", pid);
    f = fopen(fname, "r");

    char buff[255];
    fgets(buff, 255, f);

    long long unsigned int base_addr;

    sscanf(buff,"%Lx",&base_addr);

  unsigned char buf[sizeof(int)];

  int n = 1;

  for (int i = 0; i < sizeof(int)*2; ++i){
    buf[i] = n>>i*sizeof(int)*2;
  }

  local[0].iov_base = buf;
  local[0].iov_len = sizeof(int);

  remote[0].iov_base = (void*)base_addr+FLAG_OFFSET;
  remote[0].iov_len = sizeof(int);

  nread = process_vm_writev(pid, local, 1, remote, 1, 0);

  if (nread){
    printf("Captured the flag!\n");
  }
  return 0;
}
```

Listing 6: ctf.c

Running ctf.c:

```
$ gcc ctf.c -o ctf && sudo ./ctf 18401
Captured the flag!
```

The output of the continuously running flag2 is now changed without interrupting the execution!

```
Flag = 0
Flag = 0
Flag = 0
Flag = 0
Flag = 0
Flag = 0
Flag = 1
Flag = 1
Flag = 1
Flag = 1
Flag = 1
```

Building off this simple example where the flag is just an integer in the data section, I would like to conclude my work here with an example comparable to video game hacks in the wild. The file externalhack.c is a health hack written for AssualtCube, a cross-platform first-person shooter video game. The value I'm changing is the player's health, which starts at 100 by default. This is not an integer in the data section, it lives on the stack. However, there is a player class initialized in the data section and my program follows pointers to player1's location and uses that address to calculate the offset (see the code appendix for the source code of externalhack.c).

Running assault cube:

Notice the health value at 100 in the bottom left hand corner. Running externalhack.c:

```
$ gcc externalhack.c -o externalhack
$ sudo ./externalhack
Found PID:  14238
Found Base Address:  0x55c0b687b000
Player Pointer:  0x55c0b6bf3130
Player Address:  0x55c0b7a3c2f0
Player V Health:  0x55c0b7a3c400
Health: 100
Hacked Health
```

This is a more complete program in which you don't even need to pass a PID over– the hack will parse the /procs directory for the correct PID. The result is infinite health! Every tick the value of the players health is overwritten, leaving the player invincible.


AssaultCube
RISE
1337
MTP-57
7 40
fps 144

Boan: an HTTP(S) Man-in-the-Middle Proxy

Project hosting and documentation:
`https://github.com/0x64796c616e/Boan`

Boan is the English version of the Korean word 보안, meaning security or preservation of public peace. The project is an HTTP/HTTPS proxy that allows for the modification of requests to aid in manual web application penetration testing.

Background

The idea to code a web proxy for my Plan project comes from multiple sources. One common reason to code your own tool like Boan is fear of becoming a "script kiddie"– a person who uses existing computer scripts or code to hack, lacking the expertise to write their own. I wanted to demonstrate knowledge of web security and protocols' implementation. In my time as an intern at a software security firm it became abundantly clear that the majority of web vulnerabilities revolve around breaking encoding. A proxy that allows for the manipulation of requests is a the tool for the job. Also, a standard project for a computer science student is to implement an HTTP server of some kind– making one that can handle encryption was my way of taking it one step further. The addition a graphical user interface (GUI) as the front-end was a way of satisfying the software developer in me.

How Boan Works

To safeguard sensitive information from potential eavesdroppers or man-in-the-middle attacks, TCP/IP includes secure protocols (SSL/TLS) which have been designed specifically to attain end-to-end security. These protocols follow a handshake protocol to establish a shared encryption key which is known only to the client and the secure server. Once the handshake is complete, all successive data transferred between the client and server are encrypted. Since the client and secure server alone have the shared encryption key, they exclusively are able to decipher the data. A special case occurs when circumstances require the client to use a proxy server to connect to a remote secure server. To preserve pure end-to-end encryption the proxy server must not see the shared encryption key. However, the proxy server must still connect the client to the remote server so the handshake can even take place.

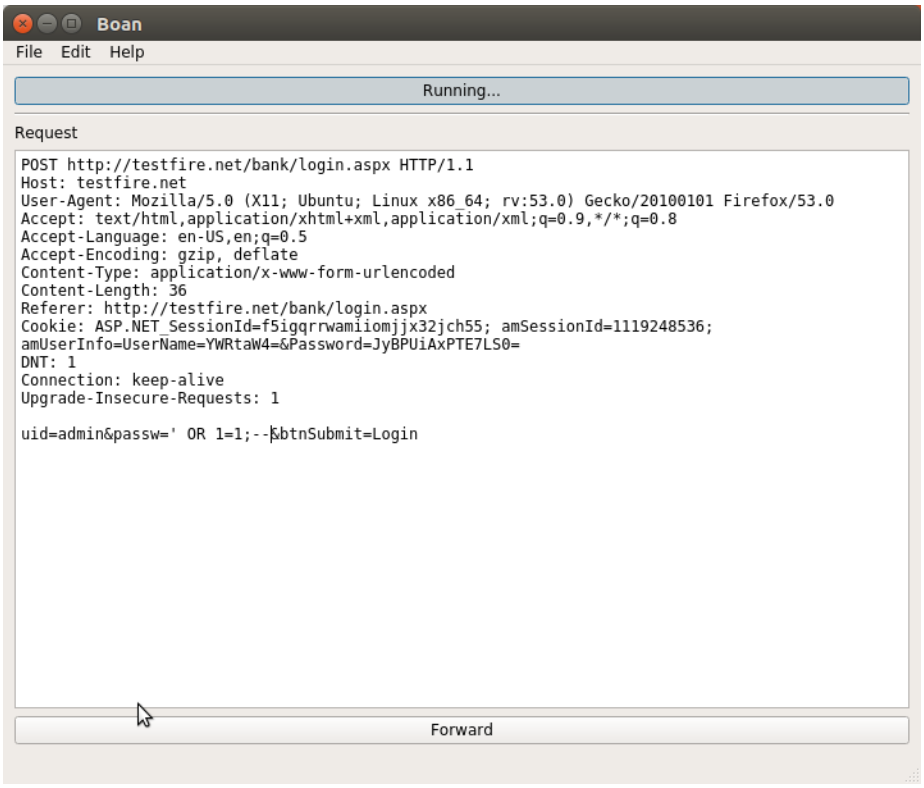
To accommodate this special case, HTTP has a method: CONNECT. This method is used by a client to instruct a proxy server to establish a connection with a remote server so that handshaking between the client and the remote server can take place. After that connection is established, the handshake takes place. All consecutive data transfer between the client and the remote secure server are encrypted and sent to the proxy server– the proxy acts only as a relay point between the client and the remote secure server. Using a relay point like this is known as tunneling. Since the proxy server does not know the encryption key, it cannot examine the data in the communications, and end-to-end security is preserved.

When the client opens an SSL/TLS connection to a secure server, it verifies the server's identity by checking two conditions: First, it checks whether its certificate was signed by a certificate authority (CA) known to the client. Second, it makes sure that the common name (CN), host name, of the server matches the one it connects to. If both conditions are true, the client assumes the connection is secure. In order to be able to intercept traffic on the proxy connection, Boan acts as a certificate authority, however, not a very trustworthy one. Instead of issuing certificates to actual persons or organizations, Boan dynamically generates certificates to whatever hostname is needed for a connection. If, for instance, a client wants to connect to https://www.facebook.com, Boan generates a certificate for "www.facebook.com" and signs it with its own CA. This CA certificate is generated the first time Boan is run, and installed as a trusted root in your browser.

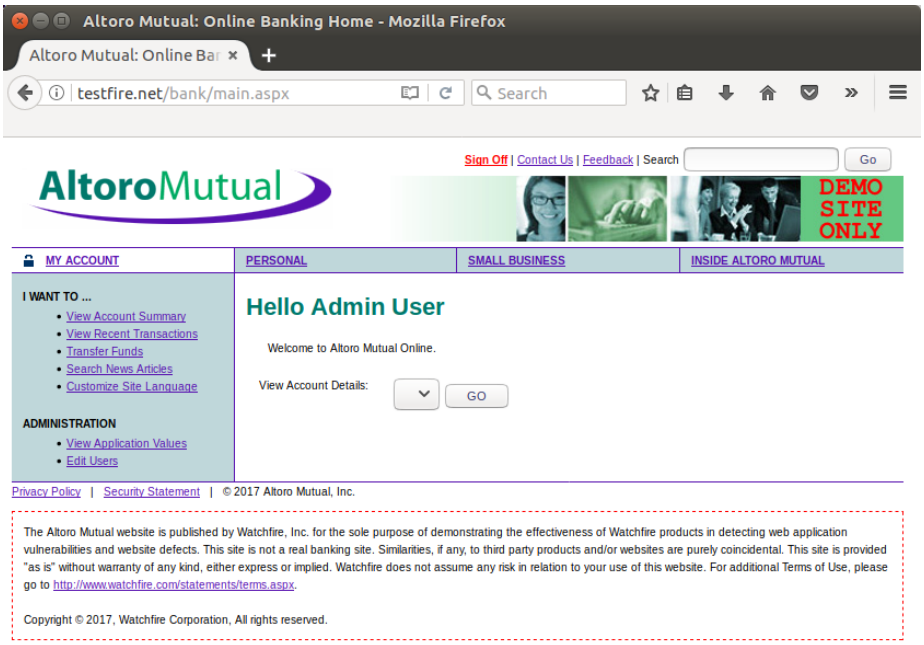
Results

To gauge my development I have been testing Boan on `http://demo.testfire.net/`, and an example website of my own creation `http://demofire.000webhostapp.com/`. The website I created is a PHP example of GET/POST forms, and SESSION data to demonstrate Boan working. Testfire.net is an intentionally vulnerable fake-banking website entitled Altoro Mutual. The Altoro Mutual website is published by Watchfire, Inc. for the sole purpose of demonstrating the effectiveness of Watchfire products (IBM's AppScan) in detecting web application vulnerabilities and website defects. Watchfire produced a security report for testfire.net, (`http://blog.watchfire.com/files/demo.testfire.net-security-report-8.6.pdf`), of which I have gone through and reproduced the vulnerabilities with Boan that deal with breaking encoding or inserting malicious code. Vulnerabilities such as, cross-site scripting (hosted and reflected), SQL injection, privilege escalation (horizontal and vertical), faulty session management, forced browsing, and HTTP response splitting. [20]

To document a serious finding, here is an example of Boan being used to detect privilege escalation via SQL injection on Altroro Mutual. Although this finding is listed in Watchfire's security report, the actual payload. On `http://testfire.net/bank/login.aspx` you are prompted to log in with a username and password.



Because the username for the admin account is predictable, and there is poor input validation on the SQL query– thus I am able to modify the SQL statement to return with a positive username/password match.



The result is being able to log in as a bank administrator. The Watchfire security report doesn't mention remediation for this vulnerability but my recommendation would be to use parameterized queries.

Moving Forward

Boan was not always a MitM proxy. It started out as web crawler written in C++/QT. Due to the learning curve of developing in C++ and working with GUI frameworks, and also the fact I still had to learn about HTTP implementation, I opted to move the code base to Python. Developing in Python was faster for me and provided a lot of built-in libraries for things like socket connections and compression. Developing Boan has been a learning experience in HTTP(S), web security, and software development, but moving forward there are a couple goals I would like to accomplish. For one, I would like to port the code back to C++ for speed. Although the main bottleneck for a proxy is the internet connection speed, C++ would provide a speed up on the compression/decompression and the encryption/decryption. Feature wise, the Python proxy module within Boan actually supports more functionality than the GUI allows. Currently it is possible to import the proxy module and override the request/response handlers to script automated changes to web traffic; i.e implementing SSL split or DNS redirection. The proxy module also allows for sending arbitrary requests

and viewing the responses. Moving forward I would like to incorporate these into the GUI, as for actual security report generation the response would be something you want to include in findings.

Examinations

Programming Languages

Out: Tue April 11 2017
Due: Tue April 18 by midnight

This is open take-home exam: books or web sources are OK as long as the problem doesn't set other constraints, you cite them explicitly, and that they aren't a drop-in solution to the problem. However, the more your answer is a summary of someone else's article, the less we will be impressed. Don't ask other people for help. Don't just give a numerical result, give an explanation. Your job is to convince us you understand this stuff.

So in terms of a grading rubric, what you should think about is

- technical merit - correctness & thoroughness of response
- clarity of expression (including docs & tests for code)
- demonstration of understanding (vs summarizing others work)

Be very explicit about which sources you used for each problem. As always with my exams, if you think there's a mistake in one of the questions or it doesn't make sense, you can (a) ask for clarification, and/or (b) make and state an explicit interpretation and do the problem that way. (Again: the point is to show your mastery, not to get the "right answer" per se.)

Good luck.

Paradigms	Variables & Scope	Functions	Data Structure
imperative functional object oriented	variable type immutable global scope name macro pointer	function iterate lambda closure callback recursion pass by reference pass by value argument side effect stack overflow dynamic	data structure array list vector link collection hash

Object Oriented	Syntax	Multithreading	Compiler Design
static object overload namespace method throw class inheritance interface exception package	syntactic sugar comment	concurrent fork thread	lazy parse lexical symbol compiled interpreted link bind

Software Design
pattern test

Paradigms

Paradigms are the conceptual representation of data and execution in a programming language. The three main abstractions are imperative, functional, and object-oriented. We use them to classify programming languages although most languages support more than one paradigm. (I provide an example of a multi-paradigm language in question two by using Python to implement a program using a functional, imperative, and object-oriented style).

The imperative paradigm is a style of programming wherein you describe how to get what you want. An example imperative language would be C. Declarative programming is similar in its procedural style, however the difference from imperative is that you are describing what you want and not how to get it. An example declarative language would be SQL. In SQL you write queries for what you want without writing procedures for it. An example of me using imperative, object-oriented, and declarative all together is the course scheduling web application I wrote for Marlboro College's registrar. I used objected oriented PHP and declarative SQL in the back-end and javascript to produce a calendar front end.

A prime example of a functional language would be either LISP or HASKELL. In a pure functional programming language, a program's execution is the evaluations of expressions, whereas in imperative languages you are writing statements which change a global state. Functional programming in essence is using functions as the building blocks.

Object-oriented programming is a paradigm based on the notion of objects. Objects have attributes (data) and methods (functions) and interact with one another as the logic of the program. This paradigm focuses more on the representation of data rather than procedural logic. Python, Java, C++, and PHP are some of the most widely used languages that encourage this paradigm. C does not have support for objects, and PHP only in the last couple of years starting supporting it.

Compiler Design and syntax

The languages I will be talking about the mostly are Python, Javascript, PHP, and C. The C language is considered statically typed, meaning the variable type is known at compile time. The big advantage for statically typed languages is that type checking can be done by the compiler, and therefore a lot of trivial bugs are caught before execution. Dynamically typed languages like Python do their type checking at run time. Although bugs may exist, development speed is increased. The syntax of PHP and Javascript play off the popularity of C syntax.

Command lines end in semicolons, white space for the most part doesn't matter, and curly brackets are every where. Python is the language with synatic sugar in mind. Python uses white space to encapsulate its functions and conditionals. It uses more human readable keywords too like "not", "is in", "until".

Programming languages can be interpreted or compiled. Interpreted languages execute code directly usually composed of subroutines in machine code. Compiled languages are converted into machine code or bytecode. The advantage of interpreted languages is platform independence. Compiled languages like C, although not platform independent, enjoy an increase in speed thanks to their static type checking. Javascript, Python, and PHP are interpreted languages. They are compiled into intermediate code for their respective JIT-compiler and executed.

Multithreading

Multithreading is all about concurrent execution. A single CPU can execute a process on each of its cores, speeding up a program utilizing this. When programming concurrent processes, you will either be forking or threading. Forking creates a child process that shares no memory with the parent. For a concrete example of this please refer to my ptrace example in Modifying a Process on Linux. In threading, a sibling is created that shares the same program data but its own stack. For a concrete example of threading please refer to Boan: an HTTP(S) MitM Proxy. Forking is more resource intensive than threading, although a new process can run independently.

Variables & Scope

Variables are the symbolic references to some data in a computer program. The name of the variable is that literal symbol. Languages usually have restrictions on what characters or numbers can be in a name. For instance, valid variable names in python must start with a letter or an underscore, then consist of letters, numbers and underscores, and are case sensitive. Variables in PHP follow the same rules except they must start with a $. A variable's type is the classification of its data. This type information tells the compiler or interpreter how the program intends to treat the data. For example, some built-in types in Python are: integer, boolean, float, bytes, and string. In C, there is no built-in string type. Strings in C are represented as an array of chars. Some languages like C or D require the type as part of the variables definition, whereas in PHP or javascript the type is just optional.

Example variable declaration in Python:

```
x = "Hello"
y,z = 0
n = True
```

Example variable declaration in C:

```
int d = 3, f = 5;
double z = 0x8;
char string[] = "hello, world";
char x = 'x';
```

In C, a variable must be defined before it is initialized. In contrast, Python is interpreted so we can create variables on the fly with the only rule being you cannot reference a variable that hasn't been assigned a value. Variables can be mutable or immutable, meaning they can be changed or remain a constant state. Without specifying an int as const in C, it will by default be mutable. In Python and Javascript, strings are immutable. Strings in PHP are mutable but the common practice is to treat them as mutable.

Example of immutability in javascript:

```
var string = "I can not be changed";
var piece = statement.slice(2, 5);
```

The variable 'string' still contains the same value even after the slice function is invoked. If strings were mutable in javascript the string variable would now have the value "I can be changed". Such is the case in languages like Ruby, where strings are in fact mutable.

Variables associate a name to a value, whereas pointers store the location in memory. Example pointer declaration in C:

```
int foo = 2;
int *foo_ptr = &foo;
```

The variable foo_ptr contains the memory address of foo. To access the value of foo, we would have to deference the pointer. Python and other interpreted languages abstract away from having to deal with pointers.

A scope in any programming is the region of the program where a variable exists and its symbol is accessible. There are three places where variables can be declared: 1. Local- inside a function. 2. Global- Outside of all functions. 3. Formal- in the definition of function parameters.

Example of scope in C:

```c
#include <stdio.h>

/* global variable declaration */
int var = 1;

/* function scope */
void printnum(int var) {
  var = 3;
  printf ("%i\n",  var);
}

int main () {

  printf ("%i\n",  var); // global

  int var = 2;

  printf ("%i\n",  var); // local

  printnum(var); // function

  printf ("%i\n",  var); // proof

  return 0;
}
```

The output of the above example would be 1, 2, 3, 3. Even though the variable var uses the same name throughout the program, it exists in different scopes. Some languages utilize a global keyword to use a global variable inside a functions scope. PHP goes as far as to include a global associative array as part of the language spec.

Example of global variables in PHP:

```php
<?php
$a = 1;

function printnum(){
  $a = 2;
  echo $GLOBALS['a'];
  echo $a;
}

printnum();
?>
```

This program would output 1, 2 as it first prints the global $a and then prints the local $a.

Functions

These are example function declarations in C, Python, and PHP, of a function maxnum() which returns the greater value of two integers.

Function in C:

```
int maxnum(int num1, int num2) {
   int result;

   if (num1 > num2)
      result = num1;
   else
      result = num2;

   return result;
}
```

Function in Python:

```
def maxnum( num1, num2 ):
   return max(num1,num2)
```

Function in PHP:

```
function maxnum($num1,$num2) {
    return max($num1,$num2);
}
```

PHP shares some syntax with C in the curly brackets and semi-colons but there is no need to declare argument type as with C. In a lot programming languages there exists a special type of function lamba, an anonymous function not bound to a symbol. C has no support for lambda and closures.

Example lambda in PHP:

```
$input = array(1, 2, 3, 4, 5);
$output = array_filter($input, function ($v) { return $v > 2; });
```

Example lambda in Python:

```
>>> foo = [1, 2, 3, 4, 5]
>>> filter(lambda x: x > 2, foo)
[3, 4, 5]
```

Another cool type of functions are closures. Closures attach data to code

therefore remembering values passed in by enclosing their scope. The use case for closures include eliminating dependencies on constants and global variables.

Example closure in Python:

```
def say_hi(msg):
    def talk():
        print("Hello, "+msg)
    return talk

hello = say_hi("Dylan")
hello()
hello()
```

This program would say "Hello, Dylan" twice without having to pass in "Dylan". A callback function is a type of closure. In the simplest terms, a callback function is any function that is called by another function that takes the first function as a parameter. A callback function is invoked after something happens.

Example callback in javascript/jQuery

```
$("#btn_1").click(function() {
    alert("Btn 1 Clicked");
  });
```

Here, once the selected DOM element is clicked, an anonymous function is called. Even without a symbol, the function could still be examined via the arguments object of the containing function. The main languages I've been discussing, PHP, javascript, Python and C all support recursive functions. In C, the maximum recursion depth is the operating systems stack size (probably 1MB). In Python and PHP you have environment variables that guard against this impending stack overflow. In python you can not pass primitives by reference as arguments. Python uses call by object reference.

Example:

```
def append_one(li):
    li.append(1)
l = [0]
append_one(l)
print l

def append_one(li):
    li = [0, 1]
l = [0]
append_one(l)
print l
```

```
[0, 1]
[0]
```

The reason the second append_one function doesn't change l, is because l points to one array, but inside the function a new array is created and l still points to the old one. Such is not the case in PHP or C.

Example in PHP:

```
function foo(&$var){
    $var++;
}

$a=5;
foo($a);
```

In the above code snippet, the variable $a ends up as 6. Both PHP and C use the ampersand to denote passing by reference.

Data Structure

Data structures are a way of organizing data to to be more useful. Structures like hash tables, linked lists, stacks, and queues are usually created from smaller language collections like arrays, structs, tuples, and lists. In python you have a hefty amount of built in structures. Something that always threw me off coming from C was Python's use of the word dictionary. In other languages a "dictionary" is is an associative array. You have an array of arrays, and you construct that explicitly. In python, this dictionary is its own type.

Example dictionary in Python:

```
records = {
  'Dylan': [1,2,3],
  'Matt': [2,3,4]
}

for student,grades in records.items():
  print("{} - {}".format(student,grades))
```

This iteration would produce:

```
Matt - [2, 3, 4]
Dylan - [1, 2, 3]
```

Achieving this same output with an associative array in PHP or C is more

complicated. We have to create an array of arrays, and iterate over them using indices.

Example dictionary in PHP:

```
$records = array("Dylan"=>array(1,2,3),"Matt"=>array(2,3,4));
print_r($records);
```

This would ouput:

```
Array
(
    [Dylan] => Array
        (
            [0] => 1
            [1] => 2
            [2] => 3
        )

    [Matt] => Array
        (
            [0] => 2
            [1] => 3
            [2] => 4
        )

)
```

In C things get even more complicated. We have to make choices about string size and use structs that pointer to one another. PHP and Python, although using different collections, have built-in methods for appending and removing elements of an array. In C, you would have to use malloc and remember to free when done.

Object-oriented

Both Python and PHP support object oriented programming. In fact even the variables are objects in these languages. C has no support for it at all. In my personal opinion, object-oriented programming is all about good exception handling (C doesn't even support exception handling). Beyond simple things like inheritance and class definitions, one feature of object-oriented programming that differs wildly among languages is interfacing. An interface in object-oriented programming is a method defined in some abstract class that requires children to define it. For instance, a vehicle class would require a start engine method, but the different car classes might implement the method differently. Python doesn't support interface per se, but you will see people doing something along the lines of this:

Interface in python:

```
class vehicle( object ):
    def startengine( self ):
        raise NotImplementedError( "This method is required" )
```

In php you don't need to use a work around because it supports interfacing with its own template class type.

Interface in PHP:

```
interface vehicle{
    public function startengine($key);
}
class Car implements vehicle{
    private $vars = array();

    public function startengine($key){
        return;
    }
}
```

Interfacing requires you to define a function, whereas overloading is the ability to define methods with the same name and different arguments. There is no method overloading in python however you can in C++ or Java. In PHP overloading requires some hook trickery that I've never truly understood. Common practice is to avoid it.

Question 2

Write six programs implementing solutions to the following two problems across the three languages with different styles. (These may be the same three languages from question 1 but don't need to be.)

In each case, include docs and tests appropriate to the style of that language, including explicitly what version of what language you ran, in what environment, what steps compiled and/or ran the code, and what the input and output looked like.

Use these programs to illustrate some different currently popular programming paradigms, as well as your mastery of the vernacular within these programming language communities. Standard libraries and extensions are allowed, but

the more you can do yourself the better. As a post scipt, discuss which languages you found well suited to which problem, and why. The two problems are:

A) fraction sum search

What permutation of the digits from 1 to 9 will add to 1 when arranged in a form similar to 1/23 + 4/56 + 7/89 ? Find the answer with a brute force search.

B) web crawler visualization

Write a program which will first build a network of URLs and the links between them by starting at a given web page and following its links outwards, and then generate a visual representation of that network.

All the details are up to you, including which page to start at, how far to go, and how to display the result.

Fraction Sum Search

I have written 5 solutions to this problem in three languages. Inside the attached code folder are the following programs:

File: fraction_sum_search.c
Language: C
Style: imperative
Running: Compile with gcc and run. Tested on gcc v6.3

```
$ gcc fraction_sum_search.c
$ ./a.out
```

Output:

```
5/34 + 7/68 + 9/12 = 1
5/34 + 9/12 + 7/68 = 1
7/68 + 5/34 + 9/12 = 1
7/68 + 9/12 + 5/34 = 1
9/12 + 5/34 + 7/68 = 1
9/12 + 7/68 + 5/34 = 1
Execution time: 0.016011 seconds
```

File: fraction_sum_search.js
Language: Javascript
Style: imperative
Running: Run with nodejs or include in an HTML page. I ran with nodejs v4.7.2

```
$ nodejs fraction_sum_search.js
```

Output:

```
[ 5, 3, 4, 7, 6, 8, 9, 1, 2 ]
[ 5, 3, 4, 9, 1, 2, 7, 6, 8 ]
[ 7, 6, 8, 5, 3, 4, 9, 1, 2 ]
[ 7, 6, 8, 9, 1, 2, 5, 3, 4 ]
[ 9, 1, 2, 5, 3, 4, 7, 6, 8 ]
[ 9, 1, 2, 7, 6, 8, 5, 3, 4 ]
Execution time: 328ms
```

File: fraction_sum_python_imperative.py
Language: Python 3
Style: imperative
Running: Run with python3. I used python v3.5.3

```
$ python3 fraction_sum_search_imperative.py
```

Output:

```
(5, 3, 4, 7, 6, 8, 9, 1, 2)
(5, 3, 4, 9, 1, 2, 7, 6, 8)
(7, 6, 8, 5, 3, 4, 9, 1, 2)
(7, 6, 8, 9, 1, 2, 5, 3, 4)
(9, 1, 2, 5, 3, 4, 7, 6, 8)
(9, 1, 2, 7, 6, 8, 5, 3, 4)
Execution time 0.3413383960723877 seconds.
```

File: fraction_sum_python_functional.py
Language: Python 3
Style: functional
Running: Run with python3. I used python v3.5.3

```
$ python3 fraction_sum_search_functional.py
```

Output:

```
[(5, 3, 4, 7, 6, 8, 9, 1, 2),
 (5, 3, 4, 9, 1, 2, 7, 6, 8),
 (7, 6, 8, 5, 3, 4, 9, 1, 2),
 (7, 6, 8, 9, 1, 2, 5, 3, 4),
 (9, 1, 2, 5, 3, 4, 7, 6, 8),
 (9, 1, 2, 7, 6, 8, 5, 3, 4)]
Execution time: 0.36745166778564453
```

File: fraction_sum_python_OOP.py
Language: Python 3
Style: Object-oriented
Running: Run with python3. I used python v3.5.3

```
$ python3 fraction_sum_search_OOP.py
```

Output:

```
[(5, 3, 4, 7, 6, 8, 9, 1, 2),
 (5, 3, 4, 9, 1, 2, 7, 6, 8),
 (7, 6, 8, 5, 3, 4, 9, 1, 2),
 (7, 6, 8, 9, 1, 2, 5, 3, 4),
 (9, 1, 2, 5, 3, 4, 7, 6, 8),
 (9, 1, 2, 7, 6, 8, 5, 3, 4)]
Execution time 0.38771629333496094 seconds.
```

Web Crawler Visualization

File: network_grpahy.py
Language: Python 3
Style: imperative
Usage: A valid website address, hostname, is the only required command line argument to the program. There is an optional flag parameter -n, which is the number of urls to process from the built up queue of URLs.

