Analysis of Algorithms Homework 4

USC ID: 3725520208
Name: Anne Sai Venkata Naga Saketh
Email: annes@usc.edu

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

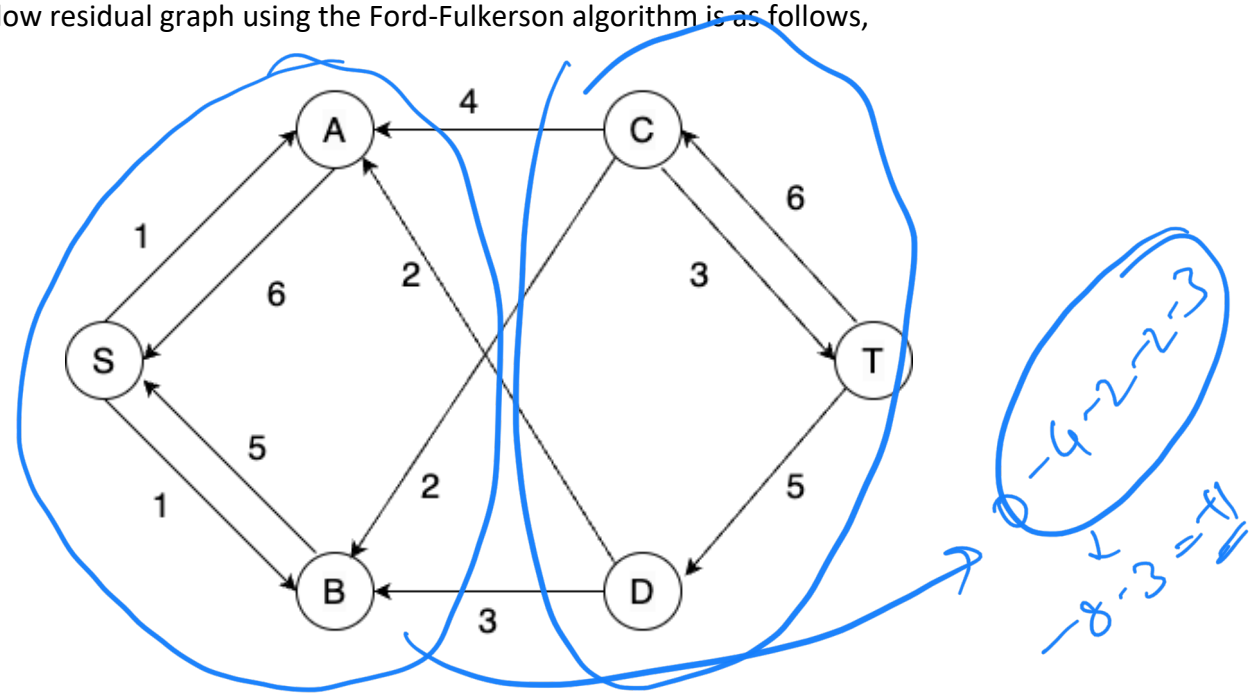
   (a) Find a max-flow in G using the Ford-Fulkerson algorithm. Draw the residual graph G_f corresponding to the max flow. You do not need to show all intermediate steps.

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

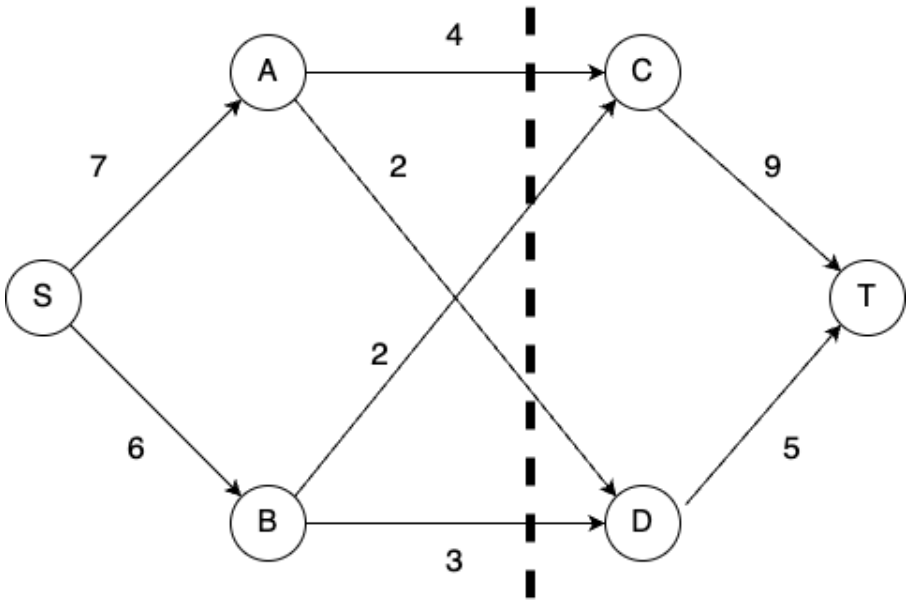
   (c) Prove or disprove that increasing the capacity of an edge that belongs to a min cut will always result in increasing the maximum flow.

   Solution:

   a. The max-flow residual graph using the Ford-Fulkerson algorithm is as follows,



   b. The max-flow value in the graph is **'11'**, and the min-cut for the network flow graph is as follows,

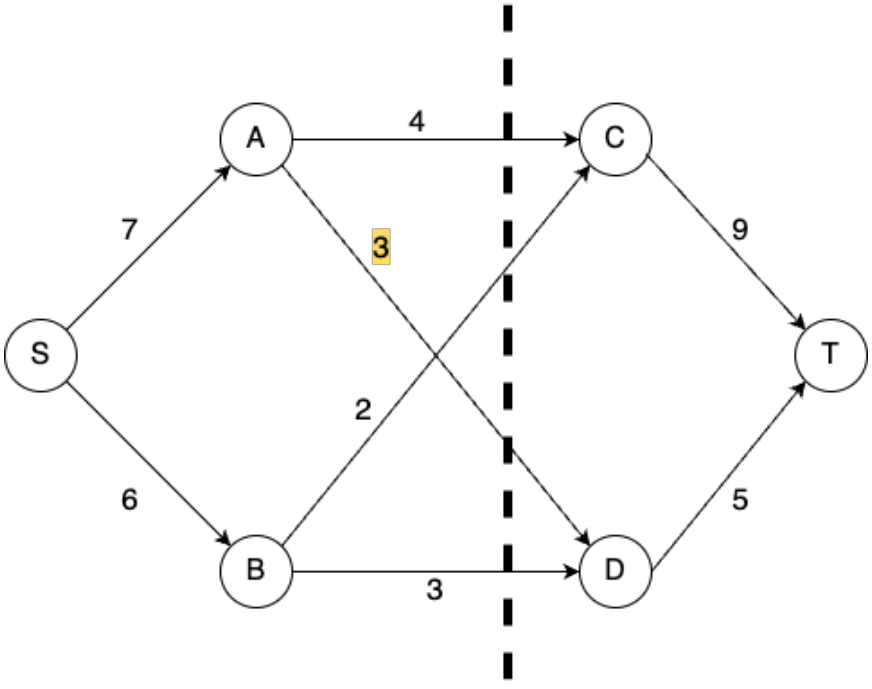


The Nodes S, A, and B lie in one partition, and the Nodes C, D, and T lie in the other partition of the min-cut.

Partition 1: {**S**, A, B}

Partition 2: {C, D, **T**}

c. Increasing the capacity of an edge on the min-cut may not increase the max-flow.



Changing the edge capacity of an edge in the min-cut, may not always impact the max flow value because the max flow in a given network flow shall also depend on the capacities of the edges that are out of the min-cut and are directed towards the Sink **'T'**.

