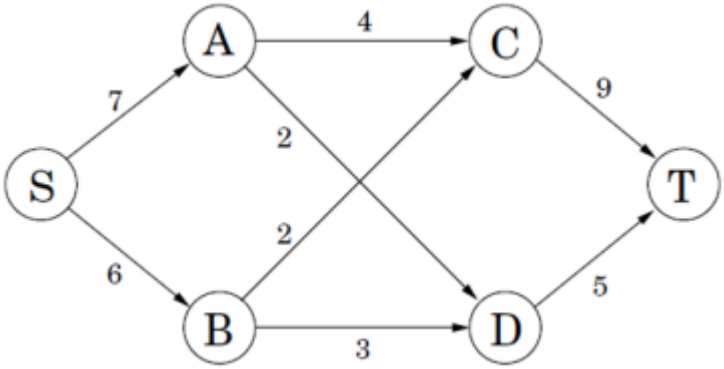
dnabar@usc.edu

Deven Nabar
USC ID: 7229446568

CSCI570 Homework 4

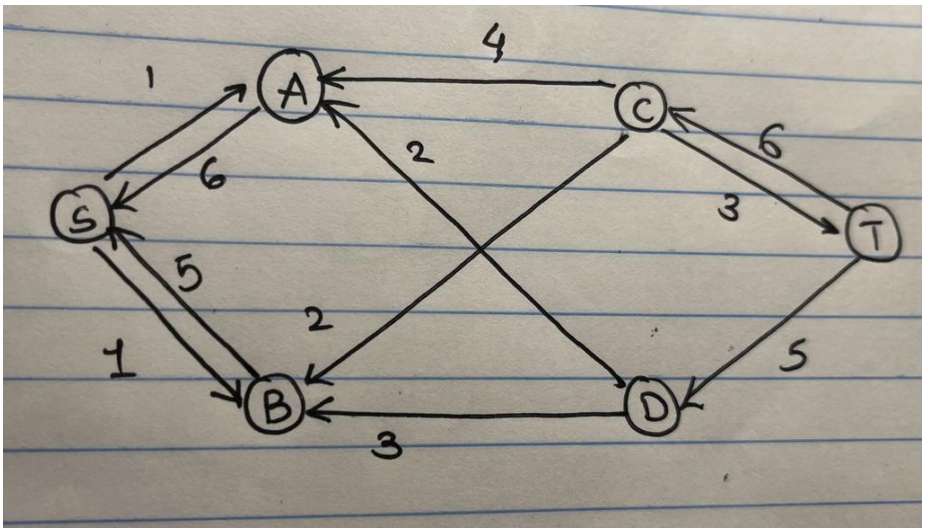
1. SOLUTION:

We have been given the following graph G:



a) We will run Ford-Fulkerson Algorithm of the residual graph G_f.

- We start with a zero flow and $G_f = G$. We find an augmenting path S-A-C-T with bottleneck 4. We push 4 units of flow and augment the flow along that path and update the residual graph. Total Flow is 4
- Next, we find another augmenting path S-B-D-T and push 3 units flow as it is the bottleneck value. We update the residual graph. Total flow now is 4+3 = 7
- Next, we pick path S-A-D-T and push 2 units of flow and update the residual graph. Total flow now is 7+2 = 9
- Next, we pick S-B-C-T and push 2 units of flow through it as it is the bottleneck value and update the residual graph. Total flow is 9+2=11
- The residual graph after this iteration is as follows:



- As the residual graph is disconnected i.e., there is no path available from S-T, therefore we have found the maximum flow. Also, the residual graph above is the final iteration of the algorithm.

dnabar@usc.edu

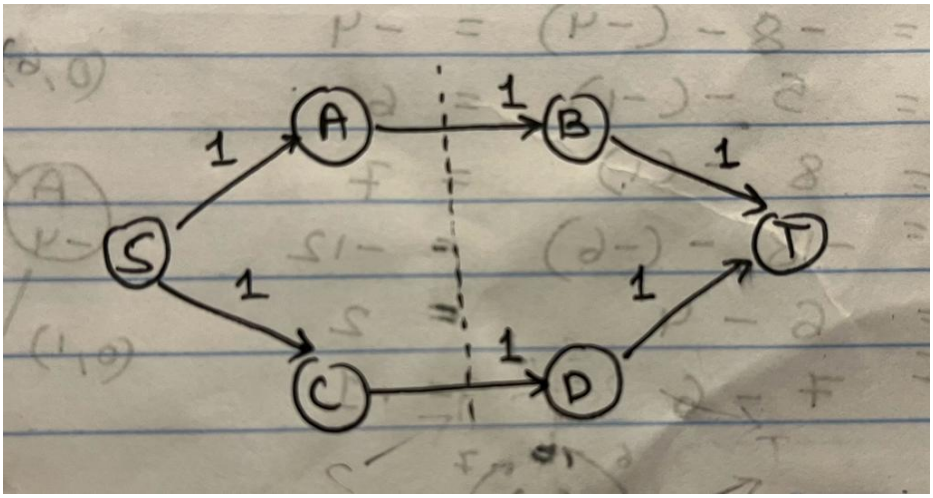
Deven Nabar
USC ID: 7229446568

b) Max-flow and Min-cut:

- The max- flow value was found using Ford Fulkerson algorithm and it comes out to be 11 units of flow.
- The min-cut in the graph is the cut that divides the graph into two Sets containing the following vertices:
  Set 1 = {S, A, B}
  Set 2 = {C, D, T}

c) Proof by Counter-example:

- We can consider a graph containing vertices A, B, C, D along with a source and a sink. Let all the edges be of capacity 1 and the max-flow for the network flow is 2.



- Now, the min-cut can be as shown above along the edges A-B and C-D.
- If we increase the capacity of one of these edges to 2, i.e., capacity of A-B was increased to 2. This change would not result in the max-flow being increased as the bottleneck is still occurring at edges S-A and B-T.
- Hence, we have shown that increasing the capacity of an edge that belongs to a min cut will not always result in increasing the maximum flow.

