Assignment 2 (due Monday, February 10, 6:00pm)

Instructions:

- Hand in your assignment using Crowdmark. Detailed instructions are on the course website.
- Give complete legible solutions to all questions.
- Your answers will be marked for clarity as well as correctness.
- For any algorithm you present, you should justify its correctness (if it is not obvious) and analyze the complexity.

1. [6 marks] Recurrence relations.
   Consider the following recurrence:
   
   \[ T(n) = \begin{cases} 
   T(\lfloor n/3 \rfloor) + T(\lfloor n/5 \rfloor) + 4 & \text{if } n \geq 2 \\
   1 & \text{if } n = 0, 1 
   \end{cases} \]

   Prove that \( T(n) \in \Omega(n^{0.4}). \)

2. [10 marks] Recursion tree method.
   Consider the following algorithm that gets as an input an array \( A \) of \( n \) integers, and two indices \( 1 \leq p, r \leq n \):

   \[
   \text{Alg}(A, n, p, r)
   \]
   (a) If \( (r - p + 1 \leq n^{1/3}) \)
       i. \( \text{Proc1}(A, n, p, r) \)
   (b) Else
       i. \( d = \log n, t = \frac{r-p+1}{d} \)
       ii. For \( i = 1 \) to \( d \)
           \[ \bullet \text{Alg}(A, n, p + (i - 1)t, p + it - 1) \]
       iii. \( \text{Proc2}(A, n, p) \) \hspace{1cm} \text{(Note: this line is outside of the For loop)}

   \( \text{Alg} \) uses the following two procedures:

   (a) \( \text{Proc1}(A, n, p, r) \) runs in time \( c \cdot s \cdot \log(s) \) for any input array, where \( c \) is a constant and \( s = r - p + 1 \) is the size of the subarray \( A[p, \ldots, r] \).
   (b) \( \text{Proc2}(A, n, p, r) \) runs in time \( c' \cdot s \cdot \log(n) \) for any input array, where \( c' \) is a constant and \( s = r - p + 1 \) is the size of the subarray \( A[p, \ldots, r] \).

   Let \( T(n) \) denote the run time of \( \text{Alg}(A, n, 1, n) \). What is \( g(n) \) such that \( T(n) = \Theta(g(n)) \)?
Guidance: In your answer, use the recursion tree method. In particular, work according to the following steps:

(a) [2 marks] What is the height of the tree?

(b) [3 marks] What is the contribution of each level of the tree, excluding the level of the leaves?

(c) [3 marks] What is the contribution of the level of the leaves?

(d) [2 marks] What is the total runtime of the algorithm?

You can assume that \( \log n \) is an integer, \( r - p + 1 / \log n \) is a positive integer, and the recurrence always stops when \( r - p + 1 = n^{1/3} \). It is beneficial to first express the run time as a function of \( n \) and \( d \), and only after substitute \( d = \log n \) (recall that if \( a^b = c \) we can take \( \log \) from both sides and get a solution for \( b \)).

Find an asymptotic \( \Theta \)-bound for the solution to the following recurrence relation by applying the Master Theorem. Show your work.

\[
T(n) = \begin{cases} 
5T(n/4) + 9 \log_7 n & \text{if } n > 1 \\
1 & \text{if } n = 1.
\end{cases}
\]

Define the following sequence of numbers: \( F_0 = 0, F_1 = 1, \) and

\[
F_{2n} = (F_n + F_{n-1})^2 - F_{n-1}^2 \\
F_{2n+1} = (F_n + F_{n-1})^2 + F_n^2
\]

(This is in fact the Fibonacci number sequence, but you are not required to prove it.)

(a) [2 marks] Give a pseudocode description of an efficient divide-and-conquer algorithm to compute \( F_n \) for a given integer \( n \geq 0 \), based on the above definition.

(b) [4 marks] Prove that \( F_n \leq 2^n \) by induction, using the usual recurrence for \( F_n \), namely, \( F_n = F_{n-1} + F_{n-2} \).