```
$ python3 network_graph.py -h
usage: network_graph.py [-h] [-n N] hostname

Network Graph

positional arguments:
  hostname    The website URL from which to spidering.

optional arguments:
  -h, --help  show this help message and exit
  -n N        Number of urls to parse from the queue
```

Running: Run with python3. I used python v3.5.3

```
python3 network_graph.py http://cs.marlboro.edu/ -n 20
```

Output:

```
Visiting #19: http://www.marlboro.edu/
Urls 295 found. The queue is now: 40
Visiting #18: http://cs.marlboro.edu/
Urls 41 found. The queue is now: 329
Visiting #17: http://cs.marlboro.edu/courses/spring2017/
Urls 34 found. The queue is now: 367
Visiting #16: http://cs.marlboro.edu/courses/spring2017/algorithms/
    home
Urls 20 found. The queue is now: 398
Visiting #15: http://cs.marlboro.edu/courses/spring2017/info/home
Urls 19 found. The queue is now: 412
Visiting #14: http://cs.marlboro.edu/courses/spring2017/3d/home
Urls 18 found. The queue is now: 425
[...]
```

The program produces a dot file named "network_graph.dot" in the current working directory. This dot file can be compiled into an image format with graphviz like such:

```
$ dot network_graph.dot -Tpdf -o graph.pdf
```

Sample output for different number of URLS processed:

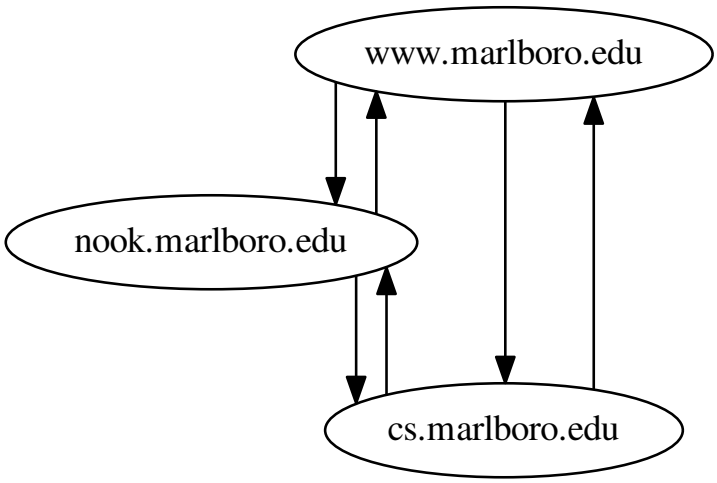


Figure 1: -n 30

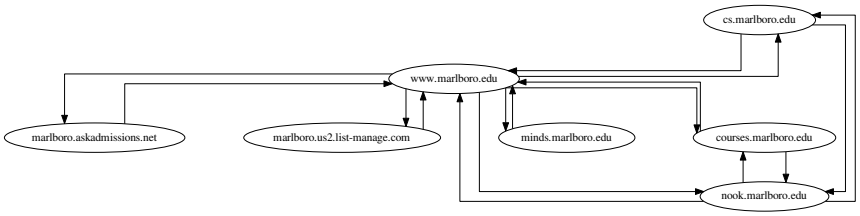


Figure 2: -n 300

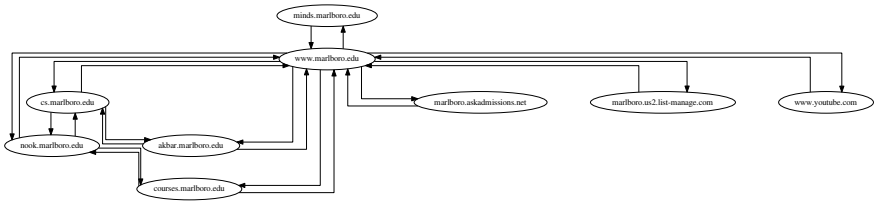


Figure 3: -n 1000

Notes

The brute force solutions to the fraction sum problem are pretty straightforward. Even though execution time isn't the best measure of what's going on, C and its compiled nature runs faster than Javascript and Python. There are two approaches to making an object oriented solution for the fraction sum problem: one, each permutation is it's own permutation or fraction class and those objects are then checked and two, a single a 'permutation generator' object is created and passed data. I chose the later for my object-oriented approach– it is essentially a wrapper class to Python's built in itertools permutation class. For all the python solutions I used a default library to generate permutations. In Javascript and C I generated the permutations myself. I believe the better paradigm here is functional or imperative depending on which you work faster in. The fraction problem does not lend itself to an object-oriented solution because that adds a lot of bloat memory and execution on top of a brute force method. In brute-force the name of the game is speed and memory management.

For the web crawler problem I went the route of creating one program that works better than three that are less featured given my time frame. When implementing HTTP and parsing HTML there is a lot to think about. First, you need to make requests and receive responses. This was made easier in Python with some built-in networking classes but in languages like C, you'd have to deal with buffering and TCP connections all on your own. Once you have a HTTP response parsing it for more links is a game of trade-offs. You can use regular expressions but that doesn't follow redirects, or javascript/vbscript code, or submit forms to potential new pages. Rendering the page and submitting forms, or processing javascript and faking clicks takes time but is ultimately a more complete a web crawl. HTTP responses can be compressed (in a number of different formats) or sent streamed (chunked) so you have to parse headers and have solutions for all of that. There might be file formats such as pdfs and images you don't want to download when crawling so you have to worry about black-listing or white-listing extensions. Because the question is pretty

open ended in what kind of graph to show with the collected data I chose to display a graph only of websites that link to one another. If a website A links to a website B, but B does not link to A, than that connection is not included in my graph.

Question 3

Discuss the strengths and weaknesses of these programming languages as you see them. Feel free to discuss a few other languages that you're familiar with as well. What sorts of problems or situations are good fits to these languages, and why? Which do you personally like, and why? Be specific, giving examples that justify your comparisons and conclusions. (This may well cover some ground you've already discussed in the previous two problems. If so, you don't have to repeat any of that, just refer back to it and bring up anything that you feel hasn't yet been brought forward.)

I subscribe to the notion that programming is 90% decision and 10% typing. There is a right tool for the job, and languages usually have an area they shine in. Every problem I'm trying to solve and each piece of software I code is a battle between efficiency and development time. Is learning GO to code a bleeding-edge web server worth the time? Is programming it in Python worth the loss in efficiency?

That being said I do have a favorite programming language: C. Furthermore, I prefer to code imperatively. I believe learning C makes you a better programmer in other languages. C is very minimal and close to computer architecture. There is no garbage collection and you have to worry about memory management yourself. The advantage to knowing C is that you have a very good idea of how a computer works. Not just how your programming executes, but how memory is laid out. The only level below C is the assembly spoken by a specific CPU. Although coding in assembly might provide an ounce of speedup, development would be slow. Even developing in C can be tiresome. When I am prototyping applications it is much faster for me to develop in an interpreted language like Python with dynamic typing. Such is the case with my Plan project. I started by using C++ but the learning curve was steeper than time allowed so I switched to Python– which beside being faster to develop in, has a bunch of useful built-in libraries that helped me out. I believe the speedup in development was more beneficial than an enterprise speed application.

I believe the superiority battle between imperative and functional program is simply answered by which one you develop in faster. Although learning both will help you become a better programmer and problem solver. Object-oriented

programming should be reserved for certain problems that lend themselves to that abstraction, such as: video games, web applications, and GUI development. Applications where you are interacting with "things". I do believe it is easier to write OOP MVC where a web page is a page object, than to code imperatively some event driven web server. OOP in my opinion is also more maintainable and easier to add features in. However, there is a reason neural net research is not done representing matrices as objects: it adds too much memory use and execution time. This is where statically compiled languages like C shine, and why you see tools like Cpython trying to compile python into C.

Python, javascript and PHP are all powerful languages. Javascript has grown from this cute scripting language to a back-end technology. I think its powerful for students because you can use it in a multitude of situations for web development. You can replace the whole linux-apache-mysql-php stack with just javascript now. That being said javascript/node has limitations in speed, like still being able to only utilize a single core of a web server. PHP has a special place in my heart as the first language I learned. It is ubiquitous on the web but I have my gripes with it. For one, it is sort of scattered brained. PHP is one of those languages people added to with a 'wouldn't it be cool if' mentality. Therefore things like functions don't follow a standard pattern– like which comes first the needle or haystack in a search function. There are about 1000 different ways to do the same thing in PHP which take it as you will, I'm not a fan of. Pythons main benefit in my mind is its readability. Although it is a personal preference, I find Python to be the cleanest language I've used. It is definitely more ubiquitous than javascript too. Besides being on computers and servers, Python is the poster child for the Internet of things. The major con to python for me is speed. Without using some sort of py to c third party tool, python is almost never the fastest solution. With that being said, Python is definitely easier to implement and prototype algorithms in than C. Although C is faster, Pythons built-in data structures and libraries of science related functions allow for rapid development.

Besides the execution times of my fraction sum solutions, a favorite language benchmark of mine to share is Gregory Hildstrom's done in 2015.[9] The test program he came up with dynamically allocates some heap memory once, initializes an array/vector, and then performs many 32-bit integer and 64-bit floating point calculations using that array. The algorithm does not calculate anything useful and it does not measure the performance of any single operation. It is just a low-level workload that was easy to port to various languages. For the intended number of calculations, a program in C ran in 206.1 seconds. A program in Cpython (python compiled to C) ran in 382.4 seconds. That's close to regular C, but regular interpreted Python took 24970 seconds. PHP beat out Python but not by anything to write home about at 19195 seconds. Javascript was surpisngly to me, the fastest of the interpreted languages at 1678.6 seconds.

Algorithms

out: Sun April 23 2017, due: Sun April 30 by midnight

This is open take-home exam: books or web sources are OK as long as the problem doesn't set other constraints, you cite them explicitly, and that they aren't a drop-in solution to the problem. However, the more your answer is a summary of someone else's article, the less we will be impressed. Don't ask other people for help. Don't just give a numerical result, give an explanation. Your job is to convince us you understand this stuff.

So in terms of a grading rubric, what you should think about is

- technical merit - correctness & thoroughness of response
- clarity of expression (including docs & tests for code)
- demonstration of understanding (vs summarizing others work)

Be very explicit about which sources you used for each problem. As always with my exams, if you think there's a mistake in one of the questions or it doesn't make sense, you can (a) ask for clarification, and/or (b) make and state an explicit interpretation and do the problem that way. (Again: the point is to show your mastery, not to get the "right answer" per se.)

Good luck.

Question 1

Explain, implement, explore numerically the O() behavior either Prim's or Kruskal's algorithm, which find the minimum spanning tree of a weighted graph.

The core of this work should be a numerical experiment in which you randomly generate graphs of different sizes, find either the time or number of steps the algorithm takes, use those to create plots of its behavior, and compare the shape of the plot with the expected result.

As usual with my assignments, your code should be clearly documented with explicit tests.

Be clear that you should choose one of these to do, not both.

Prim's Algorithm

Prim's is a greedy algorithm that finds a minimum spanning tree (MST) for a weighted undirected graph.[8] The MST is a subset of the graph that contains all the vertices without cycle with the least amount of edge weight. Frequently MST algorithms are used in random graph generation, however, I will be using the networkx library– installed by default in the Anaconda suite.[10]

Prims works as follows:

Start at an arbitrary vertex and add it to a list of visited vertices (which just contains this root one for now) Find the least costly edge from a vertice in the visited list to a vertice that hasn't been visited Add the vertice that you just visited to your list of visited and repeat until all vertices are marked visited

There are of course many different data strucures to represent the graph and implement the algorithm, and they effect the run-time complexity. I've chosen to represent my graph as an adjacency matrix– purely because of personal preference. Adjacency matrices use O(n*n) memory. They have constant look-up time to check for the presence or absence of a specific edge, but are slow to iterate over all edges. Adjacency lists on the other hand, use memory in proportion to the number edges, which might save a lot of memory if the adjacency matrix is sparse.

```
# Some library inclusion
import random
import networkx as nx # for random graph generation
```

Below is my graph generation function. Another reason I chose to represent the graph as adjacency matrix was so that I wasn't replying on networkx's class methods for retrieve edges or finding neighbor vertices. I wanted the algorithm part to be all my own code.

```
def gen_graph(n,p=1):
    """
    n = The number of nodes.
    p = Probability for edge creation.

    Returns an adjacency matrix representation of a random
    undirected graph weighted.
    """
    G = nx.fast_gnp_random_graph(n,1)
    for (u, v) in G.edges():
        G.edge[u][v]['weight'] = random.randint(0,10)
    return nx.adjacency_matrix(G).todense()

G = gen_graph(4)
print(G)
```

Output:

```
[[0  1  5  2]
 [1  0  7  0]
 [5  7  0  6]
 [2  0  6  0]]
```

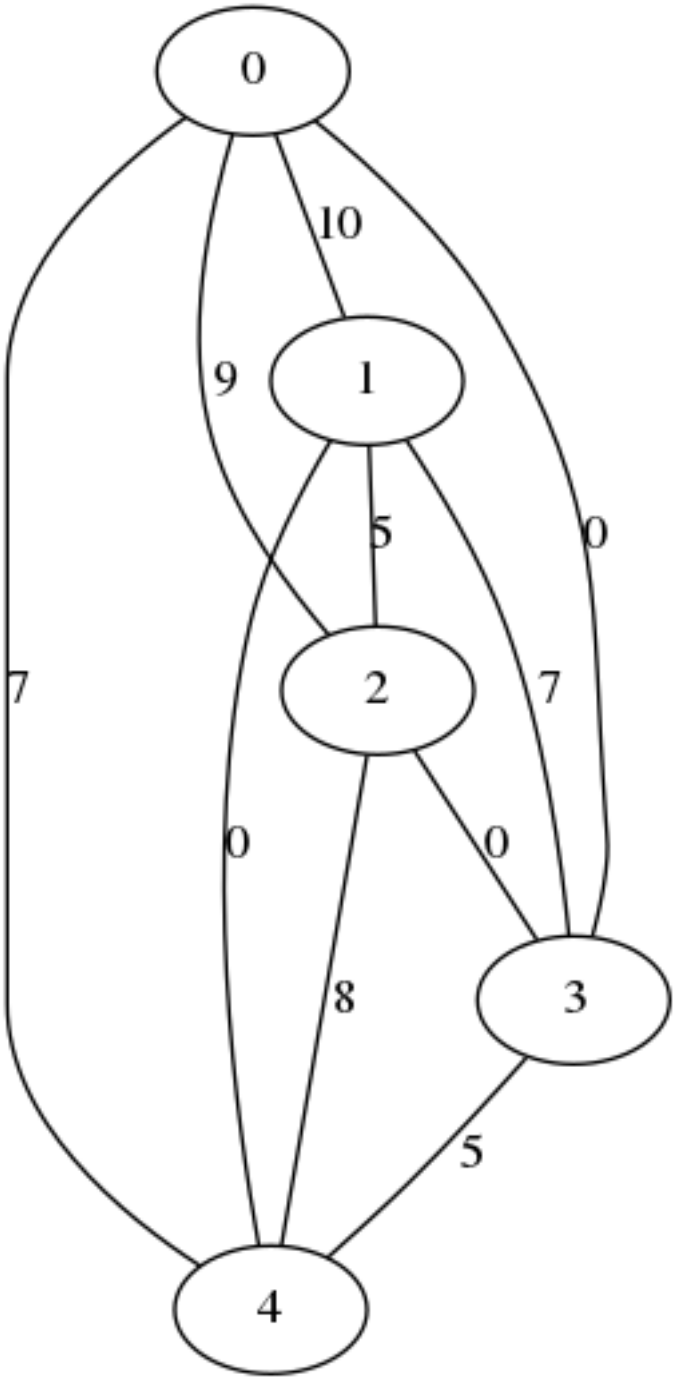
Following is my Prim's Algorithm implementation. Some implementations use a priority queue or two lists to manage nodes visited and not reached– I am using one list to mark nodes as seen or unseen (False or True). The lookup if a node has been reached is done in constant time: unreached[node]. While there are still unreached nodes, I find an edge from a visited node to one that has not been seen with the lowest cost. The MST ouputted is a graph in edge list form. The edges have the structure (node1,node2,weight).

```
def prim(G, start=0):
    unreached = [[True]]*len(G)
    unreached[start] = False

    MST = []

    while any(unreached):

        e = min([(x,y,G[(x,y)]) for x in range(len(unreached)) for
    y in range(len(unreached)) if not unreached[x] and unreached[y
    ]], key=lambda x: x[2])

        unreached[e[1]] = False

        MST.append(e)

    return MST

print(prim(G))
```

Output:

```
[(0, 1, 1), (1, 3, 0), (0, 2, 5)]
```

To visualize an MST I am using my code on a previously generated graph. The starting graph:

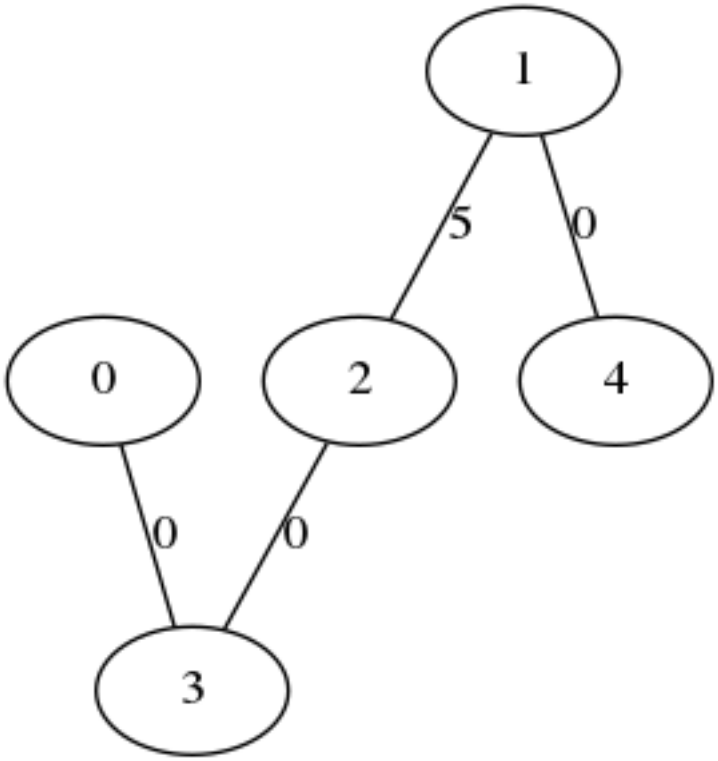

0
10
9
1
5
0
7
2
7
0
0
8
3
5
4

```
G = nx.adjacency_matrix(nx.convert.from_dict_of_dicts({0: {1: {'
    weight': 5}, 2: {'weight': 7}, 3: {'weight': 0}}, 1: {0: {'
    weight': 5}, 2: {'weight': 2}, 3: {'weight': 8}}, 2: {0: {'
    weight': 7}, 1: {'weight': 2}, 3: {'weight': 4}}, 3: {0: {'
    weight': 0}, 1: {'weight': 8}, 2: {'weight': 4}}, 4: {1: {'
    weight': 0}, 2: {'weight': 8}, 3: {'weight': 5}, 0: {'weight':
    7}}})).todense()
print("This is the adjacency matrix\n")
print(G)
print("\nThis is MST edge list (Node,Node,Weight)\n")
print(prim(G))
```

Output:

```
This is the adjacency matrix

[[0 5 7 0 7]
 [5 0 2 8 0]
 [7 2 0 4 8]
 [0 8 4 0 5]
 [7 0 8 5 0]]

This is MST edge list (Node,Node,Weight)

[(0, 3, 0), (3, 2, 4), (2, 1, 2), (1, 4, 0)]
```

The produced MST:

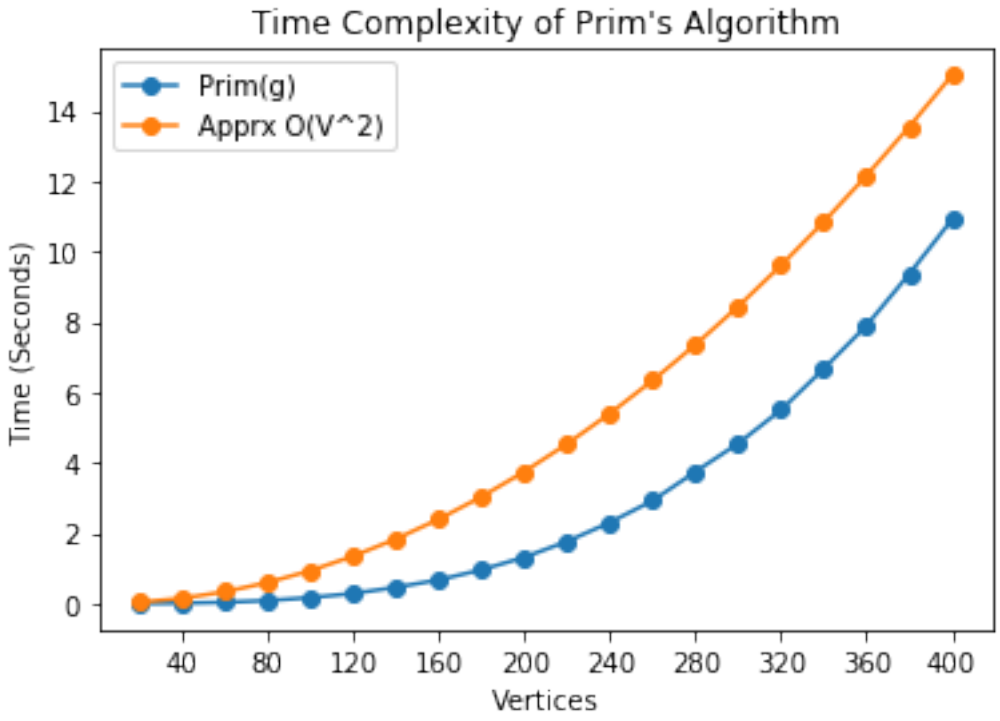


The expected time complexity of my implementation is O(V$\hat{2}$) using the adjacency matrix. An improvement could be made by using a priority heap to store all edges of the input graph, ordered by their weight. That would lead to an O(E*log(E)) worst-case running time.[1]

```
# This will take about two minutes on a 4-core processor
from matplotlib import pyplot
import timeit, functools

x,y = [],[]

for n in range(20,420,20):
    x.append(n)
    G = gen_graph(n)
    t = timeit.Timer(functools.partial(prim, G))
    y.append(t.timeit(1))

pyplot.title('Time Complexity of Prim\'s Algorithm')

l1=pyplot.plot(x, y, 'o-', label='Prim(g)')

avg = sum([y[i]/x[i] for i in range(len(x))])/len(x)

l2=pyplot.plot(x, [(avg*v)**2 for v in x], 'o-', label='Apprx O(V
```

```
    ^2)')

pyplot.legend()
pyplot.ylabel('Time (Seconds)')
pyplot.xlabel('Vertices')
pyplot.xticks(range(40,440,40))
pyplot.xticks(range(40,440,40))
pyplot.show()
```

Output:



Question 2

Illustrate a depth-first and breadth-first tree search, preferably using a stack and queue, using the tree of possible moves in a triangular peg solitaire game as the tree.

The game I have in mind starts with this configuration;

```
    .
   * *
  * * *
 * * * *
* * * * *
```

where each * is a peg in a hole, and the . is an empty hole. The goal is to remove pegs by jumping as in checkers, leaving one peg in the center as the final position.

Switching from a breadth-first search (BFS) to a depth-first search (DFS) is as simple as switching from a queue to stack. In a BFS, from the root node you add children to a queue. Those children are then visited at that depth and the process repeated for their children at another depth. If you switch to a stack you now have a DFS. You will search down an entire branch(s) before reaching a second child from the root. Arguably the hard part of this problem is the generation of the tree and representation of the game. I've chose to represent it in a pseudo matrix form, where each hole is a coordinate:

```
(0,0),(1,0),(2,0),(3,0),(4,0)
(0,1),(1,1),(2,1),(3,1)
(0,2),(1,2),(2,2)
(0,3),(1,3)
(0,4)
```

This would make the corners points (0,0),(4,0) and (0,4). The center is (1,2). Pegs are represented as astriks and holes are periods.

```
problem_board = [['*','*','*','*','*'], #0 - 0,1,2,3,4
                   ['*','*','*','*'],    #1 - 0,1,2,3
                     ['*','*','*'],      #2 - 0,1,2
                       ['*','*'],        #3 - 0,1
                         ['.']]          #4 - 0
```

Below are some utility functions for things like making moves, displaying the board, finding neighbors, and determining if a point is a peg or hole.

```
import copy

def print_board(board):
    """
    board = Array or arrays representing board.

    Given a board as
    [['*','*','*','*','*'],['*','*','*','*'], ['*','*','*'], [
    '*','*'],['.']],
    it pretty prints the board:
    * * * * *
     * * * *
```

```
      * * *
       * *
        .
    """
    i = 0
    for row in board:
        print("{}{}".format(" "*i,' '.join(row)))
        i+=1

def neighbors(p):
    """
    p = A point in tuple form (x,y)

    Returns a list of points surrounding it.

    >>>neighbors((1,2))
    [(2, 2), (0, 2), (0, 3), (1, 3), (1, 1), (2, 1)]
    """
    x = p[0]
    y = p[1]
    return list(filter(lambda c: c[0] >= 0 and c[0] <= 4-c[1] and c
    [1] >= 0 and c[1] < 5, [(x+1,y),(x-1,y),(x-1,y+1),(x,y+1),(x,y
    -1),(x+1,y-1)]))

def ispeg(p,b):
    """
    p = point (x,y)
    b = board in the form
    [['*','*','*','*','*'],['*','*','*','*','*'],['*','*','*'], [
    '*','*'],[  '.']]

    Returns true if the point is a '*' peg
    """
    return [(x,y) for x,y in filter(lambda c: b[c[1]][c[0]] == '*',
     p)]

def valid_moves(b):
    """
    b = board in the form
    [['*','*','*','*','*'],['*','*','*','*','*'],['*','*','*'], [
    '*','*'],[  '.']]

    Returns a list of valid moves on the board. Moves are in the
    tuple form (peg to move, peg to remove, peg to land on)
    where each of those is points in tuple form ((x,y),(x1,y1),(x2,
    y2))

    >>>valid_moves(board in the form
    [['*','*','*','*','*'],['*','*','*','*','*'],['*','*','*'], [
    '*','*'],[  '.']])
    [[(0, 2), (0, 3), (0, 4)], [(2, 2), (1, 3), (0, 4)]]
    """
    moves=[]
    for y in range(len(b)):
        for x in range(len(b[y])):
            if b[y][x] != '*':
                continue
```

```
            p = (x,y)
            n = neighbors(p)
            pn = ispeg(n,b)
            for m in pn:
                nx = 0
                ny = 0

                if m[0]==p[0]:
                    nx=m[0]
                if m[0] < p[0]:
                    nx=m[0]-1
                if m[0] > p[0]:
                    nx=m[0]+1

                if m[1]==p[1]:
                    ny=m[1]
                if m[1] < p[1]:
                    ny=m[1]-1
                if m[1] > p[1]:
                    ny=m[1]+1

                if (nx < 0 or ny < 0) or ((nx,ny) not in neighbors(
    m)) or b[ny][nx] == '*':
                    continue

                moves.append([p,m,(nx,ny)])
    return moves

def move(m,b):
    """
    m = Move to be made in the form ((x,y),(x1,y1),(x2,y2))
    b = board

    Returns a new board instance with the move made.
    """
    nb = copy.deepcopy(b)
    nb[m[0][1]][m[0][0]] = '.'
    nb[m[1][1]][m[1][0]] = '.'
    nb[m[2][1]][m[2][0]] = '*'
    return nb


def pegcount(b):
    """
    b = board

    Returns the number of pegs on the board.
    """
    return sum(x.count("*") for x in b)

def gen_tree(b):
    """
    b = Board

    Returns a reference to a root node of a tree of all possible
    moves.
```

```
    Nodes are in dict form:
    Node {
        board:b
        valid_moves:[]
        children:[]
    }
    Children of a node are references to nodes with the parent
    moves made on their board and so on.
    """
    n={"board":b}
    n["valid_moves"] = valid_moves(n["board"])
    n["children"] = []

    for m in n["valid_moves"]:
        child = gen_tree(move(m,b))
        child["move"] = [m]
        n["children"].append(child)

    return n
```

Generating the tree takes up most of the time – I average around 55 seconds. This is mainly due to me copying the board every move.

