Homework: 03

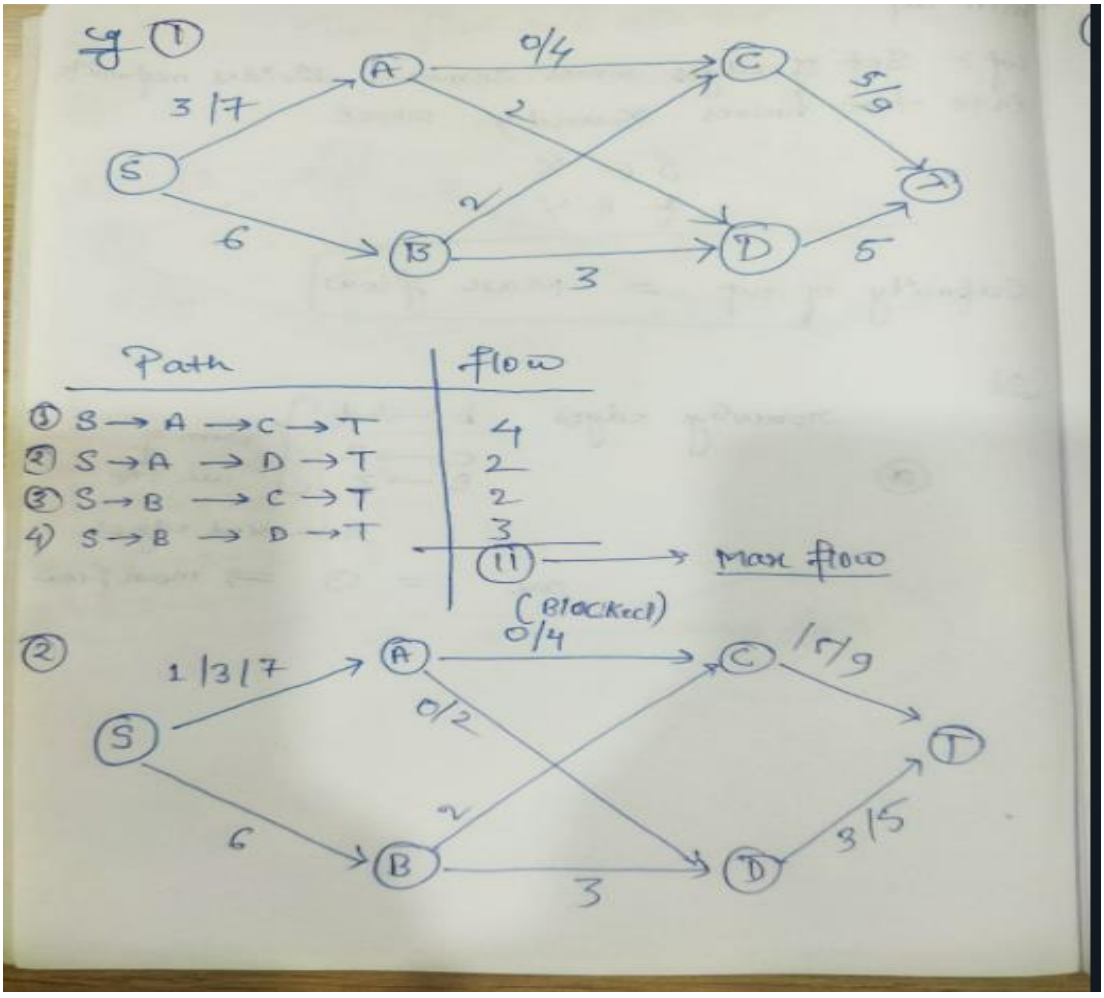
Name: Sudhir Sharma RollNo: 12041500 email: sudhirsharma@iitbhlai.ac.in
Collaborators Names:

