CS 4820, Spring 2014	**Homework 5, Problem 2**
Name: Piyush Maheshwari	
NetID: pm489	
Collaborators: Eeghan Wagh	

(2) *(15 points)* Let G be a flow network with vertex set V, edge set $E = \{e_1, e_2, \ldots, e_m\}$, source $s \in V$ and sink $t \in V$. If x is a vector in the vector space $\mathbb{R}^m$, we will say that:

- x is a *flow vector* if there exists a flow f such that $x_i = f(e_i)$ for all i.
- x is a *path vector* if there exists a directed path from s to t such that $x_i = 1$ if e_i belongs to the path and $x_i = 0$ otherwise.
- x is a *cycle vector* if there exists a directed cycle such that $x_i = 1$ if e_i belongs to the cycle and $x_i = 0$ otherwise.

(2.a) Prove that every flow vector can be represented as a non-negative weighted sum of path vectors and cycle vectors. To prove this, you should design a polynomial-time algorithm to compute such a representation. As usual, you should analyze your algorithm's running time and prove its correctness.

(2.b) Give an example of a flow network having a flow vector that cannot be represented as a non-negative weighted sum of path vectors. Your example should specify both the flow network G and the flow vector x.

(2.c) Prove the following statement, or give a counterexample: *if a flow f is computed using the Ford-Fulkerson algorithm, then the corresponding flow vector x can always be represented as a non-negative weighted sum of path vectors.* (If you are giving a counterexample, your counterexample should specify not only the flow network G and the flow f, but also the sequence of augmenting paths used in the Ford-Fulkerson execution that computes f.)

Solutions

(2.a)

Let a valid flow vector be F. Let a simple path from s-t be P such that all edges e on that path P have f(e) > 0. Let the minimum cost edge along the simple path be the bottleneck edge and its flow value be b(P).

Algorithm

Phase 1

In one iteration of the algorithm we will find a simple path P from source to sink such that edges on that path have a positive flow. Then we will subtract b(P) from all edges on that path. Record the path P and its bottleneck b(P). We will keep on performing the steps above until there are no such paths.

Phase 2

Then pick any edge with a positive flow remaining and run DFS start from any node of that edge to find a cycle. Traverse only the edges with positive flow. We will prove the claim that a cycle will always exist below. Now find the bottleneck of this cycle b(C) and subtract this value from all edges of this cycle. Record each cycle C and b(C). Continue performing this operation unless there is no edge left with a positive flow.

For every such path P, create a path vector pv(P) and let its weight be b(P).
For every such cycle C, create a cycle vector cv(P) and let its weight be b(C).

The flow F can be expressed as

F = $\sum_{allP}$ b(P) * pv(P) + $\sum_{allC}$ b(C) * cv(P)

Proof of correctness

Lemma 1. *After each iteration of phase 1, each node $v \neq s,t$ of the graph has $f^{in}(v) = f^{out}(v)$*

Proof. This claim is true before phase 1 since f is a valid flow. Consider any iteration of phase 1. When we subtract the bottleneck flow from any path, the we subtract the same amount entering the node and the same amount exiting any node since P is a simple path. Hence the lemma will hold after every iteration for all nodes except s,t. □

Lemma 2. *After phase 1, any edge with positive flow is part of a cycle*

Proof. When we end phase 1, there is no path from s-t. From lemma 1, we can conclude that all the nodes which part of a positive flow after phase 1 has $f^{in}(v) = f^{out}(v)$. Now when we perform DFS, suppose we start on node x. At the first step suppose we go to node y. Because we know that there is some flow entering y, there would be an equal flow leaving y. Suppose we go to some node z after that. We can use the same argument on z and every other node that the DFS traverses after that. Since there are a finite number of nodes in the graph, the DFS traversal will eventually come back to x since we already have some flow leaving it. Hence we will always have a cycle. □

Lemma 3. *After each iteration of phase 2, each node $v \neq s,t$ of the graph has $f^{in}(v) = f^{out}(v)$*

Proof. From lemma 1 we know that this property holds before beginning of phase 2. From lemma 2 we know that each iteration of phase 2 results in subtracting a bottleneck flow from a cycle. Since we are removing the same amount entering a node and leaving the node and the path is a cycle, this property will remain true after every iteration. □

Let us first prove that phase 1 and phase 2 take finite number of steps. In each iteration of phase 1, we make the flow value of one edge as zero. This means after at most 'm' iterations all edges should have zero flow and phase 1 would terminate. Similarly in phase 2, we are making flow value of some edge zero in each iteration. Hence it would execute at most 'm' times.

From lemma 2 we know that whenever there is a positive flow edge, its part of a cycle. Hence when the algorithm terminates, we would have found all the cycles. Also we have all the simple

paths from phase 1. Also once we subtract a flow, we never add it again. So when we will add all the paths and all the cycles weighted by their bottleneck flows, these will sum up to the flow itself.

