CptS 223 - Homework #1

Big-O and general Git topics

Please complete the homework problems on the following page. Note that this is an individual assignment and all work must be your own. Be sure to show your work when appropriate.

You may use any editor you like (or even print it out, *legibly* write in answers, and scan it in), but convert it to *PDF* for submission. I have provided MS Word (doc) and LibreOffice (ODF) versions for your platform of choice.

Once you have your PDF file, put it into your Git repository in the HW1 directory, commit and push it. Once you've pushed your PDF file up, put something onto Blackboard so the TA knows to grade your work.

1. [5] Order the following set of functions by their growth rate:

Unordered Complexities	Ordered Complexities
N	2/N
√N	37
N^1.5	√N
N^2	N
N log N	N log(log(N))
N log(log(N))	N log N
N log^2 N	N log^2 N
2/N	N^1.5
2^N	N^2
2^(N/2)	N^2 log(N)
37	N^4
N^2 log(N)	2^(N/2)
N^4	2^N

2. [5] 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 run-time complexities? Chapter 2.1 notes that: T(N) ≤ cf(N). For this you'll need to find the c (constant scaling factor) for a given Big-O growth rate.

1. O(N)
   O(20) = c*N = c*20 = 35s -> c = 35s/20, O(100) = c*N = (35s/20)*100 = 175s
2. O(N + log N)
   O(20) = c*(N + log N) = c*(20 + log 20) = 35s -> c = 35s/(20 + log 20),
   O(100) = c*(N + log N) = (35s/(20 + log 20))*(100 + log 100) ~= 167.598s
3. O(N^3)
   O(20) = $c*N^3 = c*20^3$ = 35s -> c = $35s/20^3$, O(100) = $(35s/20^3)*100^3$ = 4375s
4. O(2^N)[1]
   O(20) = $c*2^N = c*2^{20}$ = 35s -> c = $35s/2^{20}$, O(100) = $(35s/2^{20})*2^{100}$ = $4.231x10^{25}$s

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

3. [8] Given the following two functions:

```
int g(int n)
{
   if(n <= 0)
   {
      return 0;
   }
   return 1 + g(n - 1);
}
```

```
int f(int n)
{
   int sum = 0;
   for(int i = 0; i < n; i++)
   {
      sum += 1;
   }
   return sum;
}
```

A. [2] State the runtime complexity of both f() and g()

g() is O(N) runtime, f() is O(N) runtime.

B. [2] State the memory (space) complexity for both f() and g()

g() is O(1) space, f() is O(1) space.

C. [4] Write another function called "int h(int n)" that does the same thing, but is significantly faster.

```
int h(int n) {

      (n > 0) ? return n : return 0;

}
```

4. [5] State g(n)'s runtime complexity:

```
int f(int n){
   if(n <= 1){
      return 1;
   }
   return 1 + f(n/2);
}

int g(int n){
   for(int i = 1; i < n; i *= 2){
      f(i);
   }
}
```

$O(\log_2(N)*\log_2(N)) = O(\log^2 N)$

5. [5] What is the runtime complexity of Adam's famous string splitter code? Hint: Make sure to look into the source code for string.find() in the C++ std library. I've included that code (downloaded from GNU).

```
static vector<String> split(String text, String delimiter)
{
        vector<String> pieces;
        int location = text.find(delimiter);
        int start = 0;

      //while we find something interesting
        while ( location != String.Length() ){

            //build substring
                string piece = text.substring(start, location - start);
                pieces.push_back(piece);
                start = location + 1;

            //find again
                location = text.indexOf(delimiter, start);
        }too
        string piece = text.substr(start, location - start);
        pieces.push_back(piece);
        return pieces;
}
```

```
// Excerpt from OpenJDK's implementation of string searching in
//  src/java.base/share/classes/java/util/regex/Pattern.java
//  The key component is how long it takes to run to find the next match
boolean match(Matcher, int i, CharSequence seq) {
            if (i > matcher.to - minLength) {
                matcher.hitEnd = true;
                return false;
            }
            int guard = matcher.to - minLength;
            for (; i <= guard; i++) {
                if (next.match(matcher, i, seq)) {
                    matcher.first = i;
                    matcher.groups[0] = matcher.first;
                    matcher.groups[1] = matcher.last;
                    return true;
                }
            }
            matcher.hitEnd = true;
            return false;
```