```
import time

s = time.time()
root = gen_tree(problem_board)
gentime = time.time()-s

print("It took {} seconds to generate the tree of moves".format(
    gentime))
```

Output:

```
It took 58.63520812988281 seconds to generate the tree of moves
```

Here is my depth-first search. Without a specific goal in mind, my search returns a dictionary containing a lot of information found along the way. I keep track of end games found, the moves to reach them, and also tally them based on the number of pegs left.

```
def dfs(root):

    stack = root["children"]
    results = {"info":{}}
    paths = {}

    while stack:
        node = stack.pop()

        children = node["children"]
        for c in children:
```

```
            c["move"]= node["move"]+c["move"]

        if not children:
            c = pegcount(node['board'])
            if c not in results["info"]:
                results["info"][c] = 0
                paths[c] = []
            results["info"][c] += 1
            paths[c].append(node["move"])

        stack.extend(children)

    results['paths'] = paths
    return results
```

My breadth-first search is the same code with the stack exchanged for a queue. I'm using python's collection library because a default list's method of pop(0) is not as efficient as deque's popleft().

```
from collections import deque

def bfs(root):

    queue = deque(root["children"])
    results = {"info":{}}
    paths = {}

    while queue:
        node = queue.popleft()

        children = node["children"]
        for c in children:
            c["move"]= node["move"]+c["move"]

        if not children:
            c = pegcount(node['board'])
            if c not in results["info"]:
                results["info"][c] = 0
                paths[c] = []
            results["info"][c] += 1
            paths[c].append(node["move"])

        queue.extend(children)

    results['paths'] = paths
    return results
```

Running the BFS:

```
# BFS
import time

s = time.time()
root = gen_tree(problem_board)
```

```
gentime = time.time()-s

s2 = time.time()
results = dfs(root)
dfstime = time.time()-s2

print("Tree generation took {} seconds.".format(gentime))
print("The BFS took {} seconds.".format(dfstime))

print("\nPegs\t\tNumber")
print(" Remaining\t of Games")
print("==========================")
for p,n in sorted(results["info"].items()):
    print("{}\t\t{}".format(p,n))
print("==========================")
print("Total: \t\t{}".format(sum(results["info"].values())))
```

Output:

```
Tree generation took 55.356807231903076 seconds.
The BFS took 3.0062286853790283 seconds.

Pegs     Number
 Remaining    of Games
==========================
1    29760
2    139614
3    259578
4    123664
5    14844
6    844
7    324
8    2
==========================
Total:     568630
```

Running the DFS:

```
# DFS
import time

s = time.time()
root = gen_tree(problem_board)
gentime = time.time()-s

s2 = time.time()
results = bfs(root)
bfstime = time.time()-s2

print("Tree generation took {} seconds.".format(gentime))
print("The BFS took {} seconds.".format(bfstime))

print("\nPegs\t\tNumber")
print(" Remaining\t of Games")
```

```
print("============================")
for p,n in sorted(results["info"].items()):
    print("{}\t\t{}".format(p,n))
print("============================")
print("Total: \t\t{}".format(sum(results["info"].values())))
```

Output:

```
Tree generation took 62.894432067871094 seconds.
The BFS took 2.132901668548584 seconds.

Pegs      Number
 Remaining    of Games
============================
1    29760
2    139614
3    259578
4    123664
5    14844
6    844
7    324
8    2
============================
Total:     568630
```

The time complexity of my queue implementaion is O(V+E) as the total time spent looking in the adjacency list is E (the number of edges) and all of V (vertices) are added to the queue in constant time. This is the same big O behavior for the DFS. Push and pop on the stack are constant so there is V nodes added, and E edges are visited. In my implementation both algorithms run about the same pace.

Interestingly, in all the games with 1 peg left not a single one ends with it in the center of the board (point (1,2)). My results are in agreement with Keith Wannamaker's results– he is a software engineer who published a paper on the number of games and various endings starting from different positions.[12]

```
for path in results["paths"][1]:
    if path[-1][-1] == (1,2):
        print(path)
# No games end in the center
```

Also in my returned results are the games themselves. Attached is results.txt which contain all 29760 games ending in one peg. Below is an example game of a random 1 peg solution.

```
b = problem_board
print("Starting position")
print_board(b)
for m in results["paths"][1][0]:
```

```
    b = move(m,b)
    print("Move {} to {} and remove {}".format(m[0],m[2],m[1]))
    print_board(b)
```

Output:

```
Starting position
* * * * *
 * * * *
  * * *
   * *
    .
Move (2, 2) to (0, 4) and remove (1, 3)
* * * * *
 * * * *
  * * .
   * .
    *
Move (0, 2) to (2, 2) and remove (1, 2)
* * * * *
 * * * *
  . . *
   * .
    *
Move (0, 4) to (0, 2) and remove (0, 3)
* * * * *
 * * * *
  * . *
   . .
    .
Move (3, 1) to (1, 3) and remove (2, 2)
* * * * *
 * * * .
  * . .
   . *
    .
Move (0, 1) to (0, 3) and remove (0, 2)
* * * * *
 . * * .
  . . .
   * *
    .
Move (3, 0) to (1, 2) and remove (2, 1)
* * * . *
 . * . .
  . * .
   * *
    .
Move (2, 0) to (0, 2) and remove (1, 1)
* * . . *
 . . . .
  * * .
   * *
    .
Move (1, 3) to (1, 1) and remove (1, 2)
* * . . *
```

```
 . * . .
  * . .
   * .
    .
Move (0, 3) to (0, 1) and remove (0, 2)
* * . . *
 * * . .
  . . .
   . .
    .
Move (0, 0) to (0, 2) and remove (0, 1)
. * . . *
 . * . .
  * . .
   . .
    .
Move (0, 2) to (2, 0) and remove (1, 1)
. * * . *
 . . . .
  . . .
   . .
    .
Move (1, 0) to (3, 0) and remove (2, 0)
. . . * *
 . . . .
  . . .
   . .
    .
Move (4, 0) to (2, 0) and remove (3, 0)
. . * . .
 . . . .
  . . .
   . .
    .
```

More interesting I find is the worst-case senario from the starting position. There are only two "solutions" that end with 8 pegs and end in the same position. For example:

```
b = problem_board
print("Starting position")
print_board(b)
for m in results["paths"][8][0]:
    b = move(m,b)
    print("Move {} to {} and remove {}".format(m[0],m[2],m[1]))
    print_board(b)
```

Output:

```
Starting position
* * * * *
 * * * *
  * * *
   * *
```

```
    .
Move (0, 2) to (0, 4) and remove (0, 3)
* * * * *
 * * * *
  . * *
   . *
    *
Move (2, 0) to (0, 2) and remove (1, 1)
* * . * *
 * . * *
  * * *
   . *
    *
Move (3, 1) to (1, 1) and remove (2, 1)
* * . * *
 * * . .
  * * *
   . *
    *
Move (0, 1) to (2, 1) and remove (1, 1)
* * . * *
 . . * .
  * * *
   . *
    *
Move (2, 2) to (2, 0) and remove (2, 1)
* * * * *
 . . . .
  * * .
   . *
    *
Move (0, 4) to (2, 2) and remove (1, 3)
* * * * *
 . . . .
  * * *
   . .
    .
```

Question 3

The task of finding a given string from a collection of strings can be solved by algorithms that fit into several paradigms, including brute force and (if the list is sorted) divide-and-conquer.

- Given an example of an algorithm that fits each of these descriptions, and describe their O() behavior.
- Can you come up with a third algorithm design technique that can be applied to this problem, a representative algorithm, and its O() behavior?

(You don't need to code these - just discuss what's going on.)

In string matching the most basic approach is bruteforcing– a variation on sequential searching. With N being the search space or collection of strings, and M being pattern to look for, bruteforcing works as follows:

For each character in N, compare it to the first character of M. If they don't match, you move forward a character in N. If the two characters match, then continue matching the next character of N and M. You continue this search pattern until you find M within N or not at all.

Bruteforcing is quite slow, and the time complexity is O(N*M) as you compare each element of N against M. One way to improve runtime might be pre-processing the list to be sorted in some desired way.[11]

If the search space is already sorted, a binary search would be an optimal divide-and-conquer algorithm. Given a sorted search space N, a binary search first compares the middle element of N with search pattern M. If the pattern matches the middle element, its position is returned. If the pattern is less than the middle value, the process is repeated on the lower half of the search space until found. If the pattern is greater than the middle value, the process is repeated on the upper half. Since each comparison in the binary search halves the search space, it is easy to assert that the time complexity is a logarithm: O(log(N)).[13]

Greedy algorithms are another paradigm of pattern matching algorithms. An algorithm behind some of the fastest search algorithms out there is the Boyer-Moore algorithm. In the vanilla Boyer-Moore algorithm you search for occurrences of the pattern M in the set N by performing explicit character comparisons at different alignments. Unlike in bruteforcing, you skip as many alignments as possible using heuristic information from pre-processing the set. The shifts are calculated on two rules: the good-suffix shift and the bad-character shift. Pre-processing the set takes O(N) time but the searching happens in O(N*M). I've included the Boyer-Moore algorithm in the greedy algorithm category but it sort of lives in limbo between greedy and bruteforce. It uses heuristic information like a greedy algorithm, but pure greedy algorithms usually return a sub-optimal solution.[15][16]

Question 4

You're given a list of N names and told that you'll need to search the list for a given name M times. The following are suggested:

- using a hash table
- using a heap
- sequential search
- sorting followed by binary search

Discuss the time efficiency of these approaches as the size of N and M vary. Which one would you suggest and why? Would your answer change if you needed to change the list of names by adding and deleting names (say, P times) between searches?

A hash table would be the best solution– even more so if you wanted to add or delete names. Search, insertion, and deletion operatons all happen in constant time. The only constraint would be picking a good hash function that provides a uniform distribution of hashes. The less uniform it is, the more you need to compensate for collisions with execution time.

Sorting the array and using a binary search would leave you with a binary tree as search, insertion, and deletion operatons would happen in O(log(N)) time. Although a hastable is a faster implementation for the problem at hand, it fails for relative searchs– things like the "next smallest" or "one greater than". The binary search tree would provide this relation-type information.

My second choice would be a heap. The structure is tree-like similar to binary search tree except nodes must be ordered either greatest to least or least to greatest. Insertion and deletion happens in O(long(N)) but finding the min or max would be constant time. Again, this is not relavant to the problem but is a benefit over the binary search tree.

A sequential search (as talked about in question 3) would be my last resot option. Insertion, deletion, and search would all be O(N).

References

[1] OS Platform Statistics. (n.d.). Retrieved April 29, 2017, from https://www.w3schools.com/browsers/browsers_os.asp

[2] DLL injection. (2017, April 20). In Wikipedia, The Free Encyclopedia. Retrieved April 29, 2017, from https://en.wikipedia.org/w/index.php?title=DLL_injection&oldid=776305093

[3] Shared Libraries, TLDP, Retrieved April 29, 2017, from tldp.org/HOWTO/Program-Library-HOWTO/shared-libraries.html.

[4] Sourceware. (n.d.). Summary of GDB. Retrieved April 29, 2017, from https://sourceware.org/gdb/onlinedocs/gdb/Summary.html

[5] H., W. (n.d.). Intel 80x86 Assembly Language OpCodes. Retrieved April 29, 2017, from http://www.mathemainzel.info/files/x86asmref.html

[6] Padala, P. (n.d.). Playing with ptrace, Part I. Retrieved April 29, 2017, from http://www.linuxjournal.com/article/6100

[7] Mips42. (n.d.). Mips42. Retrieved April 30, 2017, fromhttp://mips42.altervista.org/ptrace.php

[8] Prim's algorithm. (2017, March 21). In Wikipedia, The Free Encyclopedia. Retrieved May, 2017, from https://en.wikipedia.org/w/index.php?title=Prim%27s_algorithm&oldid=771475622

[9] Hildstrom, G. (n.d.). Programming Language Performance Comparison. Retrieved May, 2017, from http://hildstrom.com/projects/langcomp/index.html

[10] Anaconda package list. (n.d.). Retrieved May, 2017, from https://docs.continuum.io/anaconda/pkg-docs

[11] Stoimen, C. (2012, March 12). Computer Algorithms: Brute Force String Matching. Retrieved May, 2017, from http://www.stoimen.com/blog/2012/03/27/computer-algorithms-brute-force-string-matching/

[12] Wannamaker, K. (2016, January). The Cracker Barrel Peg Game. Retrieved from http://wannamaker.org/math/TheCrackerBarrelPegGame.pdf

[13] Binary search algorithm. (2017, April 27). In Wikipedia, The Free Encyclopedia. Retrieved 08:41, May 1, 2017, from https://en.wikipedia.org/w/index.php?title=Binary_search_algorithm&oldid=777552439

[14] Linux Journal. (n.d.). Retrieved April 30, 2017, from http://man7.org/linux/man-pages/man2/ptrace.2.html

[15] Boyer–Moore string search algorithm. (2017, January 15). In Wikipedia, The Free Encyclopedia. Retrieved 08:44, May 1, 2017, from https://en.wikipedia.org/w/index.php?title=Boyer%E2%80%93Moore_string_search_algorithm

[16] Lecroq, T. (n.d.). Boyer-Moore Algorithm. Retrieved May, 2017, from http://www-igm.univ-mlv.fr/~lecroq/string/node14.html

[17] Bellevue Linux. (2006, April 26). System Call Definition. Retrieved April 30, 2017, from http://www.linfo.org/system_call.html

[18] TLDP. (n.d.). A Kernel Module. Retrieved April 30, 2017, from http://www.tldp.org/LDP/lkmpg/2.6/html/x40.html

[19] The art of Memory Forensics Detecting Malware and Threats in Windows, Linux, and Mac Memory. (2014). Indianapolis, IN: John Wiley & Sons, Inc.

[20] Watchfire. (n.d.). Retrieved April 30, 2017, from http://testfire.net/

Appendices

ctf.c

```
#define _GNU_SOURCE
#include <stdio.h>
#include <stdlib.h>
#include <sys/uio.h>

#define FLAG_OFFSET 0x201014

int main(int argc, char const *argv[]){
  pid_t pid;
  struct iovec local[1];
  struct iovec remote[1];
  ssize_t nread;

  pid = atoi(argv[1]);


  char fname[255];
    FILE *f;

    sprintf(fname, "/proc/%i/maps", pid);
    f = fopen(fname, "r");

    char buff[255];
    fgets(buff, 255, f);

    long long unsigned int base_addr;

    sscanf(buff,"%Lx",&base_addr);

  unsigned char buf[sizeof(int)];

  int n = 1;

  for (int i = 0; i < sizeof(int)*2; ++i){
    buf[i] = n>>i*sizeof(int)*2;
  }

  local[0].iov_base = buf;
  local[0].iov_len = sizeof(int);

  remote[0].iov_base = (void*)base_addr+FLAG_OFFSET;
  remote[0].iov_len = sizeof(int);

  nread = process_vm_writev(pid, local, 1, remote, 1, 0);

  if (nread){
    printf("Captured the flag!\n");
  }
  return 0;
}
```

externalhack.c

```
/*

Run with:
gcc hack.c -o hack && sudo hack

*/


#define _GNU_SOURCE
#include <sys/uio.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdarg.h>
#include <unistd.h>
#include <sys/types.h>
#include <linux/limits.h>

#include <stdint.h>

#define RED   "\x1B[31m"
#define GRN   "\x1B[32m"
#define YEL   "\x1B[33m"
#define BLU   "\x1B[34m"
#define MAG   "\x1B[35m"
#define CYN   "\x1B[36m"
#define WHT   "\x1B[37m"
#define RESET "\x1B[0m"

typedef void *addr; // Using void pointers to reference and compare
     memory addresses

void print_bytes(char* buffer, ssize_t len){

  for (int i = 0; i < len; ++i){
    printf("%X ", buffer[i] & 0xFF);
  }
  printf("\n");

  return;
}

pid_t getPID(char* procname){

  pid_t pid;
  char line[10];

  char str[35];
  strcpy(str, "pidof  ");
  strcat(str, procname);

  FILE *cmd = popen(str, "r");

  fgets(line, 10, cmd);
```

```
  pid = strtoul(line, NULL, 10);

  pclose(cmd);

  return pid;
}

void privcheck(){

  uid_t uid=getuid(), euid=geteuid();
  if (uid != 0 || uid!=euid) {
    printf("PROBLEM: Need to run as root.\n");
    exit(0);
  }

  return;
}

addr getbaseaddr(pid_t pid){

  char fname[PATH_MAX];
    FILE *f;

    sprintf(fname, "/proc/%i/maps", pid);
    f = fopen(fname, "r");

    char buff[255];
    fgets(buff, 255, f);
    fgets(buff, 255, f);
    fgets(buff, 255, f); /* because i need better way for assault
    cube to parse mpas file third line */

    long long unsigned int base_addr;

    sscanf(buff,"%Lx",&base_addr);

  return (addr)base_addr;
}


ssize_t read_bytes(pid_t pid, addr mem_addr, char* buf) {

}

int bytesToInt(unsigned char* b){
  int val = 0;
  int j = 0;
  for (int i = 0; i < 4; ++i)
  {
    val += (b[i] & 0xFF) << (8*j);
    ++j;
  }
  return val;
}

long bytesToLong(unsigned char* b){
```

```
  uint64_t n = 0;

  n += ((uint64_t)b[7]<<56);
  n += ((uint64_t)b[6]<<48);
  n += ((uint64_t)b[5]<<40);
  n += ((uint64_t)b[4]<<32);
  n += ((uint64_t)b[3]<<24);
  n += ((uint64_t)b[2]<<16);
  n += ((uint64_t)b[1]<<8);
  n += ((uint64_t)b[0]<<0);

  return n;
}

int intToBytes(int n, char *buf){

  for (int i = 0; i < 8; ++i){
    buf[i] = n>>i*8;
  }

  return 0;
}

long read_long(pid_t pid, addr addy){
  struct iovec local[1];
  struct iovec remote[1];

  int nbytes = 8;
  unsigned char buf[nbytes];
  ssize_t nread;

  local[0].iov_base = buf;
  local[0].iov_len = nbytes;

  remote[0].iov_base = addy;
  remote[0].iov_len = nbytes;

  nread = process_vm_readv(pid, local, 1, remote, 1, 0);

  return bytesToLong(buf);
}

int read_int(pid_t pid, addr addy){
  struct iovec local[1];
  struct iovec remote[1];

  int nbytes = 4;
  char buf[nbytes];
  ssize_t nread;

  local[0].iov_base = buf;
  local[0].iov_len = nbytes;

  remote[0].iov_base = addy;
  remote[0].iov_len = nbytes;
```

```
  nread = process_vm_readv(pid, local, 1, remote, 1, 0);

  return bytesToInt(buf);
}

int write_int(pid_t pid, addr addy, int n){

  unsigned char buf[4];
  intToBytes(n,buf);

  struct iovec local[1];
  struct iovec remote[1];

  ssize_t nread;

  local[0].iov_base = buf;
  local[0].iov_len = 4;

  remote[0].iov_base = addy;
  remote[0].iov_len = 4;

  nread = process_vm_writev(pid, local, 1, remote, 1, 0);

  return nread;
}


int main(void){

  privcheck(); // Make sure we are running as root (
    process_vm_readv and writev require this)

  char* name = "assaultcube";

  pid_t pid = getPID(name);

  if (pid == 0){
    printf("PROBLEM: Process not found.\n");
    exit(0);
  }

  printf(YEL "Found PID: \t" RESET "%i\n", pid);

  addr base_addr = getbaseaddr(pid);


    addr player1p = base_addr+0x378130;
    addr player1 = (addr)read_long(pid,player1p);
    addr player1_vh = player1+0x110;


    printf(YEL "Found Base Address: \t" RESET "%p\n", base_addr);
    printf(YEL "Player Pointer: \t" RESET "%p\n", player1p);
    printf(YEL "Player Address: \t" RESET "%p\n", player1);
    printf(YEL "Player V Health: \t" RESET "%p\n", player1_vh);
    printf("Health: %i\n", read_int(pid,player1_vh));
```

```
    printf(YEL "Hacked Health" RESET "\n");

    while (1) {
      write_int(pid,(addr)player1_vh,1337);
    }


    return 0;
}
```

flag.c

```
#include <stdio.h>

int main(int argc, char const *argv[]){

  int flag = 0;

  if (flag){
    printf("Flag is true\n");
  } else {
    printf("Flag is false\n");
  }

  return 0;
}
```

flag2.c

```
#include <stdio.h>
#include <unistd.h>

int flag;

int main(){
    flag = 0;

    while(1){
        printf("Flag = %i\n", flag);
        sleep(2);
    }

    return 0;
}
```

inspectprintf.c

```
#define _GNU_SOURCE
#include <dlfcn.h>
#include <stdio.h>
#include <string.h>

static int (*real_puts)(const char* str) = NULL;

int puts(const char* str){

    /* printing out the number of characters */
    printf("puts:chars#:%lu\n", strlen(str));

    /* resolve the real puts function from glibc
     * and pass the arguments.
     */
    real_puts = dlsym(RTLD_NEXT, "puts");
    real_puts(str);
}
```

lkm.c

```
#include <linux/module.h> /* Needed by all modules */
#include <linux/kernel.h> /* Needed for KERN_INFO */
#include <linux/init.h>   /* Needed for the macros */
#include <linux/mm.h>
#include <linux/highmem.h>
#include <linux/pid.h>
#include <linux/ptrace.h>
#include <linux/pid.h>
#include <linux/uaccess.h>
#include <linux/sched.h>
#include <linux/errno.h>
#include <linux/pagemap.h>
#include <linux/ptrace.h>
#include <linux/security.h>
#include <asm/pgtable.h>
#include <asm/uaccess.h>
#define DRIVER_AUTHOR "Dylan Murphy-Mancini"
#define DRIVER_DESC   "'Hello, World' LKM"

MODULE_LICENSE("GPL");
MODULE_AUTHOR(DRIVER_AUTHOR); /* Who wrote this module? */
MODULE_DESCRIPTION(DRIVER_DESC);  /* What does this module do */
MODULE_SUPPORTED_DEVICE("testdevice");


struct task_struct *task;
struct pid *pid_struct;
struct mm_struct *mm;
struct vm_area_struct *vma;
struct page *page;
char buf[128];
void *old_buf = buf;
pid_t pid;

static int __init init_func(void){

  printk("Hello, world init\n");

  pid = 28035;

  pid_struct = find_get_pid(pid);
  task = pid_task(pid_struct,PIDTYPE_PID);

  rcu_read_lock();

  if (task){
    mm = task->mm;

    printk("\nThis mm_struct has %d vmas.\n", mm->map_count);
    printk("\nCode  Segment start = 0x%lx, end = 0x%lx \n"
                "Data  Segment start = 0x%lx, end = 0x%lx\n"
                "Stack Segment start = 0x%lx\n",
                mm->start_code, mm->end_code,
                mm->start_data, mm->end_data,
```

```
                    mm->start_stack);
  }

  rcu_read_unlock();

  return 0;
}

static void __exit cleanup_func(void){
  printk(KERN_INFO "Hello, world cleanup\n");
}

module_init(init_func);
module_exit(cleanup_func);
```

printfecho.c

```
#include <stdio.h>

int main(int args, char *argv[]) {
  printf("%s\n", argv[1]);
  return 0;
}
```

ptest.c

```
#include <stdio.h>
#include <sys/ptrace.h>

int main(){
    if (ptrace(PTRACE_TRACEME, 0, 1, 0) == -1) {
        printf("Don't trace me!!\n");
        return 1;
    }

    printf("Normal Execution... no tracing going on here.\n");
    return 0;
}
```

random.c

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main(int argc, char const *argv[]){

  while(1) {
    printf("%d\n",rand()%100);
    sleep(1);
  }

  return 0;
}
```

target.c

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main(int argc, char const *argv[]){

  while(1) {
    printf("%d\n",rand()%100);
    sleep(1);
  }

  return 0;
}
```

tracer.c

```
#include <sys/ptrace.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <sys/syscall.h>
#include <sys/reg.h>
#include <sys/user.h>
#include <unistd.h>
#include <string.h>
#include <errno.h>

#define TARGET "/home/us3r/Desktop/plan/target"
#define NEW_UID 1337

int main(int argc,char** argv) {
  int status = 0;
  int syscall_n = 0;
  int entering = 1;
  struct user_regs_struct regs;
  int pid = fork();

  if (!pid) {
    ptrace( PTRACE_TRACEME, 0, 0, 0 );
    execv( TARGET, argv );
  } else {
    wait( &status );

    while (1) {
      ptrace( PTRACE_SYSCALL, pid, 0, 0 );

      wait( &status );

      if ( WIFEXITED( status ) ) break;

      ptrace( PTRACE_GETREGS, pid, NULL, &regs );
      syscall_n = regs.orig_rax;
      if ( syscall_n == SYS_getuid ) {
        if ( entering ) {
          entering = 0;
        }
        else {
          ptrace( PTRACE_GETREGS, pid, 0, &regs );
          regs.rax = NEW_UID;
          ptrace( PTRACE_SETREGS, pid, 0, &regs );
          entering = 1;
        }
      }
    }
  }

  return 0;
}
```

unrandom.c

```
int rand(){
    return 42;
}
```

Boan

boan.py

```
#!/usr/bin/env python3

import sys
import os
from time import sleep
### Proxy ###
import proxy
from urllib.parse import urlparse
### GUI ###
from PyQt5 import uic
from PyQt5.QtWidgets import QMainWindow, QApplication
from PyQt5.QtGui import *
from PyQt5.QtCore import *
from PyQt5.QtWidgets import *

mainwindow_ui_file = "resources/g.ui" #Qt XML ui file.
Ui_MainWindow, QtBaseClass = uic.loadUiType(mainwindow_ui_file)

waitCondition = QWaitCondition()
mutex = QMutex()

class SettingsWindow(QDialog):
    def __init__(self):
        super(SettingsWindow, self).__init__()
        uic.loadUi('resources/settings.ui', self)
        self.settings = QSettings("Boan", "Boan")
        self.loadSettings()
        self.show()

    def loadSettings(self):
        self.spinBox_port.setValue(int(self.settings.value("port",
    8080)))
        self.filter_filetypes.setPlainText(self.settings.value("
    filter_filetypes", ".css, .gif, .ico, .png"))
        self.filter_hosts.setPlainText(self.settings.value("
    filter_hosts"))

    def accept(self):
        self.settings.setValue("port",self.spinBox_port.value())
        self.settings.setValue("filter_filetypes",self.
    filter_filetypes.toPlainText())
        self.settings.setValue("filter_hosts",self.filter_hosts.
    toPlainText())
        self.close()

class AboutWindow(QDialog):
    def __init__(self):
        super(AboutWindow, self).__init__()
        uic.loadUi('resources/about.ui', self)
        self.textBrowser.setSource(QUrl("resources/about.htm"))
        self.show()
```

```
    def accept(self):
        self.close()

class MyApp(QMainWindow, Ui_MainWindow):
    def __init__(self):
        QMainWindow.__init__(self)
        Ui_MainWindow.__init__(self)
        self.setupUi(self)
        self.px = ProxyThread() # Proxy Qthread
        self.settings = QSettings("Boan", "Boan")
        self.connect()
        self.centerOnScreen()

    def connect(self):
        #Shortcuts
        self.shortcut_quit = QShortcut(QKeySequence("Ctrl+Q"), self
    )
        self.shortcut_quit.activated.connect(self.on_quit)
        self.shortcut_settings = QShortcut(QKeySequence("Ctrl+Shift
    +S"), self)
        self.shortcut_settings.activated.connect(self.do_settings)

