Flow networks and flows:

A flow network $G = (V, E)$ is a directed graph in which each edge $(u, v) \in E$ has a nonnegative capacity $c(u, v) \geq 0$. We further require that if E contains an edge (u, v), then there is no edge (v, u) in the reverse direction. If $(u, v) \notin E$, then for convenience we define $c(u, v) = 0$, and we disallow self-loops. We distinguish 2 vertices in a flow network: a source s and a sink t. For convenience, we assume that each vertex lies on some path from the source to the sink. That is, for each vertex $v \in V$, the flow network contains a path $s \to v \to t$.

Flow.

Let $G = (V, E)$ be a flow network with a capacity function c. Let s be the source of the network, and let t be the sink. A flow in G is a real-valued function $f : V \times V \to R$ that satisfies the following 2 properties:

1) Capacity constraint: For all $u, v \in V$, we require $0 \leq f(u,v) \leq c(u,v)$.

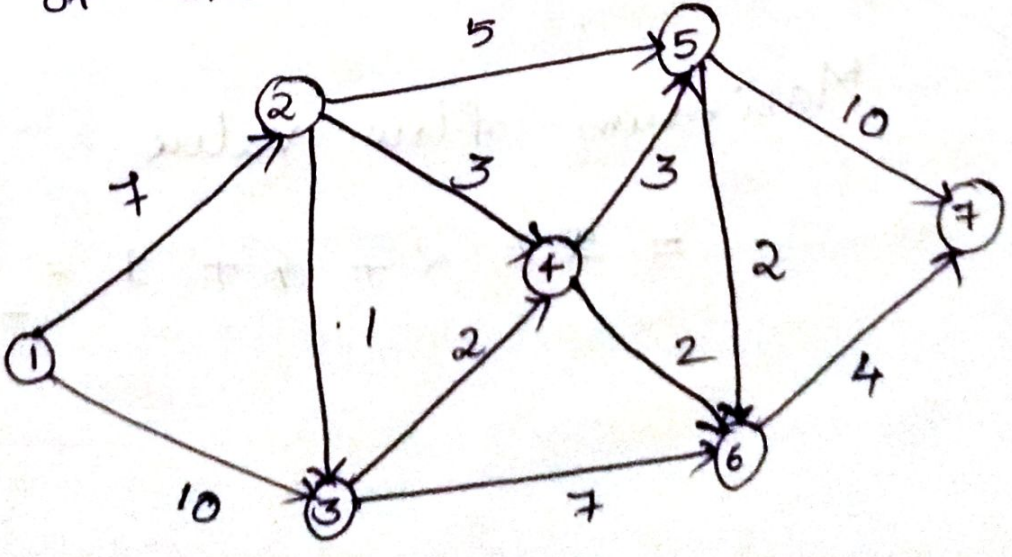
2) Flow conservation: For all $u \in V - \{s,t\}$, we require

