This booklet has 10 pages (including the cover page).

**INSTRUCTIONS**

1. No external help (books, notes, laptops, tablets, phones, etc.) or collaboration is allowed.

2. You will need to spend more time thinking than writing. When you write, make sure that the answers are brief, clear, and to the point. You will not get credit for incomprehensible answers, even if they are correct.

3. When asked for an algorithm, give a short description in English. Do not give pseudocode, proof of correctness, or running time analysis unless specifically asked for.

4. You can write on the reverse sides of the pages if you need to.

**DO NOT WRITE BELOW THIS LINE**

<table>
<thead>
<tr>
<th>Problem 1</th>
<th>Divide and conquer</th>
<th>/ 15</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem 2</td>
<td>Data Structures</td>
<td>/ 15</td>
</tr>
<tr>
<td>Problem 3</td>
<td>Dynamic Programming</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(a)</td>
<td>/ 7</td>
</tr>
<tr>
<td></td>
<td>(b)</td>
<td>/ 7</td>
</tr>
<tr>
<td></td>
<td>(c)</td>
<td>/ 2</td>
</tr>
<tr>
<td></td>
<td>(d)</td>
<td>/ 10</td>
</tr>
<tr>
<td>Problem 4</td>
<td>Greedy Algorithm</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(a)</td>
<td>/ 5</td>
</tr>
<tr>
<td></td>
<td>(b)</td>
<td>/ 15</td>
</tr>
<tr>
<td>Problem 5</td>
<td>Reductions</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(a)</td>
<td>/ 12</td>
</tr>
<tr>
<td></td>
<td>(b)</td>
<td>/ 12</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>/ 100</td>
</tr>
</tbody>
</table>
Problem 1 (Divide and conquer). [15 points] Consider a divide-and-conquer algorithm that does the following on an input of \( n \) items: in \( O(n) \) time, it creates \( n \) (overlapping) subsets containing \( \sqrt{n} \) items each, and recurses on each such subset. Write the recurrence for the running time of this algorithm and solve it to obtain its time complexity.
Problem 2 (Data Structures). [15 points] Suppose that you have an array $A$ of $n$ numbers, and you want to construct a data structure for $SUM[i, j]$ queries, i.e., you must return the value of $\sum_{k=i}^{j} A[k]$ using the data structure.

The data structure is constructed upfront before any query arrives, and the time complexity of this construction algorithm is called the construction time. When a query arrives, the time complexity of the query answering algorithm is called the query time. Finally, the size of the data structure is called its space complexity.

Give a data structure with $O(n)$ space complexity, $O(1)$ query time, and $O(n)$ construction time. Describe the data structure and its construction and query answering algorithms. No running time or space complexity proof is required.
Problem 3 (Dynamic Programming). A vertex cover of an undirected graph is a subset of vertices containing at least one endpoint of every edge. In this problem, we will develop a dynamic programming algorithm to compute the size of a minimum vertex cover for a rooted tree $T$. The following functions will be our main ingredients (for any vertex $u$, the subtree of $T$ rooted at $u$ is denoted $T_u$):

- $\text{MIN}(u)$ : the size of the minimum vertex cover of $T_u$.
- $\text{MIN-INC}(u)$ : the smallest number of vertices in a vertex cover of $T_u$ that includes $u$.

In answering the questions that follow, denote the root of $T$ by $r$, and the set of children of any vertex $u$ by $\text{CHILD}(u)$.

(a) [7 points] Recursively define $\text{MIN-INC}(u)$ in terms of the ingredients given in the problem description. Give a brief justification for your answer.

(Hint: If $u$ is included in the vertex cover, are we constrained about whether $u$’s children should be included?)
(b) **[7 points]** Recursively define $\text{MIN}(u)$ in terms of the ingredients given in the problem description. Give a brief justification for your answer.

*(Hint: What if $u$ is not included in the vertex cover? Now, combine this with the previous part.)*

(c) **[2 point]** What should you return as the final output in terms of the functions you have computed?
(d) [10 points] In what order on the vertices should the algorithm populate the entries in the dynamic programming table? Give the running time of the algorithm with a brief justification.

(Hint: A tree has n vertices and n – 1 edges.)
Problem 4 (Greedy Algorithm). In this question, we will examine a greedy algorithm for computing a minimum spanning tree (MST) of an undirected, edge-weighted graph $G = (V, E)$. (Assume that all edge weights are distinct.) We will denote the output by $T$ and its connected components by $S$.

1. Initially, $T$ is empty, and hence $S$ has each vertex in a distinct component.

2. Now, we run a series of iterations, where in any given iteration we will merge several components in $S$. Let $S = \{C_1, C_2, \ldots, C_h\}$ denote our current set of components in some iteration. For every component $C_i$, $1 \leq i \leq h$, we add the minimum weight edge $(u, v)$ such that $u \in C_i, v \notin C_i$ (denoted $\text{MIN-EDGE}(C_i)$) to $T$. Correspondingly, we update $S$ to the new set of connected components in $T$.

3. The algorithm terminates when $S$ has a single connected component comprising all vertices.

(a) [5 points] In asymptotic notation, how many iterations will the algorithm perform? Give a brief justification for your answer.
(b) [15 points] Show that $T$ is a spanning tree at the end of the algorithm.

(Hint: In any iteration, contract each connected component in $S$ to a single node. Now, use the distinctness of edge weights to argue that the edges added to $T$ in the current iteration form an acyclic graph on these contracted nodes. Use induction and the termination condition to complete the proof.)
Problem 5 (Reductions). Let $G = (V,E)$ represent a directed graph with edge weights $w : E \rightarrow \mathbb{Z}_+$. For any subset of vertices $S \subseteq V$, the directed cut $(S, \overline{S})$ is defined as the set of edges $(u,v)$ with $u \in S$, $v \notin S$. The weight of this cut is the sum of weights of edges in the cut:
\[
\delta(S) = \sum_{(u,v) \in (S,\overline{S})} w(u,v).
\]
Note that if $S = \emptyset$ or $V$, then $\delta(S) = 0$.

In this question, we will show that the following two problems are equivalent under linear-time reductions:

- **Minimum $s - t$ cut**: Given two vertices $s,t \in V$, find:
\[
\min_{S \subseteq V : s \in S, t \notin S} \delta(S).
\]

- **Maximum blocking cut**: Given vertex weights $W : V \rightarrow \mathbb{Z}$ (note that vertex weights can be negative), find:
\[
\max_{S \subseteq V} \sum_{v \in S} W(v) - \delta(S).
\]

(a) **[12 points]** Using an $O(T)$-time algorithm for the minimum $s - t$ cut problem, show that you can solve the maximum blocking cut problem in $O(T + |V| + |E|)$ time.
(b) [12 points] Using an $O(T)$-time algorithm for the maximum blocking cut problem, show that you can solve the minimum $s-t$ cut problem in $O(T + |V| + |E|)$ time.