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
Network Flow	0
Interval Scheduling	0
Assigning Teachers to Classes	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

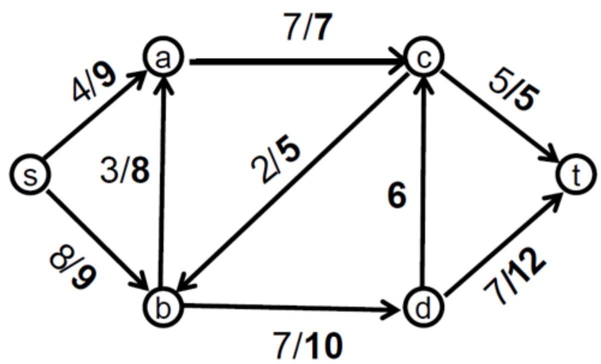
(i) If G is a weighted graph with n vertices and m edges that $does$ contain a negative-weight cycle, then for $every$ vertex v in G, the shortest path from v to t in G containing n edges is strictly shorter than the shortest path from v to t in G containing $n-1 edges$. **True** **False**

Solution:

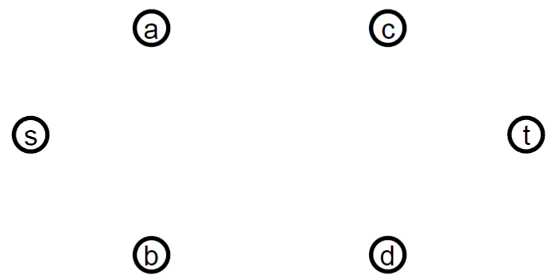
False

2. Network Flow

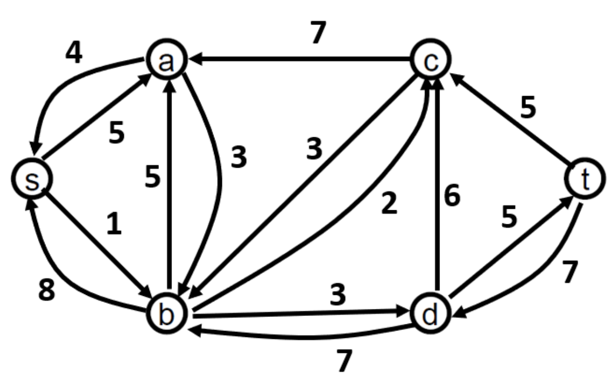
Consider the following flow network with a flow f shown. An edge labelled "b" means that it has capacity b and flow 0. An edge labelled "a/b" means that the flow on that edge is a and the capacity is b.



(a) Draw the residual graph G_f below.



Solution:

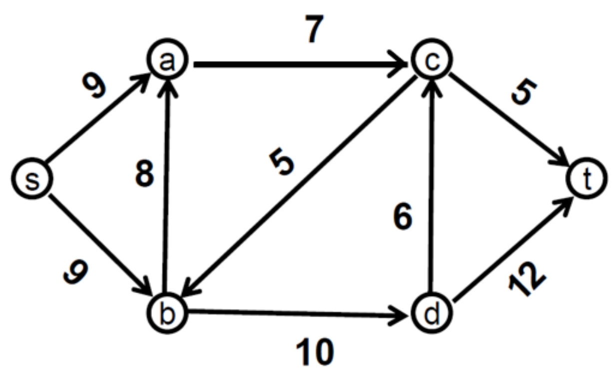


(b) What augmenting path in this graph would result in the greatest increase in flow value? (List the names of the vertices on this path in order.)

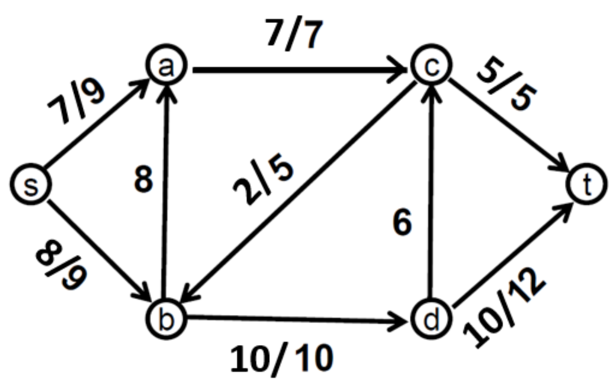
Solution:

$s, a, b, d, t.$

(c) On the diagram below, indicate the new flow values resulting from augmenting along the path you found in part (b).



