A1 MAX-FLOW MIN-CUT THEOREM

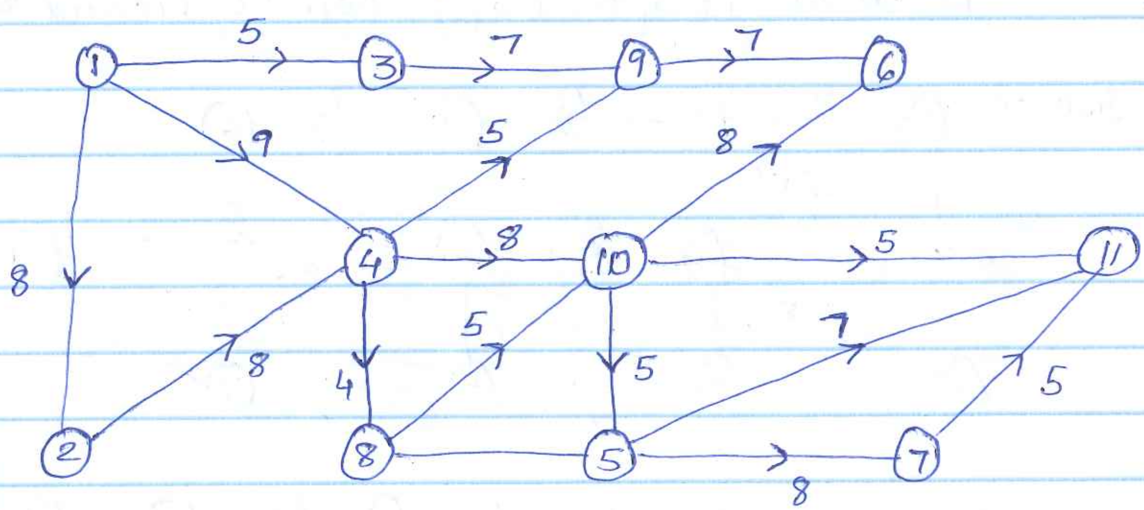
If 'f' is a flow in a flow network $G=(V,E)$ with source 's' and sink 't', then the following conditions are equivalent :→

(A) 'f' is a maximum flow in G

(B) The residual network G_f contains no augmenting paths.

(C) $|f| = c(S,T)$ for some cut (S,T) of G

We will explain the above theorem using the ~~graph~~ following graph :



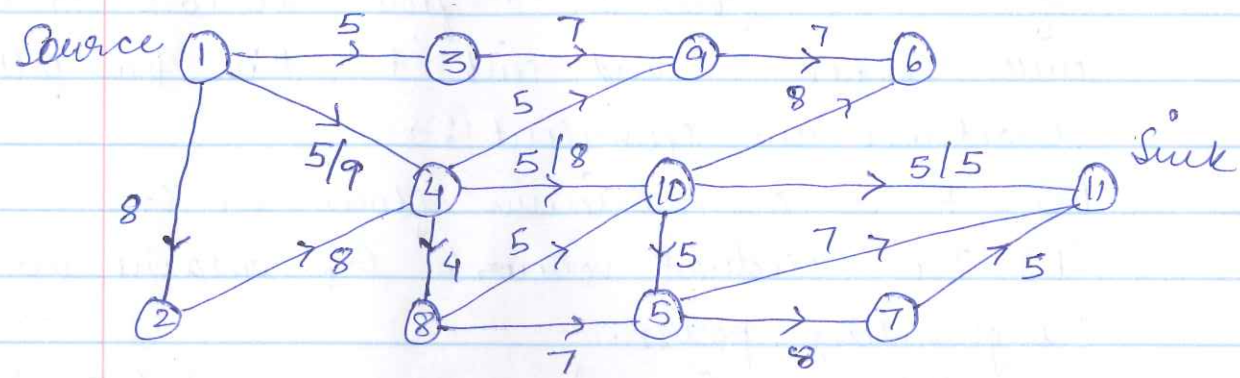
The directions and capacities for the graph have been mentioned on the graph itself.