$$\sum_{v \in V} f(u,v) = \sum_{v \in V} f(u,v).$$

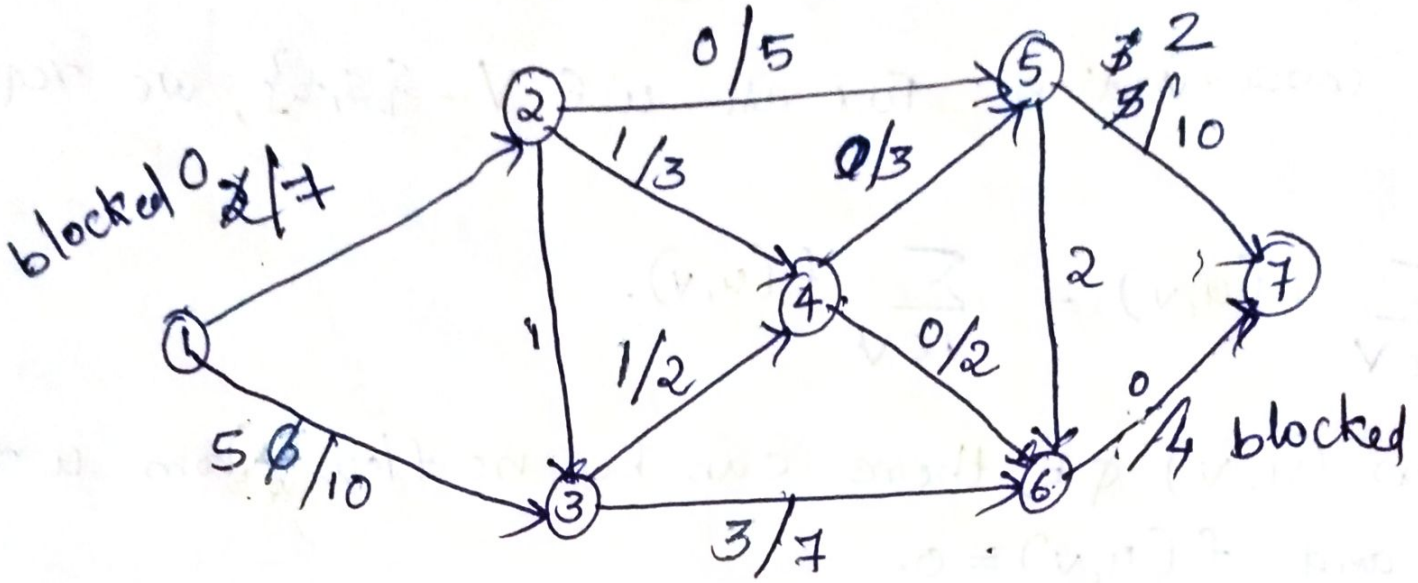
When $(u,v) \notin E$, there can be no flow from u to v, and $f(u,v) = 0$.

In the maximum-flow problem, we are given a flow network G with source s and sink t, and we wish to find a flow of maximum value. The capacity constraint simply says that the flow from one vertex to another must be nonnegative and must not exceed the given capacity. The flow conservation properly says that the total flow into a vertex other than the source or sink must equal the total flow out of that vertex ie, flow in equals flow out".

eg:



Finding the maximum flow.



Path	flow
① 1-2-5-10	5
② 1-2-4-5-7	2
③ 1-3-6-7	4
④ 1-3-4-5-7	1
	12

Maximum flow value

$= 5 + 2 + 4 + 1 = 12$

Ford Fulkerson Algorithm for Maximum Flow problem.

Problem

Given a graph which represents a flow network where every edge has a capacity. Also given two vertices Source S and Sink t in the graph. Find out the maximum possible flow from s to t with following constraints.

(a) Flow on an edge doesn't exceed the given capacity of the edge.

(b) In-flow is equal to out-flow for every vertex except s and t.

Algorithm:

Ford - Fulkerson Algorithm.

The following is a simple idea of the algorithm.

① Start with an initial flow as zero.

② While there is an augmenting path from Source to sink.
Add this path flow to flow

③ Return flow.

FORD-FULKERSON-METHOD (G, s, t)

1. initialize flow f to 0
2. while there exists an augmenting path p in the residual network G_f
3. augment flow f along p
4. return f.

Terminologies

* Residual Graph - It is a graph which indicates additional possible flow. If there is such path from source to sink then there is a possibility to add flow Denoted as G_f.
* Residual Capacity - It is original capacity of the edge minus flow. Denoted as C_f
* minimal cut - Also known as bottle-neck capacity which decides maximum possible flow from source to sink through an augmented path.

* Augmenting path - Augmenting path can be done in 2 ways

① non-full forward edges

② non-empty backward edges.

