CSE 551 Assignment 2

February 15, 2021

Submission Instructions: Deadline is **11:59pm on 02/21**. Late sub-missions will be penalized, therefore please ensure that you submit (file upload is completed) before the deadline. Additionally, you can download the submitted file to verify if the file was uploaded correctly. Submit your answers electronically, in a single PDF, via *Canvas*. **Please type up the answers and keep in mind that we'll be checking for plagiarism**.
Furthermore, please note that the graders will grade 3 out of the 6 questions randomly. Therefore, if the grader decides to check questions 1, 2 and 4, and you haven't answered question 4, you'll lose points for question 4. Hence, please answer all the questions.

1. The Fibonacci series can be computed as follows,

$$F(n) = F(n-1) + F(n-2) \tag{1}$$

In class, we showed how this can be done in $\mathcal{O}(log\ n)$ computation time. Now suppose that the definition is changed in the following way,

$$F'(n) = F'(n-1) + F'(n-2) + F'(n-3) + F'(n-4) \tag{2}$$

Can $F'(n)$ be computed in $\mathcal{O}(log\ n)$? If yes, please show how it can be done. If no, show a counterexample where this fails. Please provide your rationale for both. Assume that $F'(0) = 0, F'(1) = 1, F'(2) = 1, F'(3) = 1$. **[25 Points]**.

2. Consider the following problem. You're given an array A consisting of n integers $A[1], A[2], ..., A[n]$. You'd like to output a two-dimensional $n \times n$ array B in which $B[i, j](for\ i < j)$ contains the sum of array entries $A[i]$ through $A[j]$, i.e., the sum $A[i] + A[i+1] + ... + A[j]$. (The value of array entry $B[i, j]$ is left unspecified whenever $i \geq j$, so it doesn't matter what is output for these values.)

   A simple algorithm to solve this problem illustrated in Algorithm 1:

```
Data: Input A and n
Result: Return B
for i = 1, 2, ..., n do
    for j = i + 1, i + 2, ..., n - 1 do
        Add up array entries A[i] through A[j]
        Store the result in B[i, j]
    end
end
Return B;
```

Algorithm 1: Algorithm for Q2

   - For some function f that you should choose, give a bound of the form $\mathcal{O}(f(n))$ on the running time of this algorithm on an input of size n (i.e., a bound on the number of operations performed by the algorithm). **[8 Points]**.

   - For this same function f, show that the running time of the algorithm on an input of size n is also $\Omega(f(n))$. (This shows an asymptotically tight bound of $\Theta(f(n))$ on the running time.) **[8 Points]**.

   - Although the algorithm you analyzed in parts (a) and (b) is most natural way to solve problem-after all, it just iterates through the relevant entries of the array B, filling in a value for each - it contains some highly unnecessary sources of inefficiency. Give a different algorithm to solve this problem, with an asymptotically better running time. In other words, you should design an algorithm with running time $\mathcal{O}(g(n))$, where $lim_{n\to\infty} g(n)/f(n) = 0$. **[9 Points]**.

3. Design an algorithm to compute the 2nd smallest number in an unordered (unsorted) sequence of numbers $\{a_1, a_2, ..., a_n\}$ in $n + \lceil log_2(n) \rceil - 2$ comparisons in the worst case. If you think such an algorithm can be designed, then show how it can be done. If your answer is no, then explain why it cannot be done. **[25 points]**

4. Let $n = 2p$, $V = (v_1, ..., v_n), W = (w_1, ..., w_n)$. Then we can compute the vector product VW by the formula:
   $\sum_{1 \le i \le p} (v_{2i-1} + w_{2i}) \times (v_{2i} + w_{2i-1}) - \sum_{1 \le i \le p} v_{2i-1} \times v_{2i} - \sum_{1 \le i \le p} w_{2i-1} \times w_{2i}$

   which requires $3n/2$ multiplications. Show how to use this formula for the multiplication of two $n \times n$ matrices giving a method which requires $n^3/2 + n^2$ multiplications rather than the usual n^3 multiplications. **[25 points]**

5. Find the solution to the following recurrence relation:

   $T(n) = 16T(n/4) + n^2$,

   for n a power of 4 (or, $n = 4^k$) and $n > 1$. **Show all your work. [25 points]**

6. Find the solution to the following recurrence relation:

   $T(1) = 1$
   $T(n) = 3^n T(n/2)$,

   for n a power of 2 (or, $n = 2^k$) and $n > 1$. **Show all your work. [25 points]**