Let source for this graph be 1 and sink for the graph be 11

Now, we will deduce max flow for the graph

Consider path ① → ④ → ⑩ → ⑪

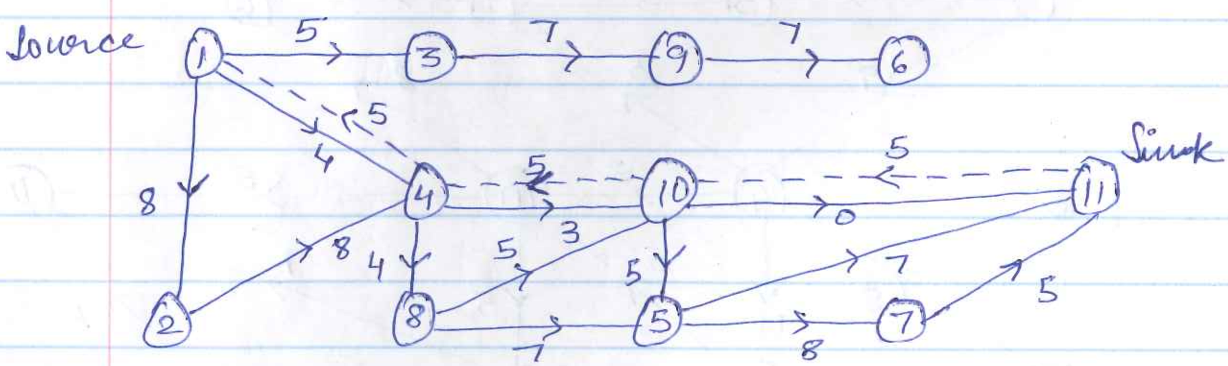
Flow b/w edges is represented as flow/capacity



Bottleneck for the above path is ⑩ → ⑪

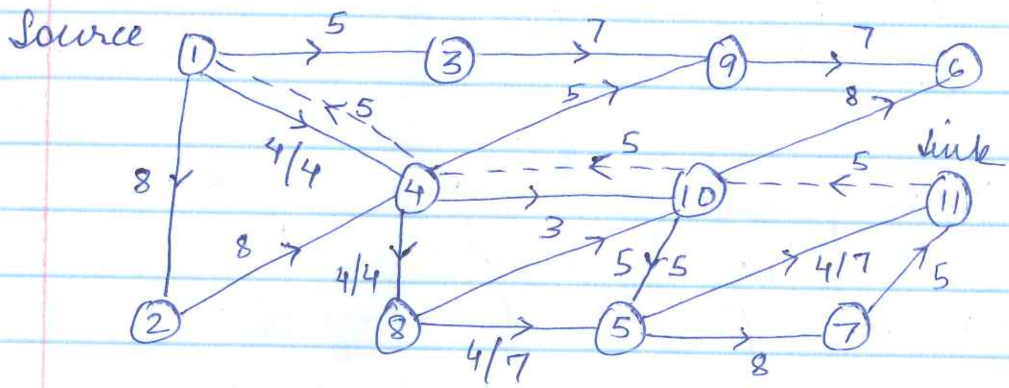
Hence flow through this path = 5

Residual Graph (G_f) can be viewed as:-



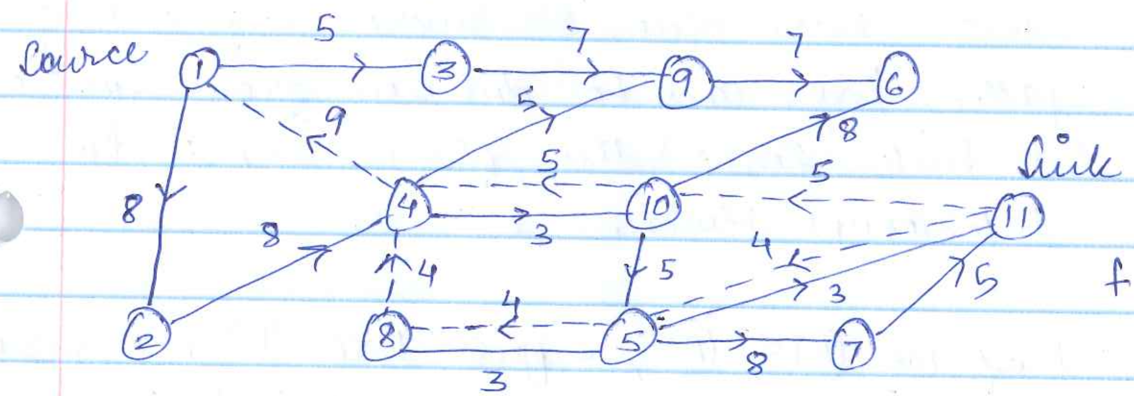
An Augmenting Path ① → ④ → ⑧ → ⑤ → ⑪ can be obtained from the Residual graph. Hence the current flow of 5 can be increased by following the augmenting path.