For example, even if the capacity of edge A-D was increased from **2 to 3**, due to the capacity constraint at the other edges in the network flow graph, the max flow that we can push through the edge D-T remains 5 which is already saturated**.**

But if the edge capacities of B-C are increased then the edge C-T can take more flow, thereby increasing the max flow in the given network flow graph.

Therefore, we can conclude that changing the edge capacity on the min-cut might not always increase the max flow of the given graph.

2. **Consider a set of mobile computing clients in a certain town who each need to be connected to one of several possible base stations. We'll suppose there are n clients, with the position of each client specified by its (x,y) coordinates in the plane. There are also k base stations; the position of each of these is specified by (x,y) coordinates as well. For each client, we wish to connect it to exactly one of the base stations. Our choice of connections is constrained in the following ways. There is a range parameter R which means that a client can only be connected to a base station that is within distance R. There is also a load parameter L which means that no more than L clients can be connected to any single base station. Given the positions of a set of clients and a set of base stations, as well as the range and load parameters, decide whether every client can be connected simultaneously to a base station.**

   (a) Describe how to construct a flow network

   (b) Make a claim of how the original problem is related to the max-flow problem.

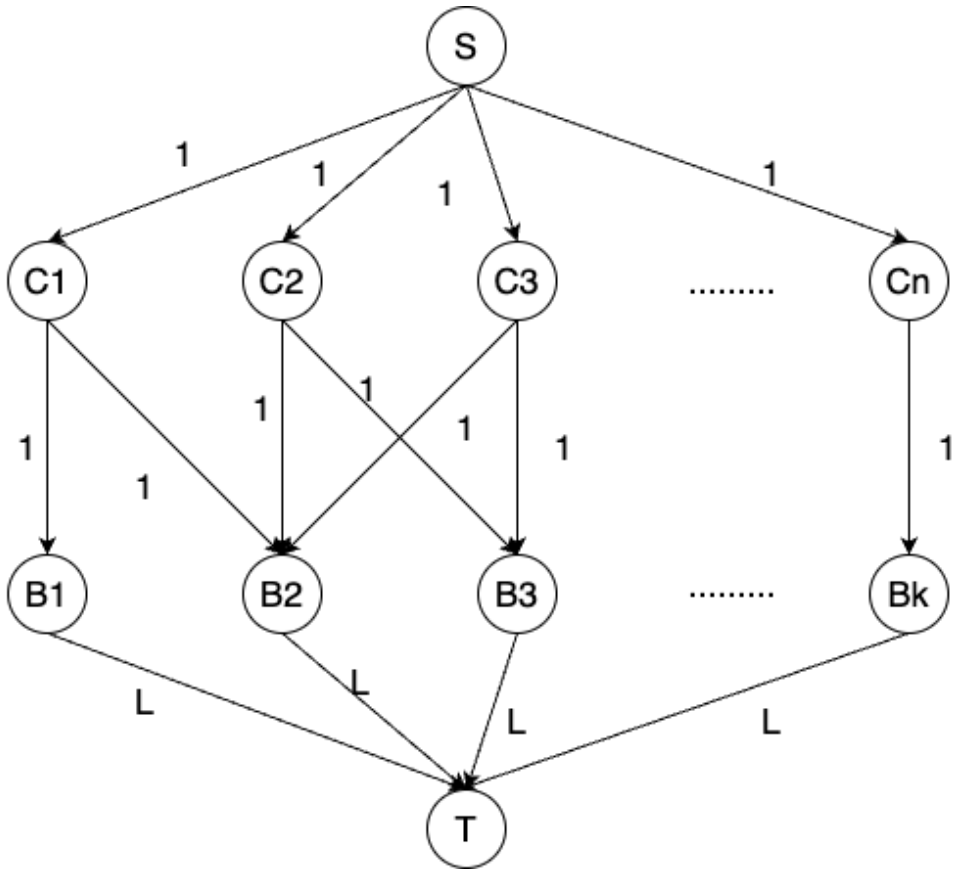
   (c) Prove the above claim in both directions

   Solution:

   a. **<u>Constructing the network flow:</u>**

   - It is given that there are '**n**' mobile computing clients C_1, C_2, C_3, C_4……..C_n and there are a set of '**k**' base stations B_1, B_2, B_3, B_4……..B_k to which these mobile computing clients can be connected.
   - The position of the base station and the position of the client is mentioned as coordinates (x,y).
   - Each client can be connected to only one base station, and it should be within the given range **'R'** (distance from the client to the base station shall be less than or equal to R). The distance between the client and the base station can be calculated using the coordinates given on the graph.

- Each base station also has a limit on the number of clients it can handle at any given point in time, which is defined by the load parameter **'L'.**



- The above network flow is constructed as per the given conditions, that a Base station can handle a load of 'L', and each mobile computing client can only be connected to one base station at any given time.
- All the clients are connected to the **Source** vertex, and all the Base stations are connected to the **Sink**/T vertex.
- The weights added on the edges from the source to the clients are **'1',** and the weights added on the edges from the base stations to the sink are **'L'**.

b. <u>Claim to the original problem:</u>

The given problem is related to a max-flow network flow graph problem since all the clients shall be connected to at least one of the base stations. So, in that case, since there are 'n' clients, the max-flow value shall also be '**n**'.

c. <u>Proof for the above claim:</u>

i. <u>Proving the claim in the forward direction. (=>)</u>

For some possible way of connections such that all the clients shall be connected to one of the base stations if they are within a given range 'R' and the base station can accept the connection and is within the given load 'L'.

In that case, all the edges, from S to the clients, i.e., S-C1, S-C2, S-C3……S-Cn, shall all be saturated as we push a value of 1 from the source to all these nodes.

All the base stations are connected to the Sink/T Node with a capacity of 'L' (as that is the maximum possible number of clients that the base station can handle at any given point in time), hence all the flow that is pushed from the source vertex shall reach the Sink vertex.

Since all the 'n' edges from the source vertex to the clients are saturated, it incurs that the max flow value in the given graph is 'n'.

ii. <u>Proving the claim in the backward direction. (<=)</u>

Here we assume that the max-flow in the above graph is 'n', and we need to prove that we can get an assignment for the clients to the base station.

The maximum flow coming to the Sink/T vertex shall only be from base stations. If the Sink is receiving a flow of 'n' if and only if the base stations are receiving a flow of 'n', it means that all the 'n' clients are connected to the base stations. This is because every client can be connected to only one of the base stations that are within the given range 'R' and if the Base station is able to handle the load 'L'.

So, it means that all the 'n' clients are connected to any one of the base stations.

Hence, we have proved our claim in both directions, which means that our claim and the network flow graph align with each other.

3. There are N students at the USC Viterbi school who want to celebrate the Indian festival of colour called Holi. There are M unique colors (call the set of colors C) available at the USC Bookstore and i^{th} color has c_i packets left at the store (e.g there are c_1 packets left of color 1, c_2 packets left of color 2, and so on till c_m packets of color M). The i^{th} student has a set $F_i \subseteq C$ representing their favourite colors and wishes to buy a total of b_i packets from their favourite color set (it doesn't matter which colors out of F_i as long as the total number of packets is b_i). However, to ensure fair availability of all colors, the USC Bookstore restricts each student to buy a maximum of 2 packets of the same color. Design an algorithm that determines if all the students can have their wish granted.

(a) Describe how to construct a flow network

(b) Make a claim of how the original problem is related to the max-flow problem.