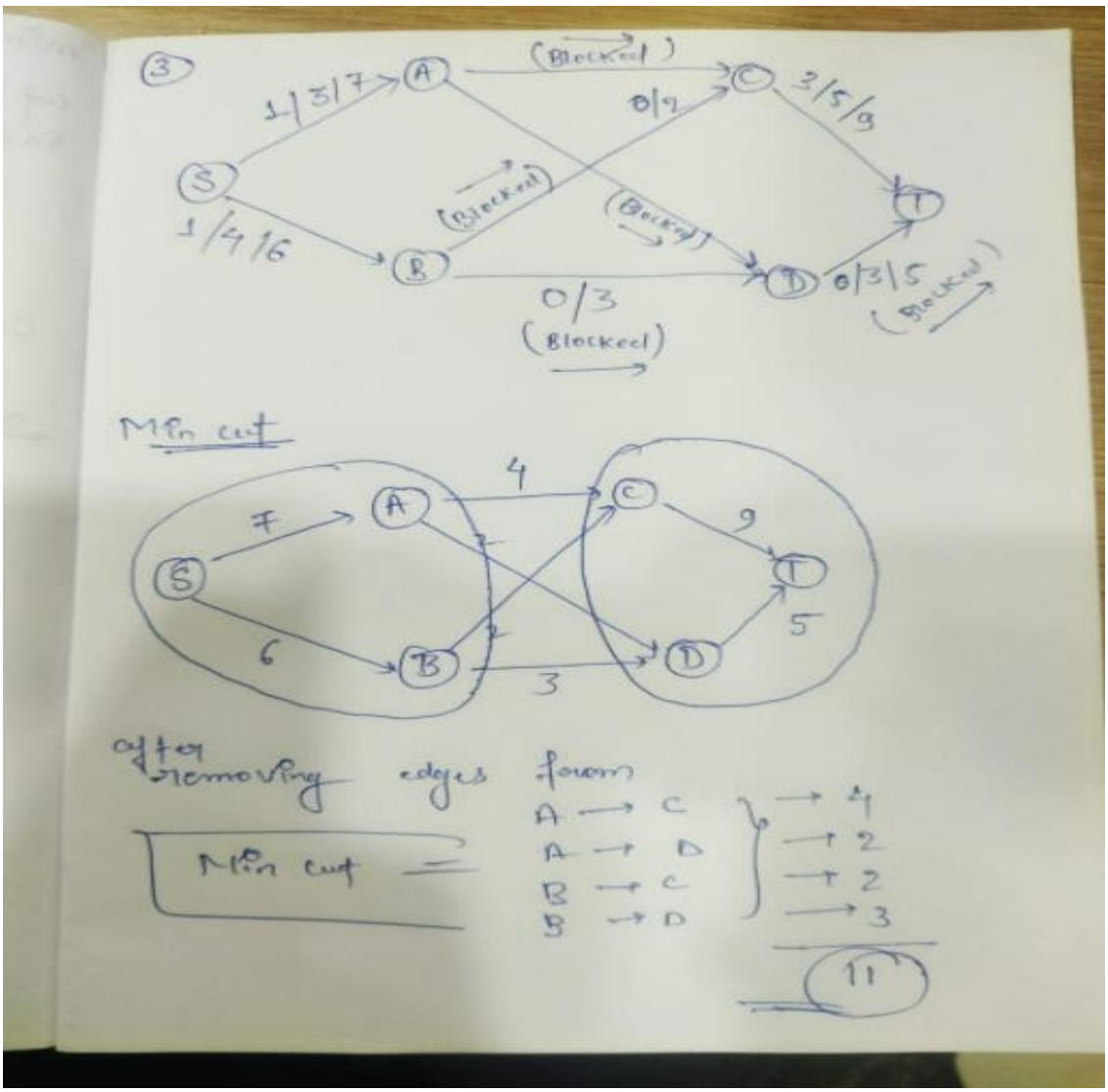
Solution of problem 1.

a.

Below image shows the max-flow as well as the min-cut. Here the max-flow = Total amount of flow pushed from source node s = 5+6 = 11 unit