        #Menu
        self.actionSettings.triggered.connect(self.do_settings)
        self.actionAbout.triggered.connect(self.do_about)
        self.actionQuit.triggered.connect(self.on_quit)

        #Buttons
        self.pushButton_forward.clicked.connect(self.
    handleButton_forward)
        self.pushButton_power.clicked[bool].connect(self.
    handleButton_power)
        self.px.statusbarsignal.connect(self.update_statusbar)
        self.px.reqsignal.connect(self.handle_reqsignal)

    def centerOnScreen (self):
        '''centerOnScreen() Centers the window on the screen.'''
        resolution = QDesktopWidget().screenGeometry()
        self.move((resolution.width() / 2) - (self.frameSize().
    width() / 2),(resolution.height() / 2) - (self.frameSize().
    height() / 2))

    def do_settings(self):
        dlg = SettingsWindow()
        dlg.exec_()

    def do_about(self):
        dlg = AboutWindow()
        dlg.exec_()

    def on_quit(self):
        self.close()

    def handleButton_forward(self):
        self.pushButton_forward.setEnabled(False)
        self.px.raw_req = self.box_body.toPlainText()
```

```
        self.box_body.clear()
        self.wakeup()
        return

    def handleButton_power(self,isPressed):
        if isPressed:
            self.px['port'] = int(self.settings.value("port", 8080)
    )
            self.px.start()
            self.pushButton_power.setText("Running...")
        else:
            self.px.stop()
            self.pushButton_power.setText("Run")
            self.pushButton_forward.setEnabled(False)

    def update_statusbar(self,msg,msec):
        self.statusbar.clearMessage()
        self.statusbar.showMessage(msg,msec)

    def handle_reqsignal(self):
        self.raise_()
        self.show()
        self.activateWindow()

        self.box_body.append(self.px.req.command+" "+self.px.req.
    path+" "+self.px.req.protocol_version);
        for h in self.px.req.headers:
            self.box_body.append(h+": "+self.px.req.headers[h])
        self.box_body.append("") # newline
        if self.px.req_body:
            self.box_body.append(self.px.req_body.decode("utf-8"))
        self.pushButton_forward.setEnabled(True)

    def wakeup(self):
        waitCondition.wakeAll()

class PX(proxy.ProxyRequestHandler):

    settings = QSettings("Boan", "Boan")

    def request_handler(self, req, req_body):

        filter_hosts= self.settings.value("filter_hosts")
        if filter_hosts and (req.headers['Host'] not in
    filter_hosts):
            return

        filter_filetypes= self.settings.value("filter_filetypes").
    split(", ")
        if any([x in req.path for x in filter_filetypes]):
            return

        self.pt.req = req
        self.pt.req_body = req_body
        self.reqsignal.emit()
```

```
        mutex.lock()
        waitCondition.wait(mutex)
        mutex.unlock()

        # Parse the raw request and map onto the httpreq object
        raw = self.pt.raw_req.split('\n')
        req.raw_req = self.pt.raw_req
        try:
            req.command, req.path, req.protocol_version = raw[0].
    split()
        except Exception as e:
            self.send_error(400)

        for c in range(1,len(raw)):
            #print(c,raw[c])
            if raw[c] == '': # the newline
                break
            try:
                h,v = raw[c].split(': ')
            except Exception as e:
                self.send_error(400)

            del req.headers[h]
            req.headers[h] = v

        req_body = bytes('\n'.join(raw[c+1:]),"utf-8")

        return req_body

    def wakey(self):
        self.wakeup()

class ProxyThread(QThread):
    port = 8080
    statusbarsignal = pyqtSignal(str,int)
    reqsignal = pyqtSignal()
    updsignal = pyqtSignal(str)
    req = None
    req_body = None
    raw_req = None

    def __setitem__(self, key, item):
        self.__dict__[key] = item

    def run(self):
        HandlerClass=PX
        ServerClass=proxy.ThreadingHTTPServer

        protocol="HTTP/1.1"
        server_address = ('', self.port)
        HandlerClass.protocol_version = protocol

        HandlerClass.pt = self
        HandlerClass.statusbarsignal = self.statusbarsignal
        HandlerClass.reqsignal = self.reqsignal

        self.httpd = ServerClass(server_address, HandlerClass)
```

```
        sa = self.httpd.socket.getsockname()

        self.statusbarsignal.emit("Listening on "+sa[0]+" port "+
    str(sa[1])+"...",0)
        self.httpd.serve_forever()

    def stop(self):
        self.httpd.server_close()
        self.terminate()

if __name__ == "__main__":
    app = QApplication(sys.argv)
    window = MyApp()
    window.show()
    sys.exit(app.exec_())
```

proxy.py

```
# -*- coding: utf-8 -*-
import sys
import os
import socket
import ssl
import select
import http.client
import urllib.parse
import threading
from urllib.parse import urlparse
import gzip
import zlib
import time
import json
import re
from http.server import HTTPServer, BaseHTTPRequestHandler
from socketserver import ThreadingMixIn
from io import BytesIO
from subprocess import Popen, PIPE
from html.parser import HTMLParser


def with_color(c, s):
    return "\x1b[%dm%s\x1b[0m" % (c, s)


class ThreadingHTTPServer(ThreadingMixIn, HTTPServer):
    address_family = socket.AF_INET6
    daemon_threads = True

class ProxyRequestHandler(BaseHTTPRequestHandler):
    cakey = 'ca.key'
    cacert = 'ca.crt'
    certkey = 'cert.key'
    certdir = 'certs/'
    timeout = 5
    #_headers_buffer = []
    lock = threading.Lock()

    def __init__(self, request, client_address, server):
        self.tls = threading.local()
        self.tls.conns = {}
        BaseHTTPRequestHandler.__init__(self, request,
    client_address, server)

    def log_message(self, format, *args):
        # Queit the http server
        #sys.stderr.write("%s - - [%s] %s\n" % (self.address_string
    (), self.log_date_time_string(),format%args))
        pass

    def log_error(self, format, *args):
        # surpress "Request timed out: timeout('timed out',)"
        if isinstance(args[0], socket.timeout):
```

```
            return
        self.log_message(format, *args)

    def do_CONNECT(self):
        if os.path.isfile(self.cakey) and os.path.isfile(self.
    cacert) and os.path.isfile(self.certkey) and os.path.isdir(self
    .certdir):
            self.connect_intercept()
        else:
            self.connect_relay()

    def connect_intercept(self):
        hostname = self.path.split(':')[0]
        certpath = "%s/%s.crt" % (self.certdir.rstrip('/'),
    hostname)

        with self.lock:
            if not os.path.isfile(certpath):
                epoch = "%d" % (time.time() * 1000)
                p1 = Popen(["openssl", "req", "-new", "-key", self.
    certkey, "-subj", "/CN=%s" % hostname], stdout=PIPE)
                p2 = Popen(["openssl", "x509", "-req", "-days",
    "3650", "-CA", self.cacert, "-CAkey", self.cakey, "-set_serial
    ", epoch, "-out", certpath], stdin=p1.stdout, stderr=PIPE)
                p2.communicate()

        self.wfile.write(bytes("%s %d %s\r\n\n" % (self.
    protocol_version, 200, 'Connection Established'),"utf-8"))
        #self.wfile.write(bytes("HTTP/1.1 200 Connection
    established\r\n\r\n","utf-8"))

        self.connection = ssl.wrap_socket(self.connection, keyfile=
    self.certkey, certfile=certpath, suppress_ragged_eofs=True,
    server_side=True)
        self.rfile = self.connection.makefile("rb", self.rbufsize)
        self.wfile = self.connection.makefile("wb", self.wbufsize)

        conntype = self.headers.get('Proxy-Connection', '')
        if self.protocol_version == "HTTP/1.1" and conntype.lower()
     != 'close':
            self.close_connection = 0
        else:
            self.close_connection = 1

    def connect_relay(self):
        address = self.path.split(':', 1)
        address[1] = int(address[1]) or 443
        try:
            s = socket.create_connection(address, timeout=self.
    timeout)
        except Exception as e:
            self.send_error(502)
            return
        self.send_response(200, 'Connection Established')
        self.end_headers()

        conns = [self.connection, s]
```

```
        self.close_connection = 0
        while not self.close_connection:
            rlist, wlist, xlist = select.select(conns, [], conns,
    self.timeout)
            if xlist or not rlist:
                break
            for r in rlist:
                other = conns[1] if r is conns[0] else conns[0]
                data = r.recv(8192)
                if not data:
                    self.close_connection = 1
                    break
                other.sendall(data)

    def do_GET(self):
        if "boan.cert" in self.path:
            print("Send the cert")
            self.send_cacert()
            return

        req = self
        content_length = int(req.headers.get('Content-Length', 0))
        req_body = self.rfile.read(content_length) if
    content_length else None

        if req.path[0] == '/':
            if isinstance(self.connection, ssl.SSLSocket):
                req.path = "https://%s%s" % (req.headers['Host'],
    req.path)
            else:
                req.path = "http://%s%s" % (req.headers['Host'],
    req.path)
        with self.lock:
            req_body_modified = self.request_handler(req, req_body)

        if req_body_modified is False:
            self.send_error(403)
            return
        elif req_body_modified is not None:
            req_body = req_body_modified
            del req.headers['Content-length']
            req.headers['Content-length'] = str(len(req_body))

        u = urllib.parse.urlsplit(req.path)
        scheme, netloc, path = u.scheme, u.netloc, (u.path + '?' +
    u.query if u.query else u.path)
        assert scheme in ('http', 'https')
        if netloc:
            req.headers['Host'] = netloc
        setattr(req, 'headers', self.filter_headers(req.headers))

        try:
            origin = (scheme, netloc)
            if not origin in self.tls.conns:
                if scheme == 'https':
                    self.tls.conns[origin] = http.client.
    HTTPSConnection(netloc, timeout=self.timeout)
```

```
            else:
                self.tls.conns[origin] = http.client.
HTTPConnection(netloc, timeout=self.timeout)
        conn = self.tls.conns[origin]

        #if req.raw_req != None:
        #    print("raw")
        #    reqq = "GET http://example.com/ HTTP/1.1\nHost:
example.com\nUser-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64
; rv:50.0) Gecko/20100101 Firefox/50.0\nAccept: text/html,
application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\nAccept-
Language: en-US,en;q=0.5\nAccept-Encoding: gzip, deflate\
nConnection: keep-alive\nUpgrade-Insecure-Requests: 1\nCache-
Control: max-age=0\n"
        #    reqq = reqq.replace('\n', '\r\n')
        #    reqq += "\r\n"
        #    print(repr(reqq))
        #    conn.connect()
        #    conn.send(bytes(reqq,"utf-8"))
        # As an alternative to using the request() method
described above, you can also send your request step by step,
by using the four functions below.
        # https://docs.python.org/3.0/library/http.client.html
        #else:
        conn.request(self.command, path, req_body, dict(req.
headers))

        res = conn.getresponse()
        version_table = {10: 'HTTP/1.0', 11: 'HTTP/1.1'}
        setattr(res, 'headers', res.msg)
        setattr(res, 'response_version', version_table[res.
version])

        # support streaming
        if not 'Content-Length' in res.headers and 'no-store'
in res.headers.get('Cache-Control', ''):
            self.response_handler(req, req_body, res, '')
            setattr(res, 'headers', self.filter_headers(res.
headers))
            self.relay_streaming(res)
            with self.lock:
                self.save_handler(req, req_body, res, '')
            return

        res_body = res.read()
    except Exception as e:
        if origin in self.tls.conns:
            del self.tls.conns[origin]
        self.send_error(502)
        return

    content_encoding = res.headers.get('Content-Encoding', '
identity')
    res_body_plain = self.decode_content_body(res_body,
content_encoding)
```

```
        res_body_modified = self.response_handler(req, req_body,
    res, res_body_plain)
        if res_body_modified is False:
            self.send_error(403)
            return
        elif res_body_modified is not None:
            res_body_plain = res_body_modified
            res_body = self.encode_content_body(res_body_plain,
    content_encoding)
            res.headers['Content-Length'] = str(len(res_body))

        setattr(res, 'headers', self.filter_headers(res.headers))

        #if req.raw_req != None:
        #    print("sent raw")
#
        #
         #reqq = "GET http://example.com/ HTTP/1.1\nHost: example.
    com\nUser-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv
    :50.0) Gecko/20100101 Firefox/50.0\nAccept: text/html,
    application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\nAccept-
    Language: en-US,en;q=0.5\nAccept-Encoding: gzip, deflate\
    nConnection: keep-alive\nUpgrade-Insecure-Requests: 1\nCache-
    Control: max-age=0\n"
         #   reqq = reqq.replace('\n', '\r\n')
         #   reqq += "\r\n"
           # print(repr(reqq))
          #  self.wfile.write(bytes(reqq,"ascii"))
           # self.wfile.flush()
           # return
            #pass

        self.send_response(res.status)
        for k,v in res.headers.items():
            self.send_header(k,v)
        self.end_headers()
        try:
            self.wfile.write(res_body)
        except Exception as e:
            #print("error")
            pass
        self.wfile.flush()

        with self.lock:
            self.save_handler(req, req_body, res, res_body_plain)

    def relay_streaming(self, res):
        self.wfile.write(bytes("%s %d %s\r\n" % (self.
    protocol_version, res.status, res.reason),"utf-8"))
        for k,v in res.headers.items():
            self.send_header(k,v)
        self.end_headers()
        try:
            while True:
                chunk = res.read(8192)
                if not chunk:
                    break
```

```
                self.wfile.write(chunk)
            self.wfile.flush()
        except socket.error:
            # connection closed by client
            pass

    do_HEAD = do_GET
    do_POST = do_GET
    do_PUT = do_GET
    do_DELETE = do_GET
    do_OPTIONS = do_GET

    def filter_headers(self, headers):
        # http://tools.ietf.org/html/rfc2616#section-13.5.1
        hop_by_hop = ('connection', 'keep-alive', 'proxy-
    authenticate', 'proxy-authorization', 'te', 'trailers', '
    transfer-encoding', 'upgrade')
        for k in hop_by_hop:
            del headers[k]

        # accept only supported encodings
        if 'Accept-Encoding' in headers:
            ae = headers['Accept-Encoding']
            filtered_encodings = [x for x in re.split(r',\s*', ae)
    if x in ('identity', 'gzip', 'x-gzip', 'deflate')]
            del headers['Accept-Encoding']
            headers['Accept-Encoding'] = ', '.join(
    filtered_encodings)

        return headers

    def encode_content_body(self, text, encoding):
        if encoding == 'identity':
            data = text
        elif encoding in ('gzip', 'x-gzip'):
            io = BytesIO()
            with gzip.GzipFile(fileobj=io, mode='wb') as f:
                f.write(text)
            data = io.getvalue()
        elif encoding == 'deflate':
            data = zlib.compress(text)
        else:
            raise Exception("Unknown Content-Encoding: %s" %
    encoding)
        return data

    def decode_content_body(self, data, encoding):
        if encoding == 'identity':
            text = data
        elif encoding in ('gzip', 'x-gzip'):
            io = BytesIO(data)
            with gzip.GzipFile(fileobj=io) as f:
                text = f.read()
        elif encoding == 'deflate':
            try:
                text = zlib.decompress(data)
            except zlib.error:
```

```
                text = zlib.decompress(data, -zlib.MAX_WBITS)
        else:
            raise Exception("Unknown Content-Encoding: %s" %
    encoding)
        return text

    def send_cacert(self):
        with open(self.cacert, 'rb') as f:
            data = f.read()

        self.wfile.write(bytes("%s %d %s\r\n" % (self.
    protocol_version, 200, 'OK'),"utf-8"))
        self.send_header('Content-Type', 'application/x-x509-ca-
    cert')
        self.send_header('Content-Length', len(data))
        self.send_header('Connection', 'close')
        self.end_headers()
        self.wfile.write(data)

    def print_info(self, req, req_body, res, res_body):
        #print(req.path)
        #print res_header_text
        pass

    def request_handler(self, req, req_body):
        #print(req.path)
        pass

    def response_handler(self, req, req_body, res, res_body):
        #print(res.status)
        pass

    def save_handler(self, req, req_body, res, res_body):
        #self.print_info(req, req_body, res, res_body)
        pass


def test(HandlerClass=ProxyRequestHandler, ServerClass=
    ThreadingHTTPServer, protocol="HTTP/1.1"):
    if sys.argv[1:]:
        port = int(sys.argv[1])
    else:
        port = 8080
    server_address = ('', port)

    HandlerClass.protocol_version = protocol
    httpd = ServerClass(server_address, HandlerClass)

    sa = httpd.socket.getsockname()
    print("Serving HTTP Proxy on", sa[0], "port", sa[1], "...")
    httpd.serve_forever()

if __name__ == '__main__':
    test()
```

proxy.py

```
# -*- coding: utf-8 -*-
import sys
import os
import socket
import ssl
import select
import http.client
import urllib.parse
import threading
from urllib.parse import urlparse
import gzip
import zlib
import time
import json
import re
from http.server import HTTPServer, BaseHTTPRequestHandler
from socketserver import ThreadingMixIn
from io import BytesIO
from subprocess import Popen, PIPE
from html.parser import HTMLParser


def with_color(c, s):
    return "\x1b[%dm%s\x1b[0m" % (c, s)


class ThreadingHTTPServer(ThreadingMixIn, HTTPServer):
    address_family = socket.AF_INET6
    daemon_threads = True

class ProxyRequestHandler(BaseHTTPRequestHandler):
    cakey = 'ca.key'
    cacert = 'ca.crt'
    certkey = 'cert.key'
    certdir = 'certs/'
    timeout = 5
    #_headers_buffer = []
    lock = threading.Lock()

    def __init__(self, request, client_address, server):
        self.tls = threading.local()
        self.tls.conns = {}
        BaseHTTPRequestHandler.__init__(self, request,
    client_address, server)

    def log_message(self, format, *args):
        # Queit the http server
        #sys.stderr.write("%s - - [%s] %s\n" % (self.address_string
    (), self.log_date_time_string(),format%args))
        pass

    def log_error(self, format, *args):
        # surpress "Request timed out: timeout('timed out',)"
        if isinstance(args[0], socket.timeout):
```

```
            return
        self.log_message(format, *args)

    def do_CONNECT(self):
        if os.path.isfile(self.cakey) and os.path.isfile(self.
    cacert) and os.path.isfile(self.certkey) and os.path.isdir(self
    .certdir):
            self.connect_intercept()
        else:
            self.connect_relay()

    def connect_intercept(self):
        hostname = self.path.split(':')[0]
        certpath = "%s/%s.crt" % (self.certdir.rstrip('/'),
    hostname)

        with self.lock:
            if not os.path.isfile(certpath):
                epoch = "%d" % (time.time() * 1000)
                p1 = Popen(["openssl", "req", "-new", "-key", self.
    certkey, "-subj", "/CN=%s" % hostname], stdout=PIPE)
                p2 = Popen(["openssl", "x509", "-req", "-days",
    "3650", "-CA", self.cacert, "-CAkey", self.cakey, "-set_serial
    ", epoch, "-out", certpath], stdin=p1.stdout, stderr=PIPE)
                p2.communicate()

        self.wfile.write(bytes("%s %d %s\r\n\n" % (self.
    protocol_version, 200, 'Connection Established'),"utf-8"))
        #self.wfile.write(bytes("HTTP/1.1 200 Connection
    established\r\n\r\n","utf-8"))

        self.connection = ssl.wrap_socket(self.connection, keyfile=
    self.certkey, certfile=certpath, suppress_ragged_eofs=True,
    server_side=True)
        self.rfile = self.connection.makefile("rb", self.rbufsize)
        self.wfile = self.connection.makefile("wb", self.wbufsize)

        conntype = self.headers.get('Proxy-Connection', '')
        if self.protocol_version == "HTTP/1.1" and conntype.lower()
     != 'close':
            self.close_connection = 0
        else:
            self.close_connection = 1

    def connect_relay(self):
        address = self.path.split(':', 1)
        address[1] = int(address[1]) or 443
        try:
            s = socket.create_connection(address, timeout=self.
    timeout)
        except Exception as e:
            self.send_error(502)
            return
        self.send_response(200, 'Connection Established')
        self.end_headers()

        conns = [self.connection, s]
```

```
        self.close_connection = 0
        while not self.close_connection:
            rlist, wlist, xlist = select.select(conns, [], conns,
    self.timeout)
            if xlist or not rlist:
                break
            for r in rlist:
                other = conns[1] if r is conns[0] else conns[0]
                data = r.recv(8192)
                if not data:
                    self.close_connection = 1
                    break
                other.sendall(data)

    def do_GET(self):
        if "boan.cert" in self.path:
            print("Send the cert")
            self.send_cacert()
            return

        req = self
        content_length = int(req.headers.get('Content-Length', 0))
        req_body = self.rfile.read(content_length) if
    content_length else None

        if req.path[0] == '/':
            if isinstance(self.connection, ssl.SSLSocket):
                req.path = "https://%s%s" % (req.headers['Host'],
    req.path)
            else:
                req.path = "http://%s%s" % (req.headers['Host'],
    req.path)
        with self.lock:
            req_body_modified = self.request_handler(req, req_body)

        if req_body_modified is False:
            self.send_error(403)
            return
        elif req_body_modified is not None:
            req_body = req_body_modified
            del req.headers['Content-length']
            req.headers['Content-length'] = str(len(req_body))

        u = urllib.parse.urlsplit(req.path)
        scheme, netloc, path = u.scheme, u.netloc, (u.path + '?' +
    u.query if u.query else u.path)
        assert scheme in ('http', 'https')
        if netloc:
            req.headers['Host'] = netloc
        setattr(req, 'headers', self.filter_headers(req.headers))

        try:
            origin = (scheme, netloc)
            if not origin in self.tls.conns:
                if scheme == 'https':
                    self.tls.conns[origin] = http.client.
    HTTPSConnection(netloc, timeout=self.timeout)
```

```
            else:
                self.tls.conns[origin] = http.client.
HTTPConnection(netloc, timeout=self.timeout)
        conn = self.tls.conns[origin]

        #if req.raw_req != None:
        #    print("raw")
        #    reqq = "GET http://example.com/ HTTP/1.1\nHost:
example.com\nUser-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64
; rv:50.0) Gecko/20100101 Firefox/50.0\nAccept: text/html,
application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\nAccept-
Language: en-US,en;q=0.5\nAccept-Encoding: gzip, deflate\
nConnection: keep-alive\nUpgrade-Insecure-Requests: 1\nCache-
Control: max-age=0\n"
        #    reqq = reqq.replace('\n', '\r\n')
        #    reqq += "\r\n"
        #    print(repr(reqq))
        #    conn.connect()
        #    conn.send(bytes(reqq,"utf-8"))
        # As an alternative to using the request() method
described above, you can also send your request step by step,
by using the four functions below.
        # https://docs.python.org/3.0/library/http.client.html
        #else:
        conn.request(self.command, path, req_body, dict(req.
headers))

        res = conn.getresponse()
        version_table = {10: 'HTTP/1.0', 11: 'HTTP/1.1'}
        setattr(res, 'headers', res.msg)
        setattr(res, 'response_version', version_table[res.
version])

        # support streaming
        if not 'Content-Length' in res.headers and 'no-store'
in res.headers.get('Cache-Control', ''):
            self.response_handler(req, req_body, res, '')
            setattr(res, 'headers', self.filter_headers(res.
headers))
            self.relay_streaming(res)
            with self.lock:
                self.save_handler(req, req_body, res, '')
            return

        res_body = res.read()
    except Exception as e:
        if origin in self.tls.conns:
            del self.tls.conns[origin]
        self.send_error(502)
        return

    content_encoding = res.headers.get('Content-Encoding', '
identity')
    res_body_plain = self.decode_content_body(res_body,
content_encoding)
```

```
        res_body_modified = self.response_handler(req, req_body,
    res, res_body_plain)
        if res_body_modified is False:
            self.send_error(403)
            return
        elif res_body_modified is not None:
            res_body_plain = res_body_modified
            res_body = self.encode_content_body(res_body_plain,
    content_encoding)
            res.headers['Content-Length'] = str(len(res_body))

        setattr(res, 'headers', self.filter_headers(res.headers))

        #if req.raw_req != None:
        #    print("sent raw")
#
        #
         #reqq = "GET http://example.com/ HTTP/1.1\nHost: example.
    com\nUser-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv
    :50.0) Gecko/20100101 Firefox/50.0\nAccept: text/html,
    application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\nAccept-
    Language: en-US,en;q=0.5\nAccept-Encoding: gzip, deflate\
    nConnection: keep-alive\nUpgrade-Insecure-Requests: 1\nCache-
    Control: max-age=0\n"
         #   reqq = reqq.replace('\n', '\r\n')
         #   reqq += "\r\n"
           # print(repr(reqq))
          #  self.wfile.write(bytes(reqq,"ascii"))
           # self.wfile.flush()
           # return
            #pass

        self.send_response(res.status)
        for k,v in res.headers.items():
            self.send_header(k,v)
        self.end_headers()
        try:
            self.wfile.write(res_body)
        except Exception as e:
            #print("error")
            pass
        self.wfile.flush()

        with self.lock:
            self.save_handler(req, req_body, res, res_body_plain)

    def relay_streaming(self, res):
        self.wfile.write(bytes("%s %d %s\r\n" % (self.
    protocol_version, res.status, res.reason),"utf-8"))
        for k,v in res.headers.items():
            self.send_header(k,v)
        self.end_headers()
        try:
            while True:
                chunk = res.read(8192)
                if not chunk:
                    break
```

```
                self.wfile.write(chunk)
            self.wfile.flush()
        except socket.error:
            # connection closed by client
            pass

    do_HEAD = do_GET
    do_POST = do_GET
    do_PUT = do_GET
    do_DELETE = do_GET
    do_OPTIONS = do_GET

    def filter_headers(self, headers):
        # http://tools.ietf.org/html/rfc2616#section-13.5.1
        hop_by_hop = ('connection', 'keep-alive', 'proxy-
    authenticate', 'proxy-authorization', 'te', 'trailers', '
    transfer-encoding', 'upgrade')
        for k in hop_by_hop:
            del headers[k]

        # accept only supported encodings
        if 'Accept-Encoding' in headers:
            ae = headers['Accept-Encoding']
            filtered_encodings = [x for x in re.split(r',\s*', ae)
    if x in ('identity', 'gzip', 'x-gzip', 'deflate')]
            del headers['Accept-Encoding']
            headers['Accept-Encoding'] = ', '.join(
    filtered_encodings)

        return headers

    def encode_content_body(self, text, encoding):
        if encoding == 'identity':
            data = text
        elif encoding in ('gzip', 'x-gzip'):
            io = BytesIO()
            with gzip.GzipFile(fileobj=io, mode='wb') as f:
                f.write(text)
            data = io.getvalue()
        elif encoding == 'deflate':
            data = zlib.compress(text)
        else:
            raise Exception("Unknown Content-Encoding: %s" %
    encoding)
        return data

    def decode_content_body(self, data, encoding):
        if encoding == 'identity':
            text = data
        elif encoding in ('gzip', 'x-gzip'):
            io = BytesIO(data)
            with gzip.GzipFile(fileobj=io) as f:
                text = f.read()
        elif encoding == 'deflate':
            try:
                text = zlib.decompress(data)
            except zlib.error:
```

```
                text = zlib.decompress(data, -zlib.MAX_WBITS)
        else:
            raise Exception("Unknown Content-Encoding: %s" %
    encoding)
        return text

    def send_cacert(self):
        with open(self.cacert, 'rb') as f:
            data = f.read()

        self.wfile.write(bytes("%s %d %s\r\n" % (self.
    protocol_version, 200, 'OK'),"utf-8"))
        self.send_header('Content-Type', 'application/x-x509-ca-
    cert')
        self.send_header('Content-Length', len(data))
        self.send_header('Connection', 'close')
        self.end_headers()
        self.wfile.write(data)

    def print_info(self, req, req_body, res, res_body):
        #print(req.path)
        #print res_header_text
        pass

    def request_handler(self, req, req_body):
        #print(req.path)
        pass

    def response_handler(self, req, req_body, res, res_body):
        #print(res.status)
        pass

    def save_handler(self, req, req_body, res, res_body):
        #self.print_info(req, req_body, res, res_body)
        pass


def test(HandlerClass=ProxyRequestHandler, ServerClass=
    ThreadingHTTPServer, protocol="HTTP/1.1"):
    if sys.argv[1:]:
        port = int(sys.argv[1])
    else:
        port = 8080
    server_address = ('', port)