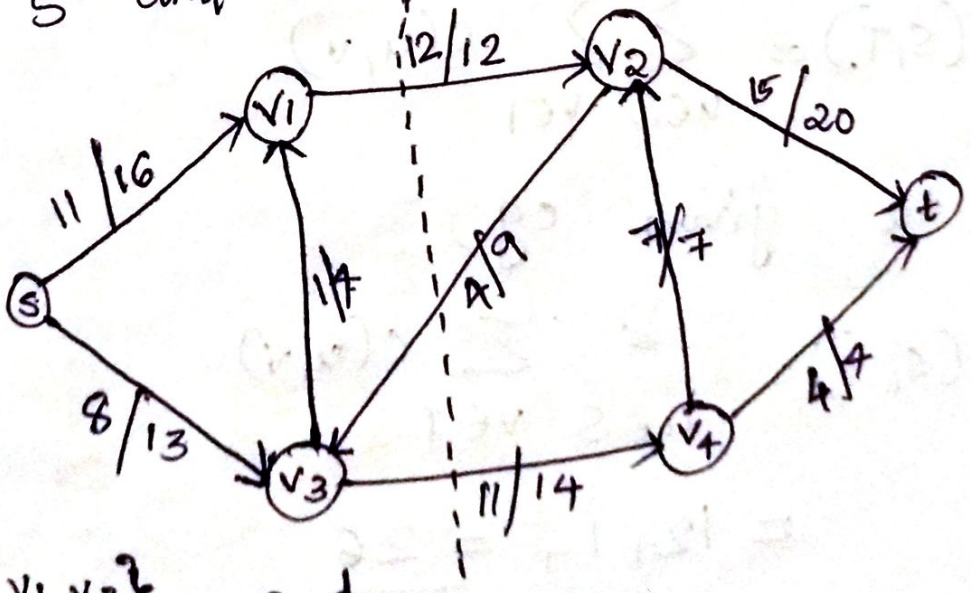
Given a flow network $G = (V, E)$ and a flow f, an augmenting path p is a simple path from s to t in the ~~xed~~ residual network G_f.

Residual capacity of p given by

$$c_f(p) = \min \{ c_f(u, v) : (u, v) \text{ is on } p \}.$$

* Cuts of flow network - (S, T) cut.

A cut (S, T) of flow network $G = (V, E)$ is a partition of V into S and $(T - S)$ such that $s \in S$ and $t \in T$.



$S = \{ s, v_1, v_3 \}$ and

$T = \{ v_2, v_4, t \}$

If f is a flow, then the net flow $f(S,T)$ across the cut (S,T) is defined to be

$$f(S,T) = \sum_{u \in S} \sum_{v \in T} f(u,v) - \sum_{u \in S} \sum_{v \in T} f(v,u).$$

In given fig:, the net flow across (S,T) is $f(S,T) = 19$.

ie,

$$f(S,T) = \sum_{u \in S} \sum_{v \in T} f(u,v) - \sum_{u \in S} \sum_{v \in T} f(v,u)$$

$$= 12 + 11 - 4 = 23 - 4 = 19$$

The Capacity of the cut (S,T) is

$$c(S,T) = \sum_{u \in S} \sum_{v \in T} c(u,v)$$

For the given eg:

$$c(S,T) = \sum_{u \in S} \sum_{v \in T} c(u,v)$$

$$= 12 + 14 = 26$$

Max-flow Min-cut theorem

If f is a flow in a flow network $G = (V, E)$ with source s and sink t, then the following conditions are equivalent:

1. f is a maximum flow in G
2. The residual graph or residual network G_f contains no augmenting paths.
3. $|f| = c(S, T)$ for some cut (S, T) of G.

We want to prove,

$1 \Leftrightarrow 2 \Leftrightarrow 3 \Leftrightarrow 1$

ie, In a network flow, maximum amount of flow passing from source to sink is equal to the minimum of the capacities of all possible cuts.

proof: Let N be a given network and 'S' be the source and 'T' be the sink.

Let S-T be any cut in the network, so in any S-T of the network there will be an edge from the path connecting source and sink.

Thus every flow from Source to Sink must pass through one or more edges of S-T. Hence the total flow between source and Sink can't exceed the minimum capacity of S-T.

Since this is true for every cut the flow can't exceed but can be equal to the minimum of the capacities of all the possible cuts Hence the proof.

The basic Ford-Fulkerson algorithm.

FORD - FULKERSON (G, s, t).

1. for each edge $(u,v) \in G.E$
2. $\quad (u,v).f = 0$
3. While there exists a path p from s to t in the residual network G_f
4. $\quad c_f(P) = \min\{c_f(u,v) : (u,v) \text{ is in } P\}$
5. $\quad$ for each edge (u,v) in p
6. $\quad\quad$ if $(u,v) \in E$
7. $\quad\quad\quad (u,v).f = (u,v).f + c_f(P)$
8. $\quad\quad$ else $(v,u).f = (v,u).f - c_f(P)$.

