Feedback — Assignment 6

You submitted this quiz on **Thu 11 Apr 2013 10:38 AM PDT -0700**. You got a score of **28.00** out of **28.00**.

Question 1

A polyhedron is said to be full-dimensional if it has positive volume. Consider a bounded polyhedron $\mathcal{P} = \{x \in \mathbb{R}^n \mid Ax \leq b\}$with A having non-zero rows. Of the following statements select the ones that are equivalent to the statement "$\mathcal{P}$ is full-dimensional".

Your Answer		Score	Explanation
☐ There exist n distinct feasible points that do not lie on a common hyperplane in $\mathbb{R}^n$	✔	1.00	This is not possible since any set of n points in $\mathbb{R}^n$ always lie on some common hyperplane.
☑ There exist $n+1$ distinct feasible points that do not lie on a common hyperplane in $\mathbb{R}^n$	✔	1.00	
☐ There exists a feasible point x such that $a_i^T x < b_i$ for at least one inequality in the system $Ax \leq b$	✔	1.00	One counterexample could be a point on one side of a triangle that lies in 3 dimensions (which is not full-dimensional).
☑ There exists an x such that $Ax < b$	✔	1.00	
Total		4.00 / 4.00	

Question Explanation

First let's see that the full-dimensionality of $\mathcal{P}$ is equivalent to the existence of a feasible x such that $Ax < b$. Suppose there exists such an x, then we define $\epsilon := \min_i \frac{b_i - a_i^T x}{\|a_i\|}$ which is strictly positive. Consider the n-dimensional ball with center x and radius ϵ, $\mathcal{B}(x, \epsilon)$. Pick a point $y \in \mathcal{B}(x, \epsilon)$. Then, we can write y as

$x + \mu \hat{l}$ where $\mu \leq \epsilon$ and $\hat{l}$ is a unit vector. So, for every $j \in \{1, \ldots, m\}$, $a_j^T y = a_j^T (x + \mu \hat{l}) \leq a_j^T x + \epsilon \|a_j\|$ But by our choice of ϵ, we know that the last expression is at most b_j. Hence each such y is feasible and $\mathcal{B}(x, \epsilon) \subseteq \mathcal{P}$. Conversely, if $\mathcal{P}$ is full-dimensional, then we can find a ball of positive radius within $\mathcal{P}$ whose center satisfies the condition $Ax < b$.

Now we show the equivalence of the existence of $n+1$ feasible solutions that do not lie on a common hyperplane to the existence of an x such that $Ax < b$. Suppose there are $n+1$ feasible points $p_1, \ldots, p_{n+1}$ that do not lie on a common hyperplane in $\mathbb{R}^n$. Then define x as $\frac{1}{n+1} \sum_{i=1}^{n+1} p_i$, which is feasible. If $Ax < b$ then we are done (by the first part). If not, then there exists a $j \in \{1, \ldots, m\}$ such that $a_j^T x = \frac{1}{n+1} \sum_{i=1}^{n+1} a_j^T p_i = b_j$ Then, we know that $a_j^T p_i = b_j$ for all $i \in \{1, \ldots, n+1\}$ Thus, we have constructed a hyperplane that contains all the points p_i. So there exists no such j. Conversely, if we have an x such that $Ax < b$ then we can pick $n+1$ feasible points from an appropriately defined ball inside $\mathcal{P}$ such that they don't lie on a common hyperplane.

Question 2

In this question we will see if it is possible to make money in the currency exchange market (in an idealized scenario where we have no brokerage costs). The setting is that you have n different currencies and every 1 unit of currency i can be converted into r_{ij} units of currency j. We are trying to find a series of currency conversions that starts with 1 unit of some currency i and returns > 1 units of currency i. If you rephrased this problem as one of finding negative weight cycles in directed graphs with the currencies as vertices then what would be the weight of a directed edge uv be in this graph?

Your Answer		Score	Explanation
◉ $-\log r_{uv}$	✓	1.00	
○ $\log r_{uv}$			
Total		1.00 / 1.00	

Question Explanation

Consider a sequence of k currency conversions $i_0 \mapsto i_1 \mapsto \ldots i_{k-1} \mapsto i_k (= i_0)$ starting and ending with the same currency. It is possible to make money through this sequence of conversions if and only if $\prod_{j=0}^{k-1} r_{i_j i_{j+1}} > 1$. In this case, $-\log \prod_{j=0}^{k-1} r_{i_j i_{j+1}} = \sum_{j=0}^{k-1} -\log r_{i_j i_{j+1}} < 0$. Hence, we should set the edge weights as $-\log r_{uv}$ for an edge uv in the directed graph.

Question 3

Given a directed graph $G = (V, A)$ and edge costs specified by $\{c_e\}_{e\in A}$ your friend claims that she has found a shortest path $P := (v_0, v_1, \ldots, v_{k-1}, v_k)$ connecting vertices v_0 and v_k. To prove her claim she shows a set of potentials $\{y_v\}_{v\in V}$ to you. Which of the following would you verify to accept her claim?

