**FLOW ALGORITHMS**

BPM

1. O(E \* V) Solution

//Implementationof BPM O(E \* V)

//matchL[] keeps who are connected with Right side nodes

//matchR[] keeps who are connected with Left side nodes

//visited[] keeps who are visited

int matchL[MAX+5],matchR[MAX+5];

int visited[MAX+5];

bool dfs\_BPM\_EV(int u)

{

visited[u] = 1;

for(int i=0;i<graph[u].size();i++)

{

int v = graph[u][i];

if(matchR[v] == -1)

{

matchL[u]=v;

matchR[v]=u;

return 1;

}

else if(visited[matchR[v]]==0 && dfs\_BPM\_EV(matchR[v])==1)

{

matchL[u] = v;

matchR[v] = u;

return 1;

}

}

return 0;

}

int BPM\_EV(int m, int n)

{

//initialization

for(int i=1;i<=m;i++)

{

matchL[i] = -1; //they are connected with none

}

for(int i=1;i<=n;i++)

{

matchR[i] = -1; //they are connected with none

}

int result = 0;

/\*

Try with all nodes and run dfs so that there can be matched with

matchL[] array and matchR[] array

The solution is something like that

Start with a node and run dfs with that such that the DFS is

always running with left side nodes

liearly traversing so it can be said running with free nodes

then trying from right side match nodes like dfs(matchR[v])

if matchR[v] == -1 then it's a free node we get an augmenting path

which will increase the matching by one

else traverse

if by traversing we don't do dfs with same nodes twice

else traverse from right side dfs(matchR[v])

if reached it's a free node but in left side so no solution

else it's in right side and it's give a solution

return 1 else return 0

by counting them get's a solution

\*/

for(int i=1;i<=m;i++)

{

for(int j=1;j<=m;j++)

{

visited[j] = 0;

}

bool counter = dfs\_BPM\_EV(i);

result = result + counter;

}

return result;

}