Ritika Kumari
A20414073

HOMEWORK - 5

Problem 1:

Soln: let $G = (V, E, c, s, t)$ be a flow network with integer capacities, and f be a feasible flow with integer values. Suppose there is an edge e with head s such that $f(e) = 1$

If $f(e) = 1$ then there must exists a cycle with edge e assuming that each edge has flow, set to 1. So, if we will reduce the edge-flow by 1 in cycle, then reducing $f(e)$ from 1 to 0 will not affect the value of the flow.

Therefore, we have to prove that if the graph $G(V, E, c, s, t)$ be a flow network with integer capacities & f be a feasible flow such that $f(e) = 1$, then there exists $f'(e) = 0$ if and only if there is no flow b/w u to v (edge e)

We can prove this by using flow conservation

Algorithm:

① lets assume a graph $G(V, E, c, s, t)$ having flow $f(e) = 1$.

② let $M \in m$ such that s is flow-connected to m.

(3) Apply the flow-conservation to vertices in $(V-M)$ and generate the equation.

(4.) Now, partition V into M and $(V-M)$.

(5) Apply the flow-conservation again and generate the equation.

(6) Combine the equations generated in step 3 and step 5 and Check for below:

(7) if two vertices $(u, v) \in$ same partitions

(8) then $\sum f(u, v) = \sum f(v, u)$

(9) else $\sum f'(u, v) = 0$ // u & v are in different partition. ∴ it is not connected.

Example of above alogorithm:

let's $M \in m$ such that S is flow-connected to m.

Apply the flow conservation to vertices in $(V-M)$, we will get

$$\sum_{\substack{n \in V-M \\ l \in V}} f(l, n) = \sum_{\substack{n \in V-M \\ l \in V}} f(n, l) \quad \text{—— (I)}$$

Now, partition V into M and (V−M), we get :

$$\sum_{\substack{n \in V-M \\ l \in V-M}} f(l,n) + \sum_{\substack{n \in V-M \\ l \in M}} f(l,n) = \sum_{\substack{n \in V-M \\ l \in V-M}} f(n,l) + \sum_{\substack{n \in V-M \\ l \in M}} f(n,l) \quad \text{--- (II)}$$

Also, we know that

$$\sum_{\substack{n \in V-M \\ l \in V-M}} f(l,n) = \sum_{\substack{n \in V-M \\ l \in V-M}} f(n,l) \quad \text{// n \& l are in same partition}$$

&

$$\sum_{\substack{n \in V-M \\ l \in M}} f(l,n) = 0 \quad \text{// n \& l are in different partion}$$

∴ Computing the eq^n I & eq^n II considering the two facts mentioned above we will get

$$\sum_{\substack{n \in V-M \\ l \in M}} f(n,l) = 0$$

Thus, we can say that $f'(e) = 0$ if and only if they ~~are~~ are in different partitions which implies that they are not connected. ie, there is no flow b/w ~~u~~ $v \in V - Y$ and $s \in Y \Rightarrow f'(v, s) = 0$

<u>Time - complexity:</u> $O(|V|) + O(|E|)$

$\downarrow$ for step (4) ie, for partitioning

$\downarrow$ for step (7) ie, for each eadge

$= O(|V| + |E|)$

<u>Proof of Correctness:</u>

We are claiming that $f'(e) = 0$ ($f'(v, s)$) if s is not flow connected to v.

<u>Proof by contradiction:</u> let's assume that

$f'(v, s) > 0$ which means s is flow connected to v through some path p.

let p' be the path which forms a cycle that has positive flow on each edge.