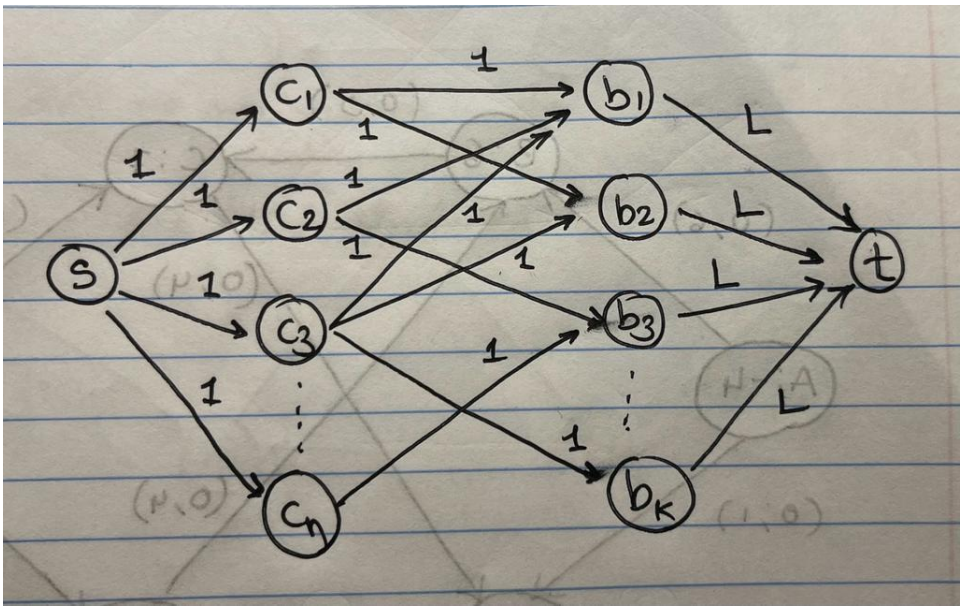
dnabar@usc.edu

Deven Nabar
USC ID: 7229446568

2. SOLUTION: Reduction to network flow

a) Construction of flow network

- We start by setting the problem as a bipartite graph problem. In this graph one partition is a set of vertices c_i representing all n clients
- Another partition is a set of k base stations b_j.
- We then connect each client to base station/s that are in their respective range R by directed edges of capacity 1.
- We now extend the bipartite graph to a network flow.
- We add source s and connect it to every client vertex c_i by an edge (s, c_i) of capacity 1.
- We now add target t and for every base station b_j , we add an edge (b_j , t) of capacity L which is the maximum load that the base station can handle (the maximum number of clients)



b) Claim:

- The original problem has a valid solution (given the positions of a set of clients and a set of base stations as well as the range and load parameters, a valid assignment is indeed possible) if and only if the constructed network has a max-flow of value n which is the total number of clients.

dnabar@usc.edu

Deven Nabar
USC ID: 7229446568

c) Proof:

==>)

- Assume that there is a solution. It means that every client has been connected to a base station.
- We can therefore push a flow of 1 to each client. (There are n clients)
- On the edges between clients and base stations we assign a flow of 1.
- Since all clients need to be connected to a base station and base stations cannot accommodate more than L members, on the edges between the base stations and the sink/target we assign a flow value equal to the number of clients connected to the base station($\leq$L).
- This must be possible as we have a valid assignment. (The max flow is n)

Conversely, <==)

- Assume that there exists a max-flow of value n.
- This means that each client vertex will get a flow of 1 as there are n clients in total.
- Due to capacity constraints (each edge (c_i, b_j) has a capacity of 1), no two base stations can have the same client connecting to them.
- No base stations are overloaded with more than L clients because of the capacity condition on them.

- Lastly, we get a suitable assignment by running a network flow algorithm and picking edges (c_i, b_j) with a unit flow.

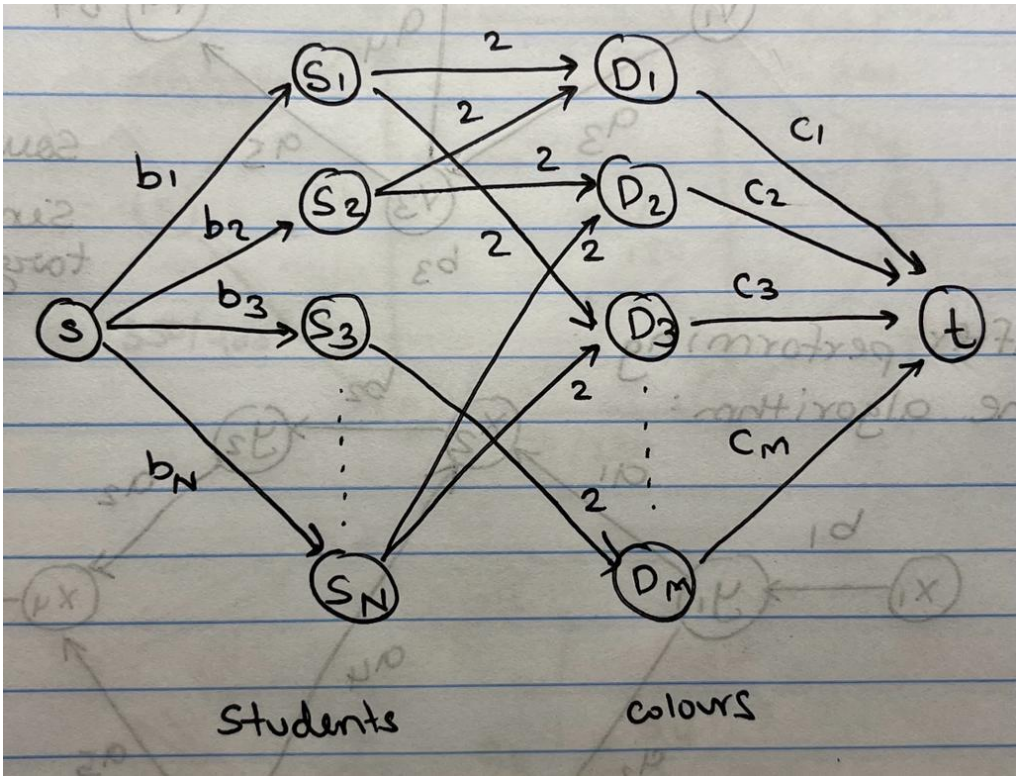
dnabar@usc.edu

Deven Nabar
USC ID: 7229446568

3. SOLUTION: Reduction to network flow

a) Construction of network flow

- We start by setting the problem as a bipartite graph problem. In this graph one partition is a set of vertices S_i representing all N students (S_1, S_2, S_3, … S_N)
- Another partition is a set of M unique colours D_j (D_1, D_2, D_3,…, D_M)
- We then connect each student to their favourite colour/s following the set F_i by directed edges of capacity 2 as one student can buy maximum 2 packets of the same colour.
- We now extend the bipartite graph to a network flow.
- We add source s and connect it to every student vertex S_i by an edge (s, S_i) of capacity b_i.
- We now add target t and for every colour D_j, we add an edge (D_j , t) of capacity c_j which is the number of packets left for each unique colour in the bookstore.



