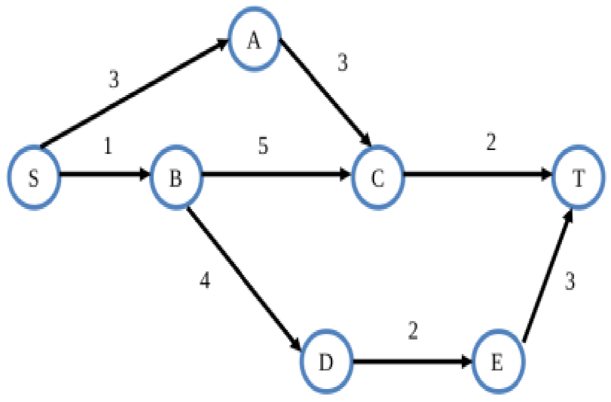
# CSCI 570 : HW-04 (Spring 2021)

April 4, 2021

## Question 1

**You are given the following graph G. Each edge is labeled with the capacity of that edge.**



**a. Find a max-flow in G using the Ford-Fulkerson algorithm. Draw the residual graph $G_f$ corresponding to the max flow. You do not need to show all intermediate steps. (10 pts)**

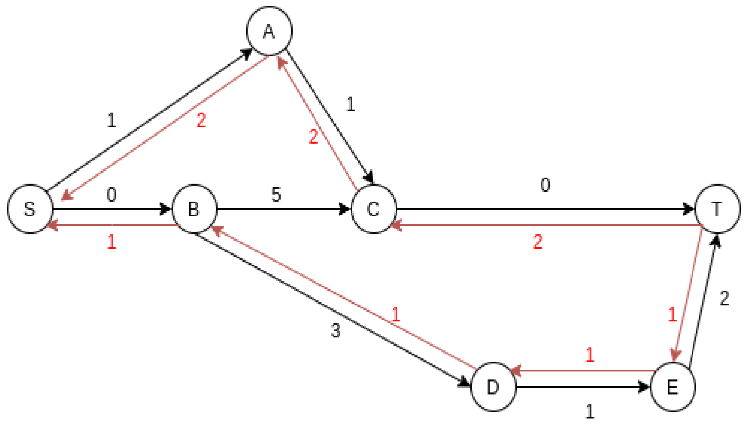
**b. Find the max-flow value and a min-cut.**

**Answer**

**Part a**

We will follow the following (Ford-Fulkerson) algorithm to find the max-flow and get a residual graph.

```
Start with f(u,v)=0 and G_f = G
while exists an augmenting path in G_f.
    find bottleneck
    augment the flow along this path
    update the residual graph G_f.
```



Residual Graph

### Part b

The max-flow is that is found using Ford-Fulkerson algorithm is 3.

To find the min-cut of the graph G, we will run a BFS from S in residual graph $G_f$. We will not traverse the edges that are saturated. Here we will get nodes {S,A,C} or {B, D, E, T} as min-cuts.

