1. Solve the following recurrences using the Master Theorem (MT), if it is applicable, and show your work. If the MT is not applicable, then state clearly why this is so. Assume throughout that $T(1) = 1$ and that $n$ is a power of 4, 8, and 2, in recurrences (i), (ii), and (iii), respectively.

(i) $T(n) = 4T(n/4) + \sqrt{n} \log n$
(ii) $T(n) = 2T(n/8) + \sqrt{n}$
(iii) $T(n) = 2T(n/2) + n/\log n$

2. Let $S$ be a set of $n$ intervals on the real line, where each interval is specified by its two endpoints. The union of the intervals in $S$ is a collection, $U$, of disjoint intervals.

For instance, if $S = \{[1, 4], [2.5, 3.5], [2, 5], [6, 7], [6.5, 8]\}$, then $U = \{[1, 5], [6, 8]\}$. Give an $O(n \log n)$-time divide-and-conquer algorithm to find the union of the intervals in $S$.

Your answer should include (a) a description of the key ideas, from which the correctness of your approach should be evident, (b) pseudocode, and (c) analysis of the running time.

3. Assume that you have somehow obtained a database containing the stock price of Elgoog, Inc. for the next $n$ trading days. Let these prices be $p(1), p(2), \ldots, p(n)$.

You wish to find a day $i$ on which to buy the stock and a day $j$ on which to sell it ($i \leq j$), such that your return on investment, $p(j) - p(i)$, is maximized. This can be done in $O(n^2)$ time, by trying all pairs $i$ and $j$, but you would like to do better to impress your boss. Design an $O(n)$-time algorithm, based on divide-and-conquer, for this problem.

Your answer should include (a) a clear description of the main ideas, (b) pseudocode, and (c) an analysis of the running time.

4. Give a top-down, memoized version of the algorithm $\text{LCS-Length}(X, Y)$ to compute, in $\Theta(mn)$ time, the length of a longest common subsequence of strings $X$ and $Y$, where $m = |X|$ and $n = |Y|$. (You do not have to retrieve the LCS itself; just compute its length.) Give a careful analysis of the running time.

5. Let $A$ be an $n \times n$ matrix of positive integers. A path, $P$, in $A$, from a cell $A[i, j]$ to a cell $A[k, m]$ is a sequence of cells starting at $A[i, j]$ and ending at $A[k, m]$. The cost of $P$ is the sum of the integers in the cells comprising it. $P$ is valid if it satisfies the following condition: Any cell on the path is visited from only its western, northern, or north-western neighbor (provided the neighbor exists).

Design a $O(n^2)$-time, bottom-up dynamic programming algorithm to compute the minimum cost of a valid path which begins at $A[1, 1]$ and exits $A$ from some cell in the rightmost column. (Note that the exit cell is not given; you have to find the best one.) You do not have to retrieve the optimal path; just compute its cost.

Your answer should include: (a) a short description of the main ideas, including the recurrence and a justification for it, (b) pseudocode, and (c) an analysis of the running time.

6. The Republican Party wishes to establish offices for its candidate along Interstate 35, from the Twin Cities down to Dallas. There are $n$ potential sites, $s_1, s_2, \ldots, s_n$, for the offices, in sorted order along the route. Each site $s_i$ is at distance $d_i$ from the first site $s_1$ (so $d_1 = 0$). By law, successive offices on the route must be at least $m$ miles apart. The party believes that an office
located at site \( s_i \) will generate \( v_i > 0 \) votes for its candidate. The goal is to determine a subset of these \( n \) sites where offices can be located so as to maximize the total number of votes generated.

For instance, if \( n = 4 \) and \( d_1 = 0, d_2 = 200, d_3 = 600, \) and \( d_4 = 800 \) miles, \( m = 500 \) miles, and \( v_1 = 5000, v_2 = 6000, v_3 = 5000, \) and \( v_4 = 1000 \), then the optimal solution is to locate offices at sites \( s_1 \) and \( s_3 \), which generates a total of 10000 votes.

Design a bottom-up dynamic programming algorithm for this problem, whose worst-case running time is \( O(n^2) \). You need only compute the maximum number of votes generated; the office locations are not needed.

Your answer should include: (a) a short description of the main ideas, including the recurrence and a justification for it, (b) pseudocode, and (c) an analysis of the running time.

**Hint:** Let \( u_i \) be the total number of votes generated when picking office locations optimally from \( s_1, \ldots, s_i \). Set up your dynamic programming recurrence for \( u_i \).

**Bonus:** Show how to solve the problem in \( O(n) \) time, still using dynamic programming.