CSE 421 : Sample Final Exam Solutions

Name:	**NetID:** @uw.edu

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.

Question	**Max points**
True or False	0
Interval Scheduling	0
Group Test	0
Number of Paths in a DAG	0
Vertex Cut	0
Number Partition	0
Total	**???**

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** **Solution:**

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** **Solution:**

True

(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** **Solution:**

False

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

False

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

True

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

False

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

True

(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** **Solution:**

True

(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** **Solution:**

```
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 $A_1, A_2 \subset [n]$ 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
```

Solution:

Greedy is not optimal in this case. Consider the following example: (1,2), (3,4), (1,5). These jobs are sorted by increasing finish time. The optimum solution sends (1,2), (3,4) to the first machine and (1,5) to the second machine so it schedules all jobs.

The above Greedy algorithm first allocates (1,2) to A_1, then it allocates (3,4) to A_2. Then it cannot allocate (1,5) to either of the machines.

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 $x_i = 1$ or
- 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

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

(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. **Solution:**

```
$i = 1, j = n$
while $i < j$ do:
    $mid = \lfloor \frac{i+j}{2} \rfloor$
    if Group-Test$(i, mid) == 1$ then:
        $j = mid$
    else
        $i = mid + 1$
return $i *$ Group-Test$(i, i)$
```

(b) Briefly justify your bound on the number of calls. **Solution:**

Correctness: Given an interval $\{i, i+1, ..., j\}$ we partition it into two (almost) equal size intervals where the first interval is $\{i, ..., \lfloor (i+j)/2 \rfloor\}$. Then we test if there is a 1 bit in the first interval using the Group-Test$(i, \lfloor (i+j)/2 \rfloor\})$; if the answer is 1 then we can recursively solve the problem on the first interval. Otherwise, if Group-Test$(i, \lfloor (i+j)/2 \rfloor\}) = 0$ there is no solution in the first interval. So, either there is a solution in the second interval or there is no solution in the second interval. In either case, we can reduce the problem to the second interval.

Runtime: The length of the interval decreases by a factor of 2 each time. So, we are only going to call Group-Test at most $O(\log n)$ many times.

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 **Solution:**

$$OPT(i) = \begin{cases} 0 & \text{if } i > t \\ 1 & \text{i = t} \\ \sum_{j:(i,j)\in E} OPT(j) & \text{otherwise} \end{cases}$$

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

```
Find a topological sorting of $G$ and rename vertices such that $i < j$ for all $(i, j) \in E$
**for** $i = t + 1 \to n$ **do**
    $M[i] = 0$
$M[t] = 1$
**for** $i = t - 1 \to 1$ **do**
    $M[i] = 0$
    **for** each edge $(i, j) \in E$ **do**
        $M[i] = M[i] + M[j]$
```

(c) Give the running time of your algorithm. **Solution:**

For $i \geq t$, we compute $M[i]$ in $O(1)$ steps. Therefore the algorithm runs in $O(n + m)$ in the worst case.

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).

Solution:

We build a flow graph and then use the minimum cut in the graph to determine the set of vertices to remove. We then split each vertex v, other than s or t into vertices v_{in} and v_{out} with a unite capacity edge from v_{in} to v_{out}. The vertex s is replaced by s_{out} and t is replaced by t_{in}. The edge (u, v) is replaced by an edge (u_{out}, v_{in}) with infinite capacity.

We compute a maximum flow between s_{out} and t_{in}. Let S be the set of vertices reachable from s_{out} in the residual graph by paths of positive capacity. We say that a vertex v is a cut vertex if v_{in} is in S, but v_{out} is not in S. The cut vertices are a minimum set of vertices to remove to separate the graph.

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. **Solution:**

Given a partition of $y_1, ..., y_n$ into two groups the verifier sums up the numbers in each group and outputs yes if the two sums are equal and no otherwise. The verifier obviously runs in time polynomial in n and logy_1, ...,logy_n because we can add up any two integers a, b in time $O($log$a$ + log $b)$.

(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. **Solution:**

Given an input $x = \{x_1, ..., x_n\}$ to the Subset Sum problem. Let $S = x_1 + ... + x_n$. Add two numbers $y = 10S$ and $z = 11S - 2$. Now, we give $f(x) = \{x_1, ..., x_n, y, z\}$ as an input to the Number Partition problem. Now, we prove the x is a yes answer to Subset Sum if and only if $f(x)$ is a yes answer to Number Partition.

Forward Direction: If x is a yes instance of Subset Sum than $f(x)$ is a yes answer of Number Partition. Since x is a yes answer of Subset Sum, there exists a set $A \subseteq \{x_1, ..., x_n\}$ such that $\sum_{x_i \in A} x_i = 1$. Then we claim $\{z \cup A, y \cup (\{x_1, ..., x_n\} - A)\}$ is a yes instance of Number Partition. It is enough to see

$$z + \sum_{x_i \in A} x_i = z + 1 = 11S - 1 = 10S + (S - 1) = y + \sum_{x \notin A} x_i$$

where the last identity uses that $x_1 + ... + x_n = S$ and $\sum_{x_i \in A} x_i = 1$.

Backwards Direction: If $f(x)$ is a yes instance of Number Partition, the n x is a yes instance of Subset Sum.

Since $f(x)$ is a yes instance of Number Partition, there exists a partition A, B of $\{x_1, ..., x_n, y, z\}$ such that the numbers in A, B sum up to the same amount. Since $x_1 + ... + x_n + y + z = 22S - 2$, the numbers in A must sum up to $11S - 1$ and similarly numbers in B must sum up to $11S - 1$. Therefore, y belongs to one of A, B, and z to the other one. Without loss of generality, suppose $z \in A$ and let $C = A - \{z\}$. Then, $\sum_{x_i \in C} x_i = 11S - 1 - z = 1$. Therefore, the answer to the Subset Sum problem is yes.