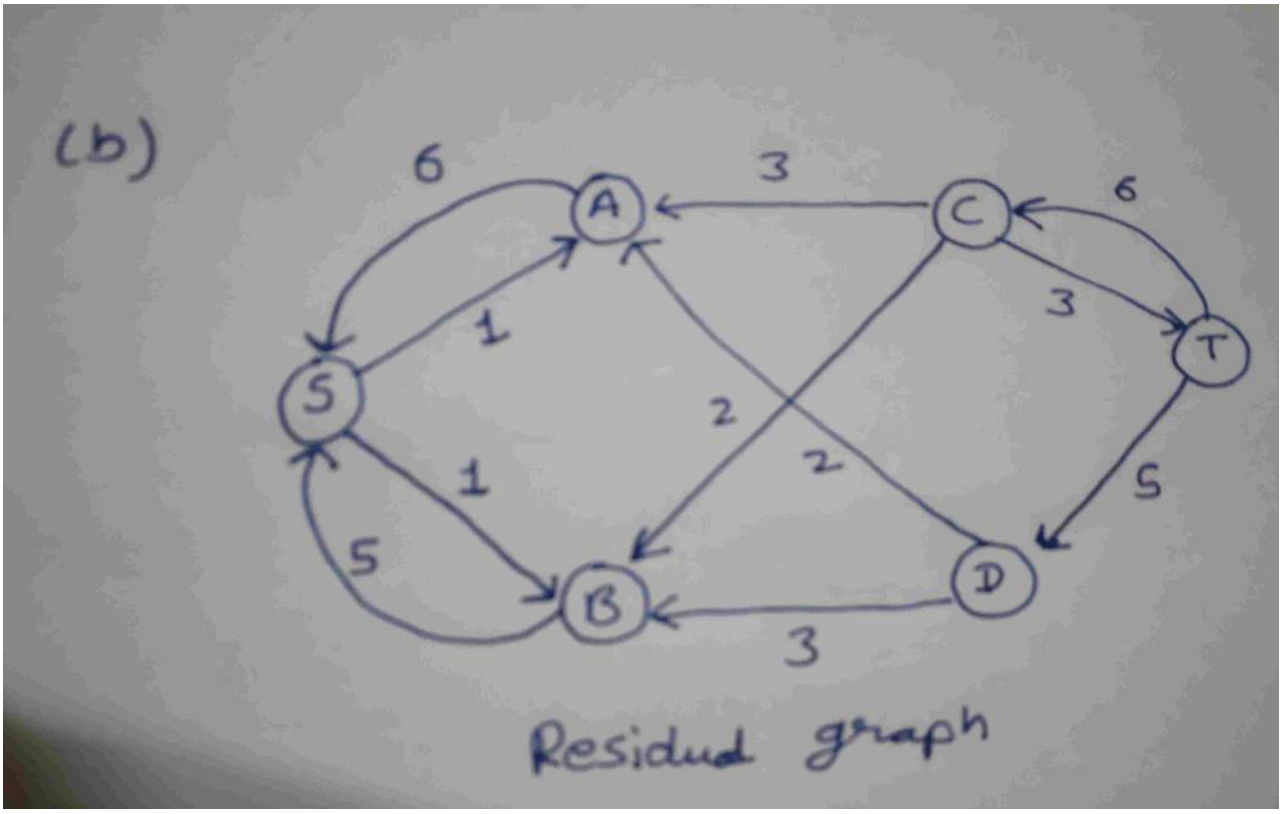
Min-cut is shown by dotted line and it consists of edges AC, AD, BC, BD whose total capacity = 4+2+2+3 = 11 unit





b.

Below image shows the residual graph of the above flow graph, such that for an edge (u,v) with flow f(u,v) and capacity c(u,v) , it is represented by backward edge (v,}u) with capacity f(u,v) and with forward edge (u,v) with capacity c(u,v)-f(u,v).



The set of vertices reachable from S are {S,A,B} since each of them has directed path from S in the residual graph and set of vertices from which T is reachable are {C,T} since each of them has directed path to T in residual graph.

c. Edge (u,v) will be bottleneck edge only if vertex u belongs to set of edges reachable from source and vertex v belongs to set of edges which has path to vertex T. Here the bottleneck edges are {(A,C), (B,C)}.

d. Consider the simple linear chain directed graph consists of vertex {S, A, B, T} and the edges {(S,A), (A,B), (B,T)} each of them having equal capacity c. Then clearly flow from S to T will be equal to c unit and all the edges will be completely saturated and none of them will be bottleneck edge because increasing capacity of only of them is not going to increase the amount of flow.

e. Bottleneck edges can be found by :-

1. Checking set of vertices S which are reachable from source vertex s in residual graph and checking set of vertices T which has path to sink vertex t.