(c) Prove the above claim in both directions

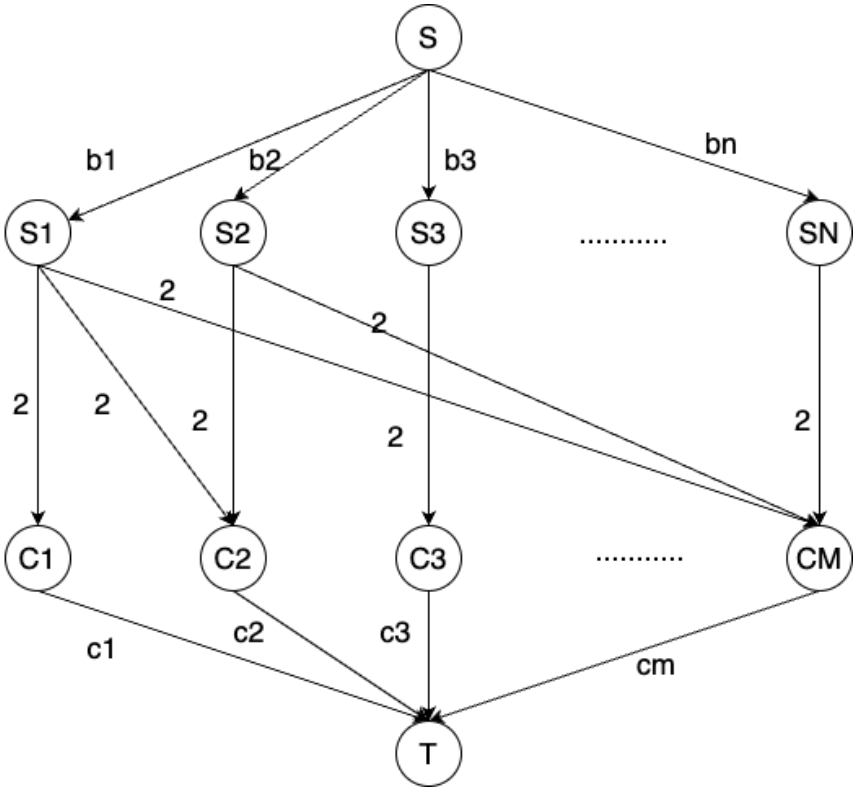
Solution:

a. <u>Construction of the network flow for the above problem:</u>

The network flow graph is constructed from the above graph using the following conditions,

- There are **'N'** students, **'M'** unique colors, and the i^{th} student buys b_i number of colors from the bookstore, but also on the condition that the maximum number of packets he can buy for a particular color is **'2'**.
- Let the students be S_1, S_2, S_3…….S_N and the unique colors be C_1, C_2, C_3…….C_M
- And each Student buys S_i and buys the set of colors F_i from all the available colors, F can be a subset or equal to set C, and b_i is the number of colors that are in the set F_i.
- Connect all the Student nodes to the Source Vertex and all the color nodes to the Sink/T vertex.
- Each student can buy a maximum of 2 packets of each color, but there are $\mathbf{c_i}$ colors for each color 'i' so, the color vertexes are joined to the Sink with the value $\mathbf{c_i}$ (the maximum possible number of packets all the students can buy).

The Network flow graph for the above problem can be as follows,



b. Claim with the original problem related to the max flow graph:

Every student buys the colors that are required for him, if and only if the max flow value of the above graph can be defined as $\Sigma\, \mathbf{b_i}$, i.e., $b_1+b_2+b_3+\ldots\ldots b_n$.

If $\Sigma\, \mathbf{c_i} < \Sigma\, \mathbf{b_i}$, in this case, all the students wish to buy the colors of their choice for the Holi festival might not be fulfilled.

c. Proving that the above claim is correct in both directions:

i. Proving the claim in the forward direction. (=>)

Assuming that we are given an assignment, and we need to find the max flow for the given network flow graph.

Assuming that there exists an assignment such that every student is able to buy the colors b_i of his requirement from the set of available colors F_i, also by following the condition that a student can only buy a maximum of 2 packets of the same color, this is taken care by having the capacity of the edge from student to color as '2'. Which does not allow pushing a flow of more than 2.

Hence, we shall be pushing a flow of b_i for each student that buys the colors, so the total flow in the graph shall be $b_1+b_2+b_3+\ldots\ldots b_n$, which is equal to the max flow.

ii. Proving the claim in the backward direction. (<=)

Assuming that we are given the max flow in the graph as $b_1+b_2+b_3+\ldots\ldots\ b_n$, we need to find the assignment such that it satisfies the max flow condition.

The max flow of the network flow graph is $b_1+b_2+b_3+\ldots\ldots\ b_n$, which means that the max flow is received from the color vertices. We have constructed the network flow graph such that every student can only buy $\mathbf{b_i}$ number of colors from the set of colors available in $\mathbf{F_i}$. Every student can buy a maximum of 2 packets of a single color. This constraint is implemented using the edge capacity of '2' from every student to each color.

This means that the maximum number of colors that the student S_i can buy is b_i. So, the max flow is $b_1+b_2+b_3+\ldots\ldots\ b_n$.

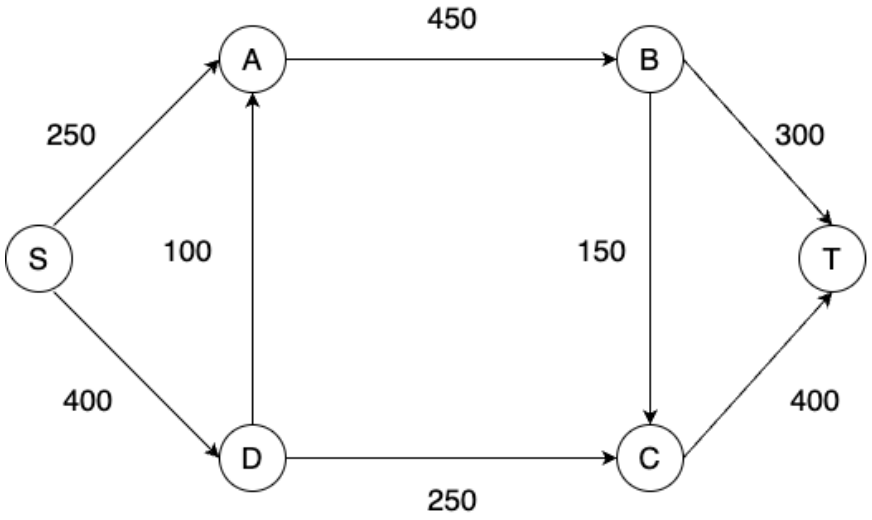
Hence, we have proved our claim in both directions, which means that our claim and the network flow graph align with each other.

4. In a museum S and T denote the entrance and exits of a hall. There are exhibits placed at the positions A,B,C,D. The hall has three gardens as shown in gray in the figure. There are hallways around the gardens that can support the movement of a specified number of people per hour which are marked in the figure. For example, people wanting to visit exhibit D can enter at S

and go to D via the hallway which supports 400 visitors to pass through every hour. Similarly, the hallway from exhibit B to C support 150 visitors per hour. The museum wants to attract 6000 visitors every day and closes when the target is reached for that day. If the museum opens at 8 am on a specific day, when is the earliest it can close on that day to support 6000 visitors?

Solution:

Converting the given diagram into a network flow graph shall result in the following graph.



From the above graph, we can derive the max-flow value to be 600, per hour.

It is given that the museum closes once there are 6000 visitors, so the max flow through the museum can only be 600 each hour. It takes at least 10 hours to attain/reach the given target of 6000 visitors. $(6000/600 = 10hrs)$.

So, assuming that there are no breaks in between, the earliest time the museum can close is **6:00 PM.**

5. **In addition to the edge capacity constraints, we introduce vertex capacities b_i constraints for each vertex v_i in a network-flow problem. The total amount of flow passing via vertex v_i cannot exceed b_i. Describe an algorithm to solve the vertex capacity constrained max-flow problem. Hint: Use reduction to edge capacity network flow problem.**

Solution:

<u>The algorithm to solve the vertex capacity constrained max-flow problem is as follows:</u>

Assume in the given vertex constrained max-flow graph, V_1 is the source node and V_n is the Sink node.