## Question 2

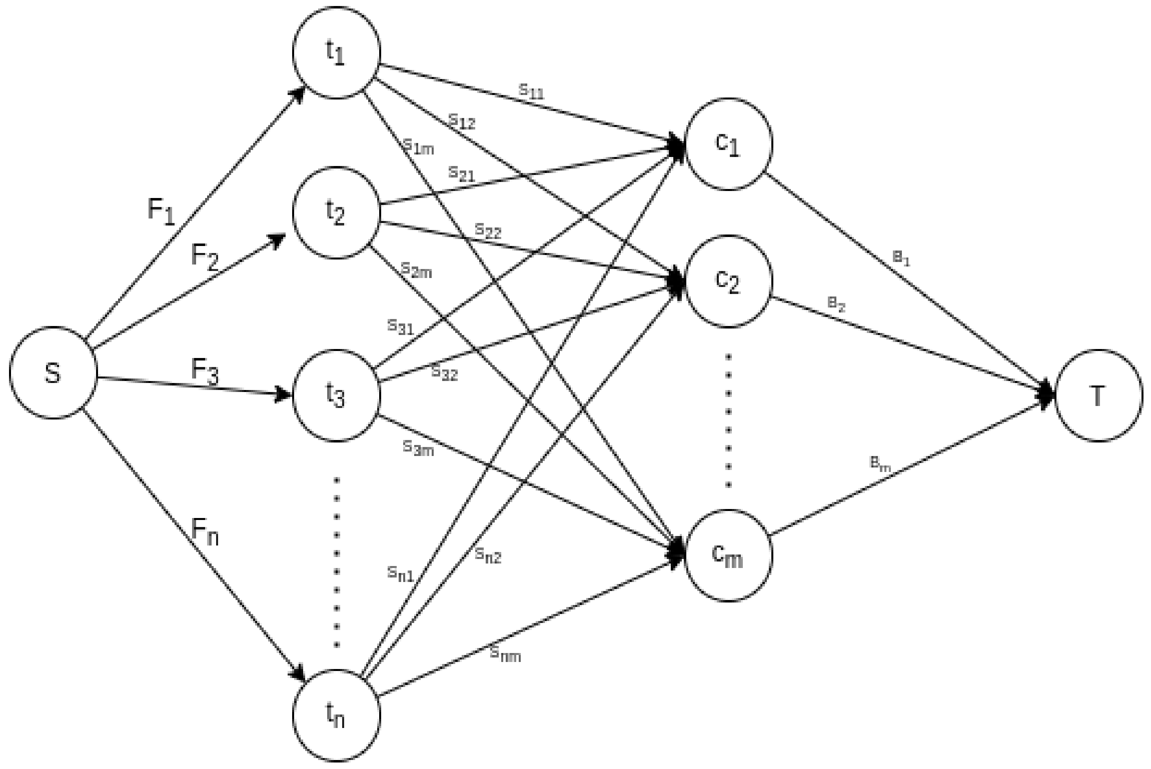
**A group of traders are leaving Switzerland, and need to convert their Francs into various international currencies. There are n traders $t_1, t_2, ..., t_n$ and m currencies $c_1, c_2, ..., c_m$. Trader $t_k$ has $F_k$ Francs to convert. For each currency $c_j$, the bank can convert at most $B_j$ Franks to $c_j$ . Trader $t_k$ is willing to trade as much as $S_{kj}$ of his Francs for currency $c_j$ .(For example, a trader with 1000 Francs might be willing to convert up to 200 of his Francs for USD, up to 500 of his Francs for Japanese's Yen, and up to 200 of his Francs for Euros). Assuming that all traders give their requests to the bank at the same time, design an algorithm that the bank can use to satisfy the requests (if it is possible).**

### Answer

We will solve this problem by designing a flow network and getting maximum flow by running Ford-Fulkerson algorithm.

Flow Network

We will construct a bipartite graph where 1st partition will represent all traders $(t_1, t_2, ..., t_n)$ and 2nd partition will represent the currencies $(c_1, c_2, ...c_m)$. Lets connect each trader with every currency where edge will represent the amount of Francs $S_{kj}$ that trader $t_k$ wants to convert to $c_j$. We will also add an edge from source node to every trader and from every currency to a sink. Edge between source node and trader will represent the frank $F_k$ that trader $t_k$ wants to convert. Edge between currency $c_j$ and sink node represent the maximum amount $B_j$ that bank can convert.



Feasible Solution

The problem has a feasible solution if and only if after running Ford-Fulkerson algorithm, maximum flow to T is sum of all edge capacity ($\sum_{k=1}^{k=n}\sum_{j=1}^{j=m} S_{kj}$).

Amount of franks available with trader $t_k$, $F_k \geq \sum_{j=1}^{m} S_{kj}$ (trader should have atleast same amount of franks that he wants to convert ) for the feasibility of solution. Also the maximum amount for currency $c_j$ that bank convert $B_j \geq \sum_{k=1}^{n} S_{kj}$ for the feasibility of solution.