Augmenting Graph:→



New flow
= 5 + 4
= 9

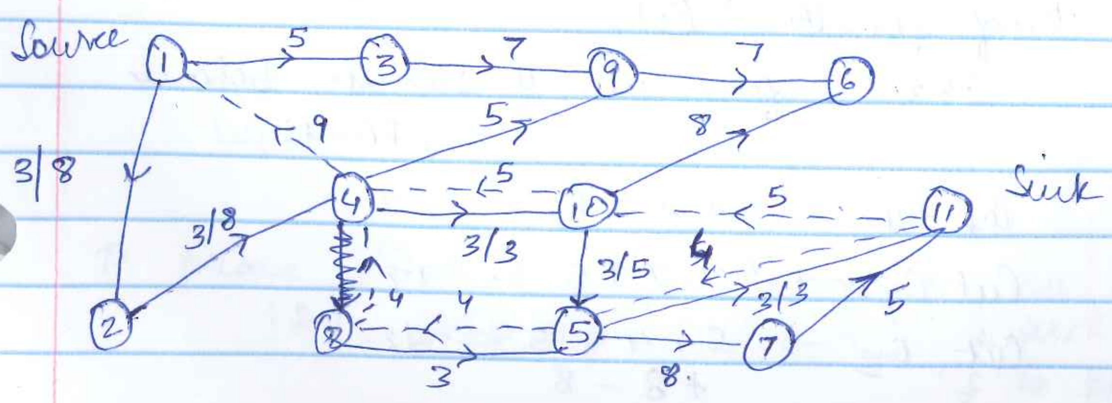
Residual Graph (G_f) :→



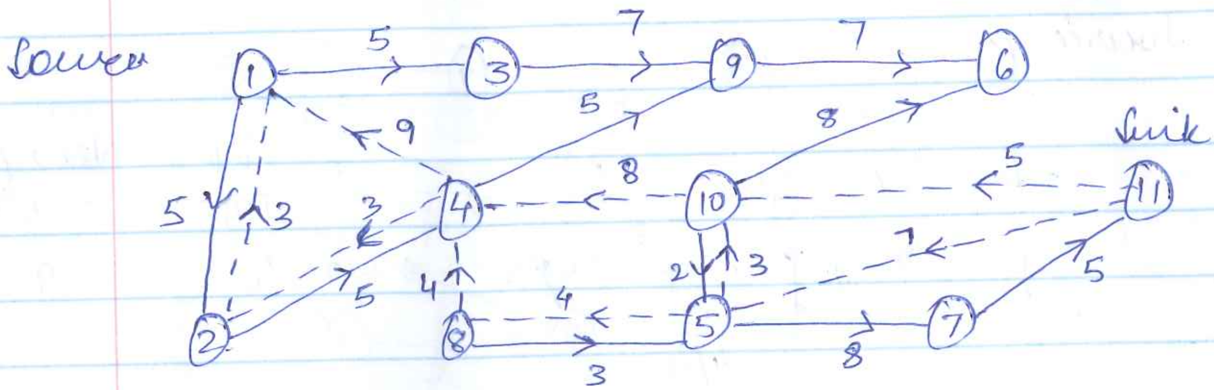
$f = 5 + 4 = 9$

New Augmenting path: 1 → 2 → 4 → 10 → 5 → 11
Bottleneck edge = 4 → 10 and 5 → 11
flow = 3

Augmenting Graph :→



Residual Graph (G_f)



New flow = 5 + 7 = 12

Since, there are no more augmenting paths that can be drawn from Source to Sink, hence the flow (12) is the MAXIMUM Flow.

