CSCI 570 - Analysis of Algorithms

Assignment 4

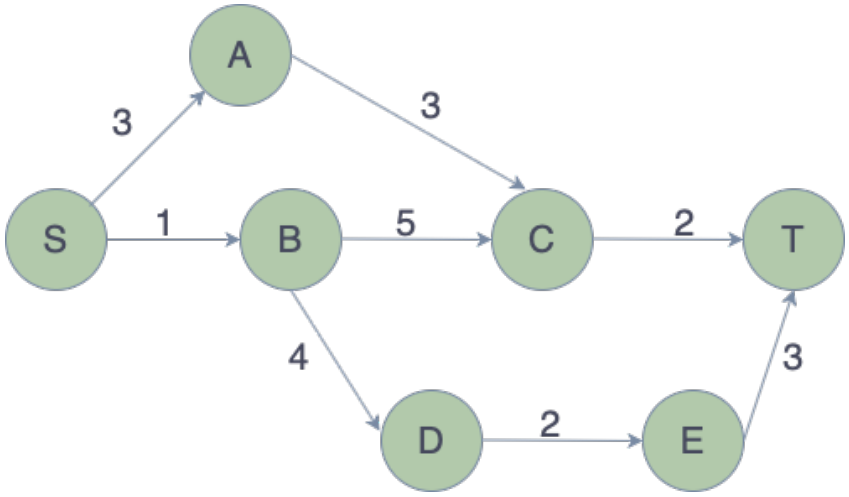
Harshitha Kurra

---

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.**

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

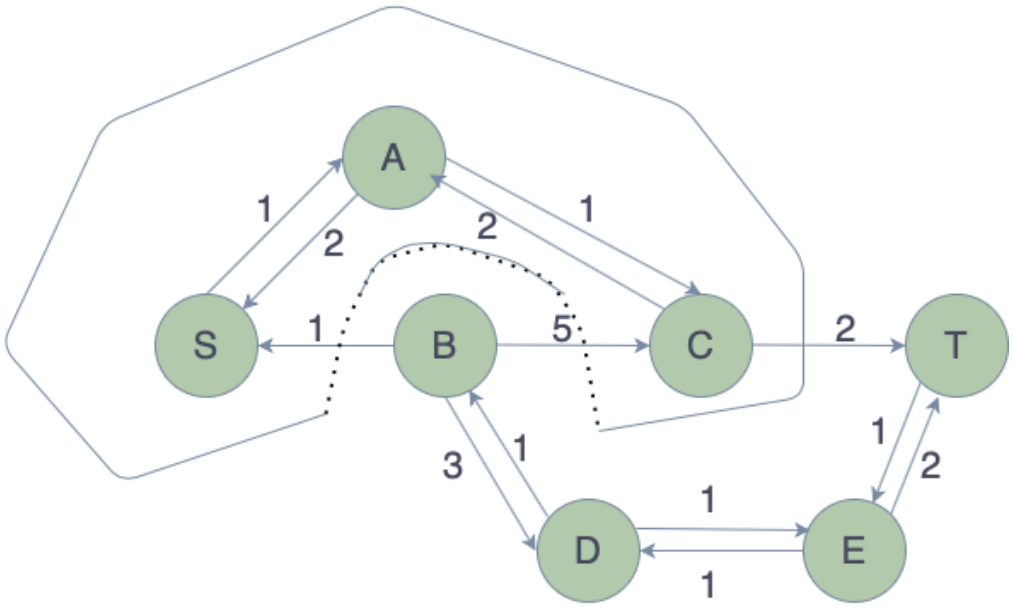


Solution:

A. Residual Graph is drawn below.

B. Max Flow is 3 and the min cut is S-A-C | B-D-E-T

Residual Graph:

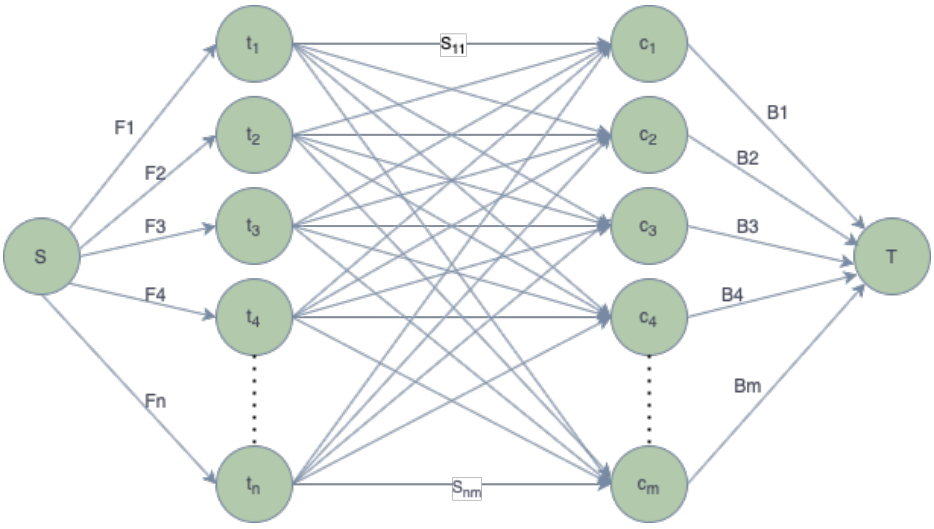


---

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).

Solution:

Running a Network Flow will give us the following graph:



Connect all the Traders $t_1, t_2, t_3, \ldots . t_n$ to the source S with edge weights as the Franks that each of the traders have to exchange/convert i.e $F_1, F_2, F_3, \ldots . F_n$.

As given in the Question, the trader t_k is willing to trade as much as S_{kj} for currency C_j. So we give the edge weights as $S_{11}, S_{12}, \ldots . S_{nm}$ connecting $t_1, t_2, t_3, \ldots . t_n$ to $C_1, C_2, C_3, \ldots . C_m$.

Finally, we connect currencies to the Target T with Edge weights as Banks limit, given as $B_1, B_2, B_3, \ldots . B_m$ as the total volume as exchange for currencies $C_1, C_2, C_3, \ldots . C_m$.

Claim:

"The given situation will have a solution only if the max flow through the graph is $\sum_k F_k$*."*

Proof:

The bank can only satisfy all the requests if the value for each of the currency is not exceeded. Otherwise it's a failure.

Assume that we have a solution. All the customers have been able to convert their currency. This means all the edges from source to nodes have been saturated with flow. Hence we have a max flow from source to sink with sum of flows as $\sum_k F_k$

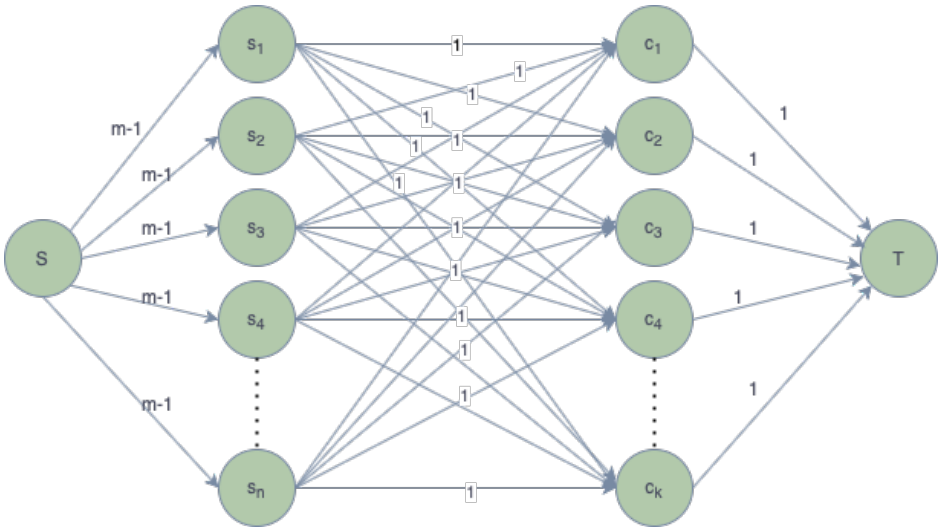
Now let's assume we have a max flow. We will prove that this only happens if we have a solution. When we have a max flow, it means either our edges from source to trader nodes are saturated with flow of $\sum_k F_k$ and this means all the traders were able to exchange their Franks.