Proof:

* Proof 1. If maximum flow to sink node T is $\sum S_{kj}$ then all the requests of traders have been satisfied by the bank. Max-flow of $\sum S_{kj}$ to T means that every edge between $t_k$ and $c_j$ are saturated. This in turn means that every request of the trader for currency $c_j$ has been satisfied by the bank.

* Proof 2. If all requests are being satisfied then maximum flow to sink node is $\sum S_{kj}$. Each trader $t_k$ want to convert $S_{kj}$ amount of Franks to currency $c_j$. To satisfy all the requests for currency (j), the flow from currency nodes $c_j$ to sink node should be $\sum_{k=1}^{n} S_{kj}$. Finally, the total flow will $\sum_{k=1}^{n}\sum_{j=1}^{m} S_{kj}$ if all the requests from all the traders for all currency are satisfied.

Time Complexity:

Time of complexity of Ford-Fulkerson algorithm is $O(|f| \cdot (E + V))$. Here $E = n + n \cdot m + m$ and $V = 2 + n + m$. Thus, time complexity for this solution is $O(|f| \cdot (n \cdot m))$

## Question 3

**After getting vaccinated, students will be able to return to in-person classes next semester. There will be n students, $s_1, s_2, ..., s_n$, return to in-person classes, and there will be k in-person classes. Each in-person class consists of several students and a**

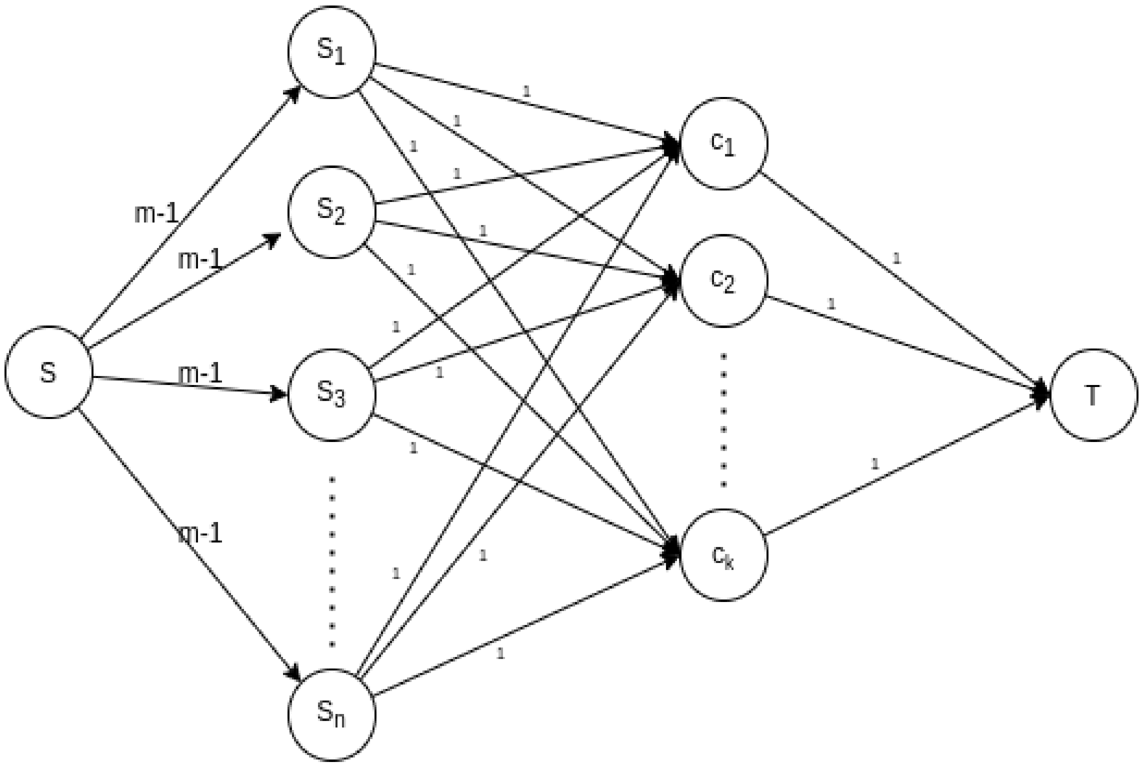
**student can be enrolled in more than one in-person class. We need to select one student from each in-person class and the maximum times a student is selected should be less than m. Design an algorithm that decides if such a selection exists, or not.**

