EECE 330 Data Structures and Algorithms
American University of Beirut, Fall 2009

November 25, 2009

Quiz I

Sections 1 & 2 – Ibrahim Abou-Faycal

Name:

ID:

Do not open this booklet until you are told to do so

- You have 90 minutes to complete the exam.
- Write all answers in this booklet. Answers written elsewhere will not be graded.
- This is a closed-book exam except for *one* double-sided *handwritten* A4 sheet that is allowed.
- Calculators are allowed.
- The problems are not necessarily in order of difficulty.
- A correct answer does not guarantee full credit and a wrong answer does not guarantee loss of credit. You should concisely indicate your reasoning and show all relevant work. The grade on each problem is based on our judgment of your level of understanding as reflected by what you have written.
- If we can't read it, we can't grade it.

Problem Q.1 (5 points) Asymptotic Growth

Sort the following functions by order of growth. Namely, find an arrangement $g_1, g_2, \ldots$ of the functions satisfying $g_1 = O(g_2), g_2 = O(g_3), \ldots$. (No explanation is necessary)

$$2n^4 + 5 \quad \big(\log n!\big)^3 \quad \frac{n^4}{\log n} \quad \big(\log n\big)^4 \quad \left(\sum_1^n j\right)^2$$

Problem Q.2 (4 points) Asymptotic Growth

Answer the following true/false questions: (No explanation is necessary)

$$\begin{aligned} 4\log n &= O(3\log n) \\ \frac{2^n}{500} &= O(n) \\ 4n+6 &= O(2^n + 24) \\ 4n+6 &= O(\log n + 10) \end{aligned}$$

Problem Q.3 (4 points) Worst vs. Best-Case Running Time

Answer the following questions and justify your answers.

(a) What is the worst-case running time of "insertion-sort"?

(b) What is the worst-case running time of "merge-sort"?

(c) What is the best-case running time of "insertion-sort"?

(d) What is the best-case running time of "merge-sort"?

Problem Q.4 (6 points) Recurrences

Write down the recurrence equation of the following: (No explanation is necessary)

(a) Binary search.

(b) Merge-sort.

(c) Post-order tree walk.

Problem Q.5 (10 points) Recurrences

Two algorithms A and A' have a worst-case running time of $T(n)$ and $T'(n)$ respectively. You are told that $T(n)$ and $T'(n)$ follow the following recurrence equations:

$$\begin{aligned} T(n) &= 7\,T(n/2) + n^2 \\ T'(n) &= a\,T'(n/4) + n^2 \end{aligned}$$

What is the largest value of a such that A' is "faster" than A? Justify your answer.

Problem Q.6 (6 points) Lists

Array Lists and Linked lists are considered among the elementary types of data structures.

(a) What are the benefits and disadvantages of using one (Array list) versus another (Linked List) in an implementation.

(b) Give an example where it may be more beneficial to use an Array list versus a Linked list, and another example where it may be better to use a Linked list.

Problem Q.7 (6 points) Lists

A friend of yours proposes the following method to implement a list: she creates an array of size n of *pointers* to objects, instead of an array of objects.

(a) What are the potential advantages of her data structure over your "regular" array implementation of the list?

(b) What are the potential advantages of her data structure over your "regular" linked implementation of the list?

Problem Q.8 (14 points) Dynamic Arrays

This problem studies the use of *dynamic* arrays to implement stacks. Once the array is full at capacity m, we allocate a new array twice the size ($2m$). We copy the data in the m array to the $2m$ array. Then we delete the m array, and we assume the $2m$ array to be current array. Assume we start with an array of only one element.

(a) How many times will we double the array to get it to be of size n?

(b) How many times do we have to copy the first element of the array if we end up with an array of size n?

(c) What about the number of times we copy the second element of the array if we end up with an array of size n?

(d) What is the total number of copy operations if we end up with an array of size n?

Problem Q.9 (10 points) Linked Lists

You are given a *Singly-Linked List* L and you would like to "shuffle" its content. More precisely, you would like to reorder the elements (nodes) of this list as follows:

- If the original order is: 1st, 2nd, 3rd, 4th, etc, ...,
- The new order is: 2nd, 1st, 4th, 3rd, 6th, 5th, 8th, 7th, etc ....

Write a pseudocode (or C++ code) for the SHUFFLE operation. Your algorithm *should not* create any new nodes.

Problem Q.10 (15 points) Intersection

The dynamic-set operation INTERSECTION takes two sets S_1 and S_2 as input, and it returns a set $S = S_1 \cap S_2$ consisting of all the elements of S_1 that *are also in* S_2.

You may assume that the elements of the sets may be ordered and that *within* each set, the elements are distinct.

(a) Show how to "efficiently" support INTERSECTION using a suitable list data structure.

(b) Would your answer change if you would like to support INSERT "efficiently" as well?

Problem Q.11 (12 points) Trees Walk

(a) Give an example of a tree with at least 3 nodes such that a pre-order walk is the same as an in-order walk and a post-order walk (if it is not possible explain why).

(b) Give an example of a tree with at least 3 nodes such that an in-order walk is the same as a post-order walk (if it is not possible explain why).

(c) Give an example of a tree with at least 3 nodes such that a pre-order walk is the same as a post-order walk (if it is not possible explain why).

Problem Q.12 (8 points) 3-Sum

Given a set $\mathcal{S}$ with n integers, describe an algorithm that determines whether there exist 3 elements in $\mathcal{S}$ such that their sum is 100. Be efficient.