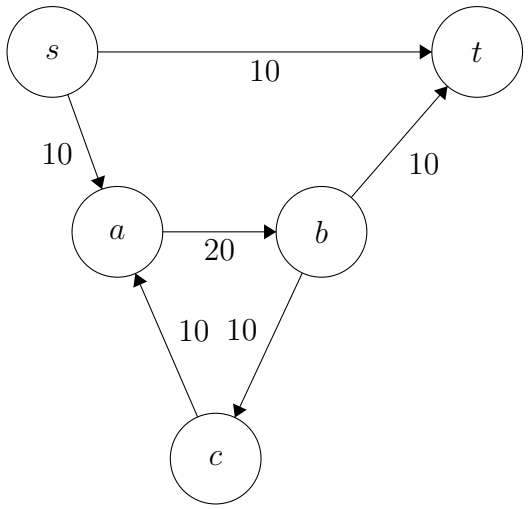
Running Time

Each step of phase 1 takes atmost O(m) time. And as proved above we can have atmost 'm' iterations. Hence phase one takes at most $O(m^2)$ time.

Similarly each step of the phase 2 takes $O(m)$ time and we can have at most 'm' such iterations. Hence phase two takes at most $O(m^2)$ time.

Hence the total running time is $O(m^2)$.

(2.b)



Let G be defined as follows
V = {s,a,b,c,t}

Capacities :

1. s-a = 10
2. a-b = 20
3. b-c =10
4. c-a = 10
5. b-t = 10
6. s-t = 10

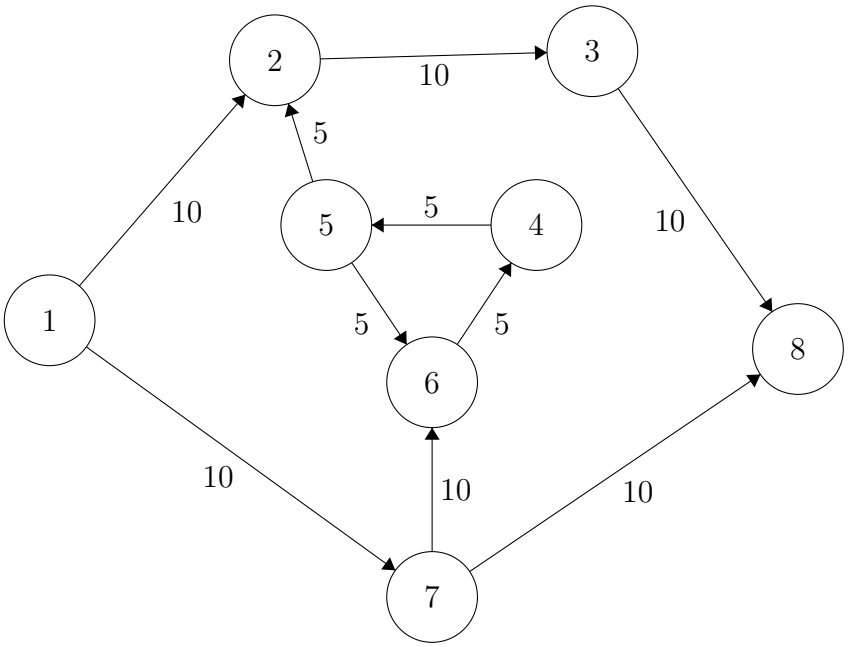
Flow vector f = [10 , 20 ,10 ,10 , 10, 10].

This cannot be represented as a sum of path vectors since there is no simple path which constitutes edges b-c and c-a. This is because the only path that b-c and c-a can be part of would have a cycle.

(2.c)

The statement is false. This is because it is possible that Ford Fulkerson during any augmentation adds flow to some cycle in G and then remove the flow from edges which connect that cycle to the remaining graph during the augment process. This will result in case where the flow f output by FF cannot be represented as only a non-negative weighted sum of path vectors.

Here is a counterexample -



Here is graph G and the number on the edges represent the maximum capacity of the edges of the network flow graph.

Here the augmented paths that are added to the graph G

1. (1 - 7 - 6 - 4 - 5 - 2 - 3 - 8) - Added flow 5
2. (1 - 2 - 6 - 7 - 8) - Added flow 5
3. (1 - 2 - 3 - 8) - Added flow 5
4. (1 - 7 - 8) - Added flow 5

FF stops after this since there are no simple paths in the residual graph since both the edges (1,2) and (1,7) from the source 1 have been saturated. The final flow on each edge is :

1. 1-2 : 10
2. 2-3 : 10

3. 3-8 : 10
4. 1-7 : 10
5. 7-8 : 10
6. 6-4 : 5
7. 4-5 : 5
8. 5-6 : 5
9. 5-2 : 0
10. 7-6 : 0

This flow vector cannot be represented as a non negative weighted sum of path vectors since the edge (5-6) can only be included in a s-t path which has a cycle.