Hence proved.

---

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.**

Solution:



Graph explanation: Connect all the students to the source with the edge weight as m-1 as the maximum number of students should be less than m. And then connect the students($S_1, S_2, S_3, \ldots S_n$) to the classes($C_1, C_2, C_3, \ldots . C_m$) each with the edge weight as 1. Finally we connect the classes to the target T with the edge weight as 1 each. Now, we run the network flow algorithm on the given graph from source to target will give us the solution for assigning students to the classes according to the selection process given in the question.

Claim:

"We will have a solution only if the max flow is k I.e; the total number of classes."

Proof:

Assuming we have a solution, I.e; a student has been selected from each class. This would mean all the edges from class nodes to the target are saturated since their weight is 1. Since the graph is no longer connected we have a max flow and the value of this is the sum of weights of edges from class nodes to target, i.e.,

1+1+1+….1(k times)

= k

Now let's assume we have a max flow of k. The min cut of the graph we built is between class nodes to target(sink), their edges are now saturated with flow of 1 on each edge. This means we selected one student from each class with taking all the constraints into consideration. Therefore, we have a solution.

Hence Proved.

---

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 = { $a_1, a_2, \ldots, a_k$ } $\subset$ V . From the starting locations the groups are taken by a guide on a path through G to some ending location in B = { $b_1, b_2, \ldots, b_k$ } $\subset$ 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.

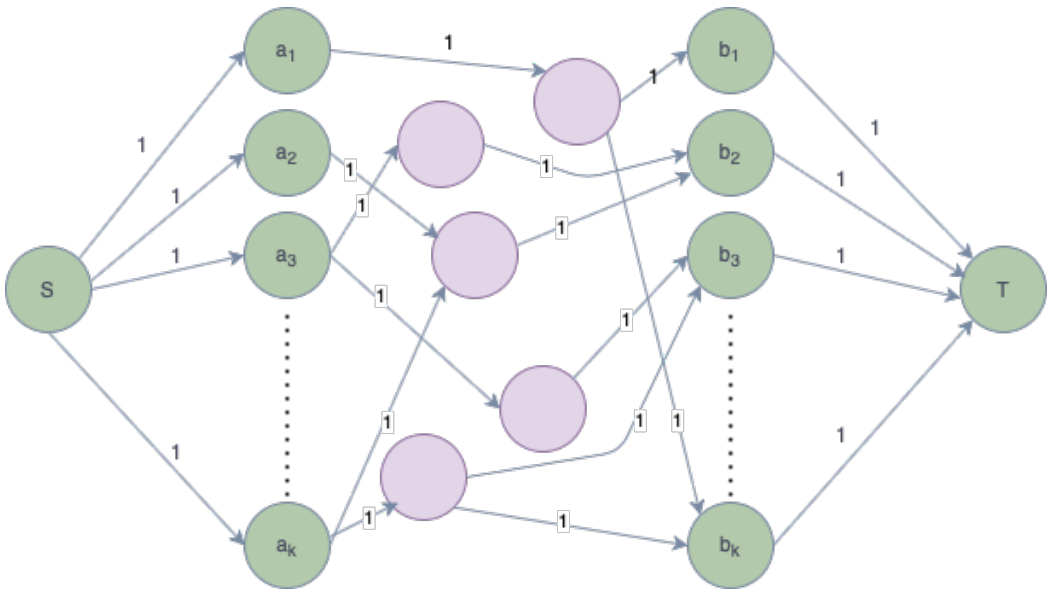
A. Design an algorithm to find k paths $a_i \rightarrow b_j$ that start and end at different vertices and such that they do not share any edges.

B. Modify your algorithm to find k paths $a_i \rightarrow b_j$, that start and end in different locations and such that they share neither vertices nor edges.

Solution:

A. Algorithm:

- Create a network flow G with Source S, starting locations, ending locations, Target(sink) T.
  We connect S to all the starting locations with edge weight 1 and all destination vertices to the sink node T.
  Connect starting nodes to intermediate nodes and intermediate nodes to destination nodes as shows below.