b) Claim:

- The original problem has a solution (a valid assignment is indeed possible, given the total number of packets a student wishes to buy, the total number of colour packets available and the set representing the favourite colours of each student) if and only if the constructed network has a max-flow of $\sum_{i=1}^{N} b_i = b_1 + b_2 + b_3 + \cdots + b_N$

dnabar@usc.edu

Deven Nabar
USC ID: 7229446568

c) Proof:

==>)

- Assume that there is a solution.
- This means that every student has their wish granted of buying b_i colour packets in total from the set of their favourite colours.
- So, we push a flow of b_i from the source s to every student.
- On the edges between the students and the colours, we assign a flow of at most 2. (0 or 1 or 2)
- Since, a student can buy a maximum of 2 colour packets of any unique colour we assign 2 on every 'student – colour' edge.
- On the edge between the colours and the target, we assign a flow value equal to the number of packets bought for each colour.
- This must be possible, since we have a valid assignment. (The max flow is the sum of all b_i)

Conversely, <==)

- Assume that there is a max-flow of value of $\sum_{i=1}^{N} b_i = b_1 + b_2 + b_3 + \cdots + b_N$.
- This means that each student vertex will get a flow of b_i.
- Due to capacity constraint set on each edge (S_i , D_j) of flow capacity 2, none of the students can have more than 2 colour packets of the same colour.
- Due to the capacity condition c_i from each colour vertex D_j to the target t, we see that a maximum of c_j packets can be bought per colour.

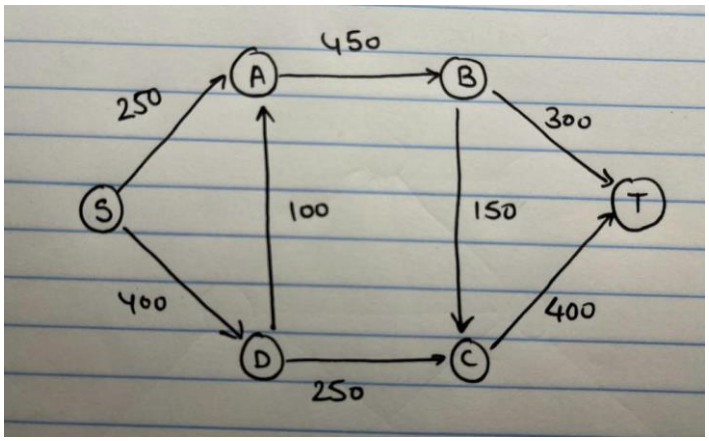
- Lastly, we get a buying arrangement by running network flow algorithm and picking edges (S_i, D_j) with flow equal to 2

dnabar@usc.edu

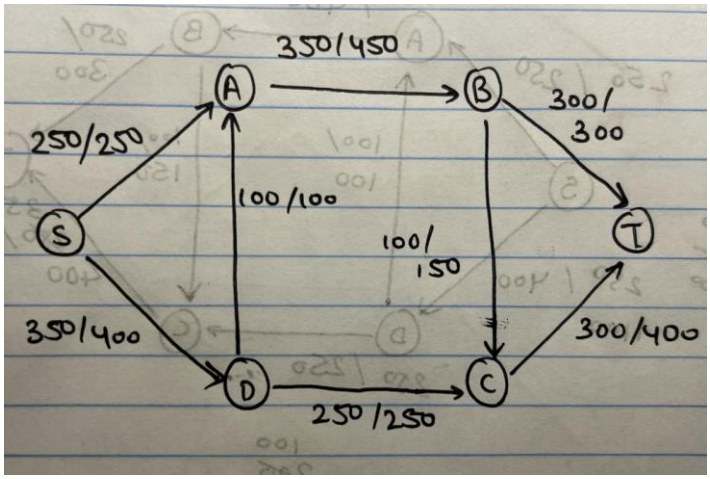
Deven Nabar
USC ID: 7229446568

4. SOLUTION:

- We can solve the given problem by treating it as a network flow problem with S and T as the Source and Target respectively and A, B, C, D as the vertices.
- The halls can be treated as edges with capacities equal to the number of people that it can support per hour. Along with this, exhibits can be considered as vertices.
- We get the following network flow:



- We can run Ford-Fulkerson Algorithm on the following graph and find the max-flow possible here. After running the algorithm, the original graph G with each edge labelled by flow/capacity is as follows:



- The max-flow is 250+350 = 600 units of flow.
- Hence the maximum number of visitors that can visit the museum is 600 visitors per hour.
- The museum wants to attract 6000 visitors and therefore it would require 6000/600 = 10 hrs to reach this target number
- If the museum opens at 8am on a specific day, the earliest it could close is 8am + 10hrs = 6pm.
- Hence, the earliest the museum can close to support 6000 visitors is 6pm.