Step 1: For every node V_i in the given original graph, create a new vertex $V_i^{|}$ and make an edge from V_i to $V_i^{|}$ with the edge capacity equal to the given vertex capacity $\mathbf{b_i}$ for the respective vertex.

Step 2: Now, as per our assumptions, V_1 shall be the Source node and $V_n^{|}$ shall be the new Sink node.

Step 3: Change all the outward edges from $\mathbf{V_i}$ **to** $\mathbf{V_i^{|}}$**.**

Step 4: Now on the new graph run Ford Fulkerson algorithm to get the max flow value.

The edge capacity from $\mathbf{V_i}$ to $\mathbf{V_i^{|}}$ (vertex capacity $\mathbf{b_i}$) shall make sure that the flow passing through the graph does not exceed the given vertex capacity $\mathbf{b_i}$.

Claim for the vertex constrained max-flow graph:

The max-flow of the modified network flow graph is the max-flow of the vertex constrained max-flow problem.

Proof for the above claim:

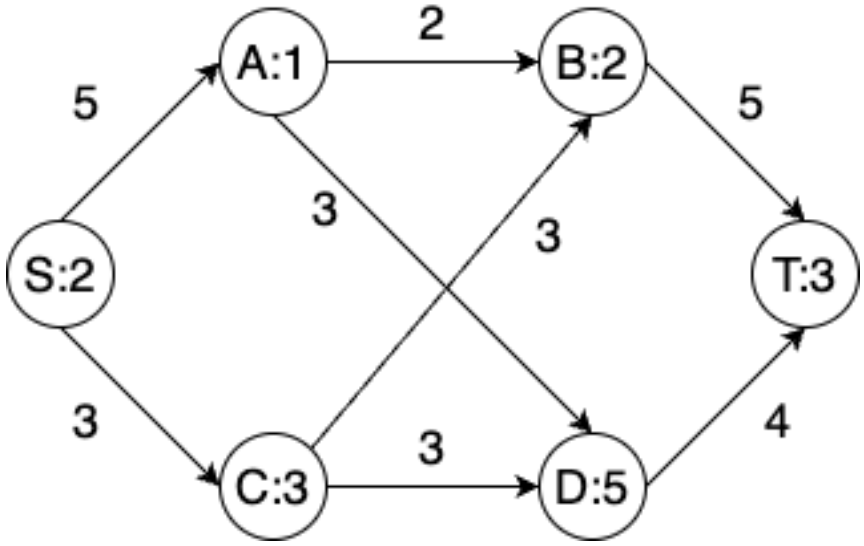
For each vertex $\mathbf{V_i}$ we have created new vertices $\mathbf{V_i^{|}}$, with the edge capacity b_i from V_i to $V_i^{|}$.

This will ensure that the flow passing from $\mathbf{V_i}$ **to** $\mathbf{V_i^{|}}$ is restricted to $\mathbf{b_i}$.

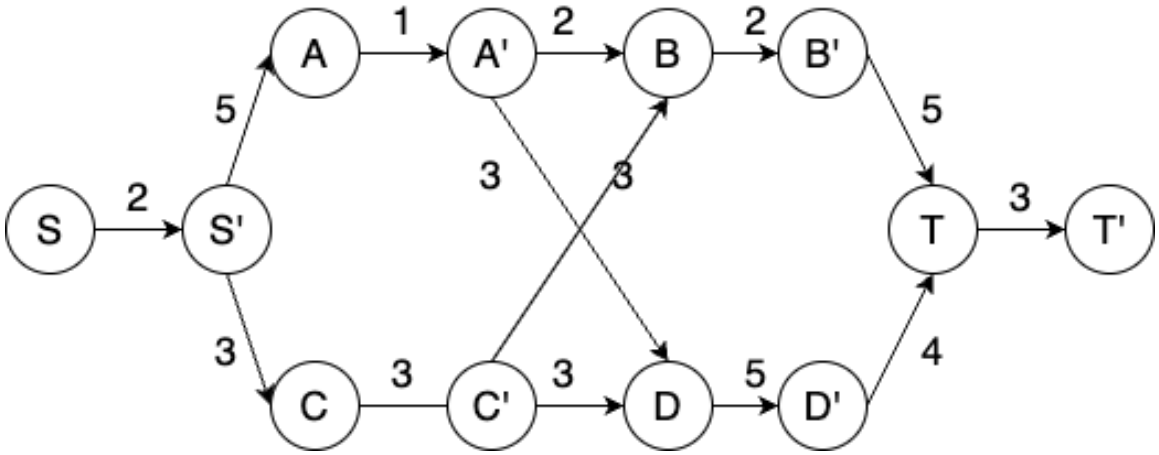
Then, we must change all the outward edges from vertex $\mathbf{V_i}$ **to** $\mathbf{V_i^{|}}$**.** So that the flow going outward from $\mathbf{V_i}$ will be directed to $\mathbf{V_i^{|}}$ and then the flow will be directed to the next connecting vertex.

Example:

If the below is the graph with the vertex constraints,



Then this graph can be converted to a max flow network graph as follows,



Now S and T' as the new Source and Sink vertices, now we can run a Ford Fulkerson algorithm on this network graph, to get the max flow value.

Time Complexity to find the max-flow:

The time complexity to find the max flow in the above vertex constrained max flow network problem using the Ford Fulkerson algorithm is $\boldsymbol{O}(|\boldsymbol{f}|.\boldsymbol{n}^2)$

6. **The edge connectivity of an undirected graph is the minimum number of edges whose removal disconnects the graph. Describe an algorithm to compute the edge connectivity of an undirected graph with n vertices and m edges in $O(m^2n)$ time.**

Solution:

Algorithm to compute the edge connectivity of an undirected graph:

Step 1: We are given an un-directed input graph G(V, E).

Step 2: Modify the original un-directed graph to be having directed edges in the same direction (either outward or inward from a given vertex)

Step 3: Choose any two vertices as Source and Sink vertices (Any vertices can be chosen; the choice does not matter.)

Step 4: Define the edge capacities of all the edges as 1.

Step 5: Let 's' be a vertex in the given list of vertices; for each vertex 't' that belongs to V – {s}, solve the min-cut problem in the given network flow (G, S, T, c).

As per the definition of a min-cut of a network graph, it shall give a partition of the vertices in the residual graph whose sum of edges in the graph is minimum, which means the edges from the min-cut of a network flow will break the connected graph.

The Time complexity of the above algorithm:

The above algorithm shall use n – 1 min-cut operations, each of which shall be solved using a max-flow computation using the Ford-Fulkerson algorithm in $O(mn)$, so the total time complexity shall be $O(n^2m)$.

7. In the network below, the demand values are shown on vertices. Lower bounds on flow and edge capacities are shown as (lower bound, capacity) for each edge.

Answer the following questions:

(a) Remove the lower bounds on each edge. Write down the new demands on each vertex A, B, C, D, E, and F in this order.

(b) Solve the circulation problem without lower bounds. Write down the max-flow value.

(c) Is there a feasible circulation in the original graph? Explain your answer.

