Assignment 2 (due Feb 3rd Friday noon, submit pdf on LEARN)

Please first read the instructions at

https://www.student.cs.uwaterloo.ca/~cs341/#assignments.

1. Comparing Rankings (15 marks)
Suppose two people rank a list of \( n \) items (say movies), denoted \( M_1, \ldots, M_n \). A conflict is a pair of movies \( \{M_i, M_j\} \) such that \( M_i > M_j \) in one ranking and \( M_j > M_i \) in the other ranking. The number of conflicts between two rankings is a measure of how different they are.

For example, consider the following two rankings:

\[
M_1 > M_2 > M_3 > M_4 \quad \text{and} \quad M_2 > M_4 > M_1 > M_3.
\]

The number of conflicts is three; \( \{M_1, M_2\}, \{M_1, M_4\}, \text{and} \{M_3, M_4\} \) are the conflicting pairs.

The purpose of this question is to find an efficient divide-and-conquer algorithm to compute the number of conflicts between two rankings of \( n \) items. You can assume \( n \) is a power of two, for simplicity.

**Hint:** Think of a MergeSort-like algorithm that solves two subproblems of size \( n/2 \). After solving the subproblems, you will also need to compute the number of conflicts between the two sublists during the “merge” step.

To simplify the notation in the algorithm, you can assume that the first ranking is \( M_1 > M_2 > \cdots > M_n \).

(a) (10 marks) Give a pseudocode description of your algorithm, briefly justify its correctness and analyze the complexity using a recurrence relation.

(b) (5 marks) Illustrate the execution of your algorithm when the second ranking is

\[
M_3 > M_1 > M_4 > M_6 > M_5 > M_2 > M_8 > M_7.
\]

2. Tiling a Square (15 marks)
An \( L \)-tile consists of three square \( 1 \times 1 \) cells that form a letter \( L \). There are four possible orientations of an \( L \)-tile:

\[
\begin{array}{ccc}
XX & XX & X & X \\
X & X & XX & XX
\end{array}
\]

The purpose of this question is to find a divide-and-conquer algorithm to tile an \( n \times n \) square grid with \( (n^2 - 1)/3 \) \( L \)-tiles in such a way that only one corner cell is not covered by an \( L \)-tile. This can be done whenever \( n \geq 2 \) is a power of two.

**Hint:** The basic idea is to split the \( n \times n \) grid into four \( n/2 \times n/2 \) subgrids. This defines four subproblems that can be solved recursively. Then you have to combine the solutions to the four subproblems to solve the original problem instance.
(a) (10 marks) Give a pseudocode description of a divide-and-conquer algorithm to solve
this problem, briefly justify its correctness and analyze the complexity using a recurrence
relation. Remember that we are assuming \( n \) is a power of two.

(b) (5 marks) Suppose we specify any single cell in the \( n \times n \) grid (this is not necessarily a
corner cell). Modify your first algorithm so the remaining \( n^2 - 1 \) cells are exactly covered
by \( (n^2 - 1)/3 \) L-tiles. You just need to describe the modifications.

3. Distributed Computers (10 marks)
Consider a distributed system with \( n \) computers \( C_1, \ldots, C_n \), each performing the same com-
putation. However, the computers are built on unreliable hardware and may produce different
results. Your goal is to determine whether strictly more than \( n/2 \) computers produced the
same result. Your only allowed operation is to query a pair of computers, \( C_i \) and \( C_j \), which
returns one of two possible information:

- \( C_i \) and \( C_j \) produced the same result; or
- \( C_i \) and \( C_j \) produced different results.

Design and analyze a divide-and-conquer algorithm that uses \( O(n \log n) \) queries to determine
whether strictly more than \( n/2 \) computers produced the same result.

Describe your algorithm. Give a brief explanation that your algorithm is correct. Finally,
analyze the running time of your algorithm, by deriving a recurrence and solving it via the
Master method.