dnabar@usc.edu

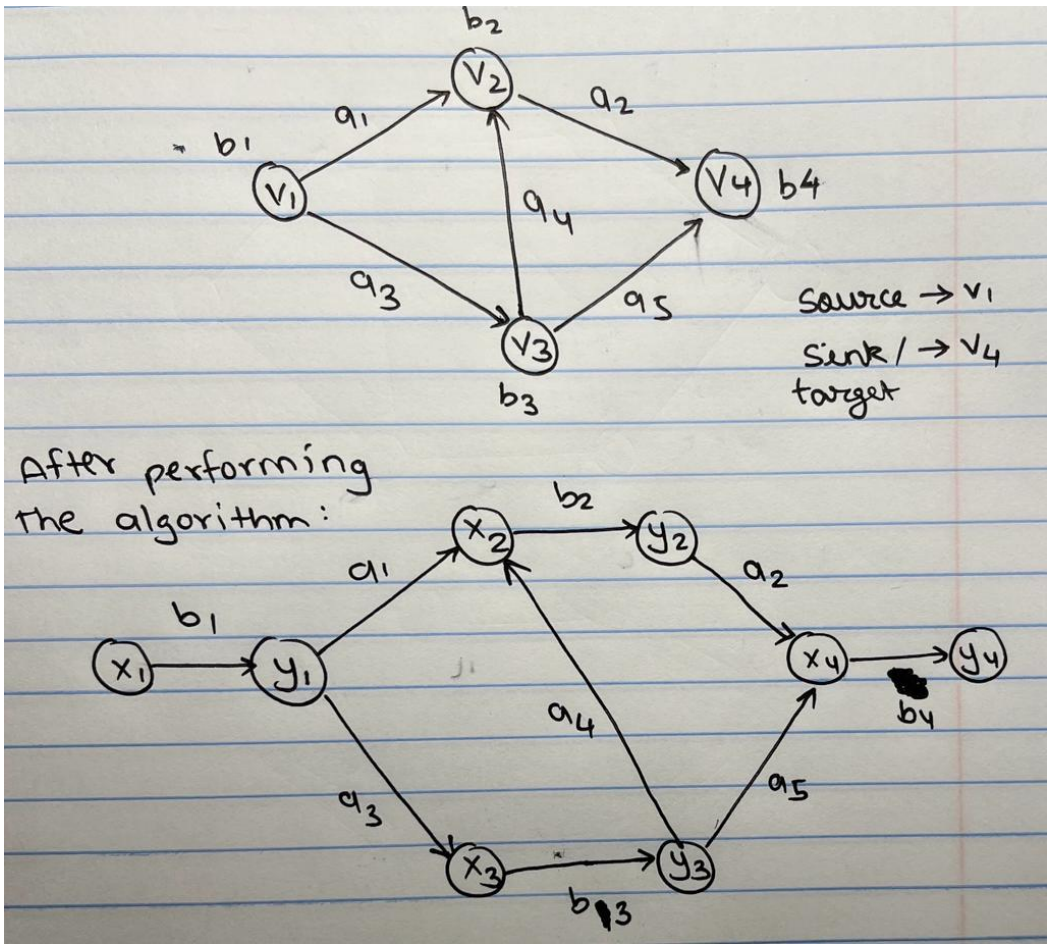
Deven Nabar
USC ID: 7229446568

5. SOLUTION:

We can solve this problem by following these steps:

- Each vertex v_i with a capacity constraint b_i needs to be converted into a directed edge with nodes (x_i, y_i) with flow capacity b_i in the new graph that would be constructed.
- All incoming edges to v_i need to be directed to x_i.
- All outgoing edges from v_i need to be directed out of y_i.
- In case of Source node, there will be no incoming edges to x_i associated with the source
- In case of the Target node, there will be no outgoing edges from vertex y_i associated with the target node.
- All other edges in the original graph remain unchanged in the new graph.
- This reduces the problem into a simple network flow problem where we need to find the maximum flow possible. This can be found easily using any of the max flow algorithms such as Fort-Fulkerson algorithm or Edmond Karp algorithm.

For example, we can use the above steps to find the max flow in the graph given below:



dnabar@usc.edu

Deven Nabar
USC ID: 7229446568

6. SOLUTION:

We can find the edge connectivity of an undirected graph by following the steps below:

- Set a variable maxflow = ∞
- Set any vertex node in the given undirected graph as the source s.
- Now, for every vertex v in the V – s:
    - Label the vertex v as Target t
    - All edges should have an edge capacity of 1 and direction of all edges should be away from s and towards t
    - Run Ford-Fulkerson Algorithm on this graph and find the max-flow value possible in the graph
    - If the max-flow value found is lesser than the max flow value previously stored, we update it.

Time Complexity:

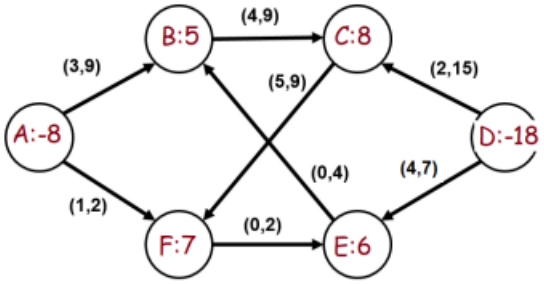
- The time complexity of Ford-Fulkerson algorithm is O(|f|(E+V)).
- Since each edge is of maximum capacity 1. In the worst case, the value of |f| is E.
- Since we are running the Ford- Fulkerson algorithm V times on the graph, the time complexity for the same would be $O(V*(E(E+V))) = O(VE^2 + V^2E) = O(VE^2)$
- As there are n vertices and m edges, the algorithm will run in $O(m^2n)$.

dnabar@usc.edu

Deven Nabar
USC ID: 7229446568

7. SOLUTION:

- We have been given the following network with demand values on the vertices.
- Along with this we have been given lower bounds and capacities of all edges



a)

- We first check the most necessary condition for a feasible circulation which is the fact that sum of demands must be equal to 0

$$Sum\ of\ demands\ = (-8) + 5 + 8 + (-18) + 6 + 7 = 0$$

- We now turn the circulation with lower bounds problem into circulation problem without lower bounds.
- We do this by pushing a flow of lower bound value $l(u, v)$ on each edge and compute excess $L(v) = f^{in}(v) - f^{out}(v)$ for each vertex v

  L(A) = 0 – (3+1) = - 4
  L(B) = (3+0) – 4 = - 1
  L(C) = (2+4) -5 = 1
  L(D) = 0 – (2+4) = - 6
  L(E) = 4 – 0 = 4
  L(F) = (1+5) – 0 = 6