2. List out all the edge (u,v) as bottleneck edges in original graph as bottleneck edges in which u belongs to S and v belongs to t.

Solution of problem 2.

Solution: (a) Algorithm

First, run the max flow algorithm. If an edge is critical, it must be filled to capacity, otherwise it would be possible to reduce its capacity to the amount of flow going through it without affecting the maximum flow. Then, for each edge (u, v) filled to capacity in the residual graph, run DFS and see if there is a path from u to v. If there isn't, then that edge is a critical edge.

(b) Pseudocode

```
define getCriticalEdge(G)
residualGraph = FordFulkerson(G)
    for edge (u, v) in residualGraph such that (v, u)
     is not in residualGraph and    (u, v) is not in G:
DFS(residualGraph, u)
    if u has no path to v:
return (u, v) return null
```

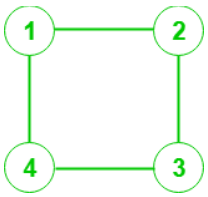
(c) Proof of Correctness

If an edge is not full when max flow is achieved, it is possible to reduce its capacity to the amount of flow going through it without affecting the maximum flow. By running the Ford-Fulkerson algorithm, we can determine which edges are full when the maximum flow is achieved. It is not enough for the edge to be full, however. We must check that the flow cannot be re-routed along another path in the residual graph. Suppose (u, v) is an edge that is full after the Ford- Fulkerson algorithm is run. If there is a path from u to v not using (u, v), then if the amount of flow allowed through (u, v) is decreased by 1, then the flow can just be routed from u to v through the other path, without affecting the max flow. If there is no path from u to v, then there is no way to re-route the flow that would have gone through (u, v), so the maximum flow must decrease. Therefore, full edges are only critical edges if there does not exist any path from u to v. Running DFS finds if there exists a path between u and v. (d) Running Time Analysis To run this algorithm, first you must run the Ford-Fulkerson algorithm, which takes $O(V E^2)$. Next, you must run DFS on at most $O(V)$ vertices, taking $O(V (V + E))$ time. This term is dominated by the running time of the Ford-Fulkerson algorithm, so the overall running time is $O(V E^2)$.

Solution of problem 3.

VERTEX COVER

A vertex cover of a graph can also more simply be thought of as a set of vertices of such that every edge of has at least one of member of as an endpoint.
The vertex set of a graph is therefore always a vertex cover. The smallest possible vertex cover for a given graph is known as a minimum vertex cover and its size is called the vertex cover number.



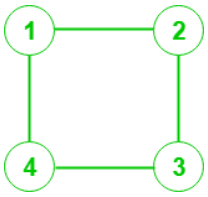
For above given graph G, Vertex cover is:

V_1 = {1, 3}, V_2 = {2, 4},
V_3 = {1, 2, 3}, V_4 = {1, 2, 3, 4}, etc.
Therefore, minimum number of vertices which can cover all edges, i.e., Vertex covering number $\beta_0(G) = 2$.

INDEPENDENT SET

- A set of vertices I is called independent set if no two vertices in set I are adjacent to each other or in other words the set of non-adjacent vertices is called independent set.
- It is also called a **stable set.**

- The parameter $\alpha_0(G) = \max \{ |I|: I \text{ is an independent set in } G \}$ is called **independence number** of G i.e the maximum number of non-adjacent vertices.
- Any independent set I with $|I| = \alpha_0(G)$ is called a maximum independent set.



For above given graph G, Independent sets are:

$I_1 = \{1\}$, $I_2 = \{2\}$, $I_3 = \{3\}$, $I_4 = \{4\}$
$I_5 = \{1, 3\}$ and $I_6 = \{2, 4\}$
Therefore, maximum number of non-adjacent vertices i.e Independence number $\alpha_0(G) = 2$.

An independent vertex set of a graph is a subset of the vertices such that no two vertices in the subset represent an edge of G.

CLIQUE

A **clique** is a subset of vertices of an undirected graph G such that every two distinct vertices in the clique are adjacent; that is, its induced subgraph is complete. Cliques are one of the basic concepts of graph theory and are used in many other mathematical problems and constructions on graphs.

