CSE 421 : Sample Final Exam

Instructions

• This Sample Final was assembled from problems given in previous 421 exams.
• The exam here is approximately the length of an 110 minute exam. Yours may be slightly longer or shorter.
• You are permitted one piece of 8.5x11 inch paper with handwritten notes (notes are allowed on both sides of the paper).
• You may not use a calculator or any other electronic devices during the exam.

Advice

• Move around the exam; if you get stuck on a problem, save it until the end.
• Proofs are not required unless otherwise stated.
• Remember to take deep breaths.

<table>
<thead>
<tr>
<th>Question</th>
<th>Max points</th>
</tr>
</thead>
<tbody>
<tr>
<td>True or False</td>
<td>0</td>
</tr>
<tr>
<td>Interval Scheduling</td>
<td>0</td>
</tr>
<tr>
<td>Group Test</td>
<td>0</td>
</tr>
<tr>
<td>Number of Paths in a DAG</td>
<td>0</td>
</tr>
<tr>
<td>Vertex Cut</td>
<td>0</td>
</tr>
<tr>
<td>Number Partition</td>
<td>0</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>??</strong></td>
</tr>
</tbody>
</table>
1. True or False

For each of the following problems, circle **True** or **False**. You do not need to justify your answer.

(a) If $f$ and $g$ are two different flows on the same flow graph $(G, s, t)$ and if $v(f) \geq v(g)$ then every edge $e$ in $G$ has $f(e) \geq g(e)$. **True**  **False**

(b) If $f$ is a maximum flow on a flow graph $(G = (V, E), s, t)$ and $B$ is the set of vertices in $V$ that can reach $t$ in the residual graph $G_f$ then $(V - B, B)$ is a minimum capacity $s - t$ cut in $G$. **True**  **False**

(c) If $f$ is a maximum flow on a flow graph $G, s, t)$ and $(S, T)$ is a minimum capacity $s - t$ cut in $G$ then every edge $e$ having endpoints on different sides of $(S, T)$ has $f(e)$ equal to the capacity of $e$. **True**  **False**

(d) If problem $B$ is $NP$-hard and $A \leq_p B$ then $A$ is $NP$-hard. **True**  **False**

(e) If problem $A$ is $NP$-hard and $A \leq_p B$ then $B$ is $NP$-hard. **True**  **False**

(f) If $P \neq NP$ then every problem in $NP$ requires exponential time. **True**  **False**

(g) If problem $A$ is in $P$ then $A \leq_p B$ for every problem $B$ in $NP$. **True**  **False**

(h) If $G$ is a weighted graph with $n$ vertices and $m$ edges that does not contain a negative-weight cycle, then the iteration of the Bellman-Ford algorithm will reach a fixed point in at most $n - 1$ rounds. **True**  **False**

(i) If $G$ is a weighted graph with $n$ vertices and $m$ edges that does contain a negative-weight cycle, then for every vertex $v$ in $G$, the shortest path from $v$ to $t$ in $G$ containing $n$ edges is strictly shorter than the shortest path from $v$ to $t$ in $G$ containing $n - 1$ edges. **True**  **False**
2. Interval Scheduling

The two processor interval scheduling problem takes as input a sequence of request intervals \((s_1, f_1), ..., (s_n, f_n)\) just like the unweighted interval scheduling problem except that it produces two disjoint sets \(A_1, A_2 \subseteq [n]\) such that all requests in \(A_1\) are compatible with each other and all requests in \(A_2\) are compatible with each other and \(|A_1 \cup A_2|\) is as large as possible. \((A_1\) might contain requests that are incompatible with requests in \(A_2\)). Does the following greedy algorithm produce optimal results? If yes, argue why it does; if no, produce a counter example.

```
Sort requests by increasing finish time
A_1 = \emptyset
A_2 = \emptyset
while there is any request \((s_i, f_i)\) compatible with either \(A_1\) or \(A_2\) do:
    Add the first unused request, if any, compatible with \(A_1\) to \(A_1\).
    Add the first unused request, if any, compatible with \(A_2\) to \(A_2\).
end while
```
3. Group Test

You are given a subsequence of $n$ bits $x_1, ..., x_n \in \{0, 1\}$. Your output is to be either

- any $i$ such that $x_i = 1$
- the value 0 if the input is all 0’s

The only operation you are allowed to use to access the inputs is a function **Group-Test** where

$$
\text{Group-Test}(i, j) = \begin{cases} 
1 & \text{if some bits } x_i, ..., x_j \text{ has value 1} \\
0 & \text{if all bits } x_i, ..., x_j \text{ have value 0}
\end{cases}
$$

(a) Design a divide and conquer algorithm to solve the problem that uses only $O(\log n)$ calls to **Group-Test** in the worst case. Your algorithm should *never* access the $x_i$ directly.

(b) Briefly justify your bound on the number of calls.
4. Number of Paths in a DAG

You are given a directed acyclic graph $G = (V, E)$ and a node $t \in V$. Design a linear time algorithm to compute for each vertex $v \in V$, the number of different paths from $v$ to $t$ in $G$. Analyze its running time in terms of $n = |V|$ and $m = |E|$.

(a) Give the optimization formula for computing the number of paths from $i$ to $t$.

(b) Give pseudocode for the (iterative) dynamic program for computing the number of different paths from $v$ to $t$ in $G$ for every vertex $v \in V$.

(c) Give the running time of your algorithm.
5. Vertex Cut

Let \( G = (V, E) \) be a directed graph with distinguished vertices \( s \) and \( t \). Describe an algorithm to compute a minimum sized set of vertices to remove to separate \( s \) and \( t \). Your algorithm should identify the actual vertices to remove (and not just determine the minimum number of vertices that could be removed).
6. Number Partition

The Number Partition problem asks, given a collection of non-negative integers $y_1, ..., y_n$, whether or not it is possible to partition these numbers into two groups so that the sum in each group is the same. Prove that Number Partition is NP-complete by solving the following problems.

(a) Show that Number Partition is in NP.

(b) Show that Subset Sum $\leq_p$ Number Partition.

Recall that in the Subset Sum problem, we are given a collection of integers (which can be positive and negative), we want to see if it is possible to find a subset that sums up to 1.

Hint: Given an input to Subset Sum include two large numbers whose size differs by $S - 2$ where $S$ is the sum of all input numbers.