1. Quickies.
   (a) Name two characteristics of problems that can be solved using greedy techniques.
   (b) Dynamic Programming and Divide-and-Conquer techniques both decompose problems into subproblems. If you find that your decomposition results in subproblems that are not repeatable, which technique should you use?
   (c) We used algebraic decision trees to establish a lower bound result for the Sorting Problem. What do the nodes of the tree represent?
   (d) What is the expected time complexity of RANDOMIZED-SELECT running on a problem of size n?
   (e) To show a problem is NP-Complete, you must demonstrate a transformation between problems. We will say the old problem is a known NP-complete problem and the new problem is the one you want to show is NP-complete. Is the new problem transformed to the old problem or is the old problem transformed to the new problem?
   (f) Given a problem of size 100, should you choose a factorial algorithm or a polynomial-time algorithm to solve it?
   (g) Name a sorting algorithm that has different average-case and worst-case time complexities.
   (h) Explain briefly why COUNTINGSORT can have worst-case time complexity $O(n)$ when the lower bound on the Sorting Problem, $O(n \lg n)$, is larger.

2. Recurrences.
   (a) Can the recurrence from the next part of this problem be solved using the Master Method?
   (b) Set up a recurrence relation for the time complexity of the following algorithm.

```c
int p, q, r;
int m(n)
    if (n == 1) return 0;
    min = ∞;
    for k = 1 to n-1 do
        temp = m(k) + m(n-k) + p*q*r;
        if (temp < min) then min = temp;
    return min;
```

3. Correctness.

If $a$ and $b \neq 0$ are integers, the program DIVI given below computes the integer quotient $q$ and remainder $r$ for the division $a/b$ by repeatedly subtracting $b$ from $a$.

If $a$ and $b \neq 0$ are integers, prove the loop invariant: At the start of the i-th iteration of the while loop (lines 4 – 6), $i \geq 1$, $a = q \cdot b + r$. Use the notation $r_i$ and $q_i$ for the values of $r$ and $q$ before the start of the $i$-th loop iteration.
\[ \text{DivI}(a, b, q, r) \]

\[ q = 0; \] (1)

\[ r = a; \] (2)

\[ \text{while } (r \geq b) \text{ do} \] (3)

\[ r = r - b; \] (4)

\[ q = q + 1; \] (5)

4. Transforming Problems.

The Hamiltonian Cycle Problem can be stated as follows: Given an undirected graph, is there a simple cycle that contains all vertices?

The Traveling Salesman Problem can be stated as follows: Given a weighted, undirected graph, and integer \( k \), is there a Hamiltonian cycle of total weight less than or equal to \( k \)?

An instance of Hamiltonian Cycle Problem can be transformed to an instance of the Traveling Salesman Problem. The complete graph on the given \( n \) vertices is constructed, the original edges are assigned weight 1, the new edges are assigned weight 2. The original graph has a Hamiltonian Cycle if and only if the new graph has a Traveling Salesman tour of weight \( n \) or less.

For the graph given below, show the transformed problem and the value of integer weight \( k \).

5. Assembly-line Scheduling. PDA’s are being assembled in a plant with two interconnected assembly lines. Each line \( i \) (\( i = 1 \) or 2) has an entry time \( e_i \), an exit time \( x_i \), and four stations, \( s_{ij}, j = 1, \ldots, 4 \). The assembly time at station \( s_{ij} \) is \( a_{ij} \) and the time to transfer from the \( j \)-th station on line \( i \) to the \((j + 1)\)-st station on the other line is \( t_{ij} \). The actual values are given in the table below.

<table>
<thead>
<tr>
<th>( i )</th>
<th>( e_i )</th>
<th>( x_i )</th>
<th>( a_{i1} )</th>
<th>( a_{i2} )</th>
<th>( a_{i3} )</th>
<th>( a_{i4} )</th>
<th>( t_{i1} )</th>
<th>( t_{i2} )</th>
<th>( t_{i3} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>3</td>
<td>8</td>
<td>5</td>
<td>3</td>
<td>7</td>
<td>3</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>5</td>
<td>7</td>
<td>6</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

Determine the stations on line 2 that should be used to make the fastest trip through the assembly plant. Ties should be broken in favor of line 1.

6. Greedy Strategy. The Coin Changing Problem can be stated as follows: Given an integer \( k \), find a minimum number of coins the values of which add up to \( k \). A greedy choice uses as many of the largest coins as possible. The strategy is repeated using the next largest coin, and so on, until the proper sum is reached.

(a) Using coins of values 1, 5, 10 and 25, solve the coin changing problem for \( k = 32 \).

(b) In a cost-cutting move, coins of value 5 are removed from circulation. Solve the coin changing problem for \( k = 32 \) using coins of values 1, 10 and 25.

(c) What can you conclude about the correctness of the greedy strategy for coin changing problems?

7. Graphs (Do part b first.)

(a) Find the sum of the edge weights in the minimum spanning tree.

(b) For the given graph, compute a minimum spanning tree using Kruskal’s or Prim’s algorithm. If you use Kruskal’s algorithm, show the order in which edges are added to the tree. If you use Prim’s algorithm, use node \( f \) as the root and show the order in which nodes are added to the tree. Indicate which algorithm you are using.