CSE 102: Spring 2021

Advanced Homework # 3

Divide and Conquer (up to 15 points)

Attempt **ONE or TWO** of the following problems. All questions do NOT carry equal weights. In all the problems below, design a Divide-and-Conquer Algorithm, establish computational complexity, and illustrate with a few examples to demonstrate how the algorithm works on some specific inputs (chosen to illustrate the working of the algorithm clearly).

1. (5 points) **Monotonically Decreasing Function** defined on natural numbers taking integer values (described below). Describe the algorithm and establish the computational complexity.

2. (5 points) **Finding the largest element in a circularly shifted array:** Suppose you are given an array $A[1,..,n]$ of sorted integers that has been circularly shifted at k positions to the right. For example, $[35, 42, 5, 15, 27, 29]$ is a sorted array that has been circularly shifted $kl = 2$ positions, while $[27, 29, 35, 42, 5, 15]$ has been shifted $k = 4$ positions. We can obviously find the largest element in $O(n)$ time. Describe an $O(\log n)$ algorithm.

3. (7 points) Design an algorithm with complexity $O(n^{\log_2(3)})$ to **search for an element in a 2D matrix** where each row and column is sorted in an increasing order from left to right and from top to bottom respectively.

4. (7 points) Design an algorithm to **merge k sorted arrays** of size n each with time complexity of $O(n * k * \log k)$

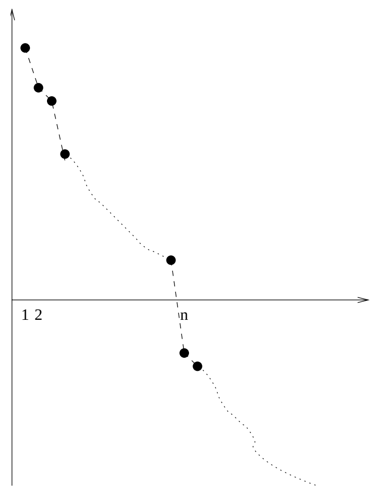
5. (10 points) Design an $O(n)$ algorithm to find the k**th smallest element** in an array.

6. (10 points) Design an algorithm to compute **median of two sorted arrays of different sizes** n and m with time complexity of $O(\log n + \log m)$.

7. (15 points) Problem 7.41 on **adders!**
8. (15 points) Problem 7.42 on **switches!**

Divide and conquer

1. In this problem we consider a *monotonously decreasing* function f : $N \rightarrow Z$ (that is, a function defined on the natural numbers taking integer values, such that $f(i) > f(i+1)$). Assuming we can evaluate f at any i in constant time, we want to find $n = \min\{i \in N | f(i) \leq 0\}$ (that is, we want to find the value where f becomes negative).

   We can obviously solve the problem in $O(n)$ time by evaluating $f(1), f(2), f(3), \ldots f(n)$. Describe an $O(\log n)$ algorithm. (*Hint:* Evaluate f on $O(\log n)$ carefully chosen values $\leq n$ and possibly at a couple of values between n and $2n$ - but remember that you do not know n initially).

Problem 7.41. An *n-tally* is a circuit that takes n bits as input and produces $1 + \lfloor \lg n \rfloor$ bits as output. It counts (in binary) the number of bits equal to 1 among the inputs. For example, if $n = 9$ and the inputs are 011001011, the output is 0101. An *(i, j)-adder* is a circuit that has one i-bit input, one j-bit input, and one $[1 + \max(i, j)]$-bit output. It adds its two inputs in binary. For example, if $i = 3$, $j = 5$, and the inputs are 101 and 10111 respectively, the output is 011100. It is always possible to construct an (i, j)-adder using exactly $\max(i, j)$ 3-tallies. For this reason the 3-tally is often called a *full adder*.

(a) Using full adders and (i, j)-adders as primitive elements, show how to build an efficient n-tally.

(b) Give the recurrence, including the initial conditions, for the number of 3-tallies needed to build your n-tally. Do not forget to count the 3-tallies that are part of any (i, j)-adders you might have used.

(c) Using the Θ notation, give the simplest possible expression for the number of 3-tallies needed in the construction of your n-tally. Justify your answer.

Problem 7.42. A *switch* is a circuit with two inputs, a control, and two outputs. It connects input A with output A and input B with output B, or input A with output B and input B with output A, depending on the position of the control; see Figure 7.8. Use these switches to construct a network with n inputs and n outputs able to implement any of the $n!$ possible permutations of the inputs. The number of switches used must be in $O(n \log n)$.

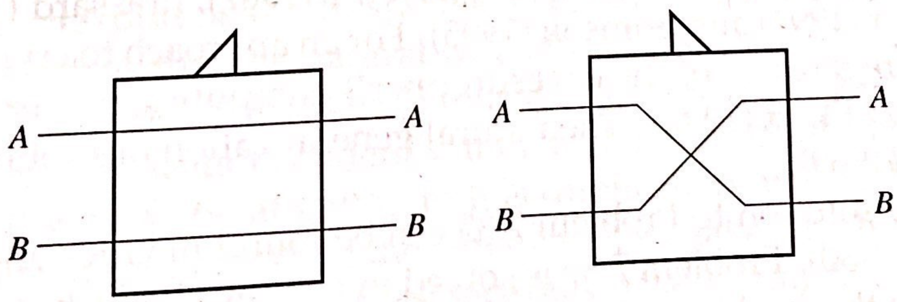


Figure 7.8. Switches