Solution:

a. Removing the lower bound values on the given graph G:

The lower bounds on the given graph can be reduced using the below formula,

New demands on the edge can be computed using the formula,

D’(v) = D(v) + Lower(F(out)) - Lower(F(in))

Therefore, the new demands are as follows:

A = -8 + 1 + 3 = -4

B = 5 + 4 – 3 – 0 = 6

C = 8 + 5 – 4 – 2 = 7

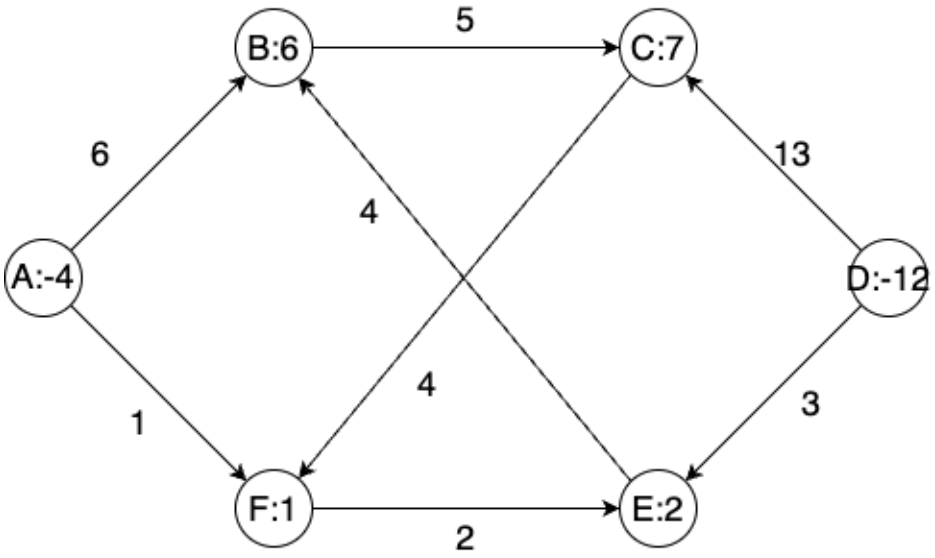
D = -18 + 2 + 4 = -12

E = 6 – 4 – 0 + 0 = 2

F = 7 – 1 – 5 + 0 = 1

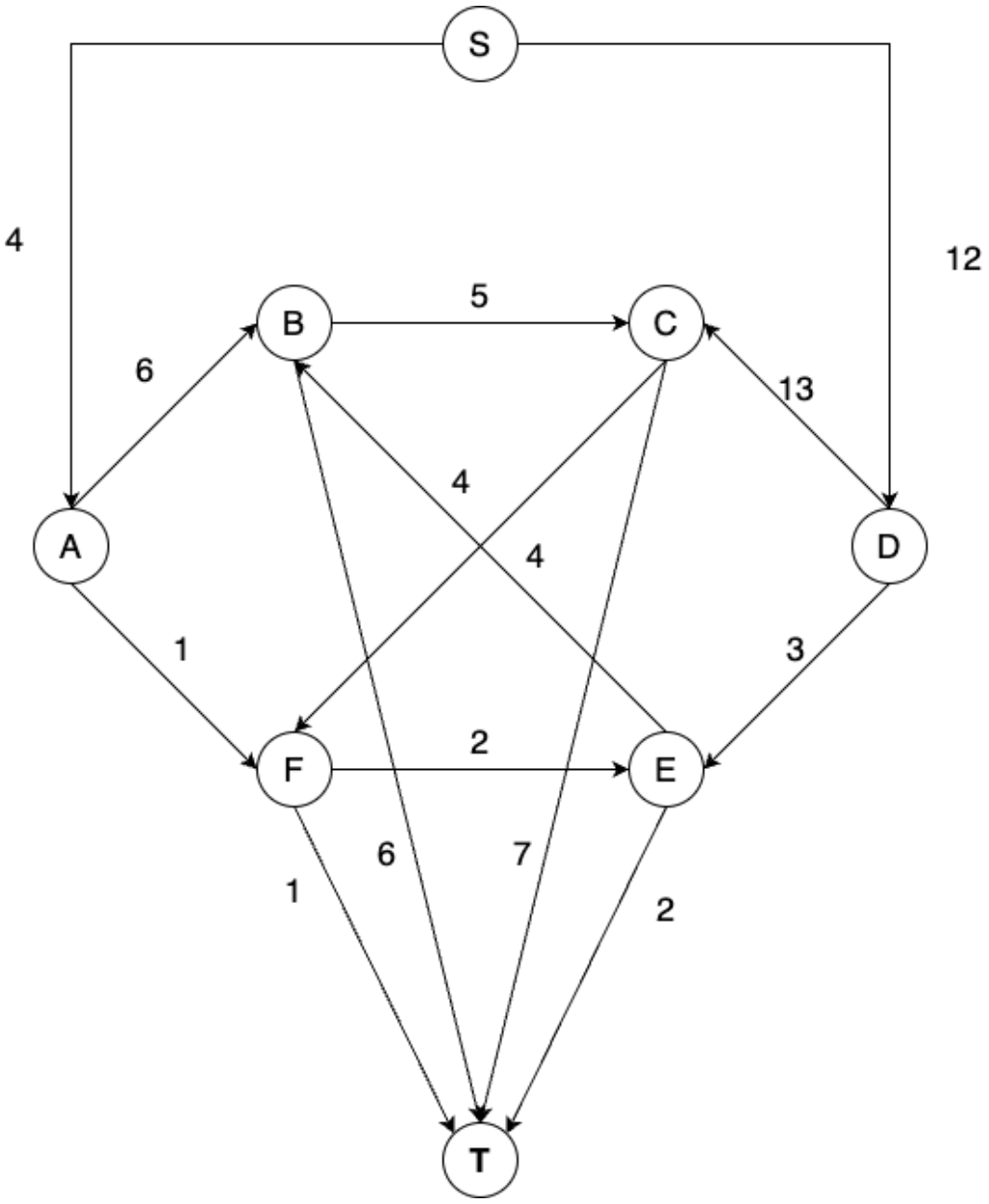
Calculating the new edge capacities after removing the lower bounds can be given by using the following formula,

New edge capacity = |(Upper bound) – (Lower bound)|



b. <u>Reducing the given problem to a network flow graph and solving for the max-flow:</u>

So, the updated diagram network flow diagram shall be as follows:



The max flow value in the above network graph is **14.**

c. <u>Checking the feasibility of the circulation problem:</u>

The above problem is feasible to solve. This can be inferred by checking the necessary condition. i.e., the sum of Demands and Supply shall be '0'.

i. Checking the above condition for the original problem: -8 + -18 + 5 + 6 + 7 + 8 = 0, therefore the problem is feasible to solve.