A **maximal clique** is a clique that cannot be extended by including one more adjacent vertex, that is, a clique which does not exist exclusively within the vertex set of a larger clique.
A **maximum clique** of a graph, G, is a clique, such that there is no clique with more vertices. Moreover, the **clique number** ω(G) of a graph G is the number of vertices in a maximum clique in G.

Independent Set problem can be reduced to an instance of Vertex Cover problem in polynomial time.

Claim. VERTEX-COVER ≡P INDEPENDENT-SET.

Proof. We show S is an independent set iff V − S is a vertex cover.

Forward

Let S be any independent set.

Consider an arbitrary edge (u, v).

S independent ⇒ u ∉ S or v ∉ S ⇒ u ∈ V − S or v ∈ V − S.

Thus, V − S covers (u, v).

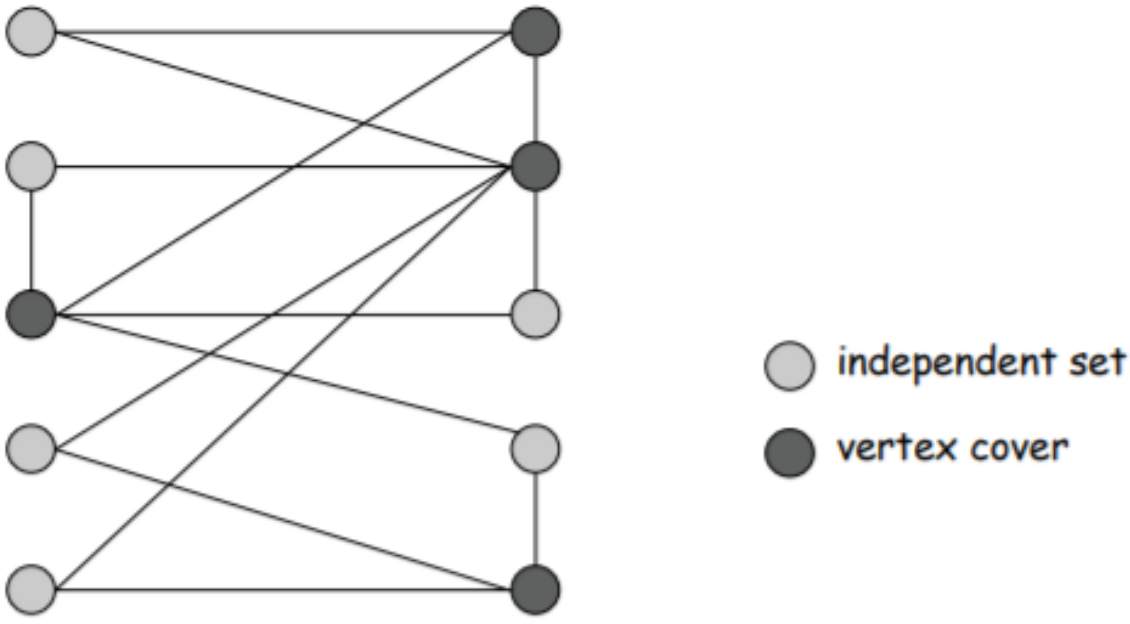
Backward

⇐ ν Let V − S be any vertex cover.

Consider two nodes u ∈ S and v ∈ S.

Observe that (u, v) ∉ E since V − S is a vertex cover.

Thus, no two nodes in S are joined by an edge ⇒ S independent set.



Connection between Clique, VC, and IS

For a graph G = (V, E), the following are equivalent

(i) S ⊆ V is an independent set in G.
(ii) S induces a clique in G(complement).
(iii) C = V \ S is a vertex cover for G.

Proof.

(i) ⇔ (ii): By definition of complement, uv ∈ E ⇔ uv not belongs to E(complement).

(ii) (i) → (iii): Assume C is not a vertex cover. Then, there is an edge uv with u, v not belongs to C. Thus, u, v ∈ S. This is in contradiction with S being an independent set.

(iii) (iii) → (i): Assume S is not an independent set. Then, there is an edge uv with u, v ∈ S. Thus, u, v not belongs to C. This is in contradiction with C being a vertex cover.