**Answer**

We will solve this problem by designing a flow network and getting maximum flow by running Ford-Fulkerson algorithm.

Flow Network

We will construct a bipartite graph where 1$^{st}$ partition will represent all students ($s_1, s_2, s_3 ... s_n$) and 2$^{nd}$ partition will represent all classes ($c_1, c_2, ..., c_k$). We will connect all students with every classes using edges of capacity 1. We will also include source and sink nodes. We will connect source node with every student node using edges having capacity $m - 1$. We will connect all class nodes with sink using edges of capacity 1.



Feasible Solution

The problem has a feasible solution if only if after running Ford-Fulkerson algorithm, the maximum flow to sink node ($T$) is $k$.

Proof

* Proof 1: If the maximum flow to sink node T is k, then each student must be selected from each class and one student cannot be selected more than m-1 times.

If the max-flow to sink is k, then a student is selected from every k classes because the capacity of edge connecting each class and sink is 1 and for max-flow to be k all of these edges must be saturated. Also, each student will not be selected more than m-1 times because incoming flow from source to student node is m-1.

* Proof 2: If a student is selected form every class and no student is selected more that m-1 times then max-flow to sink is k.

Each student cannot be selected from more than m-1 classes because the maximum flow from source to each student is m-1. Since we are selecting one student from each class and since there are k classes then the max-flow to sink node is k.

Time Complexity:

Time of complexity of Ford-Fulkerson algorithm is $O(|f| \cdot (E + V))$. Here, $|f| = k$, $E = n + n \cdot k + k$ and $V = 2 + n + k$. Thus, time complexity for this solution is $O(n \cdot k^2)$)

## Question 4

**USC Admissions Center needs your help in planning paths for Campus tours given to prospective students or interested groups. Let USC campus be modeled as a weighted, directed graph G containing locations V connected by one-way roads E. On a busy day, let k be the number of campus tours that have to be done at the same time. It is required that the paths of campus tours do not use the same roads. Let the tour have k starting locations $A = \{a_1, a_2, ..., a_k\} \subset V$. From the starting locations the groups are taken by a guide on a path through G to some ending location in $B = \{b_1, b_2, ..., b_k\} \subset V$. Your goal is to find a path for each group $i$ from the starting location, $a_i$, to any ending location $b_j$ such that no two paths share any edges, and no two groups end in the same location $b_j$.**

**a. Design an algorithm to find k paths $a_i \longrightarrow b_j$ that start and end at different vertices and such that they do not share any edges. (15 pts)**

**b. Modify your algorithm to find k paths $a_i \longrightarrow b_j$ , that start and end in different locations and such that they share neither vertices nor edges. (10 pts)**