N = text length

match: O(N)

split: O(N) + O(N)*O(N) + O(N) = $O(N^2)$

6. [10] (adapted from the 2012 ICPC programming competition) Write an algorithm to solve the following problem and specify its runtime complexity using the most relevant terms:

Given a nonnegative integer, what is the smallest value, k, such that

$$n, 2n, 3n, \ldots, kn$$

contains all 10 decimal numbers (0 through 9) at least once? For example, given an input of "1", our sequence would be:

$$1, 2(1), 3(1), 4(1), 5(1), 6(1), 7(1), 8(1), 9(1), 10(1)$$

and thus k would be 10. Other examples:

Integer Value	K value
10	9
123456789	3
3141592	5

Algorithm:

```
int kOf(int n) {
	int k = 0;
	int curr = n;
	String numbers = "0123456789";
	while (numbers.length() > 0) {
		k++;
		curr = n * k;
		String currString = Integer.toString(curr);
		int currLength = currString.length();
		for (int i = 0; i < currLength; i++) {
			String currChar = Character.toString(currString.charAt(i));
			if (numbers.contains(currChar)) {
				numbers = numbers.replace(currChar, "");
			}
```

```
            }
        }
        return k;
}
```

Runtime:

N = number of digits in n

D = number of decimals

O(k) * O(N) * O(D) * O(D) =

$O(k * N * D^2)$

If D = 10:

$O(k * N * D^2)$ =

$O(k * N * 10^2)$ =

$O(k * N * 100)$ =

$O(k * N)$

7. [18] Provide the algorithmic efficiency for the following tasks. Justify your answer, often with a small piece of pseudocode or a drawing to help with your analysis.

A. [3] Determining whether a provided number is odd or even

```
Efficiency: O(1)
Justification: (number % 2 == 0) ? positive : negative;
```

B. [3] Determining whether or not a number exists in a list

```
Efficiency: O(N) (if unsorted)
Justification:
for (int a : list) { if (a == number) return number exists in list;}
return  number doesn't exist in list;
```

C. [3] Finding the smallest number in a list

```
Efficiency: O(N) (if unsorted)
Justification:
current_smallest = list[1];
for (int a : list) { if (a < current_smallest) a = current_smallest;}
return  current_smallest;
```

D. [3] Determining whether or not two **unsorted** lists of the same length contain all of the same values (assume no duplicate values)

```
Efficiency: O(N^2)
Justification:
int same = 0;
for (a : list_1) { for (b : list_2) {
if (list_1[a] == list_2[b]) same++;  } } return same == list_1.size();
```

E. [3] Determining whether or not two **sorted** lists contain all of the same values (assume no duplicate values)

```
Efficiency: O(N)
Justification:
for (int i = 0; i < list_1.size(); i++) {
if (list_1[i] != list_2[i]) return false; } return true;
```

F. [3] Determining whether a number is in a BST

```
Efficiency: O(height)
Justification: You have to navigate down one BST branch, which is worst
case "height" tall.
```

8. [4] What is Git and what is it for?

Git is a version control tool typically used for coordinating versions of programs being developed by multiple-person teams.

9. [2] If I need to get a copy of a Git repository off of the GitLab server, what command do I use?

git clone [ssh or https link]

10. [2] Once I've created/edited/removed a file in my Git repository, what command do I use to stage it for committing?

git add [changed file path]

11. [2] Once I've staged all of my changes, which command do I use to create the next version of the repository?

git commit -m "[commit message]"

12. [2] Now that I've created at least one new update to my repository, which command do I use to send those changes to the GitLab server?

git push

13. [2] If the server has had updates from another computer, which command do I use to get these changes on my local computer without starting from a whole new copy of the Git repository?

git pull

14. [4] How does this variable get set and what is it get set with?

```
public static int main(String     args[]) {
      return(0);
}
```

args is set when you run the program, and it is set with whatever arguments are passed in.