Foundations of Algorithms, Spring 2023: Homework 2

Due: Wednesday, February 15, 11:59pm

Problem 1 (10 points)

Consider the following divide-and-conquer algorithm that assumes a global array A of integers. In the following `%` represents the modulo operation.

```
WHATDOIDO(integer left, integer right):
    if left == right:
        return (A[left] % 2, A[left] % 2, A[left] % 2)
    else:
        m = (left + right) / 2       (integer divide)
        (llstreak, lrstreak, lmaxstreak) = WHATDOIDO(left, m)
        (rlstreak, rrstreak, rmaxstreak) = WHATDOIDO(m+1, right)

        maxstreak = max(lmaxstreak, rmaxstreak, lrstreak+rlstreak)
        if lmaxstreak == m - left + 1:
            lstreak = lmaxstreak + rlstreak
        else:
            lstreak = llstreak
        if rmaxstreak == right - m:
            rstreak = rmaxstreak + lrstreak
        else:
            rstreak = rrstreak

        return (lstreak, rstreak, maxstreak)
```

Before running the algorithm, we ask the user to enter n integers that we store in the array A. Then we run `WHATDOIDO(0,n-1)`.

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

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

c) What does the algorithm do? Specify what the three returned values represent.

Problem 2 (8 points)

Use the Master Theorem to provide tight asymptotic bounds for the following recurrences. In each case, also specify the values of a, b, and $f(n)$, and indicate which case of the Master Theorem applies.

(a) $T(n) = 7T(n/8) + n$

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

(c) $T(n) = 9T(n/3) + n^2$

(d) $T(n) = 4T(n/2) + n \log^2 n$.

Problem 3 (15 points: 11 for implementation / 4 for writeup)

There is a pecking order among the kids as they wait for the school bus. They wait in a line ordered by age, with the oldest child first in line. Some of the kids are carrying their school lunch. The others have an account and buy lunch at school, and are not carrying anything with them. Among the kids, there is a universally accepted value assigned to the different lunches being brought to school.

While they wait for the bus to come, there is some bullying that occurs. If an older child has a lunch of lower value than does the younger child standing next in line, the older child will force the younger child to swap lunches.

Bullying can only occur from an older child to the younger child next in line, and can only occur if both children are carrying a lunch. This bullying continues, in no specific order, until no more bullying is possible.

Design an $O(n \log n)$ algorithm that computes the number of incidents of bullying that occur.

Problem 4 (20 points: 14 for implementation / 6 for writeup)

You have a special set of n dominoes. Each domino has two integer numbers of dots on it. Each integer number of dots is in the range $[0, n)$. There is no ordering to a domino. That is, either number of dots on a domino can be considered the 'first' or the 'second' number.

Design an $O(nT)$ algorithm that determines whether there exist two dominoes in this set, call them a and b, along with a way to order each domino, call it $a = \{a_1, a_2\}, b = \{b_1, b_2\}$, such that $|a_1 - b_1| + |a_2 - b_2| \leq T$, for some given threshold, T.

Problem 5 (15 points: 11 for implementation / 4 for writeup)

As chef at the soup kitchen, you do everything you can to make sure no food goes to waste. Perishable food items are donated at arbitrary times. Each comes with a shelf life before it goes bad. Each day you prepare a meal using one donated perishable food item. Design an $O(n \log n)$ algorithm to determine if it will be possible to use all the donated food without any going to waste.