Nathaniel Anderson
CptS 223 - Advanced Data Structures in C++

Written Homework Assignment 1: Math Review, Big-O, Recursion and General Linux/Git Topics

Assigned: Monday, February 1, 2021
Due: Sunday, February 14, 2021

I. Problem Set:

1. (15, -1 pts/rank) Order the following set of functions by their growth rate (from fastest to slowest - rank 1 - 12, where 1 is the fastest and 12 is the slowest). Hint: you can plot their curves in a X-Y axis using http://fooplot.com/:

Unordered Complexities	Ordered Complexities
N	9
√N	10
N^1.5	5
N^2	4
N log N	7
N log(log(N))	8
N log^2 N	6
2/N	12
2^N	1
2^(N/2)	2
37	11
N^2 log(N)	3

2. (15 pts) A program takes 35 seconds for input size 20 (i.e., n=20). Ignoring the effect of constants, approximately how much time can the same program be expected to take if the input size is increased to 100 given the following runtime complexities?

a. O(N)

Given = Time$_{original}$ (t_o) = 35 & Number$_{original}$ (N_o) = 20

t_o = constant (r) * N_o => 35 = r * 20

t_{new} = r * 100 (the increase in number, we need to find the increase in time)

With this we get a system of equations which can be solved as follows:

t_{new}/35 = r * 100/r * 20 (r cancels and we multiply by 35)

t_{new} = 35*(100/20) = 175seconds

ANSWER = 175s

b. O(N + log N)

Using similar methods as shown in part as we will solve to get the same answer.

O(N + log N) = O(N) So,

ANSWER = 175s

c. O(N^3)

Using the same methods as part a, with a few modifications to the solving of the system of equations:

All we have to do is set up the same equations but put the N terms (20 and 100) to the cube root, which is as follows:

t_{new} = 35*(100^3/20^3) = 4375seconds

ANSWER = 4375s

d. O(2^N)[1]

Using the same methods as part a, with a few modifications to the solving of the system of equations as follows:

[1] You might need an online calculator with arbitrarily large numbers for this one. Scientific notation and 8 significant figures is just fine.

t_{new} = 35*(2^100/2^0) = 4.2312403E25

ANSWER = 4.2312403*10^25

3. (10 pts) How many nodes in a complete trinary tree of depth 5? Hint: use geometric series.
   Number of nodes = 121

4. (15 pts) Write a simple recursive function to calculate (and return) the height of a general binary tree T. The height of a tree T is defined as the number of levels below the root. In other words, it is equal to the length of the longest path from the root (i.e., number of edges along the path from the root to the deepest leaf). Note that the term “nodes” is used to include both internal nodes and leaf nodes. You can assume the following tree node structure:

```
class Node
{
        Node *left; // points to the left subtree
        Node *right; // points to the right subtree
}
```

   Your answer can be in C++ syntax or in the form of a generic pseudocode.

```
 int height(Node* root)
{
     int depthLeft;
     int depthRight;

     If (root == nullptr) return 0;

     depthLeft = height(root->left);
     depthRight = height(root->right);

     if(depthLeft > depthRight) return 1 + depthLeft;
     else return 1 + depthRight;
}
```

5. (15 pts) Rewrite the pseudocode presented in class for the Fibonacci numbers *without* recursion (hint: use loop) and discuss the pros and cons of recursion compared to iteration.

```
Int main(n)

     Previous1 = -1
     Previous2 = 1

     for (n < 0, n--)
          Next term = Previous1 + Previous2
          Previous1 = Previous2
          Previous2 = Next term
          output next term
```

Recursion is a very helpful tool/concept in certain situations. Recursion can cut down time it takes for a program to execute, if applied correctly. Recursion is much better at traversing trees than iteration. On the other hand, if used in the wrong application, recursion could take much longer than iteration. For example, running this Fibonacci program with a loop is MUCH faster than using recursion when running high values of *n*. Both have their weaknesses and strengths, it is all in *how* we use them to maximize efficiency.

6. (10 pts) What is Git and what is the purpose of using Git in general?

Git is a useful software that makes collaboration and working on programming projects intuitive and easier (if you know your way around git commands…). Git tracks and saves the changes made to your uploaded files, making for a good record of what kind of work has been done on a project. It is very useful if applied correctly.

7. (10 pts) What is the Linux tool gdb? What is the difference between cmake and make?

gdb is a Linux tool/command that stands for GNU Debugger. It helps debug programs in C++ when called in the Linux terminal.

Make is a system that builds your code after activating the compiler. Cmake is much more in that it is more complex and high-level. Cmake was made to compile and build C++ projects. In general, bigger projects benefit from Cmake, with its cross-platform discovery and ease of use and make benefits from smaller, less demanding projects on just one platform.

8. (10 pts) How do argc and argv variables get set if the program is called from the terminal and what values do they get set with?

```
int main(int argc, char* argv[])
{
        return(0);
}
```

These two variables are command line arguments. Argc stands for argument count and contains the amount of arguments passed, and is stored as an int. Argv is a an array of strings composed of ptrs that point to characters and if the arguments passed is greater than zero it points to a character. In this case argc = 1 and argv = the letter C.

II. Submitting Written Homework Assignments:

1. On your local file system, create a new directory called HW1. Move your HW1.pdf file in to the directory. In your local Git repo, create a new branch called HW1. Add your HW1 directory to the branch, commit, and push to the remote origin which is your private GitHub repo.
2. Do not push new commits to the branch after you submit your link to Canvas otherwise it might be considered as late submission.
3. Submission: You must submit a URL link to the branch of your private GitHub repository. Please add the GitHub accounts of the instructor and two TAs (see Syllabus) as the collaborators of your repository. Otherwise, we won't be able to see your repository.

III. Grading Guidelines:

This assignment is worth 100 points. We will grade according to the following criteria:

- See above problems for individual point totals.