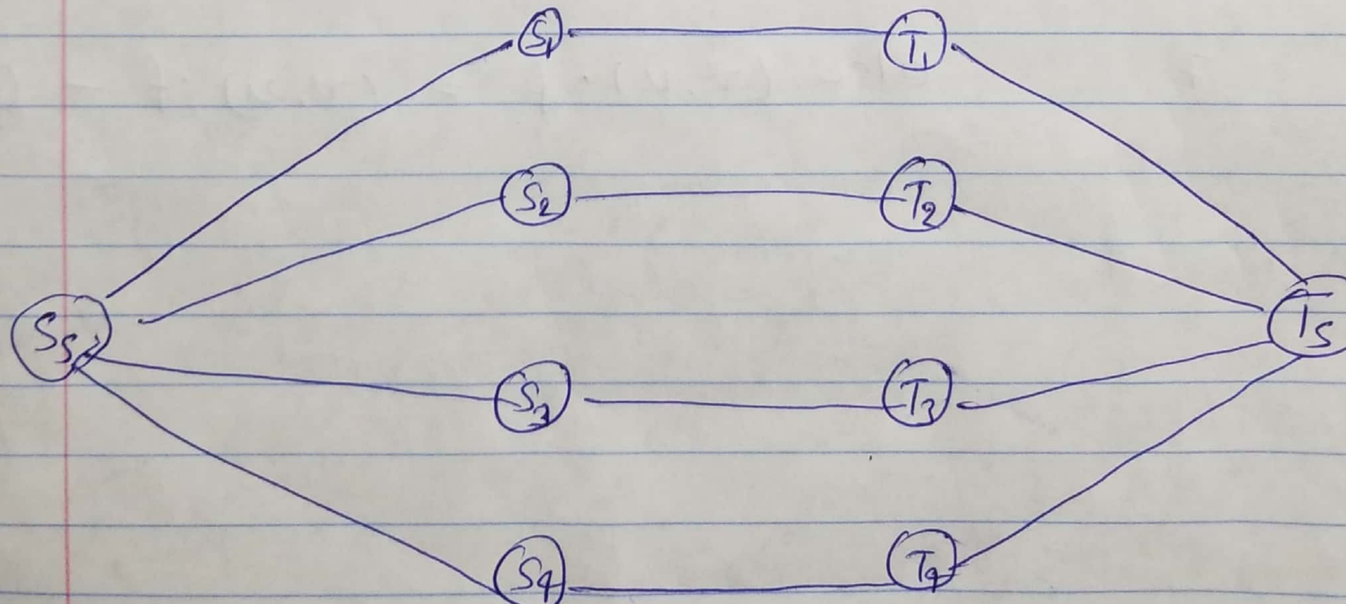
$s \xrightarrow{p} v \longrightarrow s$
$|-----p'-----|$

The flow on each edge of p' will be atleast 1, since we are assuming all edge capacities are integers. Therefore if we substract 1 from each edge then we will get the flow f' which will still satisfy the network flow properties and it has same value as $|f|$.

$\therefore$ We can say that $f'(v, s) = f(v, s) - 1$
$= 0$

Problem 2:

Soln: Consider a graph with multiple sources and sinks. We will add a single super source S_s and a single super sink T_s to convert them into a network graph like below.



FORD-FULKERSON (G, s, t)

1. for each edge $(u, v) \in G.E$
2. $(u, v).f = 0$
3. while there exists a path p from s to t in the residual network G_f
4. $c_f(p) = \min \{ c_f(u, v) : (u, v) \text{ is in } p \}$
   where $c_f(p)$ is the residual capacity having augmented flow f along p.
5. for each edge (u, v) in p
6. if $(u, v).f = (u, v).f + c_f(p)$
7. else $(v, u).f = (v, u).f - c_f(p)$

satisfied or not.

Algorithm:

SuperSourceSink ($G, V, E, c, s, t, P_s, q_t$)

1. Form a new flow network G' similar to original network G in addition of single Super Source S_S and Super Sink T_S.
2. for each $u \in S$
3. $(S_S, u) \cdot c = L1$ where $L1$ = +ve constant.
4. for each $v \in T$
5. $(v, T_S) \cdot c = L2$ where $L2$ = +ve constant

ConstraintCheck ($G, V, E, c, s, t, P_s, q_t$)

1. for each $v \in V - \{S \cup T\}$
2. $f[v] = 0$ // Initializing an array with 0
3. for each edge $(u, v) \in G.V$
4. if $f(e) \leq c(e)$ // checking the capacity constraint
5. if $u \in S$ & $v \in V - \{u\}$
6. $f_{out} = f_{out} + (u, v) \cdot f$
7. else if $v \in S$ & $u \in (V - v)$
8. $f_{in} = f_{in} + (u, v) \cdot f$ ↳ get this by using Ford-Fulkerson

```
elseif u ∈ T & v ∈ V − {u}
        fout = fout + (u,v) f
elseif u ∈ V−{v} & v ∈ T
        fin = fin + (u,v).f
elseif u ∈ V − {S ∪ T} && v ∈ V − {S ∪ T}
        f[u] = f[u] − (u, v)·f
        f[v] = f[v] + (u, v)·f
else
    return NO
Initialize Result = TRUE ;          // Checking the Flow constraint
for each u ∈ V − {S ∪ T}
    if f[u] != 0
        Result = False
if Result = TRUE && fout − fin = Ps &&
                   fin − fout = qt
    return YES
else
    return NO
```

get this (u,v)f by using Ford-Fulkerson Algo

Time-complexity : $\underbrace{O(|E| + |V|)}_{\text{for Constraint Check}} + \underbrace{O(E|f^*|)}_{\text{for Ford-Fulkerson}}$