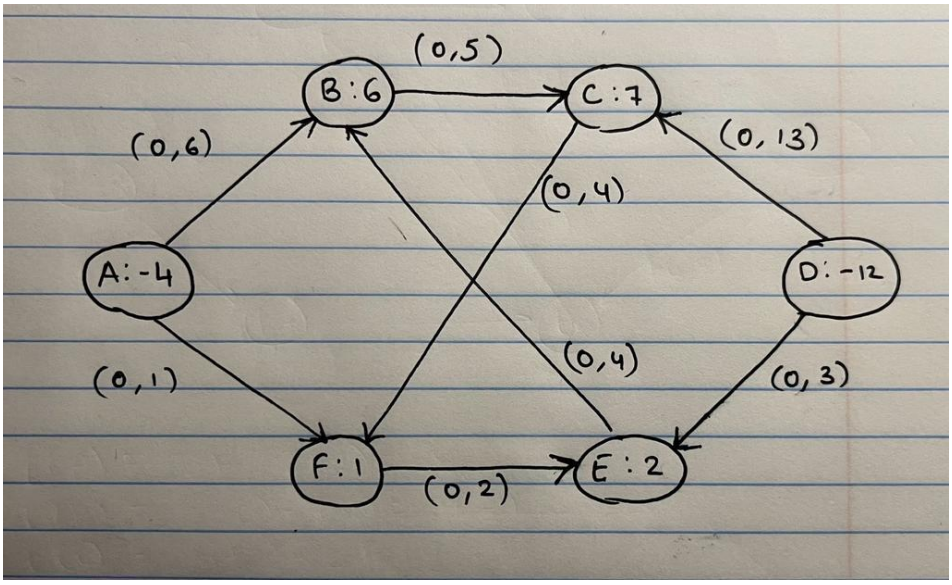
- We now recompute demands $d'(v) = d(v) - L(v)$, The new demands on each vertex are as follows:

  d' (A) = -8 – (- 4) = -4
  d' (B) = 5 – (-1) = 6
  d' (C) = 8 – (1) = 7
  d' (D) = - 18 – (- 6) = -12
  d' (E) = 6 – 4 = 2
  d' (F) = 7 – 6 = 1

- Now the original problem has been reduced to a circulation problem without lower bounds

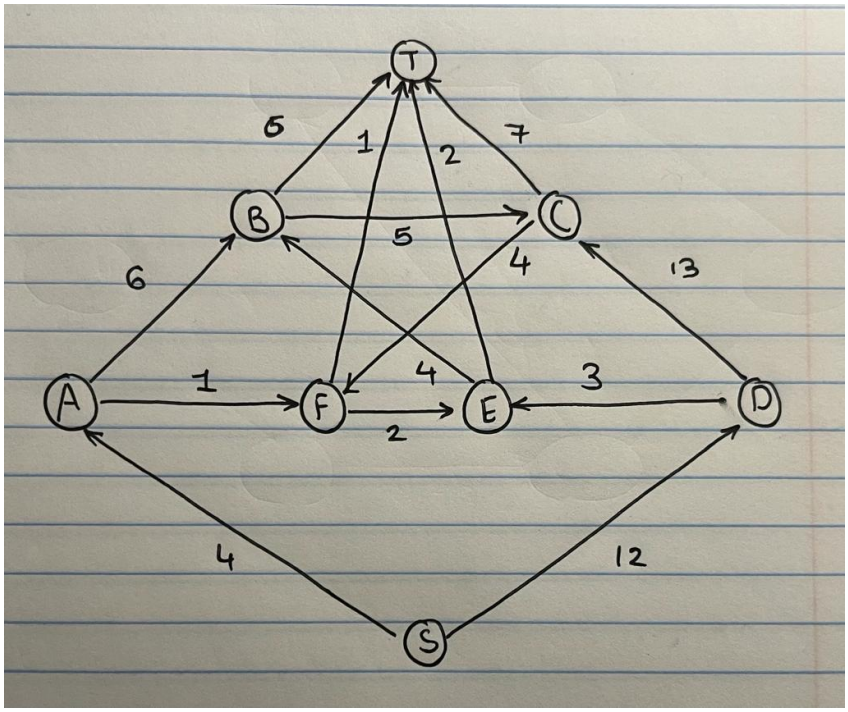
dnabar@usc.edu

Deven Nabar
USC ID: 7229446568



b)

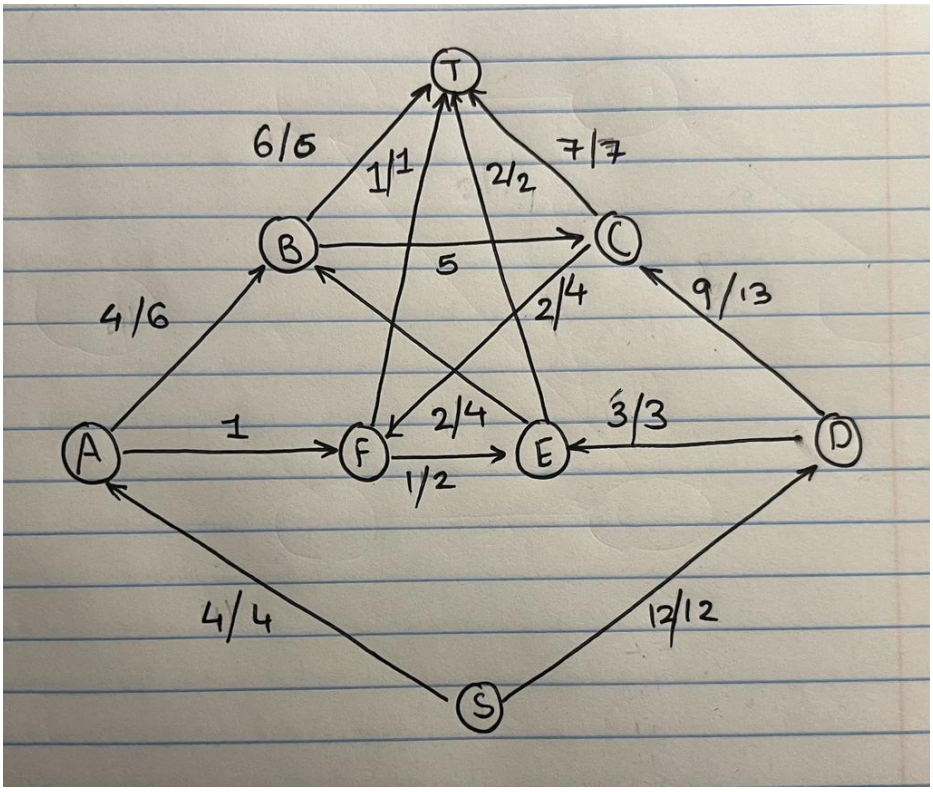
- We now construct a new graph by adding a source and a target vertex: S and T vertex
- We connect the Source with A and D with edges of capacity 4 and 12 and vertices B, C, E, F with Target with edges 6, 7, 2, 1 respectively. We get the following graph:



dnabar@usc.edu

Deven Nabar
USC ID: 7229446568

- Using Ford-Fulkerson, we can find the max flow for the above graph which has a value of 16 and saturates all the edges coming out of the Source S



c)