4. Two-stage Test Scheduling (15 marks)
A lab performs various types of blood tests. For a given sample, a chemical analysis is done
using a piece of expensive equipment; this is Stage 1 of the testing. Then a mathematical
analysis is done by a lab technician on their personal laptop; this is Stage 2. There are \( n \)
tests to be done. The stage 1 analyses must be done sequentially, but any number of stage
two analyses can be done in parallel (e.g., using different laptop computers). For each test \( T_i \)
(\( 1 \leq i \leq n \)), the time required for the stage one and stage two analyses is specified; these are
denoted by \( c_i \) and \( m_i \), respectively. These values are assumed to be positive integers.

A schedule is specified by an ordering of the \( n \) tests. The objective is to minimize the maximum
completion time. For example, suppose there are two tests to be performed, where we have
\( T_1 = (10, 4) \) and \( T_2 = (2, 1) \). If the schedule is \((T_1, T_2)\), then the completion time of \( T_1 \) is
\( 10 + 4 = 14 \) and the completion time of \( T_2 \) is \( 10 + 2 + 1 = 13 \). The maximum completion
time is 14. If the schedule is \((T_2, T_1)\), then the completion time of \( T_1 \) is \( 2 + 10 + 4 = 16 \) and
the completion time of \( T_2 \) is \( 2 + 1 = 3 \). The maximum completion time is 16. Therefore the
schedule \((T_1, T_2)\) is the optimal schedule.

(a) (6 marks) Suppose we have the following instance \( I \) comprising three tests: \( T_1 = (10, 6) \),
\( T_2 = (7, 8) \) and \( T_3 = (4, 2) \). Here are six “plausible” greedy strategies: order the jobs
in increasing or decreasing order by the values \( c_i \), \( m_i \) or \( c_i + m_i \). For the given instance
\( I \), determine the schedule arising by applying each of these six strategies, and thereby
identify which strategies are non-optimal for this instance. Show all your work.

Hint: You should be able to establish that five of the six strategies are non-optimal.
(b) (9 marks) It turns out that the remaining non-optimal strategy (from part (a)) is in fact an optimal strategy. Prove this!

**Hint:** Begin with an arbitrary optimal solution. Suppose that two tests that are adjacent in the schedule are “out of order”. Prove that they can be interchanged and the resulting schedule must still be optimal. By a sequence of interchanges of this type, the optimal solution can be transformed into the greedy schedule, preserving optimality at each step.

5. **Cross-country Driving** (10 marks)

You are planning to drive cross-country along a pre-determined route. There is a limit $L$ on the number of kilometers you can travel per day. There are $n$ hotels along the route in which you can stay overnight. For each hotel $i$, you are given its distance from your start point of the route, denoted by $d_i$. For simplicity, assume that you start from $d_0 = 0$ and your destination is $d_n$ and hotels are indexed by increasing distance from your start point, i.e., $d_0 < d_1 < ... < d_{n-1} < d_n$. At the end of each day you have to stay at a hotel (i.e., you cannot park on the side of the road to sleep after going $L$ kilometers each day). Design and analyze a greedy algorithm that reaches from $d_0$ to $d_n$ in the fewest number of days possible.

(You may assume $d_{i+1} - d_i \leq L$ for each $i$, to ensure that a solution always exists.)

*Example:* for the input $d_0 = 0$, $d_1 = 2$, $d_2 = 7$, $d_3 = 9$, $d_4 = 13$, $d_5 = 15$, $d_6 = 16$, $d_7 = 18$, $d_8 = 25$, and $L = 10$, the optimal number of days is 3. One optimal solution is to stop at hotels 2 and 6, since $7 - 0$ and $16 - 7$ and $25 - 16$ are all less than or equal to $L$. Another optimal solution is to stop at hotels 3 and 7.

Describe your algorithm, prove that your algorithm is correct, i.e., it always computes an optimal solution, and analyze its runtime.