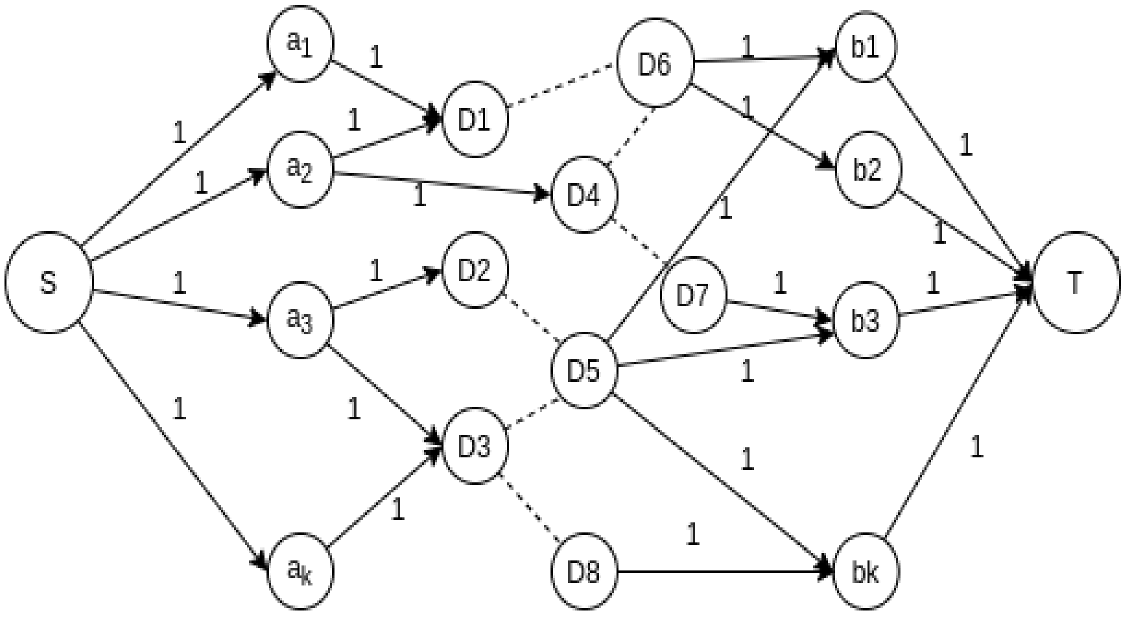
**Answer**

**Part a.**

We will solve this problem by designing a flow network and getting maximum flow by running Ford-Fulkerson algorithm.

Flow Network

We will connect a source vertex to all starting locations ($A$) using edge of capacity 1. We will also connect a sink vertex from all ending location using edge of capacity 1. All the paths connecting different location (including starting and ending locations) inside the campus have capacity 1.



Feasible Solution

The problem has a feasible solution if only if after running Ford-Fulkerson algorithm, the maximum flow to sink node ($T$) is $k$.

```
Algorithm to find k paths:

Let G be our Flow Network Graph.
Start with f(u,v)=0 and G_f = G
while exists an augmenting path in G_f from source to sink.
    paths <- augmenting path without source and sink.
    find bottleneck
    augment the flow along this path
    update the residual graph G_f.

paths will give list of required k paths
```

Proof

* Proof 1. If there are k paths that start and end at different vertices such that they do not share any edges then the max-flow to sink node is k.

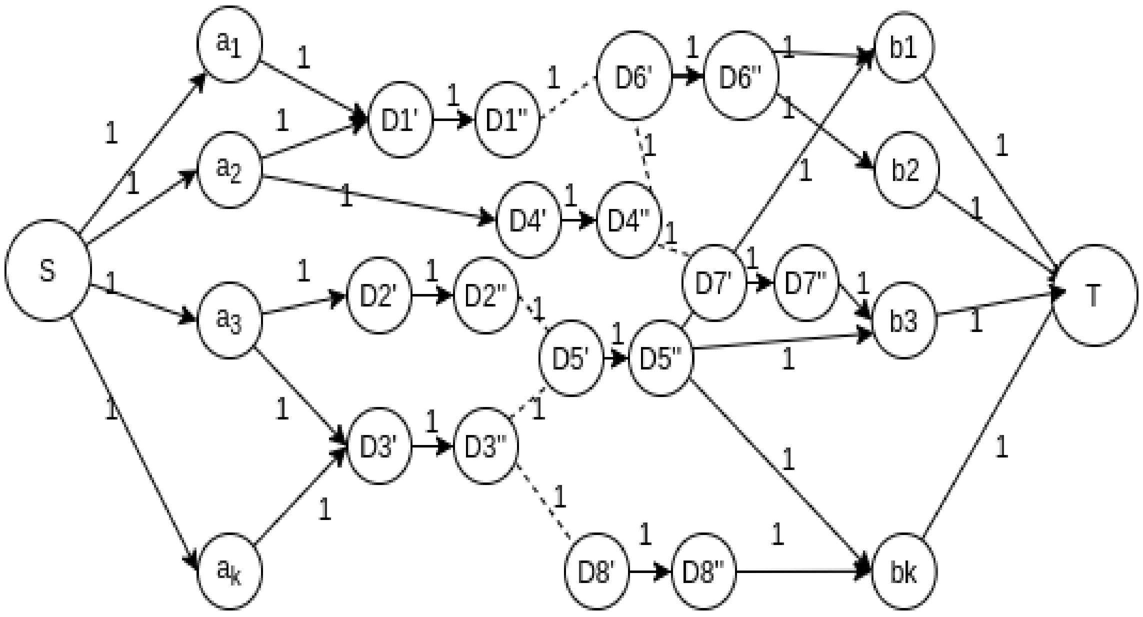
Since from source to starting locations the capacity is 1, then the algorithm will not select same starting location twice. Similarly, it will not select same ending location twice because capacity from ending location to sink is 1. The algorithm could also not select any internal edges twice (involving $D_i$) because the capacity for them is also 1. Hence if there are k paths satisfying these constraints then all edges involving source/starting locations and sink/ending locations will be saturated which implies max-flow to sink node is k.