Lines 1-2 initialize the flow f to 0. The while loop of lines 3-8 repeatedly finds an augmenting path p in G_f and augments flow f along p by the residual capacity $c_f(p)$. Each residual edge in path p is either an edge in the original network or the reversal of an edge in the original network. Lines 6-8 update the flow in each case appropriately, adding flow when the residual edge is an original edge and subtracting it otherwise. When no augmenting paths exist, the flow f is a maximum flow.

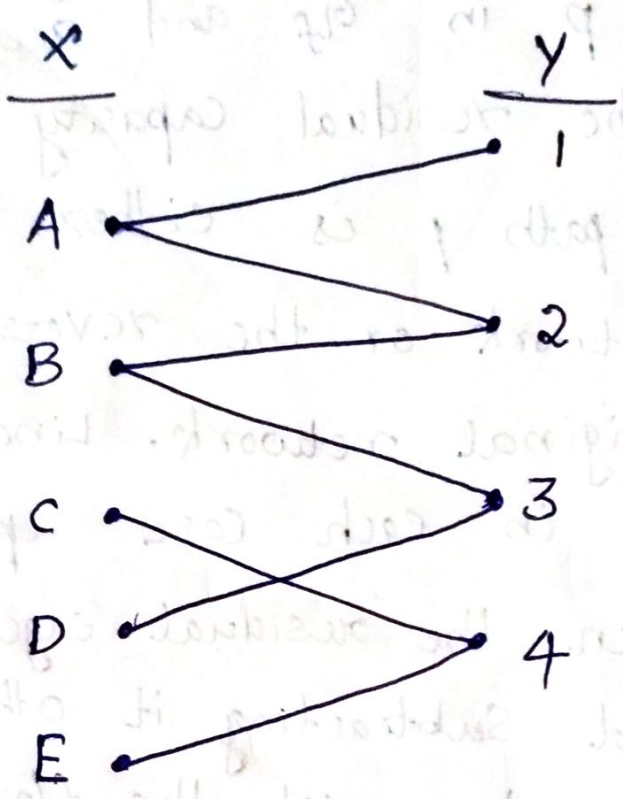
Bipartite Matching

Bipartite graph

Graph $G=(V, E)$, in which the vertex Set V is divided into 2 disjoint sets X and Y. Every edge of the graph has one end point in X and other end point in Y. And vertices of the same set are not connected.

ie, $X \cup Y = V$ and $X \cap Y = \phi$.

Example



$V = \{A, B, C, D, E, 1, 2, 3, 4\}$

$X = \{A, B, C, D, E\}$

$Y = \{1, 2, 3, 4\}$

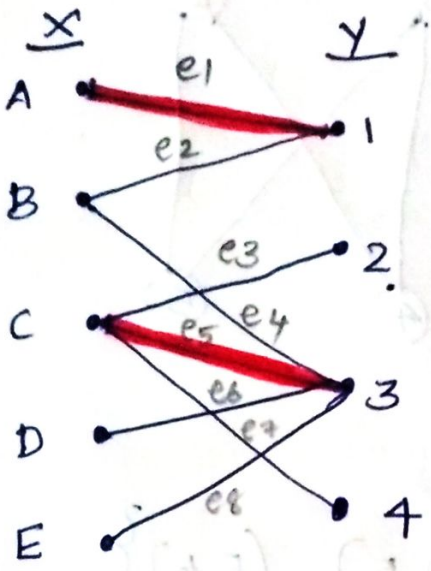
$X \cup Y = \{A, B, C, D, E, 1, 2, 3, 4\}$

$X \cap Y = \phi$

Matching (M) - pairing.

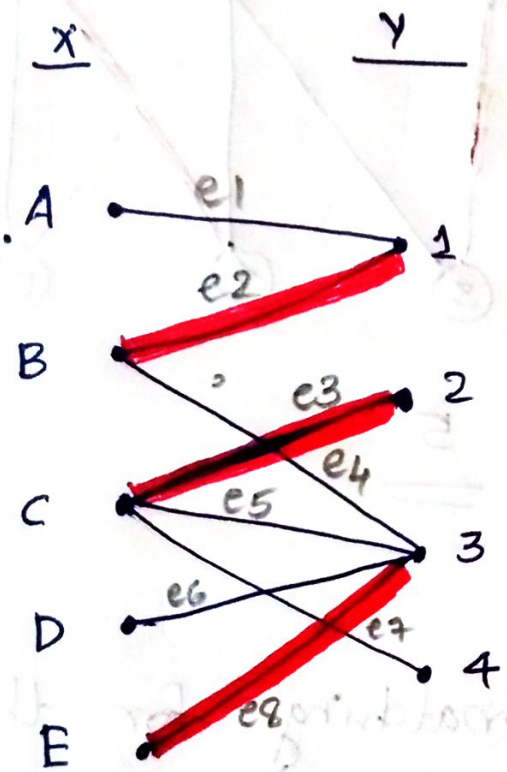
Matching in a graph is a "subset of edges" that no two edges share a vertex.