Your Answer		Score	Explanation
☐ $y_e \geq 0 \;\forall\; e \in E$	✔	0.80	
☑ $\sum_{i=1}^{k} c_{e_i} = y_{v_k} - y_{v_0}$	✔	0.80	
☐ $y_u + c_{uv} > y_v \;\forall\; uv \in E$	✔	0.80	
☐ $\sum_{i=1}^{k} c_{e_i} > y_{v_k} - y_{v_0}$	✔	0.80	
☑ $y_u + c_{uv} \geq y_v \;\forall\; uv \in E$	✔	0.80	
Total		4.00 / 4.00	

Question Explanation

Using the shortest path that connects v_0 to every other vertex v we can obtain a potential function defined such that y_v is the length of the shortest path from v_0 to v. This defines a feasible potential since across any edge uv we have that $y_u + c_{uv} \geq y_v$. From this we get that for any such shortest path with edges $\{e_1, \ldots, e_k\}$ connecting v_0 to v, $\sum_{i=1}^{k} c_{e_i} = y_{v_k} - y_{v_0}$.

Question 4

We can formulate the shortest path problem on a directed graph $G = (V, A)$ with integer edge weights $\{c_e\}_{e\in A}$, source node s and target node t as a linear programming relaxation (LP1)

$$\begin{aligned} \min \sum_{e \in A} c_e x_e \\ \text{subject to} \sum_{w \in V, wv \in A} x_{wv} - \sum_{w \in V, vw \in A} x_{vw} = b_v \ \forall v \in V \\ x_{vw} \geq 0 \ \forall vw \in A \end{aligned}$$

where b_v is -1 if $v = s$, 1 if $v = t$ and 0 otherwise. The dual (LP2) of the above program associates variables with each vertex of G as follows

$$\begin{aligned} \max\ y_t - y_s \\ \text{subject to} \ \ y_w - y_v \leq c_{vw} \ \forall vw \in A. \end{aligned}$$

Which of the following statements is correct?

Your Answer		Score	Explanation
☐ The value of an optimal solution of LP1 is a strict lower bound on the cost of any directed path connecting s and t in G	✔	1.00	
☑ Every basic feasible solution of LP1 is integral	✔	1.00	
☑ Every basic feasible solution of LP2 is integral	✔	1.00	
☑ The value of an optimal solution of LP1 is equal to the cost of shortest path connecting s and t in G	✔	1.00	
Total		4.00 / 4.00	

Question Explanation

The constraint matrix of LP1 is totally unimodular since it is the node-edge incidence matrix of a directed graph which we saw was TU in the previous assignment. Since b is an integral vector, together this implies that every basic solution of LP1 is integral. The dual solution also has integer basic solutions since the transpose of a TU matrix is also TU and c is an integral vector. Finally, the value of the optimal solution of LP1 is equal to the cost of the shortest path connecting r and s in G (from the previous observations, and the fact that we can find a satisfying assignment for the x_e variables from any path connecting s to t in G).

Question 5

In a directed acyclic graph we can speedup Bellman-Ford using the idea of topological sorting which creates an ordering of the vertices such that there is never an edge from v to u whenever u precedes v in the ordering. Suppose you were provided such an ordering with the first vertex being v_0, then what is the time complexity of Bellman-Ford that utilizes this ordering to compute shortest paths from v_0 to all other vertices in a graph with n vertices and m edges?

Your Answer		Score	Explanation
○ $O(n\sqrt{m})$			
◉ $O(m)$	✓	3.00	
○ $O(m\sqrt{n})$			
○ $O(n)$			
Total		3.00 / 3.00	

Question Explanation

Using this linear ordering we can find the single source shortest paths from v_0 to all the other vertices in just a single pass performing only one iteration of the main loop in Bellman-Ford (as opposed to doing n iterations in the general case). In this single iteration, we use the following ordering of edges: an edge e_1 is visited before an edge e_2 if the starting vertex of e_1 comes earlier in the vertex ordering than the starting vertex of e_2 (if e_0 and e_1 have the same starting vertex then the order we visit the edges is immaterial. This ordering of edges corresponds simply to visiting all the leaving edges of vertices which are themselves visited in the specified linear order. The proof of correctness of this algorithm follows by induction on the correctness of the distance labels assigned to the vertices in any shortest path from the source vertex just like in the original Bellman-Ford algorithm.

Question 6

In the Bellman-Ford algorithm, the main loop is run n times where n is the number of vertices in the graph. If in a particular iteration of the main loop, no edges were relaxed, can we then terminate the algorithm at the end of such an iteration and still retain correctness?

