CS3000 Algorithms

Recitations 02

Akshar Varma *Date: 13 July 2023*

Problem 1 (Sorting special arrays)

Consider the problem of sorting an array $A[1, \ldots, n]$ of integers. We presented an $O(n \log n)$-time algorithm in class and, also, proved a lower bound of $\Omega(n \log n)$ for any comparison-based algorithm.

1. Give an efficient sorting algorithm for a ***boolean***[1] array $B[1, \ldots, n]$.
2. Give an efficient sorting algorithm for an array $C[1, \ldots, n]$ whose elements are taken from the set $\{1, 2, 3, 4, 5\}$.
3. Give an efficient sorting algorithm for an array $D[1, \ldots, n]$ whose elements are distinct ($D[i] \neq D[j]$, for every $i \neq j \in \{1, \ldots, n\}$) and are taken from the set $\{1, 2, \ldots, 2n\}$.
4. In case you designed linear-time sorting algorithms for the previous subparts, does it mean that the lower bound for sorting of $\Omega(n \log n)$ is wrong? Explain.

Problem 2 (Local maximum)

Given an array $A = [a_1, a_2, \ldots, a_n]$ of distinct positive integers which is not necessarily sorted, find any local maximum in the array A. An element at index $1 < i < n$ is a local maximum if a_i is at least as big as elements on both side of it. That is $a_i \geq a_{i-1}$ and $a_i \geq a_{i+1}$. For $i = 1$ or $i = n$, we only compare $a_1 \geq a_2$ and $a_n \geq a_{n-1}$ respectively.

a) Give a $\Theta(n)$ time algorithm.

b) Give an $O(\log n)$ algorithm using divide-and-conquer.

Problem 3 (Counting Inversions)

An **inversion** in an array $A[1 \ldots n]$ is a pair of indices (i, j) such that $i < j$ and $A[i] > A[j]$. Describe and analyze an algorithm to count the number of inversions in an n-length array in $O(n \log n)$ time. [***Hint***: *Remember mergesort.*]

Problem 4 (Ternary Tree Track Totals)

A ternary tree is a rooted tree where each node (except the leaves) have three children each. We are given a ternary tree T with a positive integer label on each node of the tree. Further, you are given that the tree has k levels such that at level $i \in \{1, \ldots, k\}$, there are 3^{i-1} nodes since every node at level $i - 1$ has 3 children each.

You want to find the maximum path sum starting at the root of the tree and following any path on the tree from root to a leaf. From every node in your path, except the terminal leaf node, you have three options for which child to use for your path.

[1] In a boolean array $B[1, \ldots, n]$, each element $B[i]$ (for $i = 1, \ldots, n$) is either 0 or 1.

Problem 5 (Tiling checkerboards)

Suppose you are given a $2^n \times 2^n$ checkerboard with one (arbitrarily chosen) square removed. Describe and analyze an algorithm to compute a tiling of the board by without gaps or overlaps by L-shaped tiles, each composed of 3 squares. Your input is the integer n and two n-bit integers representing the row and column of the missing square. The output is a list of the positions and orientations of $(4^n - 1)/3$ tiles. Your algorithm should run in $O(4^n)$ time.

[***Hint***: *First prove that such a tiling always exists.*]