CS2223: Algorithms

Assignment I

Teams: To be done individually

Release date: 03/17/2017

Due date: 03/24/2017 (11:59 PM)

Submission: Electronic submission only

General Instructions

- ***Executable vs. Pseudocode:*** Each question will explicitly state whether the deliverable is pseudocode or an executable program that the TA will run to give you a grade.

- ***Programming Language:*** If a question asking for an executable program, then choose a language from Java, C, C++, or Python. ***You must make it clear in your report:***
  - How to compile your program
  - How to execute it and with which arguments

- ***Submissions:*** The submission of Assignment I must be done electronically through blackboard system. All programs plus your report (.doc or .pdf) should be zipped into a single file and that is the file to submit.

Question 1 (Test the Order of Growth) [20 Points]

To observe the importance of algorithm efficiency and the order of growth, you are required to perform the following experiment. Assume an input of size 10,000 (denoted as N)

(*a) Linear function (O(N):* Write a function that loops over the input items (a single loop) and print the elapsed time taken by the function in milliseconds.

(b) Quadratic function ($O(N^2)$): Write a function that loops over the input items with two-levels of nesting and print the elapsed time taken by the function in milliseconds. The function should have two nested loops (outer and inner) where each one starts from 0 to N.

(c) Cubic function ($O(N^3)$): Write a function that loops over the input items with three-levels of nesting and print the elapsed time taken by the function in milliseconds. The function should have three nested loops where each one starts from 0 to N.

Hint: **The actual code inside the loop is not important. So make it simple and define a double variable in the beginning of your program, and then increment this variable inside the inner most loop (The example below is for two-level nesting).**

```
Double sum = 0;
For (i=0;  i<= N; i++)
    For(j=0; j<=N; j++)
         Sum += j;
    End Loop;
End Loop;
```

Deliverables of Question 1

(1) A single executable program. The program should contain the three functions mentioned above. The program takes one argument with values:
-- 1 (calls the linear function),
-- 2 (calls the quadratic function),
-- 3 (calls the cubic function).

(2) In your pdf report, plot a graph (e.g., Excel graph), where the x-axis is the passed argument (1, 2, or 3), and the y-axis is the execution time that you measured (can be in log scale if needed).

Question 2 (Linear Time Sorting) [20 Points]

The sorting algorithm taken so far in class, i.e., the Bubble sort, have $O(N^2)$ worst-case complexity. That is, given an input of size N, the algorithm may need $O(N^2)$ comparisons to produce the sorted output.

In this question, you are given an array of unsorted values of size 100 (the list has 100 elements), and each value is drawn randomly from the range of [0…1000] inclusive. You are required to design an algorithm to sort the given list in a linear time (that is O(N) worst-case performance).

Hint: Make use of the domain information given to you in the problem.

Deliverables of Question 2

(1) In the report, write down a pseudocode of the algorithm and write down any assumptions you have (Only pseudocode is needed not an executable code).

(2) Analyze your algorithm and state the Best-Case and Worst-Case time complexity.

(3) Analyze your algorithm and state the Best-Case and Worst-Case space complexity.

Question 3 (Order Of Growth) [20 Points]

Put these functions in order. Put the following functions in a list by increasing order of growth. That means, in your final list, if *f* comes before *g*, then f $\in$ O(g). If f $\in$ Θ(g) meaning the have the same order of complexity, then either one can come first.

$n \log_2 n$	$12\sqrt{n}$	$1/n$	$n^{\log_2 n}$
$100n^2 + 6n$	$n^{0.51}$	$n^2 - 324$	$50n^{0.5}$
$2n^3$	3^n	$2^{32}n$	$\log_2 n$

Deliverables of Question 3

(1) In the report, write the functions in their increasing order of growth

(2) State which functions fall into the same Θ order (that is, have the same order)

Question 4 (Variation of Binary Search) [20 Points]

Write a variation of Binary Search where instead of choosing the middle element each time to compare with, you will choose the 1/3 element each time to compare with and then decide to move left or right.

Hint: In binary search, we select the middle element between index m (smaller) and n (larger) as floor of (m + (m+n)/2). In your algorithm, your 1/3 element between index m and n is computed as floor of (m+ (m+n)/3)

Deliverables of Question 4

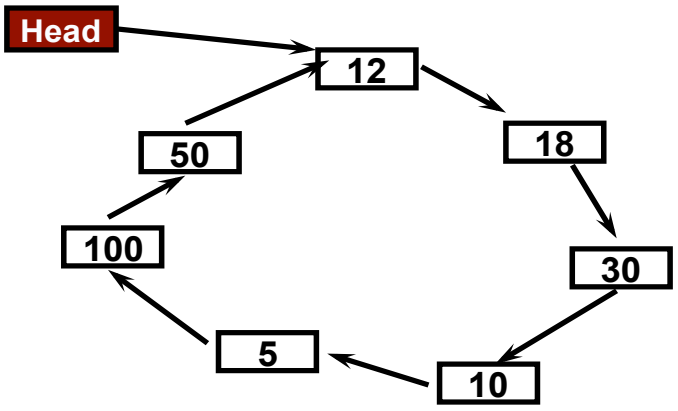
(1) Write an executable program to implement the algorithm described above. Your program should create in the beginning an array that stores only the even numbers between 1 and 400 (E.g., 2, 4, 6, …, 400) in ascending order.

(2) The program should take one argument which is the key K to be searched for. The program should print: (a) Whether K is found or not, (b) The number of comparisons done by the algorithm, and (c) The position at which K is found (if K exists).

(3) In your pdf report, analyze the above algorithm and state the Best-Case and Worst-Case time complexity in the Big O notation.

Question 5 (Operations on Circular Linked List) [20 Points]

A circular linked list is as shown in the figure below, it is a list of linked nodes, where each node has a pointer to it successor in the list (there are NO backward pointers). The list has ONE head pointer pointing to one node in the list. Assume the list is not sorted and has N nodes.



Deliverables of Question 5

(1) Write a pseudocode for an efficient algorithm for each of the following operations (no executable is needed):

- Insertion of a new node
- Deletion of a node with value *v*
- Searching for a node with value *v*

(2) State the worst case complexity (in Big O notation) for each of the three operations in 5.1

(3) Given your input list, the header pointer, and an addition pointer *P* (as indicated in the figure below). The goal is to delete the node pointed to by P and fix the pointers of the linked list. An naïve algorithm would do that in O(N). Your task is to write a pseudocode for an efficient algorithm that can this operation in O(1).