    HandlerClass.protocol_version = protocol
    httpd = ServerClass(server_address, HandlerClass)

    sa = httpd.socket.getsockname()
    print("Serving HTTP Proxy on", sa[0], "port", sa[1], "...")
    httpd.serve_forever()

if __name__ == '__main__':
    test()
```

setup_https_intercept.sh

```
#!/bin/sh

openssl genrsa -out ca.key 2048
openssl req -new -x509 -days 3650 -key ca.key -out ca.crt -subj "/
    CN=proxy2 CA"
openssl genrsa -out cert.key 2048
mkdir certs/
```

Resources

about.htm

```
<h1>Boan</h1>

<p>
<a href="https://github.com/0x64796c616e/Boan">https://github.com/0
    x64796c616e/Boan</a>
</p>

<p>
About this program—
</p>
```

about.ui

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>Dialog</class>
 <widget class="QDialog" name="Dialog">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>400</width>
    <height>300</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>About Boan</string>
  </property>
  <layout class="QGridLayout" name="gridLayout">
   <item row="0" column="0">
    <widget class="QTextBrowser" name="textBrowser">
     <property name="openExternalLinks">
      <bool>true</bool>
     </property>
     <property name="openLinks">
      <bool>false</bool>
     </property>
    </widget>
   </item>
  </layout>
 </widget>
 <resources/>
 <connections/>
</ui>
```

g.ui

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>MainWindow</class>
 <widget class="QMainWindow" name="MainWindow">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>431</width>
    <height>604</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>Boan</string>
  </property>
  <widget class="QWidget" name="centralwidget">
   <layout class="QGridLayout" name="gridLayout">
    <item row="1" column="0" colspan="2">
     <widget class="Line" name="line">
      <property name="orientation">
       <enum>Qt::Horizontal</enum>
      </property>
     </widget>
    </item>
    <item row="2" column="0" colspan="2">
     <layout class="QVBoxLayout" name="verticalLayout">
      <item>
       <widget class="QLabel" name="label_2">
        <property name="text">
         <string>Request</string>
        </property>
       </widget>
      </item>
      <item>
       <widget class="QTextEdit" name="box_body">
        <property name="focusPolicy">
         <enum>Qt::ClickFocus</enum>
        </property>
       </widget>
      </item>
      <item>
       <widget class="QPushButton" name="pushButton_forward">
        <property name="enabled">
         <bool>false</bool>
        </property>
        <property name="text">
         <string>Forward</string>
        </property>
        <property name="default">
         <bool>false</bool>
        </property>
        <property name="flat">
         <bool>false</bool>
        </property>
```

```
      </widget>
     </item>
    </layout>
   </item>
   <item row="0" column="0" colspan="2">
    <widget class="QPushButton" name="pushButton_power">
     <property name="text">
      <string>Run</string>
     </property>
     <property name="checkable">
      <bool>true</bool>
     </property>
     <property name="autoDefault">
      <bool>false</bool>
     </property>
     <property name="default">
      <bool>true</bool>
     </property>
     <property name="flat">
      <bool>false</bool>
     </property>
    </widget>
   </item>
  </layout>
 </widget>
 <widget class="QMenuBar" name="menubar">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>431</width>
    <height>25</height>
   </rect>
  </property>
  <widget class="QMenu" name="menuFile">
   <property name="title">
    <string>File</string>
   </property>
   <addaction name="actionQuit"/>
  </widget>
  <widget class="QMenu" name="menuHelp">
   <property name="title">
    <string>Help</string>
   </property>
   <addaction name="actionAbout"/>
  </widget>
  <widget class="QMenu" name="menuEdit">
   <property name="title">
    <string>Edit</string>
   </property>
   <addaction name="actionSettings"/>
  </widget>
  <addaction name="menuFile"/>
  <addaction name="menuEdit"/>
  <addaction name="menuHelp"/>
 </widget>
 <widget class="QStatusBar" name="statusbar"/>
```

```
  <action name="actionQuit">
   <property name="text">
    <string>Quit</string>
   </property>
   <property name="toolTip">
    <string>Quit (CTRL+Q)</string>
   </property>
   <property name="shortcut">
    <string>Ctrl+Q</string>
   </property>
  </action>
  <action name="actionSettings">
   <property name="text">
    <string>Settings</string>
   </property>
   <property name="shortcut">
    <string>Ctrl+Shift+S</string>
   </property>
  </action>
  <action name="actionAbout">
   <property name="text">
    <string>About Boan</string>
   </property>
  </action>
 </widget>
 <resources/>
 <connections/>
</ui>
```

settings.ui

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>Dialog</class>
 <widget class="QDialog" name="Dialog">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>400</width>
    <height>300</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>Dialog</string>
  </property>
  <layout class="QGridLayout" name="gridLayout">
   <item row="3" column="0" colspan="2">
    <widget class="QLabel" name="label_3">
     <property name="text">
      <string>In-Scope Hosts</string>
     </property>
    </widget>
   </item>
   <item row="0" column="0">
    <widget class="QLabel" name="label">
     <property name="text">
      <string>Port</string>
     </property>
    </widget>
   </item>
   <item row="4" column="3">
    <widget class="QDialogButtonBox" name="buttonBox">
     <property name="contextMenuPolicy">
      <enum>Qt::DefaultContextMenu</enum>
     </property>
     <property name="orientation">
      <enum>Qt::Horizontal</enum>
     </property>
     <property name="standardButtons">
      <set>QDialogButtonBox::Cancel|QDialogButtonBox::Ok</set>
     </property>
    </widget>
   </item>
   <item row="2" column="3">
    <widget class="QLabel" name="label_4">
     <property name="text">
      <string>Comma seperated list</string>
     </property>
    </widget>
   </item>
   <item row="3" column="3">
    <widget class="QPlainTextEdit" name="filter_hosts">
     <property name="placeholderText">
```

```
      <string notr="true">Enter a comma seperated List for ex.
    example.com, subdom.example.com, example.io</string>
     </property>
    </widget>
   </item>
   <item row="1" column="0" colspan="2">
    <widget class="QLabel" name="label_2">
     <property name="text">
      <string>Blacklist Filetypes:</string>
     </property>
    </widget>
   </item>
   <item row="1" column="3">
    <widget class="QPlainTextEdit" name="filter_filetypes"/>
   </item>
   <item row="0" column="1">
    <widget class="QSpinBox" name="spinBox_port">
     <property name="baseSize">
      <size>
       <width>0</width>
       <height>0</height>
      </size>
     </property>
     <property name="focusPolicy">
      <enum>Qt::ClickFocus</enum>
     </property>
     <property name="maximum">
      <number>65536</number>
     </property>
     <property name="value">
      <number>8080</number>
     </property>
    </widget>
   </item>
  </layout>
 </widget>
 <resources/>
 <connections>
  <connection>
   <sender>buttonBox</sender>
   <signal>accepted()</signal>
   <receiver>Dialog</receiver>
   <slot>accept()</slot>
   <hints>
    <hint type="sourcelabel">
     <x>248</x>
     <y>254</y>
    </hint>
    <hint type="destinationlabel">
     <x>157</x>
     <y>274</y>
    </hint>
   </hints>
  </connection>
  <connection>
   <sender>buttonBox</sender>
   <signal>rejected()</signal>
```

```
   <receiver>Dialog</receiver>
   <slot>reject()</slot>
   <hints>
    <hint type="sourcelabel">
     <x>316</x>
     <y>260</y>
    </hint>
    <hint type="destinationlabel">
     <x>286</x>
     <y>274</y>
    </hint>
   </hints>
  </connection>
 </connections>
</ui>
```

Algorithms Exam


```
{
 "cells": [
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "# Algorithms Exam\n",
    "Dylan Murphy-Mancini\n",
    "\n",
    "April 27, 2017"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "## Question 1\n",
    "\n",
    "Explain, implement, explore numerically the O() behavior
    either\n",
    "Prim's or Kruskal's algorithm, which find the minimum spanning
     tree\n",
    "of a weighted graph.\n",
    "\n",
    "The core of this work should be a numerical experiment in
    which you\n",
    "randomly generate graphs of different sizes, find either the
    time or\n",
    "number of steps the algorithm takes, use those to create plots
     of its\n",
    "behavior, and compare the shape of the plot with the expected
    result.\n",
    "\n",
    "As usual with my assignments, your code should be clearly\n",
    "documented with explicit tests.\n",
    "\n",
    "Be clear that you should choose one of these to do, not both."
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "### Prim's Algorithm\n",
    "\n",
    "Prim's is a greedy algorithm that finds a minimum spanning
    tree (MST) for a weighted undirected graph.<sup>[1]</sup> The
    MST is a subset of the graph that contains all the vertices
    without cycle with the least amount of edge weight. Frequently
    MST algorithms are used in random graph generation, however, I
    will be using the networkx library — installed by default in
    the Anaconda suite.<sup>[2]</sup> \n",
    "\n",
    "Prims works as follows:\n",
```

```
    "* Start at an arbitrary vertex and add it to a list of visited
     vertices (which just contains this root one for now)\n",
    "* Find the least costly edge from a vertice in the visited
    list to a vertice that hasn't been visited\n",
    "* Add the vertice that you just visited to your list of
    visited and repeat until all vertices are marked visited\n",
    "\n",
    "There are of course many different datastrucures to represent
    the graph and implement the algorithm, and they effect the run-
    time complexity. I've chosen to represent my graph as an
    adjacency matrix— purely because of personal preference.
    Adjacency matrices use O(n*n) memory. They have constant lookup
     time to check for the presence or absence of a specific edge,
    but are slow to iterate over all edges. Adjacency lists on the
    otherhand, use memory in proportion to the number edges, which
    might save a lot of memory if the adjacency matrix is sparse. "
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 1,
   "metadata": {
    "collapsed": true
   },
   "outputs": [],
   "source": [
    "# Some library inclusion\n",
    "import random\n",
    "import networkx as nx # for random graph generation"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "Below is my graph generation function. Another reason I chose
    to represent the graph as adjacency matrix was so that I wasn't
     replying on networkx's class methods for retrieve edges or
    finding neighbor vertices. I wanted the algorithm part to be
    all my own code."
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 5,
   "metadata": {
    "collapsed": false
   },
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "[[0 1 5 2]\n",
      " [1 0 7 0]\n",
      " [5 7 0 6]\n",
      " [2 0 6 0]]\n"
```

```
    ]
   }
  ],
  "source": [
   "def gen_graph(n,p=1):\n",
   "    \"\"\"\n",
   "    n = The number of nodes.\n",
   "    p = Probability for edge creation.\n",
   "    \n",
   "    Returns an adjacency matrix representation of a random
   undirected graph weighted.\n",
   "    \"\"\"\n",
   "    G = nx.fast_gnp_random_graph(n,1)\n",
   "    for (u, v) in G.edges():\n",
   "        G.edge[u][v]['weight'] = random.randint(0,10)\n",
   "    return nx.adjacency_matrix(G).todense()\n",
   "\n",
   "G = gen_graph(4)\n",
   "print(G)"
  ]
 },
 {
  "cell_type": "markdown",
  "metadata": {},
  "source": [
   "Following is my Prim's Algorithm implementation. Some
   implementations use a priority queue or two lists to manage
   nodes visited and not reached— I am using one list to mark
   nodes as seen or unseen (False or True). The lookup if a node
   has been reached is done in constant time: unreached[node].
   While there are still unreached nodes, I find an edge from a
   visited node to one that has not been seen with the lowest cost
   . The MST ouputted is a graph in edge list form. The edges have
    the structure (node1,node2,weight)."
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 6,
  "metadata": {
   "collapsed": false
  },
  "outputs": [
   {
    "name": "stdout",
    "output_type": "stream",
    "text": [
     "[(0, 1, 1), (1, 3, 0), (0, 2, 5)]\n"
    ]
   }
  ],
  "source": [
   "def prim(G,start=0):\n",
   "    unreached = [[True]]*len(G)\n",
   "    unreached[start] = False\n",
   "\n",
   "    MST = []\n",
```

```
    "\n",
    "    while any(unreached):\n",
    "\n",
    "        e = min([(x,y,G[(x,y)]) for x in range(len(unreached))
     for y in range(len(unreached)) if not unreached[x] and
    unreached[y]],key=lambda x: x[2])\n",
    "\n",
    "        unreached[e[1]] = False\n",
    "\n",
    "        MST.append(e)\n",
    "\n",
    "    return MST\n",
    "\n",
    "print(prim(G))"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "To visualize an MST I am using my code on a previously
    generated graph.\n",
    "\n",
    "<b>Starting Graph</b>\n",
    "<img src=\"imgs/ex_graph.png\">"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 7,
   "metadata": {
    "collapsed": false
   },
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "This is the adjacency matrix\n",
      "\n",
      "[[0 5 7 0 7]\n",
      " [5 0 2 8 0]\n",
      " [7 2 0 4 8]\n",
      " [0 8 4 0 5]\n",
      " [7 0 8 5 0]]\n",
      "\n",
      "This is MST edge list (Node,Node,Weight)\n",
      "\n",
      "[(0, 3, 0), (3, 2, 4), (2, 1, 2), (1, 4, 0)]\n"
     ]
    }
   ],
   "source": [
    "G = nx.adjacency_matrix(nx.convert.from_dict_of_dicts({0: {1:
    {'weight': 5}, 2: {'weight': 7}, 3: {'weight': 0}}, 1: {0: {'
    weight': 5}, 2: {'weight': 2}, 3: {'weight': 8}}, 2: {0: {'
    weight': 7}, 1: {'weight': 2}, 3: {'weight': 4}}, 3: {0: {'
```

```
   weight': 0}, 1: {'weight': 8}, 2: {'weight': 4}}, 4: {1: {'
   weight': 0}, 2: {'weight': 8}, 3: {'weight': 5}, 0: {'weight':
   7}}})).todense()\n",
   "print(\"This is the adjacency matrix\\n\")\n",
   "print(G)\n",
   "print(\"\\nThis is MST edge list (Node,Node,Weight)\\n\")\n",
   "print(prim(G))"
  ]
 },
 {
  "cell_type": "markdown",
  "metadata": {},
  "source": [
   "<b>Produced MST</b>\n",
   "<img src=\"imgs/ex_mst.png\">"
  ]
 },
 {
  "cell_type": "markdown",
  "metadata": {},
  "source": [
   "The expected time complexity of my implementation is O(V^2)
   using the adjacency matrix. An improvemnt could be made by
   using a priority heap to store all edges of the input graph,
   ordered by their weight. That would lead to an O(E*log(E))
   worst-case running time.<sup>[1]</sup>"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 22,
  "metadata": {
   "collapsed": false
  },
  "outputs": [],
  "source": [
   "# This will take about two minutes on a 4-core processor \n",
   "from matplotlib import pyplot\n",
   "import timeit, functools\n",
   "\n",
   "x,y = [],[]\n",
   "\n",
   "for n in range(20,420,20):\n",
   "    x.append(n)\n",
   "    G = gen_graph(n)\n",
   "    t = timeit.Timer(functools.partial(prim, G)) \n",
   "    y.append(t.timeit(1))"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 69,
  "metadata": {
   "collapsed": false
  },
  "outputs": [
   {
```

"data": {
"image/png": "
iVBORw0KGgoAAAANSUhEUgAAAYIAAAEWCAYAAABrDZDcAAAABHNCSVQICAgIfAhkiAAAAAlwSFlz
\nAAALEgAACxIB0t1+/
AAAIABJREFUeJzs3Xd4VGX2wPHvSSEJNdI7AaQKiBIRBRVFbIhgF0RFXVnd\
nddeyirrrT1d3XSu6uq5rZS0giIqgrhURFQtNelMQkIQSWkJLz/
n9cW9gmMwkkzIzmZnzeZ48mblz\
nZ94zk5t75r73vecVVcUYY0zsigt3AMYYY8LLEoExxsQ4SwTGGBPjLBEYY0yMs0RgjDExhKBMcbE
\nOEsEEUJE/
iwiL4c7jlASkTQRURFJqObrnCIia2sqrgra6a6iYiS0Rkn4j8sQZe73k+b+b+
aiK2miMgc\nEflNkF673O1vcRMaKyNxgtB3LqvUPZmqOiOz3uFsXyXyAeK3fu/VdV/
hDCWOsCfgSuB1sAOYDbwoKpu\
nDFUcNUVVwG6ld4XkY3Ab1R1VhCaGw98pqap9fT0oInAAUARkAd8DfxeVbf6Wl9Vb6xqICL/
c1\n/lqF5wqwHshT1Z5VjaGyPDNzQCJqloUhqhikR0R1BKqWr/0B/
gVGO6xbHKIw3kHuAAYDTQC\njgUWAkNCHEck6gCsrGCm92/c1cgFXjK10oiEl/
DsVXGqUBzoJOInBCBKBqt75GeqzhJBhBCRv4rI\
nJPd2aZfJtSKyWUT2iMiNInKCiCwTkWWedbr+
deJyGp33U9FpIOfds4EhgIjVHHBqapqeq\
nr7jrtBaR90Vkt4isEbHbS/
3IMJsRlhC6OQpwkh4rnWxrZ1FKRqjWxWwqRn0ILTDW9/QaLmeZ/
IvIHr7aWiciFfuKN\n4+
Hr3glDD4ibUsBc4D7jY/Szr\n+HkPNwKrgLbLbUcSfb8p+
N7OEzzaHus+5wYgHrgJ2FL6XtZY1gGdcY5mzZ\nh3/G+7/8
dr2Y0cEke1vqpqnqp/
h7JymqGqWmYC3wDAf4AVnzn/
epg7t8q8f7eAHnyAAA VZ0H/ILz3qaV8x6O4O5w\n3wLGuN0lo4A3/
Kze2o2x9Lklbvy+Pkt//qiqqaraRlWvVNUdHo9t9t9vssy4fn6W/v7unx3G+KX8m
\nIr+
IyN3ltHENME2dbsE8nKM0f0lf1rDrb1i3uz1WEXbSkXv15dtpTfcLwBw5HsO9P
CuCwRxIbNOF0o\nqR4/
Kar6nY91ZwH9RaStn9faAjQWkYey9rjdO9UVTuv1yrE6Y7wtBnnW2pTj/
fQUFWPKV1BRH4P\nJLkxji+nPV/fhF/DGSU1BDioq/7ee4WnKRU2qa48Vfn/
VcUW/Vf1D1i+JOqdsIZCHC7iJQ5+e/+\n3c/
ASYrbRGQbcAlwoh4J2dwjmg8txKPv2Ug20p579dKI4eIJYYLY8Dxwj4gcA4dOul7a0V1hlR
+\
nDrwnIv1EJEFEGohzMvo6d6Vd0MfAc8LwIvOPxrbc0rq04XRUTRKSh
\niMSJSGcROc19T12BvwN4ON8QO3m9/vdACTAB/0

cD4BxpDBORISKSCPwJJ0H5Sqq1\nhoicLyJHu4krB2docomPVa/
C6VPvhtOn3xfnvEgGzpGSt2nALSLSxu3uu6v0gRrYVna4MXaqaEVT\
nPZYIYoCqvgc8Ckx1R2esAM4t5ymX4IwWeQtnp7ECSMc5WgBnh5CG843vPeB+
rd6Y/DdwTopuwzmp\
n6O9CrKuBOjj94Htwhrm2ckeoTAIeVdWlqvozznUQb4hIko/XeRi41+1iusNj+
etAb8rZUanqWpxk\n8y+co5bhOEN9fZ1PqE264Pz99gPfA8+p6pc+1rvGfWyb5w
/Olwlf3UMv4SToZcBinO2miMPXwFR5\nW3G7fR4CvnX/
VgMCeqem0krPtBsTFuJcYDVJVcN+1bSIXA2MU9VB4Y4lUrlDOZ9XVZ/
Dk03tZEcE\
nxgBut9TvgBfDHUskEZEU9zqGBBFpA9yP883fRBBLBCbmicjZOP3R23HG5JvACfAATlfdYmA1znUO
\
nJoJY15AxxsQ4OyIwxpgYFxFFnpo2bappaWnhDsMYYyLKokWLdqpqs4rWi4hEkJaWxsKFC8MdhjHG
\nRBQR2VTxWtY1ZIwxMc8SgTHGxDhLBMYYE+
Mi4hyBL4WFhWRkZJCX56tSr6kNkpOTadu2LYmJieEO\
nxRhTjohNBBkZGTRo0IC0tDScOlqmNlFVdu3aRUZGBh07dgx3OMaYckRs11BeXh5NmjSxJFBLiQhN
\nmjSxIzZjqmrZNHiqF/w11fm9LOApNiotYo8IAEsCtZz9fYypomXT4IM/
QqE791DOZuc+QJ/Lary5\niD0iMMaYqPXFg4eTQKnCXGd5EFgiqIb4+
Hj69u1Lr169uPTSSzl48KBP+oyzysg\nwQfL/6
Pv2LGDc845p1Kva4yJADl+ptv2t7yaYiYRzFicycBHZtPx7v8x8JHZzFhc/
ZkFU1JSWLJk\nCStWrKBOnTo8//zzRzyuqpSVVb09jRYtWvDPP/+kmVqTb9wuGTXlL9j8H9aYd5t3x/
jd735X7jrNmjWjVatWfPTRRl3W+3R7oCAOlFbQIw3\
niyU18L28kb8ZZKsnJhLBjMWZ3DN9OZnUSiQmZ3LPdOX10gyKHXKKaewbt06Nm7cSLdu3bj66qsZ
\n1asXmzdvJi0tjZ07d7Jx40a6d+
O2LFj6dq1K1deeSWzs1i4MCBdOnShFw008/kZhSMOm\
nzhSx69evZ8CAAfTu3ZZt7772X+vUPz709cuRIJk+eXGPvwxgTZivfg/
y9IPFHLk9MgSHBqfAd0SeL\nSz3wwUpbWbdnr9/HFHzk9Ky5hcWMf2cU+b
/6vM5PVs35P7hx/h8zFtRUREff/zxpoW/+\
nmddee40BA8rOrdu3TrefvttJkyYwEmnnUSbb77J/PnzmTFjBg899BAzZsw4Yv0bb7yRP//5z7zxxhvcd
+vUPz709cuRIJk+eXGPvwxgTZivfg/
y9IPFHLk9MgSHBqfAd0SeL\nSz3wwUpbWbdnr9/HFv2ZTUHzk9Ky5hcWMf2cZU+b
/6vM5PVs35P7hx/h8zFtRUREff/zxoW6an3/+\
nmddee40BA8rOrLdu3TrefvttJk6cyAknnMCbb77J3Llzef/99/nHP/7BjBkz+
Pbbbzn++OMPPeeW\nW27hlltuYdSoUWWOOtLT07n33zMYU8ttWQLv3QRt+0
O/sTDnyZdc7qFFbJwkE4UQxREkiqIh3\nEqhoeaByc3Pp29eZ/H/2UH/
2UU07h+uuvZ8uWLbRv3/6wGYLVFhYUqQ2pgXiu8TqeRFXeuTYe/sNr/HVDNFxAtBtSiqRZGlZYz3Xz7Uz6l+
aOOw5Ps9u8eXO2bNlSrfhjKkF9m2DKaOgXl4O4YjLUxH\
nXRmSpqMiEVT0zX3gI7PPzI9dxGHZzzrl5xb/fakrdbeo7AW7169fnzfeOTxmpd0YRXVpCU4kaRS3M\
n+QkMWSRDbm5Trw8BaBWyTLwFZXgHtKS7HxZLWPLWnYGtB4mpdxIebUgVB5DPTvXeqr0aHqZVzsx1THlh
\nHteGhy/qTZvUFATnSODhi3oz8rg24Q7Np1NPPZXFixdTOo3oP//5
T5588kn69Oajnc1HjjsWrn8EUs4VXQyPEDsu5hH0H0\
nv/zyS4YNGxauUI0xfXNBFjxDpzxf9Dj/
LCEEBFzFqenp6v3xDSrV6+mIBjX+mR48eYYoo+G655RnbAbBGXkVy0aDpHgBcvvHHjFmcNd2qtnT+C\
nmWeeycGDB0lJSeHkk0/m0ksv5cwzz6RixYtnzwbdvWmyewLt1iDnx9bDNgF85S3OQfrnMEtHYRFmStbD7Hc8nO5m
+zDkaqOGr\n8UVkkaqmV7ReJiEEBYD0UUUfWRo7sG9QDN5kSbGf+0yZuwK4
vOfmTdvHgCLi5ptvRlN2sSU4ymgw1zetBs4nt0J\
ncdAmHS74V40ngcoI2hGBiEwEzgGBHGWVKkvAzNVP/Ap4AmqnqzopeKxaPCKKF/
Z2M8WHfdnjpDEDhhi+h\nQYugNBPoEUEwzxG8CpSbiACZAiN8D+
I0xJpoV5sFbV0Lubhg1JWhJoDKClghU9Wtgt4+HngLG\nA7X/5
IQxxtQkVfjgFshYAYBc+D62ODXdEQIhDYVDYnICCBTVZeGsl1jjKkVvv0/8
Feo4IdzSH\nhOxksYjUBf6M0y0UyPrjgHEA7du3D2JkxhgTAms+hx++ghvkPQK+
L4dQ7wx3NEUJ5RNAZ6AgsFZvg\
nRxFp6WtlVX1RVdNd2z3EJtM2PGDESEGDGtNuxjjJl2Z9b3f69e3f9Tf9e3f69+/
P3Llzj3j8kksu\nfuHaa6/
lhRdeOOOKxGTCNmcO655x66X1RUxLBhw2jatw2jatCkrVqw4tYt0777yT7tt2706dPHy6y688
\nL1/O2LFjg/PmjIkW21W21bAu7+B1n1hxL/
DOkLIl5AlAlVdqrqNqraWlV7qOkqNrUqa3vSFGdsF76e3v4CeRb\
nGDRoEFOmTAl3KNU2ffp0TjnllBp5rfnz5zN48GDatGlDq1at+P7774v66nmNmzb8F2r9fjj33AHB
\nlbkqeerUqYwaNQoBCo2S6yuquQQGzDlOSEfHqsc4ZsCWnzIMnoeQ9fxbomyKhU9J3f0FPzFm4uTQqcCDLj
zyjR49m2bnbnbvMlqxCjZBDk2WnHrM/
Ztuuonr3kPEIFaa2UzhJM3Lb4cq4u8wGtqz0t+sI8GbTAAMhdLqRqhKhDKoobq0nlbaNPdNnmTGnuHngZudUrQZt5cvRh
\nSWEEuOkIF3T5Tcl3nqK3KqxUbkAhUu2StlwAENIexClAagBnmM36VQYLtXSPu0CdsK5Wpm2O2VzV5VbPXyI8/
ZbXtp7JCgNxFKiS9JzqNPThc8pVA2PA02qbCwrkZUG4KSoWGWlkGjr9yW5aDr8ehA8O+4XcQaO0jbmAccbo4SSg0PtXMVwK6EN
Wkmm9bhY0s9ySdmUBH0LGWj0SYnnvEUVqMeH4eK6zjgMsPHsXk6X3JtFzADAD70Qkug2o9bakwX5z+UfxDscz6J73jo6cthIPEi
\nN/ltq/VHFA0lCVIyg3oHW2M7ZyCGkO6ijkw4g9hNktwCKBAgZLZaxL1OV5A3oEQvhq0oqfT1cn9LL+YsDRN0vP+5sF0fC
HmTMrMbvUMk98AlUpYgKBA5rGOHxr3YqFDyePDxbT8tqyHCRNOHugN6xf9PGf1jRSwc48kIhJP6Rb8cO1CMwaTzR2f3CqCMOkSN
Q8wsSv+8d8TkSA5AWiSA9xJa1aw6Crpx+Dn1fuTuXgk7VM3ZWgv9aMPebK5dZeL4sKV6+Ow0e1o3+QGwc0RI1JdSl+ZP4BKhVWTOO
XvCx6p7Lvhe6UfsOCPjk6Akw==