$= O(E|f^*|)$

Proof of correctness:

We are considering a graph with multiple sources and sinks. We will add a single super source S_s and a single super sink T_s. Now, we will set the capacities of edges (S_s, u) to large +ve constraints say L1 ~~to provi~~ which will be ~~enough~~ enough to provide the required flow to all the sources. Likewise, we set the capacities of edges (v, T_s) to some +ve constants say L2 to provide the required flow to all the sinks. After this, we are calling a function "ConstraintCheck" which will check the conditions of flow conservation, capacity constraints and also it will compare p_s and q_t. Calling Ford-Fulkerson Function to compute the flow b/w vertices u & v. We start by checking the "ConstraintCheck" function if the flow will not exceed the capacity. If it will exceed the capacity, then it is not a valid flow we can say. If it is a valid flow, we check if the current considered edge is coming from or going to a source and then increasing the total in or total out flow accordingly. We will do the same thing for the sinks. At the same time, we will check, if ~~bothe~~ both the vertices does not belongs to source neither sinks, in that case we are decreasing the total flow value of start vertex and increasing the total flow value of the end vertex. At the end, we will check for the flow of each vertex. If it is 0, then check the given equations are satisfied or not.

Problem 3

Sol^n:

The edge connectivity of an undirected multigraph is the minimum number, k of edges that must be removed to disconnect the graph.

For this, we can define a flow network G_{uv} with $s = u$, $t = v$ assuming all edges capacities set to 1 because the no. of edges of G crossing a cut equals the capacity of the cut in G_{uv}.

To determine the edge connectivity of an undirected multigraph $G = (V, E)$, we are running the FORD-FULKERSON algorithm to find the maximum-flow.

Algorithm:

```
1.  EDGE-CONNECTIVITY(G)
2.     k = ∞
3.     select any vertex u ∈ G.V
4.       for each vertex v ∈ G.V - {u}
5.           FORD-FULKERSON(G, u, v)      // It will return the max^m flow
6.           k = min(k, |f[u, v]|)
7.     return k
```

where $|f[u,v]|$ is the max^m flow obtained from FORD-FULKERSON Algorithm.

```
FORD-FULKERSON (G, s, t)
 for each edge (u, v) ∈ G.E
      (u, v).f = 0
 while there exists a path p from s to t
                  in the residual network G_f
      c_f(P) = min { c_f(u, v) : (u, v) is in p }
          where c_f(P) is residual capacity
          having augmented flow f along P.
      for each edge (u, v) in p
           if (u, v).f = (u, v).f + c_f(P)
           else (v, u).f = (v, u).f - c_f(P)
```

<u>Time-complexity:</u>

Time-complexity for FORD-FULKERSON algorithm $= O(E|f^*|)$ where f^* is the max^m flow in transformed network

$\therefore$ Total time-complexity $= O|V| * O(E|f^*|)$

$= \cancel{O(|V| + E|f^*|)}$

$= O(|V| \cdot |E| \cdot |f^*|)$

Proof of correctness:

We are claiming that edge-connectivity k is the minimum of maximum flow of all networks in the Graph, $G_{(u,v)}$.

$$k = \min \{ |f_{uv}| \}$$

Proof by Contradiction: let us assume that instead of removing k edges, we remove only (k-1) edges from G.
Since the minimum of maximum flow from u to v is k, therefore any cut has the capacity ~~at~~ atleast k which means that atleast k edges cross the cut.
This proves that we need to remove atleast k edges which is contradicting the claim.
Now, let's assume that we remove (k+1) edges from $(G_{u,v})$. Since all edge capacities we assume set to be 1, exactly $|f_{uv}|$ ie, k edges cross this cut. If these edges (k) removed, then only, the graph will become disconnected. So, we can see that, for graph to be disconnected, we are not waiting to remove till (k+1) edges. Hence, again this is ~~contrading~~ contradicting our claim made initially.

Therefore, it is proved that the above algorithm is correct to determine the edge-connectivity.