- Run Ford-Fulkerson on any other network flow algorithm called G^1 and let there be a max flow of f

- Run DFS on the max flow graph G^1 from S to T. We won't be considering the edges where there is no flow. i.e; zero.
  Remove the edges of the DFS path and the destination node part of the output path. Then repeat the same k times. We get k output paths.

Claim:

"The solution exists if the max flow is k."

Proof:

1. Assume there exists a valid solution i.e; there is no repetition in paths. Since edge weights each is 1 between intermediated nodes, once the flow is sent, it is saturated. Hence, as we have an assignment of k distinct paths, all end nodes will be connected to the sink making all those 1-weighted edges saturated. Therefore, giving us a max flow of k.

2. Now let's assume we have a max flow of k. Which implies, all end nodes $(b_1, b_2, \ldots b_k)$ are connected to the common sink node T making all these edges saturated. The end nodes can only be saturated if there are distinct-edge paths from S to T. We already guaranteed this by having edge weights as inside the graph. Having independent paths from start nodes is the solution we wanted.

Therefore, the problem has a solution.

B. Modification of Algorithm:

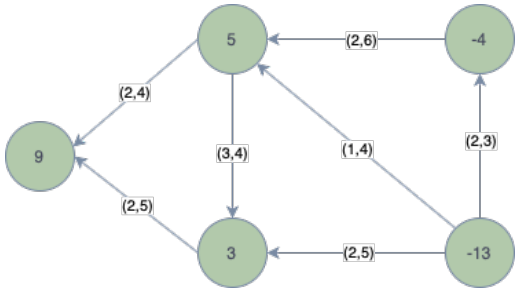
- Split each node v in the original Graph G into two nodes v_{in} and v_{out} with a directed edge between them. Don't split S and T. Connect v_{in} and v_{out} with edge weight 1. All incoming edges to v will now go to v_{in} and all the outgoing nodes from v will now go from v_{out}. This is the modification to the algorithm.
  Now we follow the same steps as A.

  - Run Folk-Fulkerson on any other network flow algorithm called G^1 and let there be a max flow of f

- Run DFS on the max flow graph G^1 from S to T. We won't be considering the edges where there is no flow. i.e; zero.
  Remove the edges of the DFS path and the destination node part of the output path. Then repeat the same k times. We get k output paths.
- Claim here is same as A.

---

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.



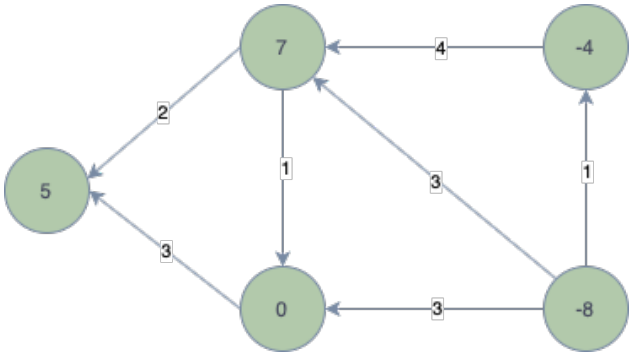
A. Turn the circulation with lower bounds problem into a circulation problem without lower bounds.

B. Turn the circulation with demands problem into the maximum flow problem.

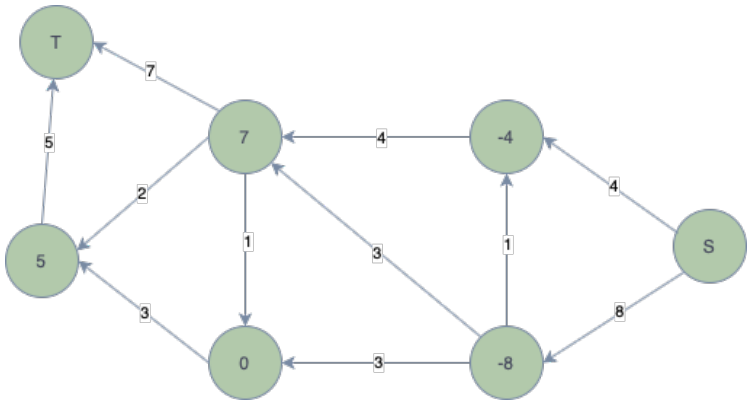
C. Does a feasible circulation exist? Explain your answer.