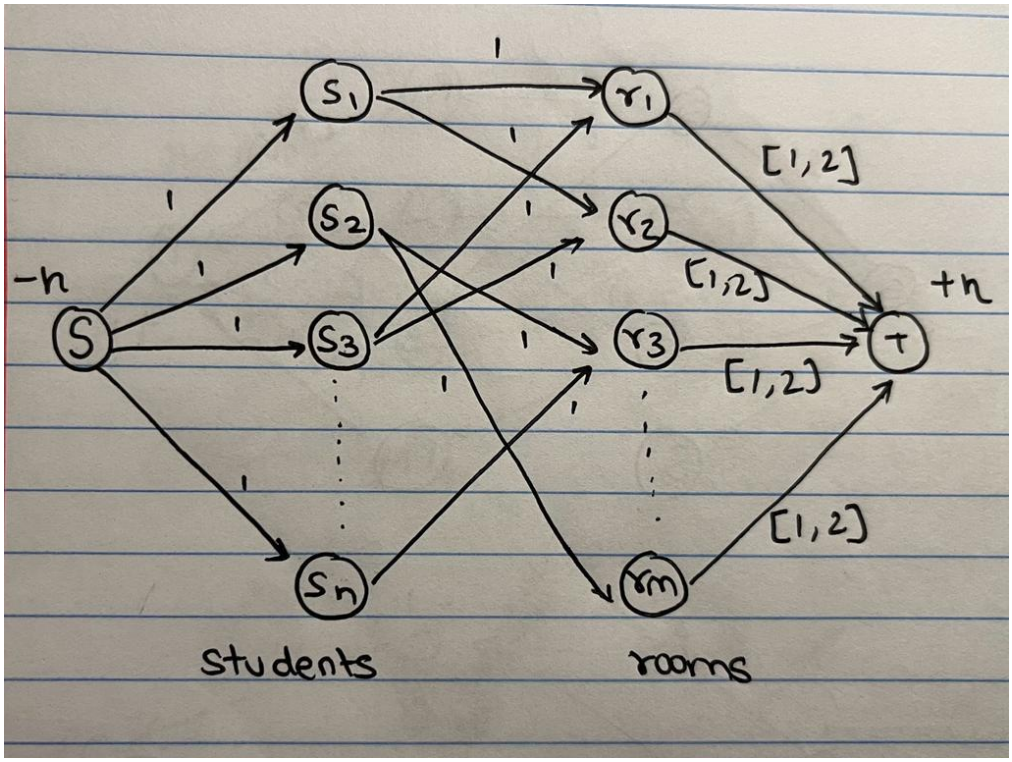
- Since all edges coming out of S (source) are saturated with a max-flow value of 16 hence feasible circulation exists in the graph.
- This is because the necessary condition for circulation in a graph with demands and lower bounds in that: There is a feasible circulation in G if and only if there is a feasible circulation in a new graph G'.
- In graph G', sum of demands = (-4) + 6+ 7 + (-12) + 2 + 1 = 0.

dnabar@usc.edu

Deven Nabar
USC ID: 7229446568

8. SOLUTION:

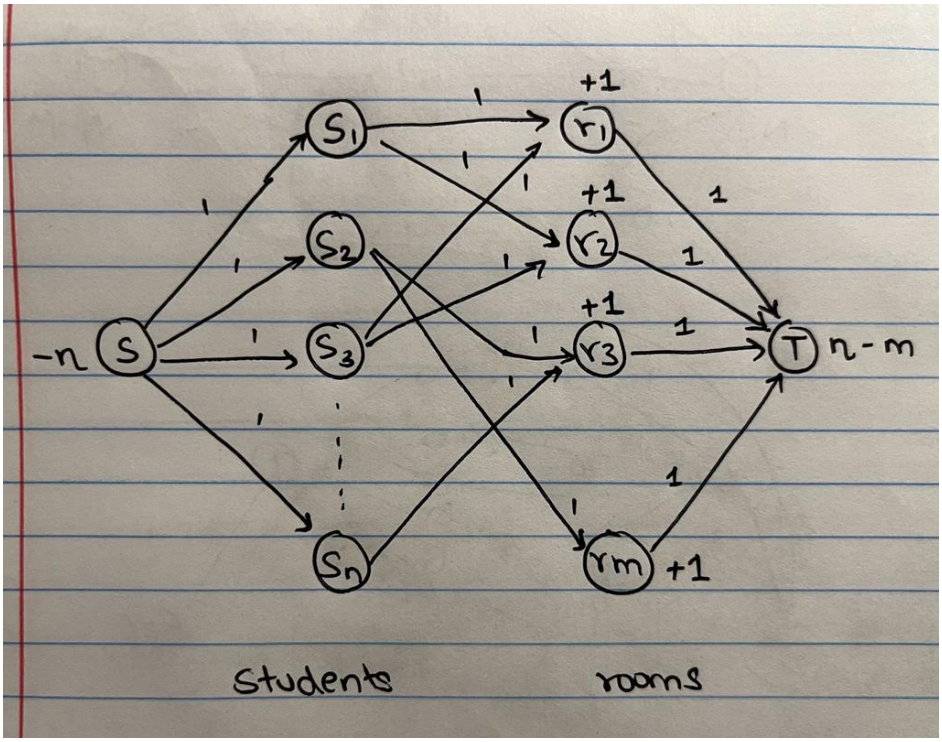
- We have been given n students and m rooms in total.
- A student can only be assigned to one room.
- A room can either house 1 or 2 students.
- We start by setting the problem as a bipartite graph problem with a set of vertices s_i representing students and a set of vertices r_j representing the rooms.
- We connect each student vertex to their preferred rooms with an edge of capacity 1
- Now, we add a source S and connect it to every student vertex with an edge (S, s_i) of capacity 1
- Since all students need to be housed, we add a supply of n units at the source.
- We now connect all the rooms to the target vertex T and add a lower bound of 1 and capacity of 2 ([lower bound/ flow] = [1,2]) on the edge (r_j , T).
- The target node gets a demand of n as we need to house n students.