(c) [8 marks] Determine a \( O \)-bound on the complexity of your algorithm from part (a) by writing down a recurrence and solving it using the Master Theorem. Here, we are interested in the bit complexity. Assume that the multiplication of two \( k \)-bit numbers requires \( O(k^{1.59}) \) time by Karatsuba's algorithm. You can use the fact that \( F_n \leq 2^n \) (which you proved in part (b)), which implies that the number of bits in \( F_n \) is at most \( n \).
5. [14 marks] Divide-and-conquer
A matrix $M$ with $r$ rows and $c$ columns containing distinct integers is said to be sorted if each row and column is sorted. Namely, for every $1 \leq i < r$ and $1 \leq j < c$ it holds that

$$M(i, j) < M(i, j + 1) \quad \text{and} \quad M(i, j) < M(i + 1, j).$$

For example, the following matrix $M$ is a sorted matrix:

$$M = \begin{pmatrix}
5 & 8 & 11 & 23 \\
6 & 9 & 14 & 25 \\
10 & 15 & 18 & 31 \\
30 & 32 & 40 & 50
\end{pmatrix}.$$

(a) [2 marks] Given a sorted matrix, which cell contains the minimum element? Prove it. Namely, if you claim that the minimum element is in cell $M(a, b)$ you need to show that for any $(a', b') \neq (a, b)$ it holds that $M(a, b) < M(a', b').$

(b) [4 marks] Given a sorted matrix $M$ with $n$ rows and $n$ columns and an integer $z$, we want to check if $z$ is in $M$. A student suggested the following algorithm:

i. Compare $z$ with an element $w$ in the middle of the matrix. Namely, $w = M(x_{\text{mid}}, y_{\text{mid}})$ where $x_{\text{mid}} = \left\lceil \frac{1+n}{2} \right\rceil$ and $y_{\text{mid}} = \left\lceil \frac{1+n}{2} \right\rceil$.

ii. Divide $M$ to four sub-matrices $A, B, C, D$ of size (roughly) $n/2 \times n/2$:

$$A = M(1, \ldots, x_{\text{mid}})(1, \ldots, y_{\text{mid}}),$$

$$B = M(1, \ldots, x_{\text{mid}})(y_{\text{mid}} + 1, \ldots, n),$$

$$C = M(x_{\text{mid}} + 1, \ldots, n)(1, \ldots, y_{\text{mid}}),$$

$$D = M(x_{\text{mid}} + 1, \ldots, n)(y_{\text{mid}} + 1, \ldots, n).$$

iii. If $z = w$, return Found.

iv. If $z < w$, we look for $z$ recursively in the following sub-matrices (fill in the blank):

v. If $z > w$, we look for $z$ recursively in the following sub-matrices (fill in the blank):

(c) [6 marks] Write a pseudo-code implementing the above algorithm and analyze its running time. Namely, let $T(n)$ denote the worst-case running time of the algorithm on a matrix of size $n \times n$. Write the recurrence relation and solve it.

(d) [2 marks] Write an algorithm checks if $z$ is an element of a sorted matrix $M$. Your algorithm should have runtime which is better than $O(n^{1.001})$.

In the HIRING problem, the input is a positive integer $t$ and a sequence of $n \geq 1$ pairs of positive integers $(p_1, l_1), (p_2, l_2), \ldots, (p_n, l_n)$ that correspond to the projected profit $p_i$ you get
if you hire candidate $i$ and the projected loss $l_i$ if you do not hire candidate $i$. You can only hire $t$ candidates; a valid solution to the problem is a subset $S \subseteq \{1, 2, \ldots, n\}$ of $|S| = t$ candidates that maximizes the total profit (accounting for losses)

$$\text{profit}(S) = \sum_{i \in S} p_i - \sum_{j \notin S} l_j$$

earned by hiring the set $S$ of candidates.

Design a greedy algorithm to solve this optimization problem. Prove that it always returns an optimal solution. Justify correctness and analyze running time.