Your Answer		Score	Explanation
No			
No provided there are no negative weight cycles			
Yes	✓	1.00	
Total		1.00 / 1.00	

Question Explanation

Call this iteration i. We know that for each edge uv, $d_i(u) + l_{uv} \geq d_i(v)$holds. First we can immediately conclude that there exist no negative weight cycles in the graph (by adding this inequality over all the edges of a cycle). In this answer we use the term "relaxing an edge" in the following way: in some iteration of Bellman-Ford it is found that the distance labels $d(u)$ and $d(v)$ assigned to the endpoints of an edge directed from u to v is such that $d(v) > d(u) + l_{uv}$then Bellman-Ford sets the distance label of $d(v)$ to be $d(u) + l_{uv}$ which we call "relaxing an edge". Towards a contradiction assume that at a later iteration some edge is actually relaxed. Let $j > i$ be the first such iteration and xy be the first edge in the jth iteration that is relaxed i.e., it is the case that $d_j(x) + l_{xy} < d_j(y)$ But since xy is the first edge we are relaxing after iteration i, it must be the case that $d_i(x) = d_j(x)$ and $d_i(y) = d_j(y)$. But we already know that $d_i(x) + l_{xy} \geq d_i(y)$leading to a contradiction.

Question 7

Suppose M' is the size of some matching in a graph G with n vertices and M is a maximum matching, such that $|M| - |M'| = k$ At least how many M'-augmenting paths are there?

Your Answer		Score	Explanation
k	✓	4.00	
$k+2$			
$k+1$			

- $2k$

Total		4.00 / 4.00

Question Explanation

Consider the subgraph H of G with only the edges $M'\Delta M$ (the symmetric difference of M' and M). As shown in the lecture, every component of H is either a simple path or a cycle with alternating edges from M' and M. Since every cycle in H has equal number of edges from both M' and M and every path has a difference of at most one, there must exist at least k disjoint paths in H that have one edge more from M compared to M'. But each such disjoint path is an M'-augmenting path. (That the other options are false can be seen by constructing small examples where this bound is tight)

Question 8

Having answered the previous question, what conclusion follows from it about the length of the shortest augmenting path for the matching M'? (we use k and n as defined in the previous question)

Your Answer		Score	Explanation
$\leq \frac{n}{2k}$			
$\leq \frac{n}{k}$	✓	3.00	
$\leq \frac{n}{k+1}$			
$\leq \frac{n}{k+2}$			
Total		3.00 / 3.00	

Question Explanation

Since there are at least k M'-augmenting paths each pair of which are disjoint, there must be at least one that is no larger than $\lfloor \frac{n}{k} \rfloor$ in length as there are only n vertices in total.

Question 9

Suppose we convert an undirected graph with arbitrary edge costs but without negative weight cycles into a directed graph by duplicating each edge (one in each direction) with the same edge weight as in the undirected graph. Do the distance labels computed by running Bellman-Ford on this directed graph correspond to the shortest path distances in the undirected graph?

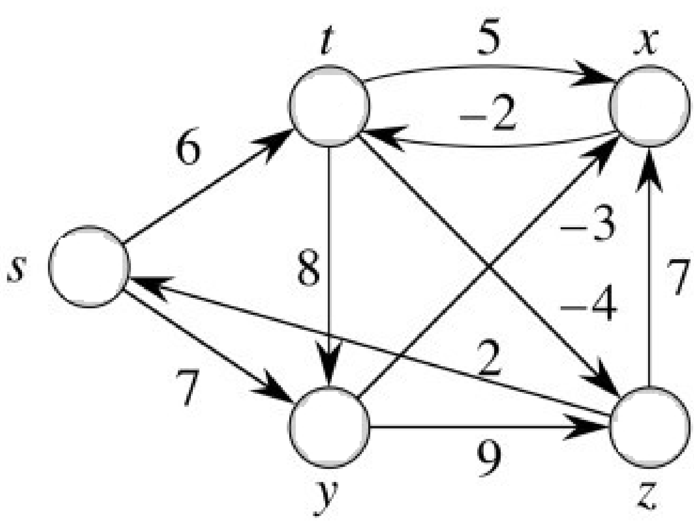
Your Answer		Score	Explanation
◉ No	✓	1.00	
○ Yes			
Total		1.00 / 1.00	

Question Explanation

The resulting directed graph can contain negative weight cycles even if the original undirected graph did not have any. So the correct answer is no.

Question 10

Solve the single source shortest paths problem from the source vertex s in the following graph.



What is the distance label that the Bellman-Ford algorithm will assign to vertex z?

You entered:

-2

Your Answer		Score	Explanation
-2	✓	3.00	
Total		3.00 / 3.00	

Question Explanation

We start with ∞ distance labels to all the vertices except for s which is 0. We then perform 4 iterations of edge relaxations to find the correct distance labels to all the vertices, and in each iteration we relax all the edges of the graph. Following this procedure, we find that in the fourth iteration the vertex z has the distance label -2 (the corresponding path is s-y-x-t-z). We perform one more iteration to make sure that there are no negative weight cycles in the graph and we find none. The distance labels on the other vertices are d(s) = 0, d(t) = 2, d(y) = 7, d(x) = 4.