Algorithm Analysis, Term I/2018-19

Assignment 2

Due: Fri, Sep 28 @ 11.59pm

Directions:

- Your solutions must be typeset. LaTeX is recommended.
- You must upload your solutions as a PDF file on Canvas before the deadline.
- You don't have to include your solutions to the programming problems in the PDF file.

Problem 1

Give asymtotic upper bounds for $T(n)$ in each of the following recurrences. Assume that $T(n)$ is constant for $n \leqslant 2$. Make your bounds as tight as possible and justify your answers.

(a) $T(n) = 2T(n/3) + 1$
(b) $T(n) = 5T(n/4) + n$
(c) $T(n) = 7T(n/7) + n$
(d) $T(n) = 9T(n/3) + n^2$
(e) $T(n) = 8T(n/2) + n^3$
(f) $T(n) = T(n-1) + 2$
(g) $T(n) = T(n-1) + n^c$, where $c \geqslant 1$ is a constant
(h) $T(n) = T(n-1) + c^n$, where $c > 1$ is a constant
(i) $T(n) = 2T(n-1) + 1$
(j) $T(n) = T(\sqrt{n}) + 1$

Problem 2

Suppose you are choosing between the following three algorithms:

- Algorithm A solves problems by dividing them into three subproblems of one-third the size, recursively solving each subproblem, and then combining the solutions in linear time.
- Algorithm B solves problems of size n by recursively solving a subproblem of size $n - 1$, and then performing additional computation in linear time.
- Algorithm C solves problems of size n by recursively solving two subproblems of one-third the size, and then combining the solutions in $O(n^2)$ time.
- Algorithm D solves problems of size n by recursively solving five subproblems of size $n/4$ and combining the results in linear time.

What are the running times of each of these algorithms (in O-notation), and which would you choose?

Problem 3

How many times does the following program print "hello", as a function of n?

```
function f(n)
    if n > 1:
        print("hello")
        f(n/2)
        f(n/2)
        f(n/2)
```

You may assume that n is a power of two. Justify and express your answer using big-O notation.

Problem 4

Suppose you are given an array A with n entries, with each entry holding a distinct number. You are told that the sequence of values $A[1], A[2], \ldots, A[n]$ is unimodal. For some index p between 1 and n, the values in the array entries increase up to position p in A and then decrease the remainder of the way until position n. (So if you were to draw a plot with the array position j on the x-axis and the value of the entry $A[j]$ on the y-axis, the plotted points would rise until x-value p, where they'd achieve their maximum, and then fall from there on.) You'd like to find the *peak entry* p without having to read the entire array – in fact, by reading as few entries of A as possible. Show how to find the entry p by reading at most $O(\log n)$ entries of A.

Problem 5

An array $A[1 \ldots n]$ is said to have a **majority element** if more than half of its entries are the same. Given an array, the task is to design an efficient algorithm to tell whether the array has a majority element, and, if so, to find that element.

The elements of the array are not necessarily from some ordered domain like the integers, and so there can be no comparisons of the form "is A[i] > A[j]?". (Think of the array elements as GIF files, say.) However you can answer questions of the form: "is A[i] = A[j]?" in constant time.

(a) Show how to solve this problem in $O(n \log n)$ time. (Hint: Split the array A into two arrays A_1 and A_2 of half the size. Does knowing the majority elements of A_1 and A_2 help you figure out the majority element of A? If so, you can use a divide-and-conquer approach.)

(b) Can you give a linear-time algorithm? (Hint: Here's another divide-and-conquer approach:

- Pair up the elements of A arbitrarily, to get $n/2$ pairs
- Look at each pair: if the two elements are different, discard both of them; if they are the same, keep just one of them

Show that after this procedure there are at most $n/2$ elements left, and that they have a majority element if and only if A does.)

Problem 6

In this problem, you will be implementing a divide-and-conquer solution the closest pair problem. You can find the problem statement here: `https://www.urionlinejudge.com.br/judge/en/problems/view/1295`

To submit your solution, you must upload your solution to the following contest: `https://www.urionlinejudge.com.br/judge/en/tournaments/register/1063`. The passphase is `muic`.