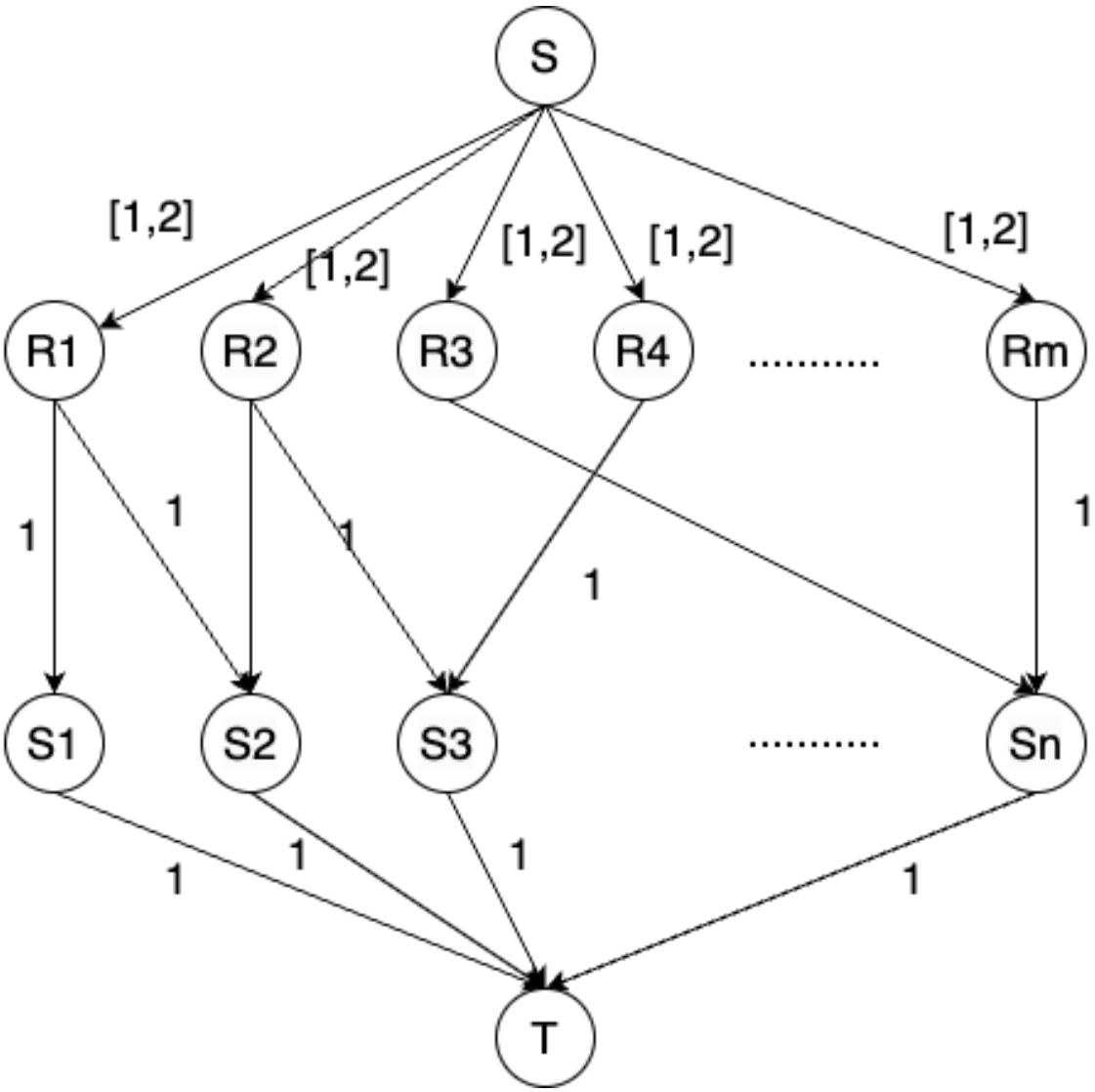
ii. Checking the above condition for the problem after removing the lower bounds: -4 + 6 + 7 -12 + 1 + 2 = 0, therefore the problem is feasible to solve.

8. **Consider a student dormitory with n students and m rooms. A student can only be assigned to one room. Each room has the capacity to hold either one or two students. Each student has a subset of rooms as their possible choice. We also need to make sure that there is at least one student assigned to each room. Give a polynomial time algorithm that determines whether a feasible assignment of students to rooms is possible that meets all the above constraints. If there is a feasible assignment, describe how your solution can identify which student is assigned to which room.**

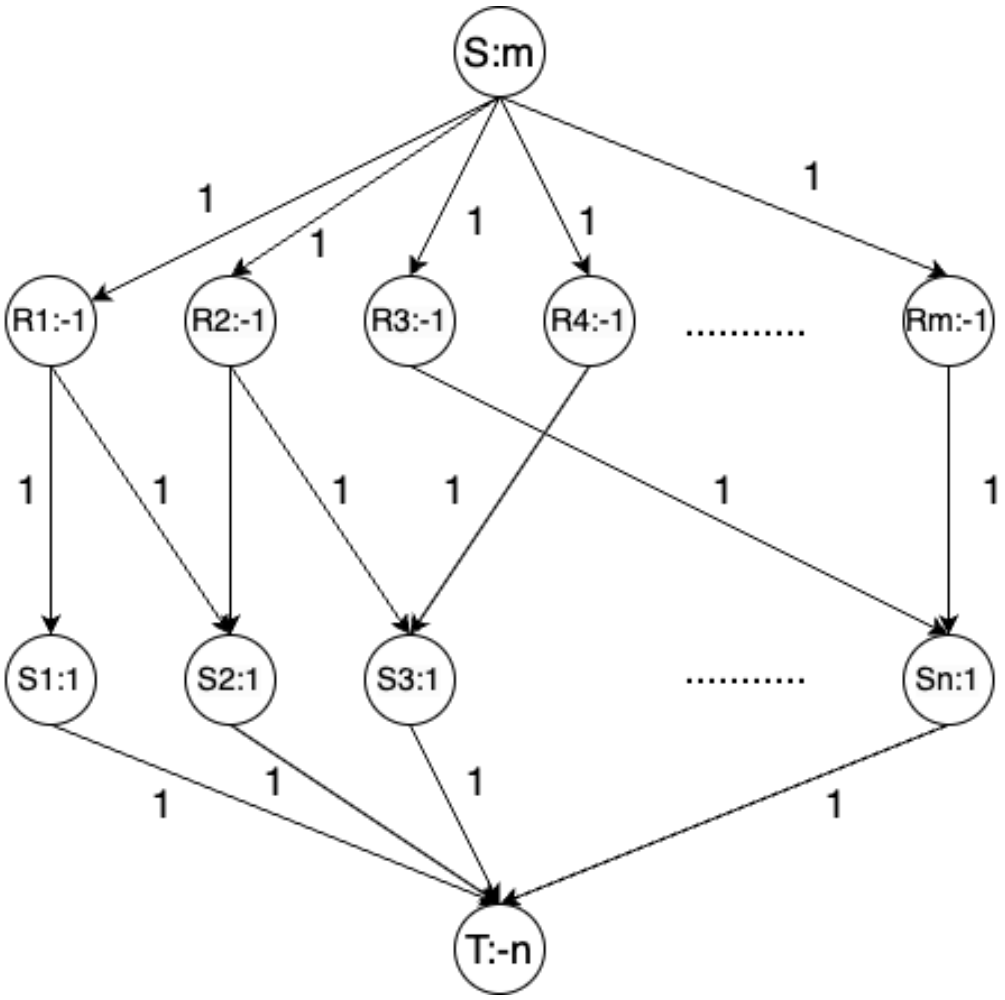
Solution:

Construction of the network flow graph is as follows:

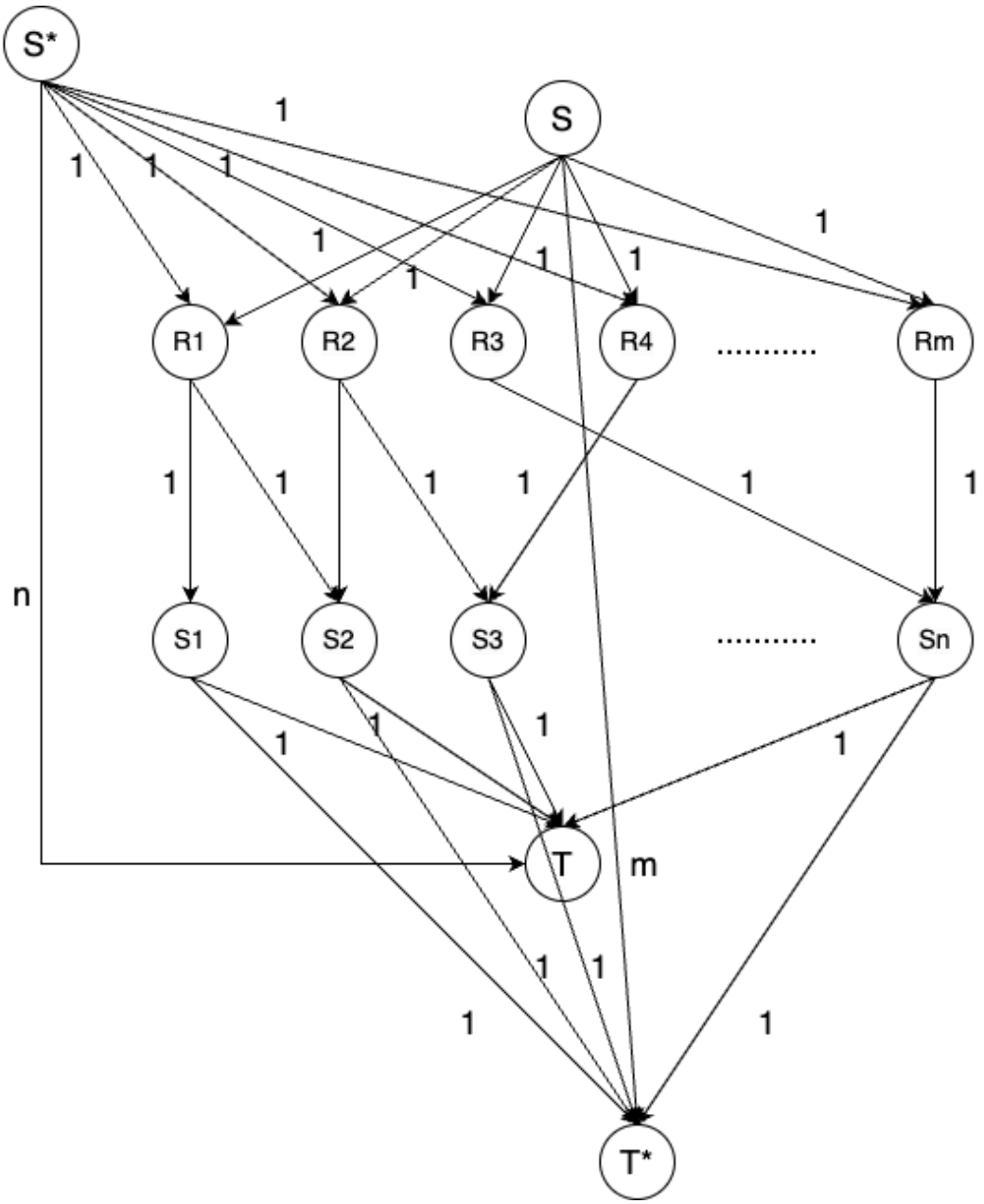
- Let S, be the Source vertex, and T be the Sink vertex.
- Let S1, S2, S3, ……Sn be the 'n' number of students that needs to be assigned to the 'm' different rooms R1, R2 R3, …….Rm.
- The assignment is made in such a way that each student can be assigned to only one room, and each room can hold a maximum of 2 students.
- The Source S is connected to the room nodes and the edge capacity between them is [1,2]. i.e., the minimum flow that needs to be sent through these edges is 1 and the maximum that can be sent through these edges is 2.
- The capacity of the edges from the Student nodes to the Sink/T node is 1, and since each student can be assigned to only one room.



Now we need to reduce this graph, so that we eliminate the lower bounds across the network flow graph which is as follows.



Now we need to reduce this to a network flow graph, by adding a new source and sink nodes (super source and super sink nodes).



Claim for the above network flow graph:

The given problem is related to a max-flow network flow graph problem since all the students shall be assigned to at least one room and every room must have at least 1 student assigned. So, in that case, since there are 'n' students and 'm' rooms, the max-flow value shall also be '**n+m**'.

Proof in both directions for the above network flow graph:

i. Proving the claim in the forward direction. (=>)

Assuming for some possible way of assignment exists, such that all the students are assigned to one of the rooms of their respective preference, and each room was assigned at least one student.

We have constructed the above network flow graph such that the source node S is connected to the room nodes such that minimum flow they need to have is 1 and the maximum flow that passes through the edges between the source and the room nodes is 2. So, the demand on the room nodes is -1 and the demand on the student nodes is 1 and the demand on the target node is n.

Since all the rooms have at least one student assigned and each student is assigned to a room, then in that case, the max flow of the above graph after multiple reductions is **n+m.**

ii. Proving the claim in the backward direction. (<=)

Here we assume that the max-flow in the above graph is 'n+m', and we need to prove that we can get an assignment for the students to the rooms and each room has at least one student in it.

The maximum flow coming to the Sink/T vertex shall be 'n+m' and which is received from the student nodes and the S node(previous Source node before reduction).

So, it means that all the 'n' students are assigned to any one of the rooms of their choice as well as each room has at least one student assigned to it.

Hence, we have proved our claim in both directions, which means that our claim and the network flow graph align with each other.

Polynomial time algorithm to find the max-flow in the above network graph:

Here, we need to use Edmond Karp's algorithm to find the residual graph in the network flow to find the max-flow value.

The above network graph that we have constructed has n+m+4 nodes, i.e., 'n' students, 'm' rooms, and a source node and a sink node.

Edmond Karp Algorithm Given(G, S, T, C):

Step 1: Start with $|f| = 0$, so $f(e)=0$

Step 2: Find the shortest augmenting path in G_f.

Step 3: Augment the flow along this path.

Step 4: Repeat the following steps until there is no remaining S-T path in the graph G_f.

Now, we can find the maximum flow in the given residual graph by summing up the capacities of all the outward edges from the Sink/T in the residual graph.

Checking the feasibility of the assignment and how we can find the student assigned to a particular room:

If the max-flow of the above graph is '**n+m**' by following the respective edge capacities as mentioned above, then we can conclude that the student-to-room assignment is feasible.

To find the room that each student has been assigned, we need to traverse the student nodes, and find the student node $\mathbf{S_i}$, and then find the saturated edge to the rooms to identify the assigned room.

Time complexity of the above algorithm is as follows:

The Edmond Karp algorithm shall take $O(V.E^2)$. here the total number of vertices is n+m+4, every student can have all the available rooms as one of his preferences, then the graph shall be a dense graph. So, the time complexity to find with room a given student is assigned is $O(V^5)$.

The final time complexity of the above algorithm is: $\boldsymbol{O((n+m)^5)}$.

9. **Suppose that you have just bought a new computer and you want to install software on that. Specifically, two companies, which you can think of like Microsoft and Apple, are trying to sell their own copy of n different products, like Operation System. Spread Sheet, Web Browser. For each product i, i ∈ {1,2,...,n}, we have**

   • the price $p_i \geq 0$ that Microsoft charges and the price $p'_i \geq 0$ that Apple charges.

   • the quality $q_i \geq 0$ of Microsoft version and the quality $q'_i \geq 0$ of Apple version.

   For example, Apple may provide a better Web Browser Safari, but Microsoft a better Word Processor. You want to assemble your favorite computer by installing exactly one copy of each of the n products, e.g., you want to buy one operating system, one Web Browser, one Word Processor, etc. However, you don't want to spend too much money on that. Therefore, your goal is to maximize the quality minus total price.

   However, as you may know, the products of different companies may not be compatible. More concretely, for each product pair (i, j), we will suffer a penalty $\tau_{ij} \geq 0$ if we install product i of Microsoft and product j of Apple. Note that τ_{ij} may not be equal to τ_{ji} just because Apple's

Safari does not work well on Microsoft Windows doesn't mean that Microsoft's Edge does not work well in Mac-OS. We assume that products are always compatible internally, which means that there is no penalty for installing two products from the same company. All pairwise penalties will be subtracted from the total quality of the system.

Your task is then to give a polynomial-time algorithm for computing which product i to purchase from which of the two companies (Apple and Microsoft) for all i ∈ {1, 2, . . . , n}, to maximize the total system quality (including the penalties) minus the total price. Prove the correctness of your algorithm.

(a) Describe how to model this problem as a min-cut problem.

(b) Make a claim of how the original problem is related to the min-cut problem.

