CSCI261 Analysis of Algorithms, Fall 2020/21, Homework 2

Due Friday, September 18, 2020, 11:59pm

Problem 1

Given is a sequence of n numbers, where each of these numbers is a non-negative integer smaller than n^2. Design an $O(n)$ algorithm that sorts the input sequence from the smallest to the largest number.

Problem 2

Consider the divide-and-conquer algorithm shown in the display below; the algorithm accesses a global array A of integers.

```
WHATDOIDO(integer left, integer right):
    if left==right:
        if A[left]<0 return (0, 0, 0, A[left])
        else return (A[left], A[left], A[left], A[left])
    if left<right:
        m = (left+right)/2 (rounded down)
        (lmaxsum, llmaxsum, lrmaxsum, lsum) = WHATDOIDO(left, m)
        (rmaxsum, rlmaxsum, rrmaxsum, rsum) = WHATDOIDO(m+1, right)
        maxsum = max{lmaxsum, rmaxsum, lrmaxsum+rlmaxsum}
        leftalignedmaxsum = max{llmaxsum, lsum+rlmaxsum}
        rightalignedmaxsum = max{rrmaxsum, lrmaxsum+rsum}
        sum = lsum+rsum
        return (maxsum, leftalignedmaxsum, rightalignedmaxsum, sum)
```

Before running the algorithm, we ask the user to enter n integers that we store in the array A. Then we run `WHATDOIDO(1,n)` and out of the four returned values, we output the first.

a) State the recurrence for $T(n)$ that captures the running time of the algorithm as closely as possible.

b) Use the "unrolling the recurrence" or the mathematical induction to find a tight bound on $T(n)$.

c) What does the algorithm do?

- For an input in A and integers `left` and `right`, succinctly describe the meaning of the return variables `maxsum`, `leftalignedmaxsum`, `rightalignedmaxsum`, and `sum`.
- Succinctly decribe the meaning of the value (the first of the four returned values) that we output after running `WHATDOIDO(1,n)`.

Problem 3

In the problem of counting inversions, you are given a permutation $a_1, a_2, \ldots, a_n$ of numbers $1, 2, \ldots, n$ and the goal is to count the number of pairs i, j, where $i < j$ and $a_i > a_j$. In this homework problem, you are given a sequence of n numbers $b_1, b_2, \ldots, b_n$ and your task is to compute the "weighted count" of inversions defined as follows: An inversion is a pair of indices i, j where $i < j$ and $b_i > b_j$. An i, j inversion has weight $b_i b_j$ and the weighted count for the input sequence is the sum of the weights of all its inversions.

For example, for $n = 5$ and input sequence $7, 3, 8, 1, 5$, we have the following inversions weights: $7 \times 3 = 21$, $7 \times 1 = 7$, $7 \times 5 = 35$, $3 \times 1 = 3$, $8 \times 1 = 8$, and $8 \times 5 = 40$. The overall weighted count is: $21 + 7 + 35 + 3 + 8 + 40 = 114$.

Design an $O(n \log n)$ algorithm which computes the weighted count of inversions for a given input sequence.

Problem 4

For each of the following recurrences, use the Master theorem to express $T(n)$ as a Theta of a simple function. State what the corresponding values of a, b, and $f(n)$ are and how you determined which case of the theorem applies. Do not worry about the base case or rounding.

1. $T(n) = 3T(n/2) + n^2$
2. $T(n) = 5T(n/2) + n \log n$
3. $T(n) = 2T(n/4) + \sqrt{n}$