Proof for (A) ⇒ (B), Suppose that 'f' is a max flow in G but Residual graph (G_f) has an augmenting path. Let its flow be f_p. Then by eq → $|f \uparrow f_p| = |f| + |f_p| > |f|$

Hence, 'f' is no more the max flow in G. Which is a contradiction. Hence A ⇒ B holds

Proof for (B) ⇒ (C)

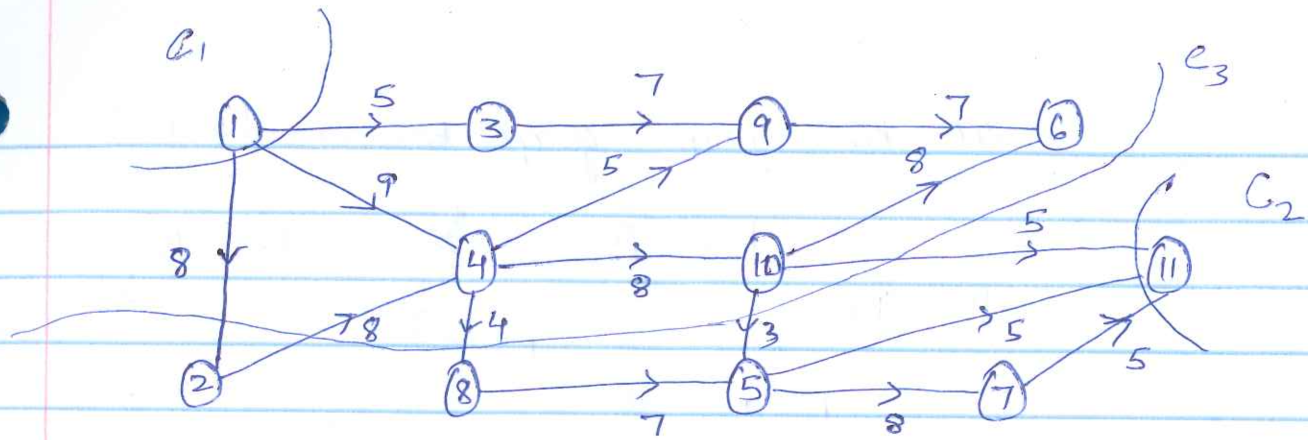
Graph for cuts is shown below

↓ Flows

Cut C_1 → 5 + 9 + 8 = 22

Cut C_2 → 5 + 7 + 5 = 17

Cut C_3 → 5 + 4 + 3 + 8 − 8 = 12



Min Cut $= 12$

Suppose no augmenting path in G_f and G_f contains no path from s to t

Let $S = \{ v \in V :$ paths exist from 1 to 11 in $G_f \}$ and ~~T_f~~ $T = V - S$.

Partition C_3 is a cut where $1 \in S$ & $11 \notin S$ because there is no path from ① to ⑪ in G_f

