CSCI3160: Midterm
Name: Student ID

Note 1: Write all your solutions in the answer book.
Note 2: You do not need to analyze algorithms that have been discussed in lectures, tutorials, and regular exercises.

Part I (Special Exercises)

Problem 1 (10%). Let \( S \) be a set of \( n \) distinct integers, and \( k_1, k_2 \) be arbitrary integers satisfying \( 1 \leq k_1 \leq k_2 \leq n \). Suppose that \( S \) is given in an array. Give an \( O(n) \) expected time algorithm to report all the integers whose ranks in \( S \) are in the range \([k_1, k_2]\) (recall that the rank of an integer \( v \) in \( S \) equals the number of integers in \( S \) that are at most \( v \)). You need to analyze the running time of your algorithm.

Problem 2 (10%). Consider the following greedy algorithm for the activity selection problem (definition: given a set \( I \) of intervals, find a maximum subset \( T \subseteq I \) such that all the intervals in \( T \) are mutually disjoint). Initialize an empty \( T \), and then repeat the following steps until \( I \) is empty:

- (Step 1) Add to \( T \) the interval \( I \in I \) with the shortest length.
- (Step 2) Remove from \( I \) the interval \( I \) and all the intervals overlapping with \( I \).

Finally, return \( T \) as the answer.

Prove: the above algorithm does not always return an optimal solution.

Problem 3 (10%). Let \( G = (V, E) \) be an undirected connected graph where each edge in \( E \) is associated with a positive weight. Consider any non-empty subset \( S \subset V \). An edge \( \{u, v\} \) in \( E \) is an \( S \)-cross edge if \( u \in S \) but \( v \notin S \). Prove: if \( e \) is an \( S \)-cross edge that has the minimum weight among all \( S \)-cross edges, \( e \) must belong to some MST of \( G \).

Part II (Algorithm Execution)

Problem 4 (10%). Consider the weighted undirected graph below (the number beside each edge indicates the edge’s weight).

![Graph Image]

Suppose that we run Prim’s algorithm to find a minimum spanning tree (MST) of this graph. Write down the edges of the MST in the order picked by the algorithm.
Problem 5 (10%). Consider an alphabet Σ that contains letters a, b, c, and d, whose frequencies are 40%, 10%, 20%, 30%, respectively. Show the code tree obtained by running Huffman’s algorithm on this input.

Problem 6 (10%). For the rod cutting problem, consider the price table below:

<table>
<thead>
<tr>
<th>length</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>price</td>
<td>3</td>
<td>5</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>17</td>
<td>17</td>
<td>20</td>
</tr>
</tbody>
</table>

For each ℓ ∈ [1, 8], define opt(ℓ) as the maximum revenue attainable from a rod of length ℓ. Give the value of opt(ℓ) for each ℓ ∈ [1, 8].

Problem 7 (10%). Define two strings s = AGXT and t = GXTG. For each i ∈ [0, 4], define s[1 : i] as the prefix of s with length i; similarly, for each j ∈ [0, 4], define t[1 : j] as the prefix of t with length j. For any i ∈ [0, 4] and j ∈ [0, 4], define len(i, j) as the length of the longest common subsequence of s[1 : i] and t[1 : j]. Give the value of len(i, j) for each i ∈ [0, 4] and j ∈ [0, 4].

Part III (Algorithm Design and Analysis)

Problem 8 (15%). Let I be a set of n intervals of the form [x, y], where x and y are integers. The union of all the intervals in I equals [0, U] (i.e., every value in [0, U] is covered by at least one interval in I). We call a subset S ⊆ I a solution if the union of the intervals in S equals [0, U]. A solution is optimal if it has the smallest size among all solutions.

For example, suppose that I = {[10, 15], [0, 35], [20, 50], [55, 60], [5, 30], [0, 25], [40, 60], [45, 50], [25, 45]} and U = 60. An optimal solution has size 3, e.g., {[0, 35], [20, 50], [40, 60]}. Another optimal solution is {[0, 25], [25, 45], [40, 60]}.

Prove that the following algorithm always returns an optimal solution.

1. Set a = 0 and S = ∅
2. Let I = [x, y] be the interval in I maximizing the length of [x, y] ∩ [a, U] among all the intervals covering value a (e.g., if a = 15, U = 60, and I = [0, 35], then [x, y] ∩ [a, U] = [15, 35], which has length 20)
3. Add I to S, and set a = y + 1
4. If a > U then return S; otherwise, repeat from Step 2

Problem 9 (15%). Let A be an array of n positive integers. You need to find the maximum possible sum that can be achieved by selecting a subset of the elements in A. However, there is a constraint: you cannot include two consecutive elements of the array into the subset. In other words, if you choose an element at index i, you cannot choose the element at index i + 1. For example, if A = (4, 1, 3, 9, 5), the maximum sum is 13, which can be achieved by selecting 4 and 9.

Design an algorithm to solve the problem in O(n) time. You need to analyze the running time of your algorithm.