Solution:

A.



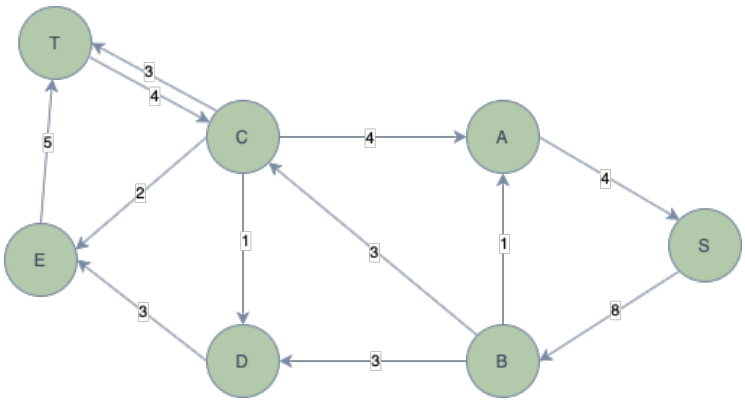
B.



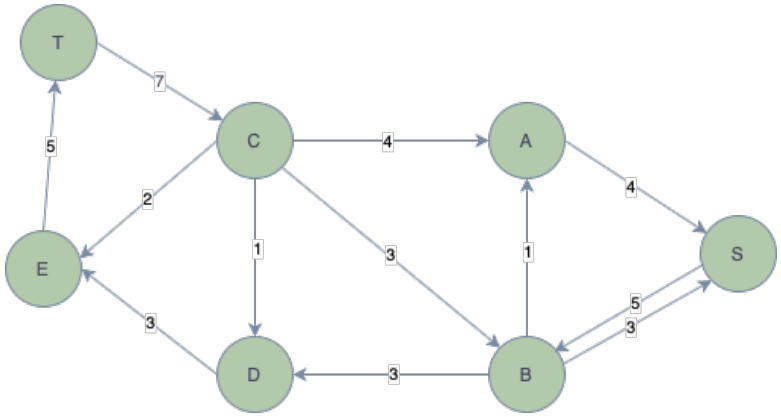
C.

From B, we can say that a feasible solution will exist if the flow is 12. By running Ford-Fulkerson (Step-by-step is shown below), we get that the max flow is 10. So, we say that the feasible circulation does not exist.

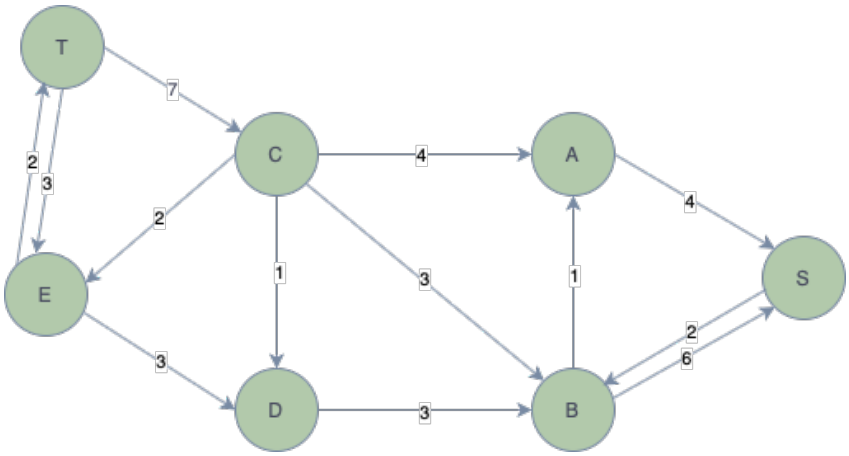
—> Bottleneck & flow = 4 (Traversing order: S-A-T)



—> Bottleneck and flow = 3 (Traverse order: S - E - B - T)



—> Bottleneck and flow = 3 (Traverse order: S - E - D - C - T)



The max flow is 10.

There will not be any feasible circulation as the max flow is less than required.