Data Structures and Algorithms — January 8, 2019
University of California, Irvine — EECS 114

Homework 1

Due on **Wednesday, January 23, 2019 by midnight**

Reading: Chapters 1-4

Mark the top of each submission with your name and course number.

You will often be called upon to give an algorithm to solve a certain problem. Your write-up should take the form of a short essay. A topic paragraph should summarize the problem you are solving and what your results are. The body of the essay should provide the following:

1. A description of the algorithm in English and, if helpful, pseudo-code.
2. Atleast one worked example or diagram to show more precisely how your algorithm works.
3. A proof (or indication) of the correctness of the algorithm.
4. An analysis of the running time of the algorithm. Remember, your goal is to communicate. Full credit will be given *only* to correct solutions which are described clearly. Convoluted and obtuse descriptions will receive low marks.

1 *(6 Points)* Asymptotic Notation

For each group of functions, sort the functions in increasing order of asymptotic (big-O) complexity:

1. *(2 Points)* **Group 1**

$$f_1(n) = n^{0.999999} \lg n$$
$$f_2(n) = 10000000n$$
$$f_3(n) = 1.000001^n$$
$$f_4(n) = n^2$$

2. *(2 Points)* **Group 2**

$$f_1(n) = 2^{2^{1000000}}$$
$$f_2(n) = 2^{100000n}$$
$$f_3(n) = n \lg n$$
$$f_4(n) = n\sqrt{n}$$

3. *(2 Points)* **Group 3**

$$f_1(n) = n^{\sqrt{n}}$$
$$f_2(n) = 2^n$$
$$f_3(n) = n^{10}.2^{n/2}$$
$$f_4(n) = \sum_{i=1}^{n}(i+1)$$

2 *(20 Points)* Recurrences

Give asymptotic upper and lower bounds for $T(n)$ in each of the following recurrences. Assume that $T(n)$ is constant for $n \leq 10$. Make your bounds as tight as possible, and justify your answers.

(a) $T(n) = 2T(n/3) + n \lg n$

(b) $T(n) = 3T(n/5) + \lg^2 n$

(c) $T(n) = T(n/2) + 2^n$

(d) $T(n) = T(\sqrt{n}) + \Theta(\lg \lg n)$

(e) $T(n) = 10T(n/3) + 17n^{1.2}$

(f) $T(n) = 7T(n/2) + n^3$

(g) $T(n) = T(n/2 + \sqrt{n}) + \sqrt{6046}$

(h) $T(n) = T(n-2) + \lg n$

(i) $T(n) = T(n/5) + T(4n/5) + \Theta(n)$

(j) $T(n) = \sqrt{n}T(\sqrt{n}) + 100n$

3 *(44 Points)* Sorting

Part A (9 Points) – Do Exercise 2.3-5 and 2.3-6 on page 39 in CLRS.
Part B (35 Points)– You will implement (a) the binary search version of the insertion sort algorithm in C/C++, (b) mergesort algorithm in C/C++, and (c) compare the asymptotic running time of the two sorting algorithms.

3.1 Getting the scaffolding code

We will use the `git` distributed version control system. The baseline code for this assignment is available at this URL: `https://github.com/EECS114/hw1.git`

To get a local copy of the repository for your work, you need to use git to clone it. To do that, run the following command .

```
$ git clone https://github.com/EECS114/hw1.git
```

If it works, you will see some output similar to the following:

```
Initialized empty Git repository in hw1
remote: Enumerating objects: 10, done.
remote: Counting objects: 100% (10/10), done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 10 (delta 1), reused 10 (delta 1), pack-reused 0
Unpacking objects: 100% (10/10), done.
```

There will be a new directory called **hw1**.

3.2 Compiling and running your code

We have provided a small program, broken up into modules (separate C/C++ files and headers), that performs sorting. For this homework, you will make all of your changes to just one file as directed below. We have also provided a `Makefile` for compiling your program. To use it, just run `make`. It will direct you with the right flags to produce the executable. For example, `make insertion-sort` will produce an executable program called `insertion-sort` along with an output that looks something like the following:

```
$ make insertion-sort
g++  -O3 -g -o driver.o -c driver.cc
g++  -O3 -g -o sort.o -c sort.cc
g++  -O3 -g -o insertion-sort.o -c insertion-sort.cc
g++ -O3 -g -o insertion-sort driver.o sort.o insertion-sort.o
```

Run `insertion-sort` on an array of size `100` as follows:

```
$ ./insertion-sort 100
```

3.3 C/C++ style guidelines

Code that adheres to a consistent style is easier to read and debug. Google provides a style guide for C++ which you may find useful: `http://google-styleguide.googlecode.com/svn/trunk/cppguide.xml`.

Part of your grade on assignments is based on the readability of your code.

3.4 Insertion Sort

Although we've given you a lot of code, to create an insertion sort using binary search, you just need to focus on editing `insertion-sort.cc`. Right now, it is mostly empty. In this file, implement the insertion sort algorithm. After implementing and testing the correctness of your insertion sort algorithm, execute the program for different powers of n such that $n = \{2^0, 2^1, 2^2, ...2^{16}, ..., 2^{26}\}$. Record the average execution times for sorting in a table. Vary n until its value is 2^{26} or the program runs out of memory.