Let vertex $u \in S$ & $v \in T$

If $(u,v) \in E, \Rightarrow f(u,v) = c(u,v)$

If $(v,u) \in E, \Rightarrow f(v,u) = 0$

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

$$= \sum_{u \in S} \sum_{v \in T} c(u,v) - \sum_{v \in T} \sum_{u \in S} 0$$

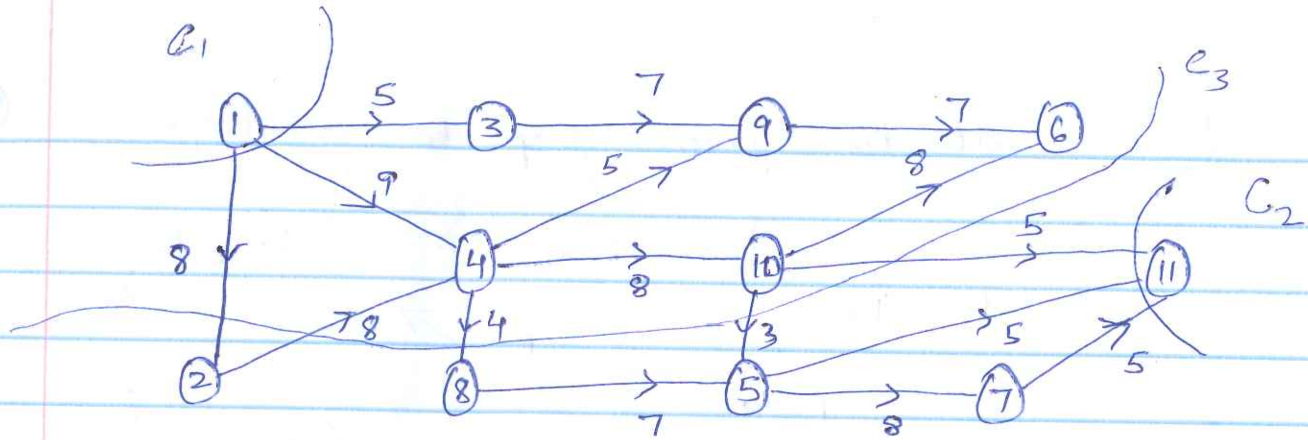
$$= c(S,T)$$

Hence $|f| = f(S,T) = c(S,T)$

Which Proves Ⓑ $\Rightarrow$ Ⓒ

To prove Ⓒ $\Rightarrow$ Ⓐ, for all cuts, we have

$|f| = \cancel{f(S,T)} = c(S,T)$ which implies



Min Cut = 12

Suppose no augmenting path in G_f and G_f contains no path from s to t

Let $S = \{ v \in V :$ paths exist from 1 to 11 in G_f
and ~~TTT~~ $T = V - S$.

Partition C_3 is a cut where $\not\, \in S$ & $11 \notin S$
because there is no path from ① to ⑪ in G_f

Let vertex $u \in S$ & $v \in T$

$$\text{if } (u,v) \in E, \Rightarrow f(u,v) = c(u,v)$$
$$\text{if } (v,u) \in E, \Rightarrow f(v,u) = 0$$

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

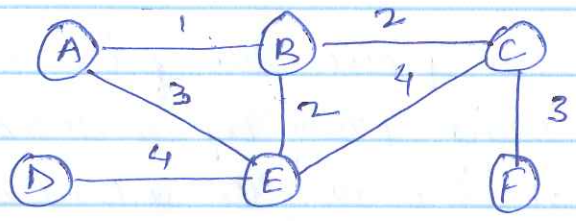
$$= \sum_{u \in S} \sum_{v \in T} c(u,v) - \sum_{v \in T} \sum_{u \in S} 0$$

$$= c(S,T)$$

Hence $|f| = f(S,T) = c(S,T)$
Which Proves (B) $\Rightarrow$ (C)

To prove (C) $\Rightarrow$ (A), for all cuts, we have
$|f| = \cancel{f(S,T)} = c(S,T)$ which implies

A2 Consider the following graph with edge weights ranging from 1 to 4.



To find a 4-approximation MST for the above graph, we will take the help of Breadth First Search (BFS) algorithm and show that the algorithm discussed is gaurantees an optimal solution within a factor of 4.

Let the weights of edges range from 1 to w

APPROXIMATE_MST (G, w)

r = random{V} // Select a random vertex to start the algo

APPROXIMATE_MST = ϕ // Initially the MST starts as null

Perform BFS from every vertex

If u not in S

APPROXIMATE_MST = APPROXIMATE_MST U(v.π, u)

CORRECTNESS OF THE ABOVE ALGORITHM

Claim:→

Let the optimal solution for the graph be OPT.

Let the solution obtained by APPROXIMATE_MST be AMST

Then, the following ~~eq~~ equation can be obtained :-

$$OPT \leq AMST \leq 4 \cdot OPT \quad — ①$$

PROOF :- Let us prove the first part of equation ① using proof by contradiction. Assume that the weights obtained using APPROXIMATE_MST procedure are less than that obtained through the OPT solution for MST. This means that the approximate algorithm is better than OPT and hence must be the OPTIMAL SOLUTION. This cannot be true, as we already have the optimal solution OPT with us.

Hence $OPT \leq AMST$ holds true.

For the second part of equation ①, we can show that $\frac{AMST}{OPT} \leq 4$, where 4 is '2' → also known as the approx. factor.

MST obtained by OPT for the graph mentioned above is 5.