* Proof 2. If the max-flow to sink node is k, then there will be k paths that start and end at different vertices such that the paths do not share any edges.

Since max-flow to sink node is k, then all edges connecting ending location to sink must be saturated means all paths have distinct ending locations. This implies we have k paths from source to sink. All k paths will also have distinct edges because every edge has capacity of 1 and can't be selected more than once. Similarly, algorithm will have distinct starting locations because from source to starting locations the capacity is 1.

**Part b.**

We will modify our network flow graph, here we will represent each department node (internal nodes) using 2 nodes which are connected by edge of capacity 1 (*Di* will be represented by *Di′* and *Di″*). All incoming nodes to *Di* will be now incoming to *Di′* and outgoing edge from *Di* will be now outgoing from *Di″*.



<u>Feasible Solution</u>

The problem has a feasible solution if only if after running Ford-Fulkerson algorithm, the maximum flow to sink node ($T$) is $k$.

```
Algorithm to find k paths:

Let G be our Flow Network Graph.
Start with f(u,v)=0 and G_f = G
while exists an augmenting path in G_f from source to sink.
    paths <- augmenting path without source/sink & replacing Di’ & Di’’ with Di.
    find bottleneck
    augment the flow along this path
    update the residual graph G_f.

paths will give list of required k paths
```

<u>Proof</u>

* Proof 1: If there are k paths that start and end at different vertices such that they do not share any edges/vertices then the max-flow to sink node is k.

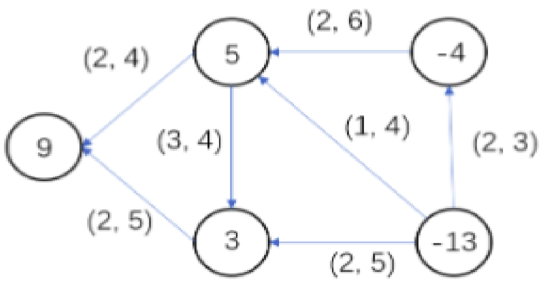
Since from source to starting locations the max capacity is 1, then the algorithm will not select same starting location twice. Similarly the it will not select same ending location twice because max capacity from ending location to sink is 1. The algorithm could also not select any internal vertices $Di'$ twice because to select $Di'$, $Di''$ must be selected and since capacity between them is 1, so they cannot appear more than once. Hence if there are k paths satisfying these constraints then all edges involving source/starting locations and sink/ending location will be saturated which implies max-flow to sink node is k.

* Proof 2: If the max-flow to sink node is k, then there will be k paths that start and end at different vertices such that the paths do not share any edges/vertices.

