Problem Set 3

Data Structures and Algorithms, Fall 2022

Due: September 29, in class.

Problem 1

Give asymptotic upper and lower bounds for $T(n)$ in each of the following recurrences. You may assume that $T(n)$ is constant for sufficiently small n. You should make your bounds as tight as possible. You do *not* need to prove your answer.

(a) $T(n) = 4T(n/3) + n \lg n$.

(b) $T(n) = 3T(n/3) + n/(\lg n)$.

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

(d) $T(n) = 3T(n/3 - 2) + n/2$.

(e) $T(n) = T(n/2) + T(n/4) + T(n/8) + n$.

(f) $T(n) = T(n-1) + \lg n$.

(g) $T(n) = \sqrt{n}T(\sqrt{n}) + n$.

Problem 2

(a) Give an asymptotic *tight* bound for the recurrence $T(n) = T(d) + T(n-d) + cn$, where $d \in \mathbb{N}^+$ is a constant and $c > 0$ is also a constant. You *need* to prove your answer.

(b) Give an asymptotic *tight* bound for the recurrence $T(n) = T(dn) + T((1-d)n) + cn$, where $d \in (0, 1)$ is a constant and $c > 0$ is also a constant. You *need* to prove your answer.

Problem 3

An array $A[0, \cdots, n-1]$ of n *distinct* numbers is *bitonic* if there are *unique* indices i and j such that $A[(i-1) \bmod n] < A[i] > A[(i+1) \bmod n]$ and $A[(j-1) \bmod n] > A[j] < A[(j+1) \bmod n]$. In other words, a bitonic sequence either consists of an increasing sequence followed by a decreasing sequence, or can be circularly shifted to become so. For example, the array $\langle 4, 6, 9, 8, 7, 5, 1, 2, 3\rangle$ is bitonic, but the array $\langle 3, 6, 9, 8, 7, 5, 1, 2, 4\rangle$ is not bitonic. Devise an algorithm to find the smallest element in an n-element bitonic array in $O(\lg n)$ time.

Problem 4

An array $A[1 \cdots n]$ 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 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, and so there can be no comparisons of the form "is $A[i] > A[j]$?". (Think of the array elements as JPEG images, say.) However, you can answer questions of the form "is $A[i] = A[j]$?" in constant time.

(a) Devise an algorithm to solve this problem in $O(n \lg n)$ time. *(Hint: use divide and conquer.)*

(b) [Bonus Question] Devise an algorithm to solve this problem in $O(n)$ time.

Problem 5

(a) Prove that in a binary heap containing n nodes, there are at most $\lceil n/2^{h+1} \rceil$ nodes of height h.

(b) Implement the HEAPUPDATE(i, val) operation in a binary max-heap containing n nodes, which changes the value of the element at index i to val. Your implementation should have $O(\lg n)$ runtime.

Problem 6

Suppose you are given k sorted linked lists, and the total number of elements in these lists is n. Devise an $O(n \lg k)$-time algorithm to merge these k sorted lists into one sorted list.

Bonus Problem

Prove that when all elements are distinct, the best-case running time of HEAPSORT is $\Omega(n \lg n)$. (Hence, when all elements are distinct, the running time of HEAPSORT is $\Theta(n \lg n)$.)