Assignment 1

COMP 3804, Fall 2021

Upload in Brighspace by 11:59 PM on September 24, 2020

1 Guidelines

General guidelines are as follows:

1. Since we are only accepting assignments via the Brighspace system, no late submissions will be entertained after the cut-off time & date.
2. Please write clearly and answer questions precisely. It is your responsibility to ensure that what is uploaded is clearly readable. If we can't read, we can't mark!
3. Please cite all the references (including web-sites, names of friends, etc.) which you used/consulted as the source of information for each of the questions.
4. The first ten questions are worth 10 points each, totalling 100. The maximum mark for this assignment is 100. Bonus problem has ten extra points.
5. When a question asks you to design an algorithm - it **requires** you to
   (a) Clearly spell out the **steps** of your algorithm in pseudo code.
   (b) **Prove** that your algorithm is correct
   (c) **Analyze** the running time.
6. This assignment is based on the material that should have been learnt in COMP 1805/2402/2804. The purpose is to ensure that you have a good grasp of the background material to understand various topics in this course. If you have difficulty answering any of these questions, it is your responsibility to review them quickly.

2 Problems

1. (a) Define the functions $O()$, $\Omega()$ and $\Theta()$ that occur frequently in analyzing the complexity of algorithms (refer to your notes/textbooks of COMP 1805/2402/2804).

   (b) Let $p(n) = a_d n^d + a_{d-1} n^{d-1} + \cdots a_1 n + a_0$, where $a_d > 0$, be a d-degree polynomial in n. Also $a_0, \cdots, a_d$ are positive constants. Let k be a positive integer. Using your definitions in Part (a) show that:

      i. If $k > d$, then $p(n) \in O(n^k)$.

      ii. If $k < d$, then $p(n) \in \Omega(n^k)$.

      iii. Is $20n^3 + 10n^2 - 100n - 5 \in O(n^4)$?

      iv. Is $20n^3 + 10n^2 - 100n - 5 \in \Omega(n^2)$?

2. Evaluate the following recurrences (You can assume that $T(1) = 1$ for each of them).

   (a) $T(n) = 5T(n/5) + O(n)$

   (b) $T(n) = T(8n/9) + O(n)$

   (c) $T(n) = T(n-2) + O(1)$

   (d) $T(n) = \sqrt{n}T(\sqrt{n}) + n$

3. Suppose you need to choose between the following algorithms which solves the same problem:

   (a) Algorithm A solves the problem by dividing it into 4 subproblems of half of the size, recursively solves each of them, and combines the solution in linear time.

   (b) Algorithm B solves the problem of size n by recursively solving three subproblems of size $n - 6$ and then combining the solutions in constant time.

   (c) Algorithm C solves the problem of size n by dividing it into 9 subproblems of size $n/3$ each, recursively solving each of them, and then combining the solution in $O(n^2)$ time.

   What are the running times of each of these algorithms? Which one will you choose and Why?

4. We want to sort $n > 0$ distinct real numbers in ascending order. Assume that these numbers are given in an array A of size n. We are also given a function double-partition(i,j) which takes as input two indices $1 \leq i < j \leq n$ of A, where $j - i \geq 2$, and returns two elements $x, y \in \{A[i], A[i+1], \ldots, A[j]\}$ that satisfy the following:

   (a) $x < y$

   (b) The number of elements in $\{A[i], A[i+1], \ldots, A[j]\}$ that are smaller than x are at most $\lceil \frac{j-i}{3} \rceil$.

(c) The number of elements in $\{A[i], A[i+1], \ldots, A[j]\}$ that are larger than y are at most $\lceil \frac{j-i}{3} \rceil$.

(d) The number of elements in $\{A[i], A[i+1], \ldots, A[j]\}$ that are larger than x but smaller than y are at most $\lceil \frac{j-i}{3} \rceil$.

(e) It takes $O(j-i)$ time to compute x and y.

Design an algorithm, running in $O(n \log n)$ time, to sort any set of n distinct real numbers using the function double-partition. (Please review Guideline #5 in the context of what is expected when you are asked to design an algorithm.)

5. You are given an array A consisting of n positive integers, where each element is $\leq 1000n$. Devise an algorithm, running in $O(n)$ time, to sort A in ascending order. Justify your answer.

6. Recall (from your COMP 2402 course) that there is a lower bound for sorting. Answer the following questions

   (a) What does it mean to have a lower bound for a problem?

   (b) State clearly what is the lower bound claim for sorting a set of n (real)-numbers.

   (c) Why this claim does not apply to the problem in Question 5?

7. You are given an array A consisting of n real numbers. Describe and analyze an algorithm, running in $O(n)$ time, that rearranges the elements of A so that A forms a binary heap. Once A is transformed into a Binary Heap, show how you can report the elements in A in sorted (ascending) order. How much time it takes to report all the elements of A in the sorted order? Justify your answer.

8. You are given a set of n real numbers which you are asked to insert incrementally in an initially empty binary search tree. Note that the time to insert an element in a binary search tree of size x is $O(\log x)$. What is the total running time of inserting all the n elements in the tree. Justify your answer. Assume that you have formed the binary search tree on n elements, show how you can report the elements in a sorted order in $O(n)$ time.

9. Reflecting on the answers to the previous two questions, is the construction of a binary heap in $O(n)$ time or reporting the elements in sorted order from a binary search tree in $O(n)$ time is in contradiction to the lower bound for sorting (refer to Question 6)? Justify your answer.

10. Discuss a couple of scenarios where you will be using a binary heap instead of a binary search tree. Justify your answer.

11. (Bonus Problem:) Assume that you have a set of three parallel and distinct lines in plane. Assume that Line 1 contains a set R of n red points, Line 2 contains a set B

of n blue points, and Line 3 contains a set G of n green points, where n is a positive integer. You need to design an algorithm that determines if there is a triplet of points (r, b, g) such that they lie on a line segment, where $r \in R$, $b \in B$ and $g \in G$. (Observe that it is straightforward to design an algorithm running in $O(n^3)$ time by considering each choice for points r, b, and g, and in $O(1)$ time testing whether they all lie on a segment.) To get Bonus points, design an algorithm running in $O(n^2)$ time.