ncdAmHS74V40ngcoI2hGBiEwEzgeyVLWX12N/Ap4AmqnqzopeKxaPCKKF/
Z2M8WHfdnjpDEDhhi+h\nQYugNBPoEUEwzxG8CpSpfyAi7YCzAN8D+
I0xJpoV5sFbV0Lubhg1JWhJoDKClghU9Wtgt4+HngLG\nA7X/5
IQxxtQkVfjgFshYABc+D62ODXdEQIhHDYnICCBTVZeGsl1jjKkVvv0nLJsKp/8
Feo4IdzSH\nhOxksYjUBf6M0y0UyPrjgHEA7du3D2JkxhgTAms+glkPQK+
L4dQ7wx3NEUJ5RNAZ6AgsFZGNQFvg\
nRxFp6WtlVX1RVdNVNd2z3EJtM2PGDESENWvWhKX9F198ke7du9O9e3f69+/P3Llzj3j8kksu\nfuHaa6/
P3Llzj3j8kksu4Zdf\nfuHaa6/
lhRdeOOKxGTNmcO655x66X1RUxLBhw2jatw0777yT7t2706dPHy688EJZbWX
\nL1/O2LFjg/PmjIkW21bAu7+B1n1hxL/
DOkLIl5AlAlVdqrqNVTVNVdOADOB4Vd0WkgCCNO3blClT\
nGDRoEFOmTKmR1wMOlZqoyIcffsLL7zA3A3LlzWbnmDNc8//
zyjR49m2zbnI125ciXF5UaNG\nlbkqeerUqYwaNQoBCo/
Ztuuonr3bszY8YMLr/8cjIy+Hzp0Dc+
Hzp0KCtWrDevXuT\

nkZHBr7/auX9jfNq/w6khlNwQrpjiVBGtZYKWCERkCvA90E1EMkTk+
mC1VaHSad9yNgN6eNq3aiaD\n/
fv3M3fuXF555ZUjdrJz5szh1FNPZdiwYXTr1o0bb7yRkhKnwF39+
vW57bbbDhWc27FjBwCDBw/m\n1ltvJT09naeffpoRI0bw+uuvA/
DCCy9w5ZVli089+uijPP7444fKVR9//PFcc801/Pvf/wZg8uTJ\
njBjh9EMOGTKENWvWsHXrVgAOHDjArFmzGDlyJAAPPPAAjRo1YsKECQwaNIiXX36ZUaNGHap5dNZZ
\
nZ5GQ4PQkDhgw4IgkMXz48DJJxpiYt2waPHUMPHG0s89Jvw4atgp3VD4F7RyBqo6q4PG0Gmvs47th
\n23L/j2csgOL8I5cV5jr1PRa95vs5LXvDuY+U2+
zMmTM555xz6Nq1K02aNGHRokX069cPgPnz57Nq\n1So6dOjAOecw/
Tp07nkkks4cOAA6enpPPXUUzz44IM88MADPPvsswAUFBRQer3EmDFjGDhwIB07\
ndmTChAn88MMPZdpfuXLlofZKpaen89vs5LXvDuY+P5+
KLL2batGnccsstfPDBBwwePJiG\nDRsCcP/99x/xOieddBLffPONz/c9ceJELr
/88iPafOSRRxg/fny5n5cxMcN7zmEU5j4JR6UFrZR0\
ndcREraEySaCi5QGaMmUKV1xxBQXBXXHHFEd1D/
fv3p1OnTsTHxzNq1KhDffdxcXGHdqJjxow5w5ok/f\nc+faokULHnzwQU4//
XQmTJhA48aNKx2fdzlrz+4h726hQD300EMkJCQccYRipbCN8RLiObKzpK\
nTFTwzZ2nerndQl4atYNr/1
elJnfv3s3s2bNZvnw5IkJxcTEiwl//+lcAEiwuOPPPw6AeJ0M8r7va7l3+erly5fT\
npEkTvzvZnj17smjRIs4444xyxYtWsQxzhluVNSUsjLyzv02Mknn8zWrVtZunQp3333XaW7c159
\n9VU+/PBDvvjiiyPitlLYxnjtxb+BoM05XF2xcUQw5L6yJ2iqOe3bO++
w1VXXcWmTZvYuHEjzmdzv\npmPHjoe6U+bPn8+bPn8+
GDRsoKSnhrbfeYtCgQQCUlJTwzjvvAPD6FmDmm28eWu5t/vz5z5fPzxxyxevJGnnniC\
nDRs2lFln/
Pjx3HXXXezatQrrx6a69i7nLWIcPnll3PNDdw7rnnkpycHPD7PHD7/eSTVDw7PHD7/
eST3js\nscd4//33qVu37hGPWSlsY7z04t+HDfG8yrbqgpUuCZbQj3EDpJFeDh/\n+OKLD3UPnXCD3UPnXDCCdx888NGDjh07Hlq3Xr16zJ8/n6ZNmzJ7924efPDBMmdNSSOSkVkEi6lGNc4b6Opb2qy3jz
z/+h2mSZDb1aiyjcfYnqFB84fzvd\n0t1CN998M/
v27WPo0KH07duXG2+8EaFBDKbAupwBaEdDeQfDwNnXSRmyHy0FX9YjPNk4vu0nbUPbvgZgU3Ze4
4sSJtG7d\
nmgkTJnDddfhXRfqggsu4LrrruPkk0+me/
fu3HDDUyaNllWrZwZmTUsOGDWPOnDlHPGFfUqFEsXbq0\n0t1CN998M/
v27WPo0KH07duXG2+88dBjVgrbGFFWapg6Bhq0goSa/
fIZTFaGOgjmzJnDE088wcYf\nfljmsfr167fjzRxybUEzEzs+HJjk/Nx+5Nk+e7+
S3x8fMVPqIL8/HxOO+005s6de3n/f/
nDIXiAvjNLPj1B+
ecQAjmHPbHylAbUlJSeOCBB8jMzdm//vorjzYuM8kYEzMyN8Hb14rtDD\
ntR9BanvnpxaOEPLF/nuDYPDgwQwePNjnY6E6Gih19tlnB/
X1u3TpQpcuXYLahEjER\n0W85Vw9HmIg+RxAJ3VqxzP4+
JuqpwvoSug3Wz4Pwnp8cPcEdUJRGbCJKTk9m1aftbGopVVWXrl2V\
nGplkTMT5ZgL8+
Dqccgf0GxvuaKoSmZrG2Zz2jlcezphqW/oWzP4b9Lkc
\
nzrg33NFUS8QmgsTERDp27BjuMVTV8RbyIrZryBhjwLq7nm7NJcsQQsqacq7Zwzf4sT7hKP7wWeNm8QRIrTGGjpqrrjhEB3cPdRegJKS1bI8n
\nxpiKFBfCtGucgwCHf1rcgHtc17Nm7YgUHMwAA3WfwEX/
Au6nBnuiGqcJQJjjPG2bNrhOkFJDSE/\nB04dD8dfHe7IgsISgTHGePKeSw/
ByQemkZvKZVgzlk8UUSyRGSFx7LHRWSNiCwTkfdEJDpOuRtj
nooev2cW0uNbOLlYTgjl89FXgHK9lnwO9VLUP8BNwTxDbN8aYyvM3i1gtnV2sJgQtEajq18Bur
\nqWqRe/cHwOoOPGGNql/otfC+
vpbOL1YRwXlB2HfCxvwdF5EYRWSgiC7Ozs6sxLGNMVEr9wAfuLQSH4bn4RgIHfalrl6dm2e8XPB1
/gIUAZP9raOqL6pquqqmN2vWLHTBGaAV6LQOsLFOB42Kq/+ZVytBWHMfRhumUmeDgvI7xZUwrl9m2ACvrVk0wT
\nzZQh
ylMNnIbn7H4LGmqOpLA/lgiMMbHl++
Ay96vpBSNhdN7zOm2hRl4wNDObUFV9vKZqI/+ZUw8+pK6uaqpjq+EaY0wWpEnRL6lPb9exCTsEKaw1yNwHD
\nEogxwRw+OgX4HugmwlIhkcV\
nUT4WvxKs9owxpIJKSmDGTbDucxW4NBwzMtwRhbxtVN5htz0Wcvwc6t8mUbY7AEBYyLqXbZxnYGBzj9kY=
\nh2DBy3DyH2PqpLA/lgiMMbHl++
fg68fhuKtgaPSWjagMSwTGmNixZAp8eg/0uMA5LxDhcw3XFEsE\

nxpjYsOYjZ8L5ToOdEUJx8eGOqNawMtTGmOi34Rt4eyy07guXT4aEpHBHVKEZizN5
/NO1bMnOpXVq\
nCnee3Y2Rx7UJSluWCIwx0cdzYpn6zZ0rhxt3hCvfgaT64Y6uQjMWZ3LP9OXkFhYDkJmdyz3TlwME
\nJRlY15AxJrqUTiyTsxlQ2L8digsg/Tqo2zjc0QXk8U/
XHkoCpXILi3n807VBac8SgTEmuviaWAaF\n7/4VlnCqYku2d/
zlL68uSwTGmOgSBRPLNExJ9Lm8dWpKUNqzRGCMiS4NWvleHiETyyzcuJu9uYXE\
neY1sTUmM586zuwWlTUsExpjosW87lBSXXR4hE8tk7c3jpsk/0qFJXf5+YS/
apKYgQJvUFB6+qLeN\nGjLGmHLt3wGvDYeC/
XDqeFg6xekOatTWSQK1fGKZgqISfjf5R/bnFTHp+hPp1rIBo/t3CEnblgiM\
nMZHvwC54/QLI/hXGvANpg+CMv4Q7qkr5x0erWbhpD8+
MOo5uLRuEtG1LBMaYyHZwN7w+Anb/AqOn\
nOUkgwry3OINXv9vI9YM6csGxrUPevp0jMMZErtw98MZI2PkTXPEmdDot3BFV2sotOdwzfTkndmzM
\n3ed2D0sMdkRgjIlMeTnwxkWQtdpJAkcPCXdElZZ9sIAbJy0iNaUOz44+
nsT48Hw3DygRiEg6cArQ\nGsgFVgCfq+qecp4zEWeS+
ixV7eUuawy8BaQBG4HLynsNY4zxKW8vTLoYti2Hy9+ALkPDHVGlFZco\
nt0xdwracPN767Uk0axC++kflph8RuVZEfgTuAVKAtUAWMAiYJSKviUh7P09/
FTjHa9ndwBeq2gX4\nwr1vjDGBy98Pky+FLYvh0leh27nhjqhKnp71E1/9
tIP7hx/D8e2PCmssFR0R1AUGqqrP65pFpC/Q\nBfjV+zFV/VpE0rwWjwAGu7dfA
+YAdwUcrTEmthUcgDcvg4wFcMlE6HF+uCOqklmrtvPM7HVc2q8t\
nV57o77t06JSbCFT13xU8vqSS7bVQ1a3u7W1AC38risg4YBxA+/bh/6
CMMWFWdBCmXAG/fg8XvRSx\
nk81v2HmA295aQq82DfnbyF5ILZgcJ6AzEyLymIg0FJFEEflCRHaIyJjqNKyqCmg5j7
+oqumqmt6s\
nWbPqNGWMiXSFeTB1tDOvwMjnofcl4Y6oSg7kF3HjG4tIiBeeH9OP5MTaMTlOoKOGzlLV8SJyIc5J
\n3ouAr4FJlWxvu4i0UtWtItIK53yDMcYcyXM+gUZtIKkRZK2Ckc/
BsZeHO7oqUVXuencZP2ft47Xr\n+
tP2qLrhDumQQMcqlSaMYcDbqppTxfbeB65xb18DzKzi6xhjopX3fAI5GZC1Eo6/
CvqODnd0VfbK\
n3A18uGwrd5zdjVO61K5ejkATwYcisgboB3whIs2AvPKeICJTgO+
BbiKSISLXA48AQ0XkZ+BM974x\nxhzmcz4BYP2XoY+lhny/
fhcPf7yGs49pwU2ndQ53OGUE1DWkqneLyGNAjqoWi8hBnBFA5T1nlJ+H\nIu+
qD2NM6ETBfAKetubkcvObTkXRJy49tlacHPZWbiIQkYt8LPO8O72mAzLGxLiGrWFvZtnlETKf
\nQKkZizN57JM1bMnJQ4Bxp3akQbLvCWfCraIjguHu7+bAycBs9/7
pwHdYIjDG1KS8vZDgYxauCJlP\noJT35PMK/
HPWOlo0TAnanALVUe45AlW9VlWvBRKBnqp6sapeDBzjLjPGmJpxcLdbSnoj9B8HjdoB
\n4vwe/kytn0/AU6gnn6+uQIePtvO4EAxgO2BXeRljasb+
LHh9JOxaB5dPhm7nwHmPhzuqKssM8eTz\
n1RVoIvhCRD4Fprj3LwdmBSckY0xMycl05hPYmwlXToNOg8MdUbX8uusgIqA+
LpcN1uTz1RXoqKGb\n3RPHp7iLXlTV94IXljEmJuze4HQH5WbDmOnQ4aRwR1Qt
+/IKuf61BSQnxFGikF9UcuixYE4+X10B\
nz0egqtOxk8PGmJqy4yfnSKAoF66eCW2OD3dE1VJaVvqXnQd447r+ZO3L5/
FP17IlO5fWqSnceXa3\nWnmiGAKfj+Ai4FGc0UPi/
qiqNgxibMaYaLVthTOzGMDY/0GLY8IbTw149JM1zF6Txd9G9uLko5sC\
n1Nodv7dAjwgeA4ar6upgBmOMiQGZi5yZxerUg6vfh6ZHhzuiant74WZe/
PoXrhrQgasGdAh3OJUW\naImJ7ZYEjDHVtuk7eG0EJDeCaz+OiiSwcONu/
vLeCgYe3YT7hvcMdzhVEugRwUIReQuYAeSXLnTP\
nGxhjTMXWz4YpoyG1nXNOoGHrcEdUbRl7DnLjpEW0Tk3m32Gcc7i6Ak0EDYGDwFkeyxQ7eWyMCcTa
\nj2Ha1dC0K1w1A+
rXruqbVXEgv4gbXl9EflEJU8edQGrdOuEOqcoCHT56bbADMcZEEc/5BOo2dq4a\
nbn0cjHnXuR/hSkqU295awtpte5k49gSOb14/3
CFVS6AzlLUVkfdEJMv9eVdEIqsClDEmNLznEzi4\nC0Sg39ioSAIAT37+E5+
t2s69w3oyuFvzcIdTbYF2aP0XZ1KZ1u7PB+4yY4w5kq/5BLQEvo7ckhGe\
nZi7J5Nkv13HFCe24dmBauMOpEYEmgmaq+l9VLXJ/XgUiv5PPGFPzomw+
AU9LNmdz5zvL6N+xMQ+O\
nqB0Tz9eEQBPBLhEZIyLx7s8YYFcwAzPGRKCiAqdktC8RNp+
At205eYx7fSHNGyTx/Jh+1EmIzBFC\
nvgT6Tq4DLgO2AVuBS4Aqn0AWkdtEZKWIrBCRKSKSXNXXMsbUEnl74c1LofAgxHlVqY