- We now remove the lower bound constraints for the flow network and accordingly add new supply and demands on the room nodes and the target

dnabar@usc.edu

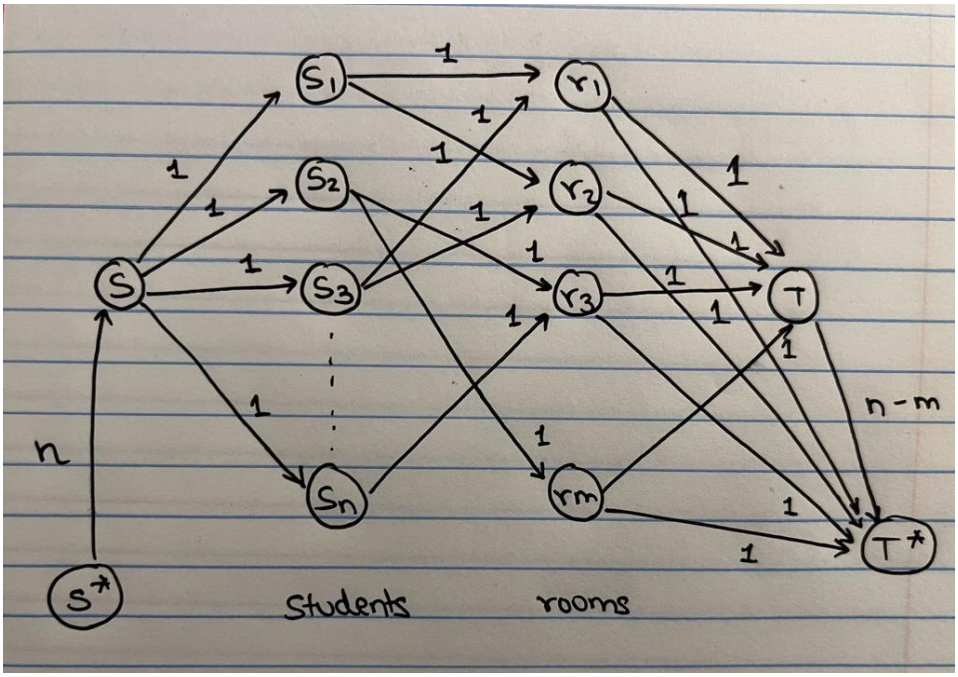
Deven Nabar
USC ID: 7229446568



- We now get a new graph with the edges from rooms to target having capacity of 1 and the rooms having a demand of 1.
- The target will now have a demand of n-m instead of n.
- We know that n > m as every room needs to be occupied by at least one student. This is not possible if m > n
- We now create a super source and a super sink/target. We connect the super source to the nodes with excess supply, that is the source and give this edge a capacity of n
- We connect all the vertices with a demand to the super target and give their respective edges the flow capacity equal to their respective demands.
- We have thus reduced the original problem into a network flow problem.

dnabar@usc.edu

Deven Nabar
USC ID: 7229446568



Claim:

- The original problem has a solution (a valid assignment is indeed possible) if and only if the constructed network has a max-flow of value n

Time Complexity:

- The given problem can be solved in polynomial time by using Edmond- Karp algorithm which has a time complexity of $O(V*E^2)$.

Assignment of students to room:

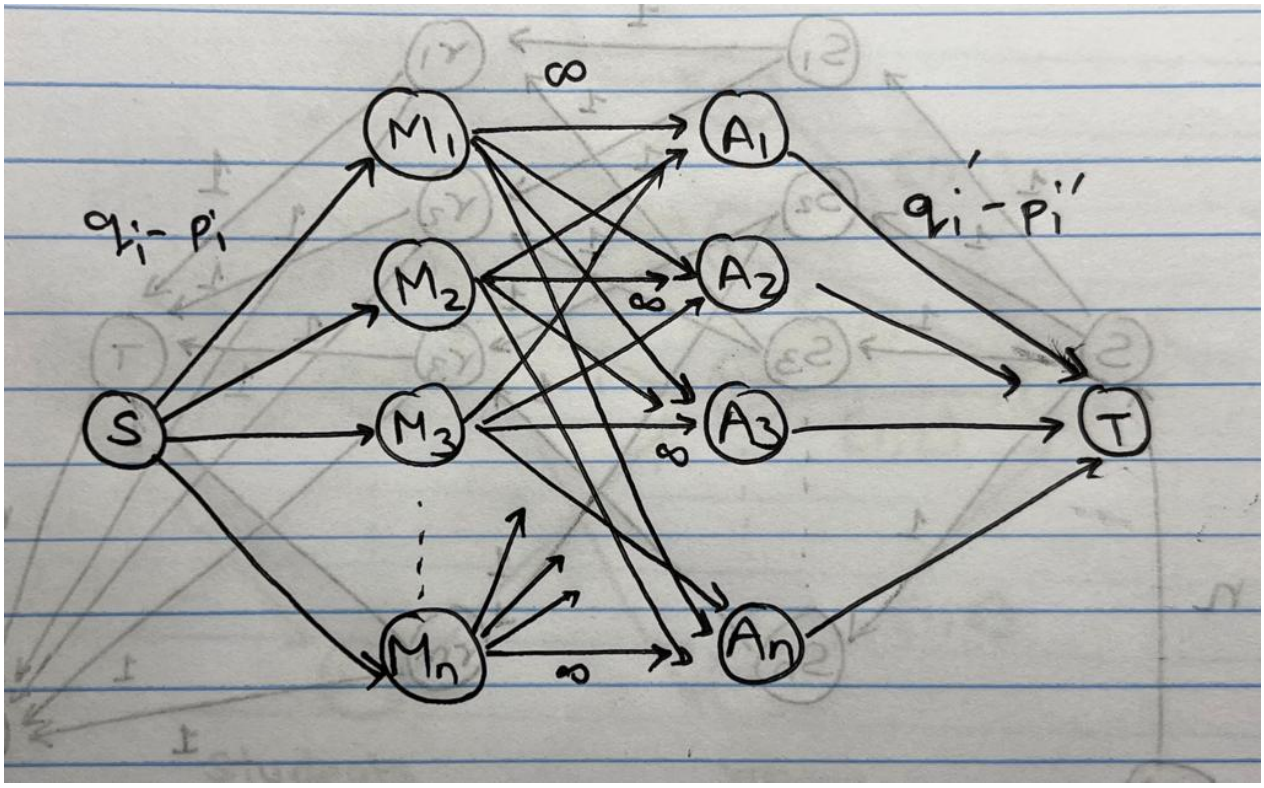
- If a feasible assignment exists, all the edges coming out of the source are saturated. We can find out which student is assigned to which room by checking which of the edges (s_i , r_j) are saturated. One student can only be assigned to a single room.

