ESO207A: Data Structures and Algorithms

Homework *1* **Due: June 7, 2017 in class**

Instructions.

1. Start each problem on a new sheet. Write your name, Roll No., the course number, the problem number, the date and the names of any students with whom you collaborated.
2. For questions in which algorithms are asked for, your answer should take the form of a short write-up. The first paragraph should summarize the problem you are solving and what your results are (time/space complexity as appropriate). The body of the essay should provide the following:
   (a) A description of the algorithm in English and/or pseudo-code, where, helpful.
   (b) At least one worked example or diagram to show more precisely how your algorithm works.
   (c) A proof/argument of the correctness of the algorithm.
   (d) An analysis of the running time of the algorithm.

   Remember, your goal is to communicate. *Full marks will be given only to correct solutions which are described clearly.* Convoluted and unclear descriptions will receive *low marks.*

Problem 1. Recurrence Relations.

1.1 Solve the following recurrence relations by unfolding the recursion tree. Assume $T(1) = 1$ as the base case for all the recurrences below.

a. $T(n) = 6T(n/4) + n^2$.

b. $T(n) = 6T(n/4) + n$.

c. $T(n) = 2T(n/4) + \sqrt{n}$.

1.2 Solve using the substitution method.

a. $T(n) = T(1) + T(2) + \ldots + T(n-1) + c$.

Problem 2. Unimodal Search (Divide-and-Conquer). An array $A[1 \ldots n]$ of numbers is unimodal if there exists some index $k \in \{1, 2, \ldots, n\}$ such that $A[1] \leq A[2] \leq \ldots \leq A[k]$ and $A[k] \geq A[k+1] \geq \ldots A[n]$. The element $A[k]$ is the maximum element of the array and it is the unique "locally maximum" element surrounded by elements $A[k-1]$ and $A[k+1]$, that are both not larger than itself. An example unimodal array: $A = \boxed{1\ 4\ 5\ 11\ 8\ 7\ 1}$. Here, $A[4] = 11$ is the maximum element.

Give an algorithm that given a unimodal array $A[1 \ldots n]$, finds the maximum element in $O(\log(n))$ time. (a) Prove the correctness of your algorithm, and (b) prove the bound on its running time.

Problem 3. Monge Arrays Problem 4-6 from CLRS. An $m \times n$ array A of real numbers is a *Monge array* if for all i, j, k and l such that $1 \leq i < k \leq m$ and $1 \leq j < l \leq n$, we have,

$$A[i, j] + A[k, l] \leq A[i, l] + A[k, j] \ .$$

In other words, whenever we pick two rows and two columns of a Monge array and consider the four elements at the intersections of the rows and columns, the sum of the upper-left and lower-right elements is less than or equal to the sum of the lower-left and upper-right elements.

a. Prove that an array is Monge if and only if for all $i = 1, 2, \ldots, m-1$ and $j = 1, 2, \ldots, n-1$, we have,

$$A[i, j] + A[i+1, j+1] \leq A[i, j+1] + A[i+1, j]$$

(*Hint*: For the "if" part, use induction separately on rows and columns.)

b. Let $f(i)$ be the index of the column containing the leftmost minimum element of row i. Prove that for any $m \times n$ Monge array, $f(1) \leq f(2) \leq \cdots \leq f(m)$.

c. The following describes a divide-and-conquer algorithm that computes the leftmost miminum element in each row of an $m \times n$ Monge array A:

> Construct a submatrix A' of A consisting of the even-numbered rows of A. Recursively determine the leftmost minimum for each row of A'. Then compute the leftmost minimum in the odd numbered rows of A.

Explain how to compute the leftmost minimum in the odd-numbered rows of A (given that the leftmost minimum of the even-umbered rows is known) in $O(m + n)$ time.

d. Write the recurrence describing the running time of the algorithm described in part (d). Show that its solution is $O(m + n \log m)$.

Problem 4. FFT Given a vector $a = (a_0, a_1, \ldots, a_{n-1})$, a certain transform with respect to a function $\phi : \{0, 1, \ldots, n-1\} \to \mathbb{N}$ and a complex number z is defined as $y = (y_0, y_1, \ldots, y_{n-1})$, where,

$$y_k = \sum_{j=0}^{n-1} (a_j z^{\phi(j)})(z^{\phi(k-j)})$$

Show how to evaluate this transform in $O(n \log n)$ time for any given complex number z by viewing it as a convolution. Assume that $z^{\phi(j)}$ can be computed in $O(\log n)$ time, for any $0 \leq j \leq n-1$.