By using APPROXIMATE_MST, we can deduce the MST in atmost 16. Hence, the approximation factor thus obtained will be $\frac{16}{5} = 3.~~2~~$

Hence, $\frac{AMST}{OPT} = 3 \cdot 6$

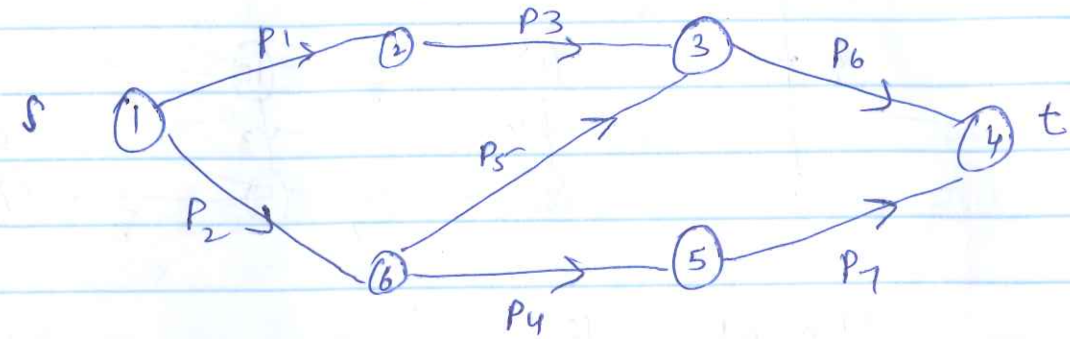
$$\Rightarrow \frac{AMST}{OPT} \leq 4$$

$$\Rightarrow AMST \leq 4 \cdot OPT.$$

Which proves our 2nd part of the equation.
Hence, ~~the~~ our algorithm is correct.

A3

Let the given graph be $G = (V, E)$



Let $p_1 \ldots p_7$ be the probability associated with each edge denoted by $r(u, v)$ from vertex 'u' to vertex 'v'

$$0 \leq r(u, v) \leq 1$$

Now, to find a reliable path b/w source 's' and sink 't' such that the probabilities are maximized, we can run Dijkstra's algorithm on this but before doing so, we must convert it into a shortest path problem. Take the inverse of the reliability value :-

$$\text{Let } w(u, v) = \frac{1}{r(u, v)}$$

The above problem can now be solved using Dijkstra's algorithm and since $0 \leq r(u, v) \leq 1$, there won't be any negative weights involved.

ALGORITHM

Let $w(u, v)$ be $\frac{1}{r(u, v)}$

Run Dijkstra' Algo on $w(u,v)$
Initialize-Single-Source (G, s)
$S \leftarrow nil$
$Q \leftarrow V[G]$
while Q is not nil
    $u \leftarrow$ Extract-min (Q)
    $S \leftarrow S \cup \{u\}$
    for each vertex $v \in Adj[u]$
        Relax (u, v, w)

TIME COMPLEXITY

Initialization $= O(E)$ time
Assuming the graph is sparse and all the vertices are reachable from source,

Dijstra takes $= O((V+E)\log V + E)$

Time Complexity $= O((V+E)\log V + E) + O(E)$
$= O(E \log V)$

$$\boxed{\text{Time Complexity} = O(E \log V)}$$

where $E = o\left(\frac{V^2}{\log V}\right)$

A4

Let us store the Transitive Closure as ~~graph~~ Graph $G^*(V, E^*)$ where $E^* = (i, j)$: there is a path from vertex i to vertex j in G.

We can compute the Transitive Closure and store it as an adjacency matrix TC[i][j]

For each vertex 'u'
    perform BFS with 'u' as root
    for every vertex 'v' discovered in the search, set TC[i][j] = 1
return TC

Since time complexity of BFS for 1 vertex is $O(E)$, performing the same operations ~~all~~ ~~ear~~ for all the vertices $v \in V$, we get the time complexity as $\underline{O(VE)}$

for $i = 1$ to n // Table for already connected vertices in G
    for $j = 1$ to n
        $t[i][j] = \begin{cases} 1 & \text{if } (v_i, v_j) \in G \\ 0 & \text{Otherwise} \end{cases}$

for $k = 1$ to n // Table to add newly discovered vertices
    for $i = 1$ to n
        for $j = 1$ to n
            $t[i][j] = t[i][j] \vee (t[i][k] \wedge t[k][j])$