+w+QS85RYU\
nc8PrCzmQX8Qr15xA43qRO0LIl0BHDW0CLqiJBkWkDfBHnPkNckVkGnAF8GpNvL4xJgz2bYfJF0PW
\
nahj5H4ic2YlOAAAgAElEQVRLODxqqFFbJwlE0HwCnlSVO95ZyootObx0VTrdWjYId0g1LtBaQ68B
\nt6hqtnv/
KGCCql5XjXZTRKQQqAtsqeLrGGPCbefPMOkiOLALRr0FXc50lkfojh+
cGcZKC8bVT05g\nX14Rd5/
bnTN7tgh3aEER6AVlfUqTAICq7hGR46rSoKpmisgTwK9ALvCZqn5WldcyxoTZ5gXw5mUg
\
ncTD2w4ivIAplp5ncl1dEvAgtGiSFObLgCfQcQZx7FACAiDSmEiWsPbmvMwLoiDMUtZ578tl7vXEi
\nslBEFu7YsaMqTRljgmntJ/DacKdu0PWfRUUSAN/
TTBar8sRnP4UpouALNBFMAL4Xkb+JyN9wJq5/\nrIptnglsUNUdqlqIU6biZO+
VVPVFVU1X1fRmzWykqjG1yqLXYOooaN4drv8cmnQOd0Q1xt90krV1\
nmsmaEOjJ4tdFZCFwhrvoIlVdVcU2fwUGiEhdnK6hIcDCKr6WMSaUVOGrx2DOP+
DoM+HS1yApsssr\neGtaP4kd+/PLLK+
t00zWhMoMhG0MHFDVZ4EdItKxKg2q6jzgHeBHYLkbw4tVeS1jTAgVF8GHtzpJ\
n4NjRMGpq1CWBZRnZ7M0rwPsysdo8zWRNCLTW0P3AXcA97qJEYFJVG1XV+1
W1u6r2UtWrVLVs+jXG\
n1B4FB2HaVbDoVTjlTzDyOYhPrPBpkWRFZg5jXp5H84bJ3De8J21SUxCgTWoKD1
/UO2JmG6uKQE/4\nXggch/MtHlXdIiLRN5jWGOPwrB7asDXEJ8GeDXDeE9D/
hnBHV+NWZOZw5cvzaJCcyJQbBtD2qLpc\
nO7BKnR4RKdBEUKCqKiIKICL1ghiTMSacSquHlhaO25vp/
D7xxqhMAiu35DDmlXnUT0pg6jgnCcSa\
nQM8RTBORF4BUEbkBmAW8FLywjDFh46t6KMCa/4U+
liBbtWUvV748j7qJ8Uy5YQDtGsdeEoDARw09\nISJDgb1AV+A+
Vf08qJEZY8IjiquHelqzbS9XvvwDKYnxTBk3gPZNYjMJQCUuClPVz0XkR+
BUYHfw\nQjLGhI0qJDWE/Jyyj0V49VBPa7ftY/
RL80hKcI4EOjSJ7d7ucruGRORDEenl3m4FrMCpRPqGiNwa\ngviMMaFSmAvv/
sZJAhJ/5GMRXj3U00/b9zH6pR9IiBOmjBtAWtPYTgJQ8TmCjqq6wr19LfC5qg4H
\nTsRJCMaYaLBvG7w6DFa8C0Puhwufh0btAHF+
D38moovIlfrZTQLxbhLoaEkAqLhrqNDj9hDcE8Sq\
nuk9ESoIWlTEmdLYsgSmjIC8HLp8EPc53lkfBjt/
Tuqz9jHppHiLCmzcMoHOz6LoYrjoqSgSbReQP\
nQAZwPPAJgIik4FxUZoyJZKtmwvTfQt0mcP2n0LJ3uCMKivU79jPqpR8AmHLDiRzd3JKAp4oSwfXA
\ngziF4i73KEU9AJu83pjIpepMJv/
lQ9C2P1wxGeo3D3dUNcZzPoFmDZLILSgiyR0ienRzuxbWW7mJ\nQFWzgBt9LP8S
+DJYQRljqgqgwF2b+3
jkf0OcKGP40JEbPbLHe8wlk7XMq2Pzu9KPp0sKSgC8VjRp6\nqXTUkI/
H6onIdSJyZXBCM8bUuH3b4L/nwYrph08KR1ESAN/zCQBM+uHXMEQTGSrqGvo3cJ
+I9MYZ\
nOroDSAa6AA2BicDkoEZojKkZnieFr5gM3YeFO6KgiMX5BKqroq6hJcBlIlIfSAda4cwhsFpV14Yg
\nPmNMVXgWjWvUFrqdBz++DvWaRvVJYYCmDZLYsS+25
hOorkBLTOwH5gQ3FGNMjfAuGpezGea/AEd1\
ncpJAFJ0U9rZkczb7cp35BNRjebTPJ1BdlZmYxhgTCfwVjSsuiOoksGDjbnc+
gRTuPb9HTM0nUF1V\nmoDeGFOL+
SsOV1pOOgp9u24nv3ltIa1Sk3nzNwNo2SiZ6wd1CndYEaNSRwTuPMPGmNqsbhPfy6Oo
\naJynL9dkce2rC+
jQpC5vjTuJlo2iaxRUKAQ6VeXJIrIKWOPeP1ZEnqtqoyKSKiLviMgaEVktIidV\
n9bWMMa7iIvj8Pji4E8Rr1t0oKhrn6ZMVWxn3xkK6tWjAlBsG0KxBUrhDikiBHhE8BZwN7AJQ1aU4
\n5air6mngE1XtDhwLrK7Gaxlj9m2HN0bCt09D+nUw4rmoLBrnaeaSTH7/5
mL6tE1l8g0nclS9OuEO\
nKWJVZj6CzXLkt4yyV2wEQEQa4SSRse7rFgAFVXktYwyw6Tt4+1rn+
oCRz0PfUc7yvqPDG1cQTVuw\nmbumL+
PEjo155ZoTqJdkpzurI9Ajgs0icjKgIpIoIndQ9W/
xHXEuTPuviCwWkZd9zYEsIuNEZKGI\nLNyxY0cVmzImiqnCd/+CV8+
HOvXghi8OJ4Eo9vr3Gxn/7jJO7dKMV6/tb0mgBgSaCG4Efg+0ATKB\
nvu79qkjAqWT6H1U9DjgA3O29kqq+

```
qKrpqprerFmzKjZlTJTKy4FpV8Fn90L382Dcl9DimHBHFXQv\nfr2e+2
auZGjPFrx4dT+SE+
MrfpKpUKAXlO0EaqqmUAaQoarz3Pvv4CMRGGP82L4S3roK9myEs/4O\
nJ91c9uRwlFFVnvliHU/N+onz+7Tiqcv7khhvl0HVlIASgYh0BP4ApHk+
R1UvqGyDqrpNRDaLSDe3\
nTMUQYFVlX8eYmLR0KnxwKyQ3grEfQoeTwx1RUHiWkW6dmkz3Vg35YnUWl/
Rry6MX9yE+LroTX6gF\n2rk2A3gF+ACoiZnJ/gBMFpE6wC8402AaY/
wpyodP7oaFE6HDILhkIjRoEe6ogsK7jHRmdh6Z2XkM\
n7NyYxy7uQ5wlgRoXaCLIU9VnaqpRt5hdek29njFRx7NoXIOWEF8HsjfBwFvhjP
+D+Og9QeqvjPTG\nXQctCQRJoFvT0yJyP/AZcKisn6r+GJSojIll3kXj9m11fg
/4HQx9IHxxhYj/MtJ5IY4kdgSaCHoD\
nVwFncLhrSN37xpia5K9o3OoP4JyHQx9PiFkZ6dALNBFcCnRyL/4yxgSTv6Jx/
pZHkWkLNrN7f76V\nkQ6xQMdfrQBSgxmIMTFPFRa84v/xKC0aB5BXWMw905cx/
t1lnNS5KX8b2cvKSIdQoEcEqcAaEVnA\nkecIKj181Bjjw4GdMPNm+
OljaN4Tdv8CRR594lFaNA4gY89Bfjf5R5Zl5PD70ztz+9BuxMcJYwZ0\
nCHdoMSPQRHB/UKMwJpb9PAtm3AR52XD2w3DijbDinSOnmhxyX9QVjQP45ucd/
HHKYoqKlRev6sdZ\
nx7QMd0gxKdAri78KdiDGxJzCPJh1P8x73jkKuOo9aNnLeazPZVG54y9VUqL856v1TPhsLV2aN
+D5\nq/rRsWmZkmMmRMpNBCIyV1UHicg+jjx3I4CqasOgRmdMtNq+
Et79DWStco4Azvyr0/0TA/bmFfKn\
naUv5fNV2Lji2NY9c3Ju6daL3uohIUNGnXw9AVRuEIBZjol9JiXMEMOt+SE6FK9
+FLmeGO6qQWbNt\nLze+
sYiMPbncP7wnY09OQ6K8TlIkqCgRaAWPG2MCtXercy7gly+
h67kw4lmo1zTcUYXMzCWZ3P3u\ncuonJzBl3ABOSGsc7pCMq6JE0FxEbvf3oKo+
WcPxGBMdPEtENGoLPYY7BeMKc+H8p6DftVFdMdSz\naFyrRs10b16fb37eSf+0
xjw7+jiaN7R5hWuTihJBPFAf55yAMSYQ3iUicjbDD+7Ukdd9Cs26hje+\
nIPMuGrclJ48tOXmc1rUpL19zgpWProUqSgRbVfXBkERiTLTwVyICjfokAP6Lxq3LOmBJoJaq6K9i
\nRwLGVJbfEhGZoY0jTPwXjfO93IRfRYlgSEiiMCZarP8SxM+/
VRSXiCj12cptfk99WNG42qvcriFV\
n3R2qQIyJaLl74NN7YckkqNfcmVO42KOCZhSXiADYfaCAv76/
kveXbqF1o2R2HSggv+jwHFZWNK52\
nsw47Y6pr1fvw7xNh6RQYdDvcutwZGtqoHSDO7+
HPRO2Vwv9btpWhT37Fxyu2cvvQrsy583QevbiP\
nFY2LIKIanksFRCQeWAhkqur55a2bnp6uCxcuDE1gxgRq33b46A5Y/T607A0j/
g2tjg13VCGzY18+\
n981cwccrttG7TSMev7QP3VtasYHaREQWqWqFs0GG87ruW4DVgG05JrKowpI34dN7nHpBQ
+6Hk/8A\
n8YnhjiwkVJWZS7bw1w9WcrCgmPHndGPcKZ1IsBFBESssiUBE2gLDgIcAvxesGVPr7NkIH9zqXB3c
\n/iS44F/QtEu4owqZ7Xvz+Mt7y5m1Oovj2qfy+
CV9OLq5VaCJdOE6IvgnMB7wuwWJyDhgHED79u1D\nFJYxfpQUw/
wXnWsEJA7OewLSr4e42PgWrKq8vSiDv324isLiEu4d1oNrB3Yk3iaTjwohTwQicj6Q
\npaqLRGSwv/
VU9UXgRXDOEYQoPGMcniUi6rdwRv3s2QBHD3VKRKS2C3eEQeVZIqJ5wySOqluHNdv2
\n0b9jYx69uI+
VjI4y4TgiGAhcICLnAclAQxGZpKpjwhCLMWV5l4jYv835nX49DJsQ1TWCoGyJiO17
\n89m+N5+Ljm/DE5ccS5wdBUSdkB/
Xquo9qtpWVdOAK4DZlgRMreKvRMTPn0V9EgD/JSLm/bLbkkCU\
nio0OTmMCtWu9UyTOF3+lI6KIqpJpJSJiTlinBVLVOcCccMZgDAD5++
GbCfD9s7gT8JVdJ8pLRGzf\
nm8c905f7fdxKREQvOyIwsU0Vlr0Nz6bD3CfhmIvgvMfLThsZxSUiVJUZizM566mv
+W79Ti46rjUp\niUfuGqxERHSziUJN7Nq6FD6+C379Hlr1hUtfg/
YnOo8lNzpyYpkh90VliYid+/P5y3vL+XTldo5v\
nn8oTlx5Lp2b1ObXr4VFDrVNTuPPsblYiIoqFrcREZViJCVOjDuyC2X+
DRa9C3cbOlcHHjYG4+HBH\nFlIfL9/KX2asYH9eEX86qyu/
OaWTXRcQZSKhxIQxoVVcBAsnwpd/d84JnHgjDL4bUlLDHVlI7TlQ\
nwP1updDebRox4bJj6drCrg6OZZYITHTynjO4z2Ww9hPIWgkdT4NzH4XmPcIdZch9sXo7d09fzp4D
\nBdw+tCs3De5ss4YZSwQmCvmaM/ibCZDSGC57w5lIPsqvB/
```

```
C8Mrh1ago3n9GZRZuyeWdRBt1bNuDV\na0/gmNaNwh2mqSUsEZjo4++
CsMS60POC0McTYt5XBmdm53LP9BUIcPPpR/PHIV2ok2BHAeYwSwQm\
nuhQc8H9B2N7YmDPY35XBTesncYcNATU+2NcCEx2KCmD+S/
DMcf7XifILwkr5uwJ45/58n8uNsURg\
nIltJsTNJzLP9nNnCmhwNp90VUxeElTqQX8Rzc9Y5F0b7YFcGG3+sa8hEJlVY/
QHM/jvsXOtcEHb+\nP6HzGc6J4CZHx8QFYQC5BcVM+mETz3+1nl0HCujRsgG/7
Dxgk8ebgFkiMJFFFdbPdnbyW5dA065w\n2evQ44IjRwL1uSxqd/yl8gqLmTr/V
/49Zz079uUz6Oim3Da0K/06HFVm1JBdGWzKY4nA1E7e1wEM\
nuQ9SOzjLNs2FRu1h5H+
g92UQH1ubcUFRCdMWbubfX65ja04eJ3ZszLOjjuPETk0OrTPyuDa24zcB\ni63/
IBMZfF0H8N5vQUugXnM493Hodw0kJIU3zhArLC5h+o8ZPPPFOjKzcw/
VBjq5cxMkyq+LMMFl\nicDUPr6uA9ASpxDcLUugTvRPk+
jZtdMqNZnTuzVj7rpdbNp1kD5tG/HQhb04rWszSwCmRlgiMLWP\
nvwlg8vbGTBLwvCBsS3Yek+dtpnWjZF66Op0zezS3BGBqlCUCUzuUlMBPn8C3T+
NzUhiImesAHv1k\njc8LwhAY2rNF6AMyUS/k1xGISDsR+
VJEVonIShG5JdQxmFqkqAAWT4LnBsDUUbB3C/S5AhJi7zqA\
nLdm5PPzRarbm5Pl8fGu27+XGVFc4jgiKgD+p6o8i0gBYJCKfq+
qqMMRiwiVvrzMfwA/Pwb6t0KI3\
nXPQyHHOhMwro6CExcx3Aks3ZvPzNL3y8YhsAKYlx5BaWlFnPLggzwRLyRKCqW4Gt7u19IrIaaANY
\nIogF+7
bBvOdhwUTIz4GOp8KIZ6HzkJi6DqCouITPVm3nlbkbWLRpDw2SErhuYBrXnJzGwo17jjhH
\nAHZBmAmusJ4jEJE04DhgXjjjMEHgfR1A/9/
Crp9h6RQoKXIuABv4R2jTL9yRhtS+vELeWrCZ/367\nkczsXNo1TuH+4
T25NL0d9ZOcf8e2R9UFsAvCTMiEbapKEakPfAU8pKrTfTw+DhgH0L59+36bNm0K
\ncYSmyryvAygl8c74/5Nuhiadwx NbiHhf2XvdoDQy9+QxbeFm9ucX0T+
tMdcN6sjQni1sekgTNIFO\nVRmWRCAiicCHwKeq+mRF69ucxRHmyZ6+
Sz43bA23rw59PCHmPfyzlAAX9G3N9YM60qdtbE2PacKj\
n1s5ZLM4A6FeA1YEkARNBtq+Ehf/1X/d/79bQxhMmj3zse/
hni4ZJPH1FOWWyjQmTcJwjGAhcBSwX\nkSXusj+
r6kdhiMVUV8FBWPkeLPovZCyA+CRnJrDCg2XXjeLrAHILivls1Tbe/TGTbXt9D/
Pcvtfm\nAzC1UzhGDc3Fb8V0EzG2r3J2/kvfckb/NOkCZ/8
Djh0F62aVPUcQhdcBqCrzN+xm+o+Z/G/5Vvbn\nF9EmNYUGSQnsyy8qs74N/
zS1lV1ZbHzzVf2zx3D32/+rsHkexNdxRv+kXwsdBh4e/lk67DNKrwPY\ntOsA03
/MZPriDDbvzqVunXjO692Ki45vw4COTXh/6RYb/mkiSthGDVWGnSwOMV+
jfuISIC4RinKd\nSV/6jYVjR0O9Jn5fJpJ5j/q5+
YzOCMK7P2awYOMeRGBg56ZcdHwbzunVkrp1Esp9vg3/NOFQq0cN\
nVZYlghB7qpfvCeATU2D025A26MiLv6KMv1E/AJ2b1ePifm0Z2beNdfWYWq/
WjhoytVhuNqz50HcS\nACjMg46nhDamENu5P5+/frDSZxJoVj+JWbefZpU/
TdSxRBDr8vbC2o9h5XRY9wWUFDoXfqmP6pdR\
nOOqnpERZuWUvs9dkMXttFssysvF3kLxzf74lAROVLBHEooIDTsnnFdPh58+
hOB8atoETfwvHXAS7\n10f8qJ/y+uj35xcx9+
cdzF6TxZdrd7BjXz4icGzbVG47sytv/LCJHfvKDvW0riATrSwRRCvvUT+D\
n74akhs43/58+dcb512/hnPTtdRG07Q9xblXytm79nwgd9ePdx5+
Znctd7y5jztosdu4vYN6GXRQW\nKw2SEzi1azPO6Nac07o1o2l9Z+
rL9o3r2qgfE1PsZHE08lfrB6BuU+g5wtn5tz8J4uJDH1+QDXxk\
nNpnZPt470KV5fc7o3pzTuzenX4ejSIz3PSWHjfox0cBOFsciVdixFj6603cSqNcMbl
/j1PuPMpnZ\nuSzYsJsFG3f7TQICfH77aQG93sjj2tiO38SM6NsjxJrcPfDLV7D
+C1g3G/b6me8X4MDOiEkC5X0j\nLylRfs7az/
yNu1m4cTcLNuxmizurV4OkBJIS4sgvsoldjAlUZOwVYpGvK3v7XAYlxZD5o7vj/
wIy\nF4KWQFIj6HQqnHoHfPWoM+uXtwgZ9eOvj/+
L1ds5WFDMwk17yMktBKB5gyRO6NiY36Y1Jj3tKLq3\
nbMgHdmWvMZViiaA28u7jz9kMM38HPzzvjOjJywYE2hwPp9zhTOvYJv3wt/069
SJ21M+B/CIe+t/q\nMuP484tK+GDZVjo1q8e5vVqSntaY/mmNadc4pcyQztIjB+
vjNyYwdrK4tikpgad6OFM6epM4p6jb\n0UOg0+lQt7H/1/
F3RBEiFZ1sLSgqYcPOA6zdvo+12/aydtt+1m7fy+bdvvv3wenj3/DIsBBEb0x0\
nsBIT4RbojvjgbshY6HTxZCyAzEWQl+PnRQX+mh3UsGuCrxINdeLjOLNHc+Lj4/
hp2z7W79hPUYmz\n7cXHCZ2a1qNrywZ0b9GA/363kd0HCsq8bpvUFL69+4
```

```
yQvQ9jIp2NGgonX107H/zRmau3aTd3p+/u\n+
PdscNaROGjeE3qOhNXvOyeBvYWwj78ywydVlV0HCti48wAbdh7ggQ9WlenaKSgu4aMV22iTmkL3
\
nlg04o0dzurdsQNcWDejUrB5JCYeHsbazcfzGhJQlgmD44sGywzcLc2HGTYfv128JbdOdOXzbpEPr
\n4yCpvvNY2qCw9vH7Oll7z/Rl7M8vokerhmzceYCNu5yd/sZdB9i086DP+
vveBAL6Rm99/MaEliUC\nfwLp2lF1hmTu+hl2/gQ7f3Z+/
BVtA7j0NScBNGzjv4JnDdTzr+wFUQfyi8ja10/W3jwe8FF0Lbew\
nhHtnrDh0P06g7VF1SWtaj+PbH0Vak3p0bFqPtKb1uPLlH9iSXXaWrsoM37Rx/
MaETvQmguqcLPXV\ntfP+H2DLYueirF3rDu/48
zz67BOSnVr9CSlO3X5vjdrBMSMDCmFG8UAez3+GLXm5tE5O4c7ibgT2\nTN/
f6Me/s4wFG3fRrnE9svbmk7Uvj6x9+exwd/4
HCnwUmfNh4th0OjSpR7uj6lInwfdVuePP7m5d\
nO8ZEkLAkAhE5B3gaiAdeVtVHarSBZdMomvkHEordb6U5m537cGQyKCl2Ttbu3w4HsmB
/lnP7q8fL\ndu0U5cEPzzm367eEpl2cMg1NuzrTNDbt4uzo4+JY8P4L9Fp0Lyly
+IRnrtZhRec/cEIA4fvakd/9\n7jK2782lX4fGZB8sJCe3kOzcQnIOFhy6nX3Q+
b0iM4fikiMHARQUlzB5nnOkUrdOPM0bJNG8QTI9\nWzdkcLdmNG+
QTPMGSbRomMzt05aQ5aPoWpvUFM7o3qLC+K1rx5jIEvJRQyISD/
wEDAUygAXAKFVd\n5e85lR01dPDR7tTNLXtBVVF8CgkdBzo7+/1
ZTreOj3LLiu9JlRXhwO2/UJRQn4LiEoqKlcLiEgqL\
nlaKSEgqLlMKSEsa9vpCTD85mfMI0WssutmgTHiu6jK+
SBvPb0zqTV1BMbqH7U1BCbmERuYeWlbAy\nM+
fQiJqKiEDD5ERS6yaSmpJIw5REvvl5p+91geUPnE39pPLzv69RPymJ8Tx8UW/
bmRsTQWrzqKH+\nwDpV/QVARKYCIwC/iaCyknN9jMEH4otyWb1+
I7skld30ZpekslMasZOj2ElDdmoqWZrK2yW30zau\n7M40s6QJg/7
xfUAxvM8g3i8YdOTC3CIe+2QtIs6ONSUxnuTEeOrWiSeljnM7NSWx3CTw+
nX9Sa2b\nSKOURFJT6tAgOYG4uCPTlr+
ia61TUypMAmDf6I2JNeFIBG0Az7OpGcCJ3iuJyDhgHED79u0r1cCW\nkia+d+
TalJe6v4KIECcQJ0JcHMSL0EqgjQhxIjz2w2U8kvgydT26dg5qHR4ruoy/
nNeDhHghIT6O\nOvFCQlwciQlxJMYJifFxJMQLd7y9lJ37y46Db9UomS/
vGExSQly5E5z425G3SU3h1K7NKnz/d57d\
nrdp99Hay1pjYUWtPFqvqi8CL4HQNVea5L9cZw/jC58rsyF+uM4YnL+9
b4fMHrhrK3Xsp07WzqOFQ\
nnjm1U4XPv3dYT5874rvO6U5yYsVln6u7I7dv9MaYyghHIsgE2nncb+
suqzF9h43jvveKuFWnHtqR\n/5MrGDRsXEDPd3bEBUd07aQkxvNwiHbENbEjt2
/0xphAheNkcQLOyeIhOAlgATBaVVf6e05VSkxU\nd2IRm5jEGBPpanWtIRE5D/
gnzvDRiar6UHnrR2StIWOMCbPaPGoIVf0I+CgcbRtjjDmS70tDjTHG\
nxAxLBMYYE+
MsERhjTIyzRGCMMTEuImYoE5EdwKYwNd8U8F28x9q39q19a792x9BBVSssRxARiSCc
\nRGRhIMOvrH1r39q39iM1BusaMsaYGGeJwBhjYpwlgoq9aO1b+9a+tR9GQY/
BzhEYY0yMsyMCY4yJ\ncZYIjDEmxlki8CIi8SKyWEQ+dO83FpHPReRn9/
dRQWz7NhFZKSIrRGSKiCQHu30RmSgiWSKywmPZ\
n4yKyRkSWich7IpLq8dg9IrJORNaKyNnBaN9d/gc3hpUi8lgQ228nIl+
KyCq3rVvc5X4/95qMwV/7\nHo//SURURJqGsn0R6SsiP4jIEhFZKCL9g9R+
sojMF5GlbvsPuMtDsg36a999LFTbYMD7nJpu+xBV\ntR+PH+B24E3gQ/f+
Y8Dd7u27gUeD1G4bYAOQ4t6fBowNdvvAqcDxwAqPZWcBCe7tR0vbBHoCS4Ek\
noCOwHogPQvunA7OAJPd+8yC23wo43r3dAGeujJ7+PveajsFf++79
dsCnOBdTNg1l+8BnwLnu8vOA\
nOUFqX4D67u1EYB4wIFTbYDnth3IbDGifE4y2S3/
siMCDiLQFhgEveyweAbzm3n4NGBnEEBKAFHEm\n76kLbAl2+6
r6NbDba9lnqlrk3v0BZxY53Fimqmq+qm4A1gH9qQZf7QM3AY+oar67TlYQ29+
qqj+6\nt/cBq3GSsr/
PvUZjKKd9gKeA8YDniI5Qta9AQ3e1RjjbYjDaV1Xd795NdH80VNugv/
YJ0TZYyX1O\njW//
pSwRHOmfOP94JR7LWqjqVvf2NqBFMBpW1UzgCeBXYCuQo6qfhar9clwHfOzebgNs9ngsg8M7
\nrZrUFThFROaJyFcickIo2heRNOA4nG+F/j73oMXg2b6IjAAyVXWp12ohaR+4
FXhcRDbjbJf3BKt9\
nt2tkCZAFfK6q87xWCeo26Kf9UG2DldnnBO1vb4nAJSLnA1mqusjfOuocnwVlvK3bDzgC55CvNVBP
```

```
\nRMaEqn0/Mf0FKAImh6pNVwLQGOcQ/
U5gmohIMBsUkfrAu8CtqrrX87FQfO6e7eN85n8G7gtmm/7a\nd9//
TcBtqtoOuA14JVhtq2qxqvbF+dbfX0R6ecQV9G3QT/tB3wbDvc/
xZIngsIHABSKyEZgKnCEi\nk+
MjSQUAAAQjSURBVIDtItIKwP2d5f8lquVMYIOq7lDVQmA6cHII2z+
CiIwFzgeudDdGcOaYbuex\nWlt3WU3LAKa7h+3zcb4tNQ1W+yKSiLMTnKyq093F
/j73Go/BR/udcb4QLHW3x7bAjyLSMkTtA1yD\
nsw0CvM3hLoigbQOqmg18CZzjxjWWEG6DXu2HYhus7D4neP9/NXGiIdp+
gMEcPnHzOEeeuHksSG2e\nCKzEOTcgOH2DfwhF+0
AaR56sPQdYBTTzWu8YjjxZ9Qs1cLLKR/s3Ag+6t7viHA5LMNp3X/d14J9e\ny31
+7jUdg7/2vdbZyOGTxSFpH+
dcwWD39hBgUZDabwakurdTgG9wdv4h2QbLaT9k26DbxmAq2OcE\nq21VtUQQwB+
lCfAF8DPOKILGQWz3AWANsAJ4w/2DB7V9YArOOYlCnG9B1+OchNoMLHF/nvdY/
y84\noxXW4o4qCUL7dYBJ7ufwI3BGENsfhHPovczj/
Z5X3udekzH4a99rnY24iSBU7bvLF7k7nnlAvyC1\n3wdY7La/
ArjPXR6SbbCc9kO2DbqvOZgA9jnBaFtVrcSEMcbEOjtHYIwxMc4SgTHGxDhLBMYYE
+Ms\nERhjTIyzRGCMMTHOEoGJKW6lzbO9lt0qIv+pxGv82ev+
dzUVnzHhYMNHTUwRkXHASap6rceyH4Dx\n6hTAK++5gnNR0V5VrR/
cSI0JHTsiMLHmHWCYiNSBQ4XWWgPfiMidIrLArYFfWhc/za39/jrOxUWv\
n4FSIXSIik911SqtXIiJ3ichyt779I+6yziLyiYgsEpFvRKS7u/
xSceaeWCoi5SYhY4IpIdwBGBNK\
nqrpbROYD5wIzgStw5n4YCnTBqakjwPsicipONdguwDWq+
gM4O3B1ipQdQUTOxSkceKKqHhSRxu5D\nLwI36v+3d/++DEVhGMe/
j0XSsBgsRn9AEwa7SAwGkwETXfwBZqvBYJMwYCJhYpIYTIhEI35sBiBx\
nicTCYngN55QSIiHpcp7Pcpvb23vHp6ened6Ia0kDwBIwSCqVG46I++
bBK2at5iCwEm2SAqARBDVg\
nnDQM5Sxf00EKgDvgthECvxgC1iLiBd5Dp4NUHrjdVF7Zno+
HwLqkLT4K3sxazkFgJdoBFiX1AZWI\nqEuaAOYjYrn5wvzT0fM/
ntUGPH23goiImbxCGAHqkvoj4vEfzzL7E+8RWHEiTaQ6AFZJqwNIIyGn\
n8zd4JPVI6v7hFq+5uvmrfWBKUiXfoytSt/+NpLF8TpKq+
XVvRJxExBzwwOeKYbOWcRBYqTaBaj4S\
naRrcBnAs6ZK0qdz5w2dXgIvGZnFDROwBu8Bpnng1m9+aBGqSzklV46P5/
ELeWL4CjkhNn2Yt57+P\
nmpkVzisCM7PCOQjMzArnIDAzK5yDwMyscA4CM7PCOQjMzArnIDAzK9wbd+THG/
gPBUkAAAASUVO\nRK5CYII=\n",
   "text/plain": [
    "<matplotlib.figure.Figure at 0x7efe8da02470>"
   ]
  },
  "metadata": {},
  "output_type": "display_data"
 }
],
"source": [
 "pyplot.title('Time Complexity of Prim\\'s Algorithm')\n",
 "\n",
 "l1=pyplot.plot(x, y, 'o-', label='Prim(g)') \n",
 "\n",
 "avg = sum([y[i]/x[i] for i in range(len(x))])/len(x)\n",
 "\n",
 "l2=pyplot.plot(x, [(avg*v)**2 for v in x], 'o-', label='Apprx
O(V^2)')\n",
 "\n",
 "pyplot.legend()\n",
 "pyplot.ylabel('Time (Seconds)')\n",
 "pyplot.xlabel('Vertices')\n",
 "pyplot.xticks(range(40,440,40))\n",
```

```
   "pyplot.xticks(range(40,440,40))\n",
   "pyplot.show()"
  ]
 },
 {
  "cell_type": "markdown",
  "metadata": {},
  "source": [
   "## Question 2\n",
   "\n",
   "Illustrate a depth-first and breadth-first tree search,
   preferably\n",
   "using a stack and queue, using the tree of possible moves in a
   \n",
   "triangular peg solitaire game as the tree.\n",
   "\n",
   "The game I have in mind starts with this configuration\n",
   "\n",
   "          .\n",
   "\t    * *\n",
   "         * * *\n",
   "        * * * *\n",
   "      * * * * *\n",
   "\n",
   "where each * is a peg in a hole, and the . is an empty hole.\n
   ",
   "The goal is to remove pegs by jumping as in checkers, leaving\
   n",
   "one peg in the center as the final position.\n"
  ]
 },
 {
  "cell_type": "markdown",
  "metadata": {},
  "source": [
   "Switching from a breadth-first search (BFS) to a depth-first
   search (DFS) is as simple as switching from a queue to stack.
   In a BFS, from the root node you add children to a queue. Those
    children are then visited at that depth and the process
   repeated for their children at another depth. If you switch to
   a stack you now have a DFS. You will search down an entire
   branch(s) before reaching a second child from the root.
   Arguably the hard part of this problem is the generation of the
    tree and representation of the game. I've chose to represent
   it in a pseudo matrix form, where each hole is a coordinate:\n
   ",
   "    \n",
   "<center>    (0,0),(1,0),(2,0),(3,0),(4,0)<br/>\n",
   "       (0,1),(1,1),(2,1),(3,1)<br/>\n",
   "          (0,2),(1,2),(2,2)<br/>\n",
   "             (0,3),(1,3)<br/>\n",
   "                (0,4)<br/>\n",
   "       </center>         \n",
   "This would make the corners points (0,0),(4,0) and (0,4). The
   center is (1,2). Pegs are represented as astriks and holes are
   periods."
```

```
 ]
},
{
 "cell_type": "code",
 "execution_count": 8,
 "metadata": {
  "collapsed": true
 },
 "outputs": [],
 "source": [
  "problem_board = [['*','*','*','*','*'], #0 - 0,1,2,3,4\n",
  "                 ['*','*','*','*'],     #1 - 0,1,2,3\n",
  "                   ['*','*','*'],       #2 - 0,1,2\n",
  "                     [ '*','*'],        #3 - 0,1\n",
  "                       [ '.']]          #4 - 0"
 ]
},
{
 "cell_type": "markdown",
 "metadata": {},
 "source": [
  "Below are some utility functions for things like making moves,
   displaying the board, finding neighbors, and determining if a
  point is a peg or hole."
 ]
},
{
 "cell_type": "code",
 "execution_count": 75,
 "metadata": {
  "collapsed": false
 },
 "outputs": [],
 "source": [
  "import copy\n",
  "\n",
  "def print_board(board):\n",
  "    \"\"\"\n",
  "    board = Array or arrays representing board.\n",
  "    \n",
  "    Given a board as
  [['*','*','*','*','*'],['*','*','*','*'],['*','*','*'], [
  '*','*'],[ '.']],\n",
  "    it pretty prints the board:\n",
  "    * * * * *\n",
  "     * * * *\n",
  "      * * *\n",
  "       * *\n",
  "        .\n",
  "    \"\"\"\n",
  "    i = 0\n",
  "    for row in board:\n",
  "        print(\"{}{}\".format(\" \"*i,' '.join(row)))\n",
  "        i+=1\n",
  "        \n",
  "def neighbors(p):\n",
  "    \"\"\"\n",
```

```
"    p = A point in tuple form (x,y)\n",
"    \n",
"    Returns a list of points surrounding it.\n",
"    \n",
"    >>>neighbors((1,2))\n",
"    [(2, 2), (0, 2), (0, 3), (1, 3), (1, 1), (2, 1)]\n",
"    \"\"\"\n",
"    x = p[0]\n",
"    y = p[1]\n",
"    return list(filter(lambda c: c[0] >= 0 and c[0] <= 4-c[1]
and c[1] >= 0 and c[1] < 5, [(x+1,y),(x-1,y),(x-1,y+1),(x,y+1)
,(x,y-1),(x+1,y-1)]))\n",
"\n",
"def ispeg(p,b):\n",
"    \"\"\"\n",
"    p = point (x,y)\n",
"    b = board in the form
[['*','*','*','*','*'],['*','*','*','*'],['*','*','*'], [
'*','*'],[ '.']]\n",
"    \n",
"    Returns true if the point is a '*' peg\n",
"    \"\"\"\n",
"    return [(x,y) for x,y in filter(lambda c: b[c[1]][c[0]] ==
 '*', p)]\n",
"\n",
"def valid_moves(b):\n",
"    \"\"\"\n",
"    b = board in the form
[['*','*','*','*','*'],['*','*','*','*'],['*','*','*'], [
'*','*'],[ '.']]\n",
"    \n",
"    Returns a list of valid moves on the board. Moves are in
the tuple form (peg to move, peg to remove, peg to land on)\n",
"    where each of those is points in tuple form ((x,y),(x1,y1)
,(x2,y2))\n",
"    \n",
"    >>>valid_moves(board in the form
[['*','*','*','*','*'],['*','*','*','*'],['*','*','*'], [
'*','*'],[ '.']])\n",
"    [[(0, 2), (0, 3), (0, 4)], [(2, 2), (1, 3), (0, 4)]]\n",
"    \"\"\"\n",
"    moves=[]\n",
"    for y in range(len(b)):\n",
"        for x in range(len(b[y])):\n",
"            if b[y][x] != '*':\n",
"                continue\n",
"\n",
"            p = (x,y)\n",
"            n = neighbors(p)\n",
"            pn = ispeg(n,b)\n",
"            for m in pn:\n",
"                nx = 0\n",
"                ny = 0 \n",
"\n",
"                if m[0]==p[0]:\n",
"                    nx=m[0]\n",
"                if m[0] < p[0]:\n",
```

```
"                    nx=m[0]-1\n",
"                if m[0] > p[0]:\n",
"                    nx=m[0]+1\n",
"\n",
"                if m[1]==p[1]:\n",
"                    ny=m[1]\n",
"                if m[1] < p[1]:\n",
"                    ny=m[1]-1\n",
"                if m[1] > p[1]:\n",
"                    ny=m[1]+1\n",
"\n",
"                if (nx < 0 or ny < 0) or ((nx,ny) not in
neighbors(m)) or b[ny][nx] == '*':\n",
"                    continue\n",
"\n",
"                moves.append([p,m,(nx,ny)])\n",
"    return moves\n",
"\n",
"def move(m,b):\n",
"    \"\"\"\n",
"    m = Move to be made in the form ((x,y),(x1,y1),(x2,y2))\n
",
"    b = board\n",
"    \n",
"    Returns a new board instance with the move made.\n",
"    \"\"\"\n",
"    nb = copy.deepcopy(b)\n",
"    nb[m[0][1]][m[0][0]] = '.'\n",
"    nb[m[1][1]][m[1][0]] = '.'\n",
"    nb[m[2][1]][m[2][0]] = '*'\n",
"    return nb\n",
"\n",
"\n",
"def pegcount(b):\n",
"    \"\"\"\n",
"    b = board\n",
"    \n",
"    Returns the number of pegs on the board.\n",
"    \"\"\"\n",
"    return sum(x.count(\"*\") for x in b)\t\n",
"\n",
"def gen_tree(b):\n",
"    \"\"\"\n",
"    b = Board\n",
"    \n",
"    Returns a reference to a root node of a tree of all
possible moves. \n",
"    Nodes are in dict form:\n",
"    Node {\n",
"        board:b\n",
"        valid_moves:[]\n",
"        children:[]\n",
"    }\n",
"    Children of a node are references to nodes with the parent
 moves made on their board and so on.\n",
"    \"\"\"\n",
"    n={\"board\":b}\n",
```

```
   "    n[\"valid_moves\"] = valid_moves(n[\"board\"])\n",
   "    n[\"children\"] = []\n",
   "\n",
   "    for m in n[\"valid_moves\"]:\n",
   "        child = gen_tree(move(m,b))\n",
   "        child[\"move\"] = [m]\n",
   "        n[\"children\"].append(child)\n",
   "\n",
   "    return n"
  ]
 },
 {
  "cell_type": "markdown",
  "metadata": {},
  "source": [
   "Generating the tree takes up most of the time -- I average
   around 55 seconds. This is mainly due to me copying the board
   every move."
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 77,
  "metadata": {
   "collapsed": false
  },
  "outputs": [
   {
    "name": "stdout",
    "output_type": "stream",
    "text": [
     "It took 58.63520812988281 seconds to generate the tree of
   moves\n"
    ]
   }
  ],
  "source": [
   "import time\n",
   "\n",
   "s = time.time()\n",
   "root = gen_tree(problem_board)\n",
   "gentime = time.time()-s\n",
   "\n",
   "print(\"It took {} seconds to generate the tree of moves\".
   format(gentime))"
  ]
 },
 {
  "cell_type": "markdown",
  "metadata": {},
  "source": [
   "Here is my depth-first search. Without a specific goal in mind
   , my search returns a dictionary containing a lot of
   information found along the way. I keep track of end games
   found, the moves to reach them, and also tally them based on
   the number of pegs left."
  ]
```

```
},
{
 "cell_type": "code",
 "execution_count": 78,
 "metadata": {
  "collapsed": true
 },
 "outputs": [],
 "source": [
  "def dfs(root):\n",
  "    \n",
  "    stack = root[\"children\"]\n",
  "    results = {\"info\":{}}\n",
  "    paths = {}\n",
  "    \n",
  "    while stack:\n",
  "        node = stack.pop()\n",
  "        \n",
  "        children = node[\"children\"]\n",
  "        for c in children:\n",
  "            c[\"move\"]= node[\"move\"]+c[\"move\"]\n",
  "        \n",
  "        if not children:\n",
  "            c = pegcount(node['board'])\n",
  "            if c not in results[\"info\"]:\n",
  "                results[\"info\"][c] = 0  \n",
  "                paths[c] = []\n",
  "            results[\"info\"][c] += 1\n",
  "            paths[c].append(node[\"move\"])\n",
  "            \n",
  "        stack.extend(children)\n",
  "        \n",
  "    results['paths'] = paths\n",
  "    return results"
 ]
},
{
 "cell_type": "markdown",
 "metadata": {},
 "source": [
  "My breadth-first search is the same code with the stack
  exchanged for a queue. I'm using python's collection library
  because a default list's method of pop(0) is not as efficient
  as deque's popleft()."
 ]
},
{
 "cell_type": "code",
 "execution_count": 79,
 "metadata": {
  "collapsed": true
 },
 "outputs": [],
 "source": [
  "from collections import deque\n",
  "\n",
  "def bfs(root):\n",
```

```
    "    \n",
    "    queue = deque(root[\"children\"])\n",
    "    results = {\"info\":{}}\n",
    "    paths = {}\n",
    "    \n",
    "    while queue:\n",
    "        node = queue.popleft()\n",
    "        \n",
    "        children = node[\"children\"]\n",
    "        for c in children:\n",
    "            c[\"move\"]= node[\"move\"]+c[\"move\"]\n",
    "        \n",
    "        if not children:\n",
    "            c = pegcount(node['board'])\n",
    "            if c not in results[\"info\"]:\n",
    "                results[\"info\"][c] = 0  \n",
    "                paths[c] = []\n",
    "            results[\"info\"][c] += 1\n",
    "            paths[c].append(node[\"move\"])\n",
    "            \n",
    "        queue.extend(children)\n",
    "        \n",
    "    results['paths'] = paths\n",
    "    return results"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "Running the BFS:"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 80,
   "metadata": {
    "collapsed": false
   },
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "Tree generation took 55.356807231903076 seconds.\n",
      "The BFS took 3.0062286853790283 seconds.\n",
      "\n",
      "Pegs\t\tNumber\n",
      " Remaining\t of Games\n",
      "=========================\n",
      "1\t\t29760\n",
      "2\t\t139614\n",
      "3\t\t259578\n",
      "4\t\t123664\n",
      "5\t\t14844\n",
      "6\t\t844\n",
      "7\t\t324\n",
```

```
      "8\t\t2\n",
      "==========\n",
      "Total: \t\t568630\n"
     ]
    }
   ],
   "source": [
    "# BFS\n",
    "import time\n",
    "\n",
    "s = time.time()\n",
    "root = gen_tree(problem_board)\n",
    "gentime = time.time()-s\n",
    "\n",
    "s2 = time.time()\n",
    "results = dfs(root)\n",
    "dfstime = time.time()-s2\n",
    "\n",
    "print(\"Tree generation took {} seconds.\".format(gentime))\n",
    "print(\"The BFS took {} seconds.\".format(dfstime))\n",
    "\n",
    "print(\"\\nPegs\\t\\tNumber\")\n",
    "print(\" Remaining\\t of Games\")\n",
    "print(\"==========\")\n",
    "for p,n in sorted(results[\"info\"].items()):\n",
    "    print(\"{}\\t\\t{}\".format(p,n))\n",
    "print(\"==========\")\n",
    "print(\"Total: \\t\\t{}\".format(sum(results[\"info\"].values\n",
    "()))"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "Running the DFS:"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 81,
   "metadata": {
    "collapsed": false
   },
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "Tree generation took 62.894432067871094 seconds.\n",
      "The BFS took 2.132901668548584 seconds.\n",
      "\n",
      "Pegs\t\tNumber\n",
      " Remaining\t of Games\n",
      "==========\n",
      "1\t\t29760\n",
```

```
      "2\t\t139614\n",
      "3\t\t259578\n",
      "4\t\t123664\n",
      "5\t\t14844\n",
      "6\t\t844\n",
      "7\t\t324\n",
      "8\t\t2\n",
      "=============================\n",
      "Total: \t\t568630\n"
     ]
    }
   ],
   "source": [
    "# DFS \n",
    "import time\n",
    "\n",
    "s = time.time()\n",
    "root = gen_tree(problem_board)\n",
    "gentime = time.time()-s\n",
    "\n",
    "s2 = time.time()\n",
    "results = bfs(root)\n",
    "bfstime = time.time()-s2\n",
    "\n",
    "print(\"Tree generation took {} seconds.\".format(gentime))\n",
    "\n",
    "print(\"The BFS took {} seconds.\".format(bfstime))\n",
    "\n",
    "print(\"\\nPegs\\t\\tNumber\")\n",
    "print(\" Remaining\\t of Games\")\n",
    "print(\"=============================\")\n",
    "for p,n in sorted(results[\"info\"].items()):\n",
    "    print(\"{}\\t\\t{}\".format(p,n))\n",
    "print(\"=============================\")\n",
    "print(\"Total: \\t\\t{}\".format(sum(results[\"info\"].values\n",
    "()))"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "The time complexity of my queue implementaion is O(V+E) as the\n",
    " total time spent looking in the adjacency list is E (the\n",
    "number of edges) and all of V (vertices) are added to the queue\n",
    " in constant time. This is the same big O behavior for the DFS.\n",
    " Push and pop on the stack are constant so there is V nodes\n",
    "added, and E edges are visited. In my implementation both\n",
    "algorithms run about the same pace.\n",
    "\n",
    "Interestingly, in all the games with 1 peg left not a single\n",
    "one ends with it in the center of the board (point (1,2)). My\n",
    "results are in agreement with Keith Wannamaker's results— he\n",
    "is a software engineer who published a paper on the number of\n",
    "games and various endings starting from different positions.<\n",
    "sup>[2]</sup> "
   ]
```

```
},
{
 "cell_type": "code",
 "execution_count": 86,
 "metadata": {
  "collapsed": false
 },
 "outputs": [],
 "source": [
  "for path in results[\"paths\"][1]:\n",
  "    if path[-1][-1] == (1,2):\n",
  "        print(path)\n",
  "# No games end in the center"
 ]
},
{
 "cell_type": "markdown",
 "metadata": {},
 "source": [
  "Also in my returned results are the games themselves. Attached
   is results.txt which contain all 29760 games ending in one peg
  . Below is an example game of a random 1 peg solution."
 ]
},
{
 "cell_type": "code",
 "execution_count": 298,
 "metadata": {
  "collapsed": false
 },
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
    "Starting position\n",
    "* * * * *\n",
    " * * * *\n",
    "  * * *\n",
    "   * *\n",
    "    .\n",
    "Move (2, 2) to (0, 4) and remove (1, 3)\n",
    "* * * * *\n",
    " * * * *\n",
    "  * * .\n",
    "   * .\n",
    "    *\n",
    "Move (0, 2) to (2, 2) and remove (1, 2)\n",
    "* * * * *\n",
    " * * * *\n",
    "  . . *\n",
    "   * .\n",
    "    *\n",
    "Move (0, 4) to (0, 2) and remove (0, 3)\n",
    "* * * * *\n",
    " * * * *\n",
    "  * . *\n",
```

```
"    . .\n",
"     .\n",
"Move (3, 1) to (1, 3) and remove (2, 2)\n",
"* * * * *\n",
" * * * .\n",
"  * . .\n",
"    . *\n",
"     .\n",
"Move (0, 1) to (0, 3) and remove (0, 2)\n",
"* * * * *\n",
" . * * .\n",
"  . . .\n",
"    * *\n",
"     .\n",
"Move (3, 0) to (1, 2) and remove (2, 1)\n",
"* * * . *\n",
" . * . .\n",
"  . * .\n",
"    * *\n",
"     .\n",
"Move (2, 0) to (0, 2) and remove (1, 1)\n",
"* * . . *\n",
" . . . .\n",
"  * * .\n",
"    * *\n",
"     .\n",
"Move (1, 3) to (1, 1) and remove (1, 2)\n",
"* * . . *\n",
" . * . .\n",
"  * . .\n",
"    * .\n",
"     .\n",
"Move (0, 3) to (0, 1) and remove (0, 2)\n",
"* * . . *\n",
" * * . .\n",
"  . . .\n",
"    . .\n",
"     .\n",
"Move (0, 0) to (0, 2) and remove (0, 1)\n",
". * . . *\n",
" . * . .\n",
"  * . .\n",
"    . .\n",
"     .\n",
"Move (0, 2) to (2, 0) and remove (1, 1)\n",
". * * . *\n",
" . . . .\n",
"  . . .\n",
"    . .\n",
"     .\n",
"Move (1, 0) to (3, 0) and remove (2, 0)\n",
". . . * *\n",
" . . . .\n",
"  . . .\n",
"    . .\n",
"     .\n",
"Move (4, 0) to (2, 0) and remove (3, 0)\n",
```

```
      ". . * . .\n",
      " . . . .\n",
      "  . . .\n",
      "   . .\n",
      "    .\n"
     ]
    }
   ],
   "source": [
    "b = problem_board\n",
    "print(\"Starting position\")\n",
    "print_board(b)\n",
    "for m in results[\"paths\"][1][0]:\n",
    "    b = move(m,b)\n",
    "    print(\"Move {} to {} and remove {}\".format(m[0],m[2],m
    [1]))\n",
    "    print_board(b)"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "More interesting I find is the worst-case senario from the
    starting position. There are only two \"solutions\" that end
    with 8 pegs and end in the same position. For example:"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 85,
   "metadata": {
    "collapsed": false
   },
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "Starting position\n",
      "* * * * *\n",
      " * * * *\n",
      "  * * *\n",
      "   * *\n",
      "    .\n",
      "Move (0, 2) to (0, 4) and remove (0, 3)\n",
      "* * * * *\n",
      " * * * *\n",
      "  . * *\n",
      "   . *\n",
      "    *\n",
      "Move (2, 0) to (0, 2) and remove (1, 1)\n",
      "* * . * *\n",
      " * . * *\n",
      "  * * *\n",
      "   . *\n",
      "    *\n",
```

```
      "Move (3, 1) to (1, 1) and remove (2, 1)\n",
      "* * . * *\n",
      " * * . .\n",
      "   * * *\n",
      "     . *\n",
      "       *\n",
      "Move (0, 1) to (2, 1) and remove (1, 1)\n",
      "* * . * *\n",
      " . . * .\n",
      "   * * *\n",
      "     . *\n",
      "       *\n",
      "Move (2, 2) to (2, 0) and remove (2, 1)\n",
      "* * * * *\n",
      " . . . .\n",
      "   * * .\n",
      "     . *\n",
      "       *\n",
      "Move (0, 4) to (2, 2) and remove (1, 3)\n",
      "* * * * *\n",
      " . . . .\n",
      "   * * *\n",
      "     . .\n",
      "       .\n"
     ]
    }
   ],
   "source": [
    "b = problem_board\n",
    "print(\"Starting position\")\n",
    "print_board(b)\n",
    "for m in results[\"paths\"][8][0]:\n",
    "    b = move(m,b)\n",
    "    print(\"Move {} to {} and remove {}\".format(m[0],m[2],m
    [1]))\n",
    "    print_board(b)"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "## Question 3\n",
    "\n",
    "The task of finding a given string from a collection of
    strings can be\n",
    "solved by algorithms that fit into several paradigms,
    including brute\n",
    "force and (if the list is sorted) divide-and-conquer.\n",
    "\n",
    " * Given an example of an algorithm that fits each of these\n
    ",
    " descriptions, and describe their O() behavior.\n",
    "\n",
    " * Can you come up with a third algorithm design technique
    that can be\n",
```

```
   " applied to this problem, a representative algorithm, and its
   O()\n",
   " behavior?\n",
   "\n",
   "(You don't need to code these - just discuss what's going on.)
   "
  ]
 },
 {
  "cell_type": "markdown",
  "metadata": {
   "collapsed": true
  },
  "source": [
   "In string matching the most basic approach is bruteforcing— a
    variation on sequential searching. With N being the search
   space or collection of strings, and M being pattern to look for
   , bruteforcing works as follows:  \n",
   "\n",
   "For each character in N, compare it to the first character of
   M. If they don't match, you move forward a character in N. If
   the two characters match, then continue matching the next
   character of N and M. You continue this search pattern until
   you find M within N or not at all.\n",
   "\n",
   "Bruteforcing is quite slow, and the time complexity is O(N*M)
   as you compare each element of N agaisnt M. One way to improve
   runtime might be preprocessing the list to be sorted in some
   desired way.<sup>[4]</sup>\n",
   "\n",
   "If the search space is already sorted, a binary search would
   be an optimal divide-and-conquer algorithm. Given a sorted
   search space N, a binary search first compares the middle
   element of N with search pattern M. If the pattern matches the
   middle element, its position is returned. If the pattern is
   less than the middle value, the process is repeated on the
   lower half of the search space until found. If the pattern is
   greater than the middle value, the process is repeated on the
   upperhalf. Since each comparison in the binary search halves
   the search space, it is easy to assert that the time complexity
    is a logarithm: O(log(N)).<sup>[5]</sup>\n",
   "\n",
   "Greedy algorithms are another paradigm of pattern matching
   algorithms. An aglorthm behind some of the fastest search
   algorithms out there is the Boyer-Moore algorithm. In the
   vanilla Boyer-Moore algorithm you search for occurrences of the
    pattern M in the set N by performing explicit character
   comparisons at different alignments. Unlike in bruteforcing,
   you skip as many alignments as possible using heuristic
   information from preprocessing the set. The shifts are
   calculated on two rules: the good-suffix shift and the bad-
   character shift. Preprocessing the set takes O(N) time but the
   searching happens in O(N*M). I've included the Boyer-Moore
   algorithm in the greedy algorithm catergory but it sort of
   lives in limbo between greedy and bruteforce. It uses heuristic
    information like a greedy algorithm, but pure greedy
   algorithms usually return a suboptimal solution.<sup>[6]</sup><
```

```
    sup>[7]</sup>"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "## Question 4\n",
    "\n",
    "You're given a list of N names and told that you'll need to
    search\n",
    "the list for a given name M times.  The following are
    suggested:\n",
    "\n",
    "  * using a hash table\n",
    "  * using a heap\n",
    "  * sequential search\n",
    "  * sorting followed by binary search\n",
    "\n",
    "Discuss the time efficiency of these approaches as the size of
     N\n",
    "and M vary.  Which one would you suggest and why?  Would your
    answer\n",
    "change if you needed to change the list of names by adding and
    \n",
    "deleting names (say, P times) between searches?  \n"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "A hash table would be the best solution— even more so if you
    wanted to add or delete names. Search, insertion, and deletion
    operatons all happen in constant time. The only constraint
    would be picking a good hash function that provides a uniform
    distribution of hashes. The less uniform it is, the more you
    need to compensate for collisions with execution time.\n",
    "\n",
    "Sorting the array and using a binary search would leave you
    with a binary tree as search, insertion, and deletion operatons
     would happen in O(log(N)) time. Although a hastable is a
    faster implementation for the problem at hand, it fails for
    relative searchs— things like the \"next smallest\" or \"one
    greater than\". The binary search tree would provide this
    relation-type information.\n",
    "\n",
    "My second choice would be a heap. The structure is tree-like
    similar to binary search tree except nodes must be ordered
    either greatest to least or least to greatest. Insertion and
    deletion happens in O(long(N)) but finding the min or max would
     be constant time. Again, this is not relavant to the problem
    but is a benefit over the binary search tree.\n",
    "\n",
    "A sequential search (as talked about in question 3) would be
    my last resot option. Insertion, deletion, and search would all
     be O(N)."
```

```
   ]
  }
 ],
 "metadata": {
  "kernelspec": {
   "display_name": "Python 3",
   "language": "python",
   "name": "python3"
  },
  "language_info": {
   "codemirror_mode": {
    "name": "ipython",
    "version": 3
   },
   "file_extension": ".py",
   "mimetype": "text/x-python",
   "name": "python",
   "nbconvert_exporter": "python",
   "pygments_lexer": "ipython3",
   "version": "3.6.0"
  }
 },
 "nbformat": 4,
 "nbformat_minor": 2
}
```