Solution:



3. 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 = ∅
A_2 = ∅
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.

4. Assigning teachers to courses

The *Teacher Assignment* problem is: given a set of teachers $T = \{t_1, \ldots, t_{n_1}\}$ and a set of courses $C = \{c_1, \ldots, c_{n_2}\}$ determine an assignment of teachers to courses. The input to the problem has a bipartite graph $G = (T, C, E)$ where an edge (t, c) indicates that teacher t can teach class c. For each teacher t_i there is a binary integer u_i giving the number of courses that t_i must teach, and for each course c_j there is a binary integer d_j indicating how many teachers must be assigned to the course. A teacher t can be assigned at most once to course c (in other words, if multiple teachers are required for a course, they must be distinct). Describe an efficient algorithm to find an assignment of teachers to courses if one exists. If no assignment is possible that meets these constraints, the algorithm should report failure. Be sure to argue the correctness of your solution and analyze its efficiency.

Solution:

Make a flow network with vertices $s, t_1, \ldots, t_{n_1}, c_1, \ldots, c_{n_2}, t$. There are three types of edges: edges of the type (s, t_i) with capacity u_i, edges of the type (t_i, c_j) with capacity 1, and edges of the (c_j, t) with capacity d_j. Find the maximum flow in this network using the capacity scaling algorithm or using the Edmonds-Karp analysis of the Ford-Fulkerson algorithm finding shortest augmenting paths. By the integrality theorem, the max flow will have a solution where all flows are integers, which ensures that instructors are not fractionally assigned to courses. If the flow fills all the edges from s and into t to capacity, then the problem can be solved; otherwise report failure. Morever, the edges (t_i, c_j) carrying flow 1 in the final solution correspond to the teacher assignments. To prove the correctness, note that every possible solution to the teacher assignment corresponds to a flow that fills all the edges from s and into t, and such a flow must be equal to a maximum flow because its value is equal to the capacity of the cut that separates s from the rest of the flow network.

Let $n = n_1 + n_2$. Write m for the number of edgees in G. Edmonds-Karp would give an $O(nm^2)$ runtime. Capacity scaling would give $O(nm \log U)$ where $U = \max(\max_i u_i, \max_j d_j)$.

5. 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.

6. 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 make it impossible to get from s to 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 unit 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.

7. 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 $\log y_1, ..., \log y_n$ 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 non-negative integers and a non-negative target t and we want to see if it is possible to find a subset that sums up to t.
Hint: Given an input to $Subset\ Sum$ include two large numbers whose size differs by $S - 2t$ where S is the sum of all input numbers.

Solution:

Given an input $(x, t) = (x_1, ..., x_n, t)$ to the Subset Sum problem. Let $S = x_1 + ... + x_n$.

If $t > S$, let the reduction function $f(x, t) = \{1, 1, 3\}$.

Otherwise, add two numbers $y = 10S$ and $z = 11S - 2t$. Now, we give $f(x, t) = \{x_1, ..., x_n, y, z\}$.

The function f is clearly polynomial-time computable.

Now, we prove the x is a yes answer to Subset Sum if and only if $f(x, t)$ is a yes answer to Number Partition.

Forward Direction: If (x, t) is a yes instance of Subset Sum than $f(x, t)$ is a yes answer of Number Partition.
Since x is a yes answer for Subset Sum, there exists a set $A \subseteq \{x_1, ..., x_n\}$ such that $\sum_{x_i \in A} x_i = t$. In this case, $t \leq S$ so the special case does not occur and the output of the reduction is the set $\{x_1, \ldots, x_n, y, z\}$. Then we claim $(z \cup A, y \cup (\{x_1, ..., x_n\} - A))$ is a certificate for Number Partition. It is enough to see

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

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

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

Since $f(x, t)$ is a yes instance of Number Partition, then it is not equal to $\{1, 1, 3\}$ so 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 $x_1 + ... + x_n + y + z = 22S - 2t$, the numbers in A must sum up to $11S - t$ and similarly numbers in B must sum up to $11S - t$. 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, $\sum_{x_i \in C} x_i = 11S - t - z = t$. Therefore, the answer to the Subset Sum problem is yes.