CS 2420-001 Algorithms and Data Structures

Fall Semester, 2018

Assignment 1: Algorithm Analysis

Due Date: 1:30 p.m., Friday, Sept. 14, 2018 (**at the beginning of CS2420 class)**

(**Note:** This assignment has **seven** questions, including both written questions and programming questions. For the programming questions, please submit only your source files (.java files) on Canvas. For written questions, you may either submit it on Canvas or write on papers and bring it to me in class before/on the due day. Note that the Canvas online submission will be automatically closed at 1:30 p.m. on Sept. 14, so please make your submission on time.)

1. For each of the following pairs of functions, indicate whether it is one of the three cases: $f(n) = O(g(n))$, $f(n) = \Omega(g(n))$, or $f(n) = \Theta(g(n))$. For each pair, you only need to give your answer and the proof is not required. **(20 points)**

   (a) $f(n) = n^2 + 35n + 6$ and $g(n) = n^4 + 12$.

   (b) $f(n) = 5 + \log^2 n + 15n$ and $g(n) = 3\log^8 n + \log^3 n$.

   (c) $f(n) = 64n^4 + 4n^2$ and $g(n) = n^4 + 3n^8 + 10n^2$.

   (d) $f(n) = n \log n$ and $g(n) = n + n^2$.

   (e) $f(n) = 2^n$ and $g(n) = 4n^{1000} + 7n^{300} - 6$.

2. For each of the following program fragments, give the running time using big-O notation of n. You only need to give the answer. **(20 points)**

   (a)
   ```
   sum = 0;
   for (i = 0; i < n; i++)
       sum++;
   ```

   (b)
   ```
   sum = 0;
   for (i = 0; i < n; i++)
       for (k = 0; k < n * n; k++)
           sum++;
   ```

   (c)
   ```
   sum = 0;
   for (i = 0; i < n; i++)
       for (k = 0; k < i; k++)
           sum++;
   ```

   (d)
   ```
   sum = 0;
   for (i = 0; i < n; i++)
       for (k = 0; k < i * i; k++)
           sum++;
   ```

(e)
```
sum = 0;
for (i = 0; i < n; i++)
    for (k = i; k < n; k++)
        sum++;
```

3. Give the time complexity of the following recursive function using big-O notation of n. You only need to give the answer. **(5 points)**

```
void fun(int n)
{
    if (n == 1)
        System.out.print(" * ");
        return;
    fun(n − 1);
    System.out.println();
    for (int i = 0; i < n; i++)
        System.out.print(" * ");
}
```

4. Consider sorting n numbers stored in an array $A[0, \ldots, n-1]$ by first finding the smallest element of A and exchanging it with the element in $A[0]$. Then, find the second smallest element of A, and exchange it with $A[1]$. Continue this manner for the first $n-1$ elements of A. This algorithm is known as **selection sort**. **(15 points)**

   (a) Write the pseudocode for this algorithm.

   (b) Give the (worst-case) running time of this algorithm using the big-O notation.

   (c) Give the **best-case** running time of this algorithm using the big-O notation.

5. Write a Java program to implement the **selection** sort algorithm introduced in the last question. **(15 points)**

   On Canvas, go to "Files" and navigate to the following directory: homework/hw1/question5. There is a "starter" java file "hw1_Q5.java" and an input file "hw1_Q5_input.txt". The program first reads the numbers in the input file and stores them in array A. After the numbers in the array are sorted, the numbers will be output to an output file called "hw1_Q5_output.txt" (they will be output on the console/screen too). You task is to complete the function "selectionSort( )". In addition, please follow the instructions below, which are also applicable to Question 6 (and programming exercises in future assignments).

   - The input and output files are only used for grading. When you test/debug your program, you may use other ways for the input and output. But when you submit your java file, you should use the same input/output format as in the starter java file (for example, you may only complete the function selectionSort() and leave everything else unchanged). When grading your code, I may use an input file with different numbers.

- Please submit only your java file to Canvas.
- Please try to write your code in a way that is easy to read. Give some appropriate comments wherever you feel necessary.

To help you check whether your program runs correctly, a file "solution_hw1_Q5_output.txt", which contains the correct output, is in the same directory. For your convenience, all above files are included in a zip file.

6. Implement the binary search algorithm discussed in class. **(15 points)**

   On Canvas, go to the following directory: homework/hw1/question6. There is a starter java file "hw1_Q6.java", a data file "hw1_Q6_data.txt", and a search file "hw1_Q6_search.txt". The program first reads the numbers from the data file and stores them in array A (the numbers are already sorted in the data file, so A is a sorted array). Then, the program reads the numbers in the search file and stores them in array B. For each number x of B, we search it in A by using binary search (i.e., if x is in A, then return its index in A; otherwise, return -1). The search results will be output in an output file "hw1_Q6_output.txt" (they will be output on the console/screen too). Your task is to complete the function "binarySearch()".

   Please also follow the instructions discussed in Question 5.

   To help you check whether your program runs correctly, a file "solution_ hw1_Q6_output.txt", which contains the correct output, is in the same directory. For your convenience, all above files are included in a zip file.

7. This exercise is to convince you that exponential time algorithms should be avoided.

   (10 points)

   Suppose we have an algorithm A whose running time is $O(2^n)$. For simplicity, we assume the algorithm A needs 2^n instructions to finish, for any input size of n (e.g., if $n = 5$, A will finish after $2^5 = 32$ instructions).

   According to Wikipedia, as of June 2018, the fastest supercomputer in the world is "Summit" (developed by IBM for use at Oak Ridge National Laboratory) and can perform about 122.3×10^{15} instructions per second.

   Suppose we run the algorithm A on Summit. Answer the following questions.

   (a) For the input size $n = 100$ (which is a relative small input size), how much time does Summit need to finish the algorithm? Give the time in terms of **centuries** (you only need to give an approximate answer).

   (b) For the input size $n = 1000$, how much time does Summit need to finish the algorithm? Give the time in terms of **centuries** (you only need to give an approximate answer).

Total Points: 100