dnabar@usc.edu
Deven Nabar
USC ID: 7229446568

9. SOLUTION:

a)

- We create vertices representing all the products of microsoft and apple.
- Now we connect the source to all the Microsoft vertices and make the capacity of these edges as qi-pi representing the quality – price for each of the products.
- Now we connect all the apple vertices to the target and add the edge capacity of these edges as qi'-pi'.
- We need to make the connect all the n Microsoft edges to all the n apple edges
- If i=j we i.e. for M1 -> A1 / M2-> A2 edge we keep the edge capacity of infinity
- For all the other values of I and j we keep the Mi->Aj capacity as $\tau\ ij$
- Since we have kept the edges of some edges as infinity, our min cut will never pass through those edges

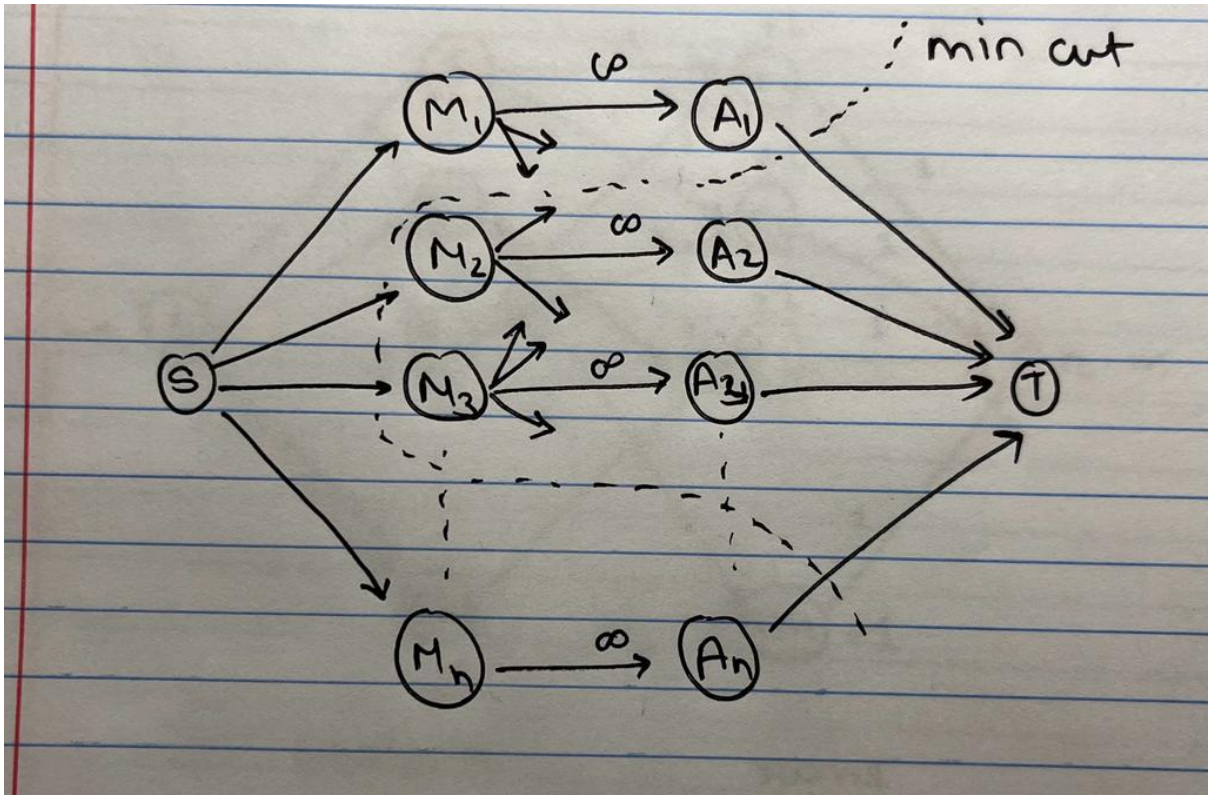


- We need two sets, set X with Microsoft products that are selected and apple products that are rejected and set Y with rejected Microsoft products and accepted apple products
- This is found by performing a min cut on the network flow graph.
- Min cut will never pass through the Mi->Ai edge as that is set to infinity
- Min cut will always pass through all the apple products -> target edges which are rejected and will also pass through Source-> Microsoft edges that are rejected.
- It will pass through all the tau edges as well which are not equal to infinity.

dnabar@usc.edu

Deven Nabar
USC ID: 7229446568

- Since we are trying to minimize everything else, we find the min cut possible here.
- We hence maximize the total system quality minus total price by minimizing penalties.
- The algorithm runs in polynomial time if we run Edmond Karp algorithm to find the min cut



b) Claim:

Minimizing the rejected values is the same as finding the min cut here and vice-a-versa.

We maximize the quality minus total price by finding the min cut

dnabar@usc.edu

Deven Nabar
USC ID: 7229446568

c) Proof:

==> Assume that a there is a solution. We have thus found an appropriate arrangement / combination of Microsoft and Apple products keeping in mind all the necessary constraints. This is only possible if we find the min cut on the graph described above.

Conversely, <==) If a min cut exists in the graph constructed above, we can say that a valid arrangement/ combination is found of Microsoft and Apple products that lead to maximum total system quality minus the total price including all the penalties

Lastly, we get a valid combination of the products by finding the min cut here.