3.5 Mergesort

To create a mergesort you just need to focus on editing `mergesort.cc`. Right now, it is mostly empty. In this file, implement the mergesort algorithm described in class. After implementing and testing the correctness of your mergesort algorithm, execute the program for different powers of n such that $n = \{2^0, 2^1, 2^2, ...2^{16}, ..., 2^{26}\}$. Record the average execution times for sorting in a table. Vary n until its value is 2^{26} or the program runs out of memory. Finally, compare and explain the difference in the running time of the two algorithms. Why is one faster than the other?

4 Login to your Unix account

For this class, you will be doing your assignments by *logging on* to a shared machine (server) running the Unix operating system. Even though you may be using a personal computer or a workstation that is capable of computation locally, you will mainly be using them as *terminals* (clients), whose job is to pass keystrokes to the server and display outputs from the server.

To use a shared machine, first you need an *account* on the machine. You need to get your username and password for the EECS account from OIT if you don't already have them. You should contact the EECS 114 TA's, if you have not received your user name / password by end of the first week.

The name of the instructional server is `laguna.eecs.uci.edu`. You can log into your account with your user name and password. Your account also comes with a certain amount of disk space. You can use this space to store homework assignment files, and you don't need to bring your own disks or other storage media.

If laguna.eecs.uci.edu is down, then you can try another machine, such as bondi.eecs.uci.edu, ladera.eecs.uci.edu, zuma.eecs.uci.edu, or crystalcove.eecs.uci.edu. You can use the same user name and password, and your files will be the same.

4.1 Software and commands for remote login

You can connect to bondi.eecs.uci.edu from on campus. If you are off campus, you will have to use the UCI VPN. See http://www.oit.uci.edu/vpn/ for more information.

Use **ssh** as the primary way to connect to the server. **ssh** stands for *secure shell*, and it encrypts your network communication, so that your data cannot be understood by snoopers. For file transfers, use **sftp** or **scp**, which are secure. You could also set up an *ssh-tunnel* so that previously unencrypted communications can be encrypted.

Depending on what computer you use, it may have a different *implementation* of **ssh**, but the basic function underneath are all the same.

- If you are logging in from a Win32 machine, you can use **PuTTY**.
- MacOS X already has this built-in (use Terminal or X11 to run a unix shell). Most Linux distributions also bundle **ssh**.
- If you are logging in from an X terminal, you can use the command
  ```
  % ssh laguna.eecs.uci.edu -X -l yourUserName
  ```
  (note: % is the prompt, not part of your command) It will prompt you for your password. Note that the -X option allows you to run programs that open X windows on your screen.

4.2 Unix Shell

By now you should be logged in, and you should be looking at the prompt

```
ladera% _
```

Note: In the following writeup, we will show just

```
%
```

for the prompt, instead of

```
ladera%
```

If you login to another machine, such as laguna, then the prompt would instead look like

```
laguna%
```

You should change your password using the **passwd** command. The password will be changed on all the EECS machines, not just on laguna.eecs.uci.edu.

Try out the following commands at the shell prompt.

ls	list files
cd	(change working directory)
pwd	(print working directory)
mkdir	(make directory)
mv	(rename/move files)
cp	(copy files)
rm	(remove files)
rmdir	(remove directory)
cat	(print the content of a file)
more	(print the content of a file, one screen at a time)
echo	(print the arguments on the rest of the command line)

Most commands take one or more file names as parameters. When referring to files, you may need to qualify the file name with directory references, absolute vs. relative paths:

`.`	(current directory)
`..`	(one level higher)
`~`	(home directory)
`/`	the root (top level) directory

5 Learn to use a text editor

There are three editors that are available on nearly all unix systems that you may choose from.
pico is the easiest to get started with. A guide for **pico** can be found at:
`http://www.dur.ac.uk/resources/its/info/guides/17Pico.pdf`.
Other editors include **vim** and **emacs**.

Learn how to edit a file, move the cursor, insert text, insert text from file, delete words, delete lines, cut/paste, save changes, save to another file, quit without saving.

There is nothing to turn in for this part. However, it is critical that you get enough practice with your editor, so that you can do the homeworks for this class.

6 Typescript

A typescript is a text file that captures an interactive session with the unix shell. Very often you are required to turn in a typescript to show that your program runs correctly. To create a typescript, use the **script** command. Here is an example:

- Type the command
  script
  into the shell. It should say
  ```
  Script started, file is typescript
  ```

```
% _
```
This means it is recording every key stroke and every output character into a file named “typescript”, until you hit **^D** or type **exit**.

- Type some shell commands; you can also enter the python prompt. But don’t start a text editor!
- Stop recording the typescript by typing **exit**.
  ```
  % exit
  Script done, file is typescript
  % _
  ```
- Now you should have a text file named `typescript`. Make sure it looks correct.
  ```
  % more typescript
  Script started on Thu Jan 5 25 23:43:56 2019
  ...
  ...
  ```

You should immediately rename the typescript to another file name. Otherwise, if you run **script** again, it will overwrite the `typescript` file.

Note: If you backspace while in script, it will show the `^H` (control-H) character in your typescript. This is normal. If you use **more** to view the typescript, then it should look normal.

7 Submission

When you’ve written up answers to all of the above questions, turn in your write-up for questions 1, 2 and 3-Part A in a separate file from the answers to 3-Part B. Your code should be zipped or turned into a tarball. Submit the three submission pieces (2 write-ups and .tar/.zip of your code by uploading it to eee.uci.edu dropbox.