Since max-flow to sink node is k, then all edges connecting ending location to sink must be saturated means all paths have distinct ending locations. This implies we have k paths from source to sink. All k paths will also select distinct internal edges/vertices because to select a internal vertex both $Di'$ and $Di''$ must be selected but since capacity between them is only 1 hence $Di$ can be selected at most 1 time. Similarly, algorithm will select distinct starting location because from source to starting locations the capacity is 1.

## Question 5

**In the network below, the demand values are shown on vertices (supply value if negative). Lower bounds on flow and edge capacities are shown as (lower bound, capacity) for each edge. Determine if there is a feasible circulation in this graph. You need to show all your steps**



**a. Turn the circulation with lower bounds problem into a circulation problem without lower bounds. (8 pts)**

**b. Turn the circulation with demands problem into the maximum flow problem. (8 pts)**

**c. Does a feasible circulation exist? Explain your answer. (4 pts)**

**Answer**

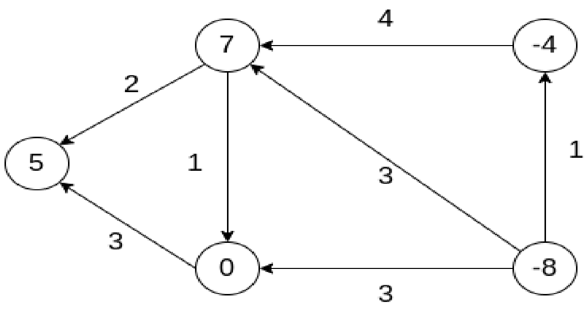
**Part a.**

We will turn the given circulation graph into graph without lower bounds by updating the demands values of each node as follows.

$$d'(v) = d(v) - f_{in}(v) + f_{out}(v)$$

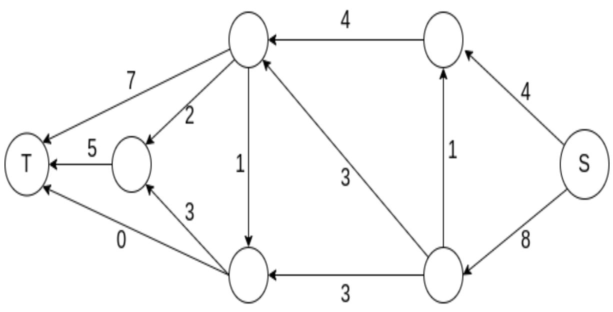
We will update the capacity of each edge as

$$Capacity(edge) = UpperBound(edge) - LowerBound(edge)$$



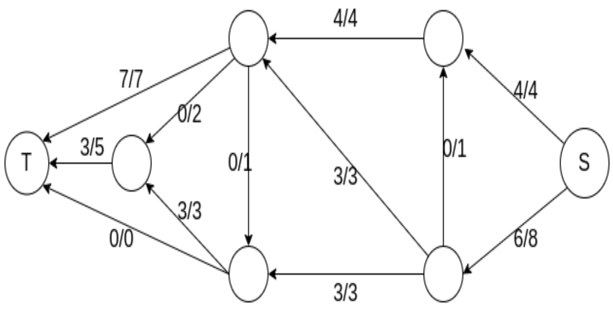
**Part b.**

We will remove the demand from each node by adding source and sink nodes. Source node will be connected by all nodes with negative demands using edge of capacity that is equal to absolute value of demand on that node. Sink node will be connected by all nodes with positive demands using edge of capacity that is equal to demand on that node.



**Part c.**

We will use Ford-Fulkerson algorithm to check feasibility of circulation. The circulation is feasible if and only if the edges connecting sink nodes are saturated or flow into the sink node is same as capacity of edges out of source.



Here, after completion of Ford-Fulkerson algorithm we got the max-flow at sink node as 10 and the edges connecting sink node are not saturated. And we also dont have any augmenting path from source to sink to make those edges saturated. This means that there has been insufficient supply. Hence there is no feasible solution to this circulation problem.