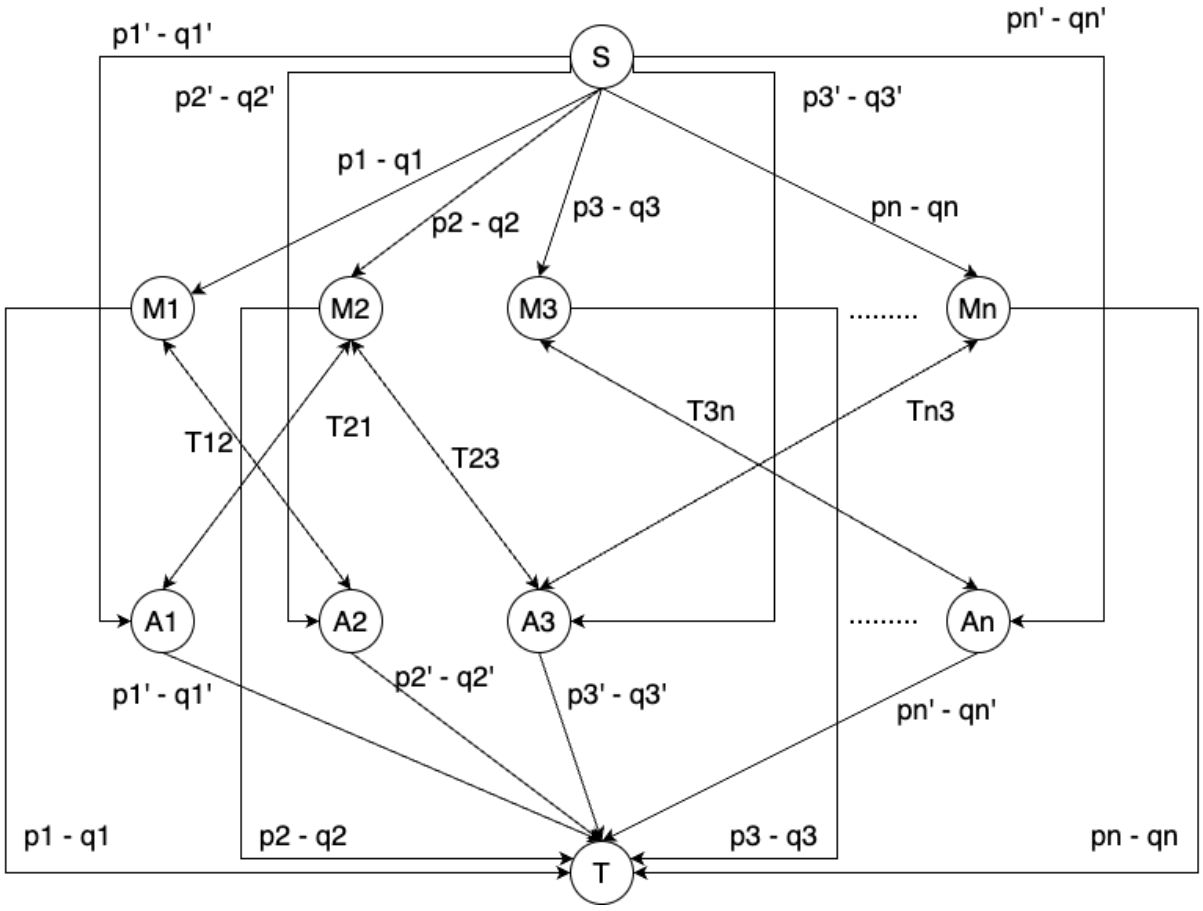
(c) Prove the above claim in both directions

Solution:

a. Describing how to model the problem as a min-cut problem:

- Let S be the source node and T be the Sink node.
- Create a Bipartite graph, with one set of nodes for the Microsoft nodes and one set of nodes for the Apple products.
- Let M1, M2, M3…….Mn be the Microsoft products and let A1, A2, A3……..An be the Apple products.
- The capacity of the edges connecting the Source to the Microsoft Nodes is **pi-qi** for i^{th} Microsoft products.
- The capacity of the edges connecting the Source to the Apple Nodes is **pi'-qi'** for i^{th} Apple products.
- The capacity of the edges connecting the Microsoft Nodes to the Sink is **pi-qi** for i^{th} Microsoft products.
- The capacity of the edges connecting the Apple Nodes to the Sink is **pi'-qi'** for i^{th} Apple products.
- The capacity of the edges connecting the i^{th} Microsoft product to the j^{th} Apple products is $\mathbf{T_{ij}}$.
- This shall be a double sided edge because the We can traverse from the Microsoft product to the apple product and then to the Sink as well as similarly, we can traverse from the Apple product to the Microsoft product and then to the Sink.

  The Network flow graph for the above problem is as follows:



b. Making a claim with the original problem statement:

The flow that has been built must have the flow across the min-cut with the following value in order to maximize the total quality minus the total price:

$$\sum_{i \in A} pi - qi + \sum_{j \in B} pi' - qi' + \sum_{i \in A, j \in B} Tij$$

$$Where\ the\ partitions\ created\ by\ the\ min-cut\ are\ A\ and\ B.$$

i.e.,

$$\sum_{i \in A} pi - qi + \sum_{j \in B} pi' - qi' + \sum_{i \in A, j \in B} Tij\ \ is\ minimized(\ from\ the\ min-cut)$$

Relating the above problem to the min-cut problem:

Given that we must maximize the qi-pi in the aforementioned query
In the mathematical format, we have to maximize the following,

$$Max(\sum_{i \in A}(qi - pi) + \sum_{j \in B}(qi' - pi') + \sum_{i \in A, j \in B} Tij)$$

From the above network flow that we have constructed, this can be discovered using the min-cut, we need to minimize/reduce the below following,

$$Min(\sum_{i \in A}(pi - qi) + \sum_{j \in B}(pi' - qi') + \sum_{i \in A, j \in B} Tij)$$

Therefore, the ultimate value—quality minus total price that we must minimize is as follows.

$$-(Min\left(\sum_{i \in A}(pi - qi) + \sum_{j \in B}(pi' - qi') + \sum_{i \in A, j \in B} Tij\right))$$

c. Proof for the above claim:

1. **Proving the claim in the forward direction. (=>)**

   Here, we need to assume that we are given an assignment such that the quality minus the total price of the purchasing the products is minimum and we need to find the min cut.

   We have constructed the graph in such a way that the source is connected to all the Microsoft product and the Apple Product nodes with the capacities, **pi-qi** and **pi'-qi'** respectively.

   Assuming the solution exists,

$$Max\left(\sum_{i \in A}(qi - pi) + \sum_{j \in B}(qi' - pi') + \sum_{i \in A, j \in B} Tij\right)$$

   If the above equation is maximized, then the below is minimized,

$$Min\left(\sum_{i \in A}(pi - qi) + \sum_{j \in B}(pi' - qi') + \sum_{i \in A, j \in B} Tij\right)$$

   The above equation is the definition of the min-cut.

2. **Proving the claim in the backward direction. (<=)**

Here, we assume that we are given the min cut, and we need to prove that there exists some assignment such that the quality minus the total price is maximized.

$$Min\left(\sum_{i \in A}(pi - qi) + \sum_{j \in B}(pi' - qi') + \sum_{i \in A, j \in B} Tij\right)$$

If the above equation is minimized, then it means that the below equation is maximized.

$$Max\left(\sum_{i \in A}(qi - pi) + \sum_{j \in B}(qi' - pi') + \sum_{i \in A, j \in B} Tij\right)$$

Since our goal is to maximize the quality minus the total price, this above equation is what we require to confirm that we have maximized the quality minus the total price.

Hence, we have proved our claim in both directions, which means that our claim and the network flow graph align with each other.