Programming Languages Exam

fraction_sum_search.c

```
/*

fraction_sum_search.c

DESC:

A brute-force solution to the problem: what permutation of the
    digits from 1 to 9 will add to 1 when arranged in a form
    similar to 1/23 + 4/56 + 7/89 ?

RUN:

$ gcc fraction_sum_search.c -o fss
$ ./fss
5/34 + 7/68 + 9/12 = 1
5/34 + 9/12 + 7/68 = 1
7/68 + 5/34 + 9/12 = 1
7/68 + 9/12 + 5/34 = 1
9/12 + 5/34 + 7/68 = 1
9/12 + 7/68 + 5/34 = 1
Execution time: 0.018168 seconds

*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>

#define ARRAY_LEN 9

unsigned cc(unsigned x, unsigned y) {
    unsigned pow = 10;
    while(y >= pow){ pow *= 10; }
    return x * pow + y;
}

void swap(int *x, int *y){
  if (x != y) {
    *x ^= *y;
    *y ^= *x;
    *x ^= *y;
  }
}

void printArray(int array[]){
  for (int i = 0; i < ARRAY_LEN; ++i){
    printf("%i", array[i]);
  }
  printf("\n");
}
```

```
void pp(int a[]){
  printf("%i/%i%i + %i/%i%i + %i/%i%i = 1\n", a[0], a[1], a[2], a
    [3], a[4], a[5], a[6], a[7], a[8]);
}

int testArray(int a[]){
  if ((float)a[0]/cc(a[1],a[2])+(float)a[3]/cc(a[4],a[5])+(float)a
    [6]/cc(a[7],a[8]) == 1){
    return 1;
  }
  return 0;
}


void permute(int *array,int i,int length) {

  if (length == i){
    if (testArray(array)==1){
      pp(array);
    }
  }

  int j = i;
  for (j = i; j < length; j++) {
    swap(array+i,array+j);
    permute(array,i+1,length);
    swap(array+i,array+j);

  }
}

int main(int argc, char const *argv[]){
  clock_t begin,end;
  double time_spent;
  int digits[ARRAY_LEN];

  begin = clock();

  for (int i = 0; i < ARRAY_LEN; ++i){
    digits[i] = i+1;
  }

    permute(digits, 0, ARRAY_LEN);

    end = clock();

    printf("Execution time: %f seconds\n", (double)(end - begin) /
    CLOCKS_PER_SEC);

  return 0;
}
```

fraction_sum_search.js

```
/*

fraction_sum_search.js

DESC:

A brute-force solution to the problem: what permutation of the
    digits from 1 to 9 will add to 1 when arranged in a form
    similar to 1/23 + 4/56 + 7/89 ?

RUN:

nodejs fraction_sum_search.js
[ 5, 3, 4, 7, 6, 8, 9, 1, 2 ]
[ 5, 3, 4, 9, 1, 2, 7, 6, 8 ]
[ 7, 6, 8, 5, 3, 4, 9, 1, 2 ]
[ 7, 6, 8, 9, 1, 2, 5, 3, 4 ]
[ 9, 1, 2, 5, 3, 4, 7, 6, 8 ]
[ 9, 1, 2, 7, 6, 8, 5, 3, 4 ]
Execution time: 703ms

*/

var permArr = [],usedChars = [];

function permute(input) {
  var i, ch;
  for (i = 0; i < input.length; i++) {
    ch = input.splice(i, 1)[0];
    usedChars.push(ch);
    if (input.length == 0) {
      permArr.push(usedChars.slice());
    }
    permute(input);
    input.splice(i, 0, ch);
    usedChars.pop();
  }
  return permArr
};

function cc(x,y) {
  return x*10+y;
}

console.time("Execution time");

var a = [];
for (var i = 1; i <= 9; i++) {
    a.push(i);
}

perms = permute(a);

for (var i = 0; i < perms.length; i++) {
```

```
    if (perms[i][0]/cc(perms[i][1],perms[i][2])+perms[i][3]/cc(perms[
      i][4],perms[i][5])+perms[i][6]/cc(perms[i][7],perms[i][8])==1){
      console.log(perms[i])
    }
}

console.timeEnd("Execution time");
```

fraction_sum_search_functional.py

```
"""

fraction_sum_search_functional.py

DESC:

A brute-force solution to the problem: what permutation of the
    digits from 1 to 9 will add to 1 when arranged in a form
    similar to 1/23 + 4/56 + 7/89 ?

RUN:
$ python3 fraction_sum_search_functional.py
[(5, 3, 4, 7, 6, 8, 9, 1, 2),
 (5, 3, 4, 9, 1, 2, 7, 6, 8),
 (7, 6, 8, 5, 3, 4, 9, 1, 2),
 (7, 6, 8, 9, 1, 2, 5, 3, 4),
 (9, 1, 2, 5, 3, 4, 7, 6, 8),
 (9, 1, 2, 7, 6, 8, 5, 3, 4)]
Execution time: 0.6779532432556152

"""


import itertools
import time

def cc(x,y):
  return x*10+y

def check(p):
  return float(p[0])/cc(p[1],p[2])+float(p[3])/cc(p[4],p[5])+float(
    p[6])/cc(p[7],p[8])==1

t = time.time()

print(list(filter(lambda x: check(x),list(itertools.permutations(
    range(1,10))))))

print("Execution time: {}".format(time.time()-t))
```

fraction_sum_search_imperative.py

```
"""

fraction_sum_search_imperative.py

DESC:

A brute-force solution to the problem: what permutation of the
    digits from 1 to 9 will add to 1 when arranged in a form
    similar to 1/23 + 4/56 + 7/89 ?

RUN:

$ python3 fraction_sum_search_imperative.py
(5, 3, 4, 7, 6, 8, 9, 1, 2)
(5, 3, 4, 9, 1, 2, 7, 6, 8)
(7, 6, 8, 5, 3, 4, 9, 1, 2)
(7, 6, 8, 9, 1, 2, 5, 3, 4)
(9, 1, 2, 5, 3, 4, 7, 6, 8)
(9, 1, 2, 7, 6, 8, 5, 3, 4)
Execution time 0.6378731727600098 seconds.

"""

import itertools
import math
import time

def cc(x, y):
    return x*10+y

digits = range(1,10)

perms = list(itertools.permutations(digits))

t = time.time()
for p in perms:
  if (float(p[0])/cc(p[1],p[2])+float(p[3])/cc(p[4],p[5])+float(p
    [6])/cc(p[7],p[8])==1):
    print("{}".format(p))

print("Execution time {} seconds.".format(time.time()-t))
```

fraction_sum_search_OOP.py

```
import itertools
import time

class Permuter(object):
  """docstring for ClassName"""
  def __init__(self, r):
    super(Permuter, self).__init__()
    self.r = r
    self.p = itertools.permutations(self.r)

  def next(self):
    return self.p.next()

  def perms(self):
    return self.p

  def listAll(self):
    return list(itertools.permutations(self.r))

def cc(x, y):
    return x*10+y

t = time.time()

p = Permuter(range(1,10))

results = []

for l in p.listAll():
  if (float(l[0])/cc(l[1],l[2])+float(l[3])/cc(l[4],l[5])+float(l
    [6])/cc(l[7],l[8])==1):
    results.append(l)

print(results)

print("Execution time {} seconds.".format(time.time()-t))
```

network_graph.py

```
import argparse, sys
from urllib.error import URLError
from urllib.parse import urlparse,urljoin
from urllib.request import urlopen
from html.parser import HTMLParser
import gzip
import zlib
from io import BytesIO

def validate_url(url):
  p = urlparse(url)
  return True if all([p.scheme, p.netloc]) and p.scheme != 'mailto'
     else False

class LinkParser(HTMLParser):
  def __init__(self):
    HTMLParser.__init__(self)
    self.links = []

  def handle_starttag(self, tag, attrs):
    if tag != 'a':
      return
    attr = dict(attrs)
    try:
      self.links.append(attr["href"])
    except KeyError:
      return

  def feed(self, data):
    self.output = []
    HTMLParser.feed(self, data)
    return self.links

def decode_content_body(data, encoding):
  if encoding == 'identity':
    text = data
  elif encoding in ('gzip', 'x-gzip'):
    io = BytesIO(data)
    with gzip.GzipFile(fileobj=io) as f:
      text = f.read()
  elif encoding == 'deflate':
    try:
      text = zlib.decompress(data)
    except zlib.error:
      text = zlib.decompress(data, -zlib.MAX_WBITS)
  else:
    return None

  return text

def process_url(url):
  parser = LinkParser()

  try:
```

```
    res = urlopen(url)
  except URLError:
    return None

  content_encoding = res.headers.get('Content-Encoding', 'identity
    ')
  res_body_plain = decode_content_body(res.read(), content_encoding
    )

  if not res_body_plain:
    return None

  links = parser.feed(res_body_plain.decode("utf-8"))


  for i in range(len(links)):
    if links[i] is None:
      del links[i]
    if validate_url(links[i]) == False:
      links[i] = urljoin(url, links[i])

  return links


def main(args):

  visited = []

  nodes={}

  links = process_url(args.hostname)

  total = int(args.n);

  while (total and links):
    link = links.pop(0)
    total-=1

    cur_netloc = urlparse(link)[1]

    if cur_netloc and cur_netloc not in nodes:
      nodes[cur_netloc] = []

    print("Visiting #{}: {}".format(total,link))
    if (link not in visited):
      visited.append(link)
      more_links = process_url(link)
      if more_links is not None:
        print("Urls {} found. The queue is now: {}".format(len(
    more_links),len(links)))
        for new_link in more_links:
          if new_link not in visited:
            links.append(new_link)
            new_netloc = (urlparse(new_link)[1])
            if new_netloc and new_netloc != cur_netloc and
    new_netloc not in nodes[cur_netloc]:
              nodes[cur_netloc].append(new_netloc)
```

```
  # Dot file
  f = open('network_graph.dot', 'w')
  f.write('digraph G {\n')
  f.write('graph [splines=ortho, nodesep=1, rankdir = "TB"]\n')
  for node,connected in nodes.items():
    for c in connected:
        if c in nodes:
          if node in nodes[c]:
            f.write('"{}" -> "{}";\n'.format(node,c))
  f.write('}\n')
  f.close()

  print(total)
  return

if __name__ == '__main__':
  parser = argparse.ArgumentParser(description = 'Network Graph')
  parser.add_argument('hostname', help='The website URL from which
    to spidering.')
  parser.add_argument('-n', metavar='N', type=int, help='Number
    of urls to parse from the queue',default=200)
  args = parser.parse_args()

  if validate_url(args.hostname) == False:
    parser.print_help()
    sys.exit()

  main(args)

""" python3 network_graph.py http://cs.marlboro.edu && dot
    network_graph.dot -Tpng -o output.png && eog -f output.png """
```

network_graph.dot

```
digraph G {
graph [splines=ortho, nodesep=1, rankdir = "TB"]
"minds.marlboro.edu" -> "www.marlboro.edu";
"cs.marlboro.edu" -> "www.marlboro.edu";
"cs.marlboro.edu" -> "nook.marlboro.edu";
"cs.marlboro.edu" -> "akbar.marlboro.edu";
"courses.marlboro.edu" -> "www.marlboro.edu";
"courses.marlboro.edu" -> "nook.marlboro.edu";
"nook.marlboro.edu" -> "courses.marlboro.edu";
"nook.marlboro.edu" -> "www.marlboro.edu";
"nook.marlboro.edu" -> "cs.marlboro.edu";
"www.marlboro.edu" -> "marlboro.askadmissions.net";
"www.marlboro.edu" -> "minds.marlboro.edu";
"www.marlboro.edu" -> "nook.marlboro.edu";
"www.marlboro.edu" -> "marlboro.us2.list-manage.com";
"www.marlboro.edu" -> "www.youtube.com";
"www.marlboro.edu" -> "cs.marlboro.edu";
"www.marlboro.edu" -> "courses.marlboro.edu";
"www.marlboro.edu" -> "akbar.marlboro.edu";
"marlboro.askadmissions.net" -> "www.marlboro.edu";
"akbar.marlboro.edu" -> "cs.marlboro.edu";
"akbar.marlboro.edu" -> "www.marlboro.edu";
"www.youtube.com" -> "www.marlboro.edu";
"marlboro.us2.list-manage.com" -> "www.marlboro.edu";
}
```