Examples

① 

← Matching with cardinality 2

$M = \{e_1, e_5\}$.

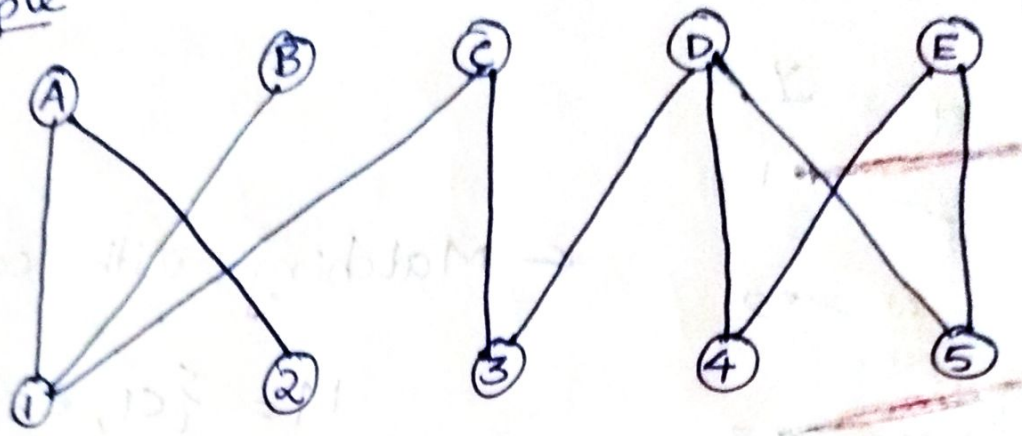
② 

← Matching with cardinality 3

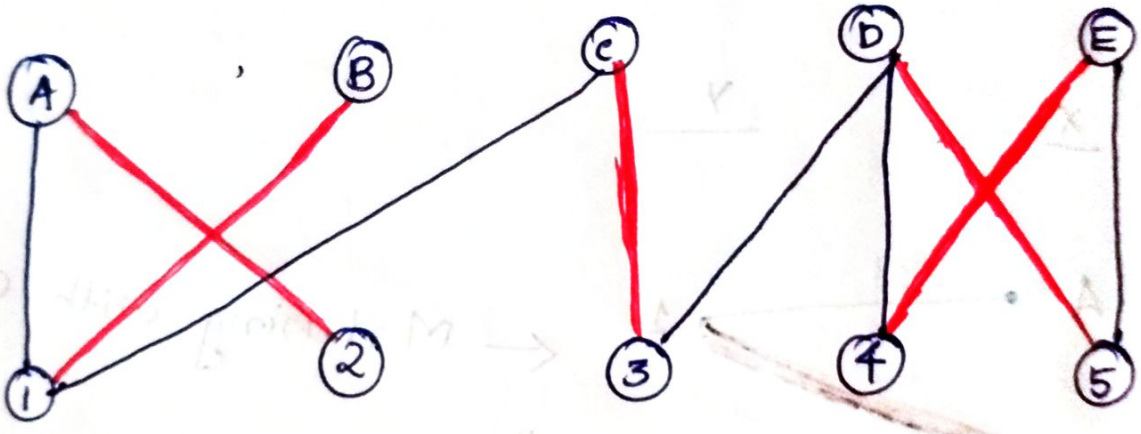
$M = \{e_2, e_3, e_8\}$.

Maximum matching in bipartite graph problem is to find out the maximum matching (M) edges from the graph. We can use the Ford - Fulkerson method to find a maximum matching in an undirected graph $G = (V, E)$.

Example



$M = \{(A,2), (B,1), (C,3), (D,5), (E,4)\}$



M with cardinality 5

① Find the maximum matching for the following bipartite graph.