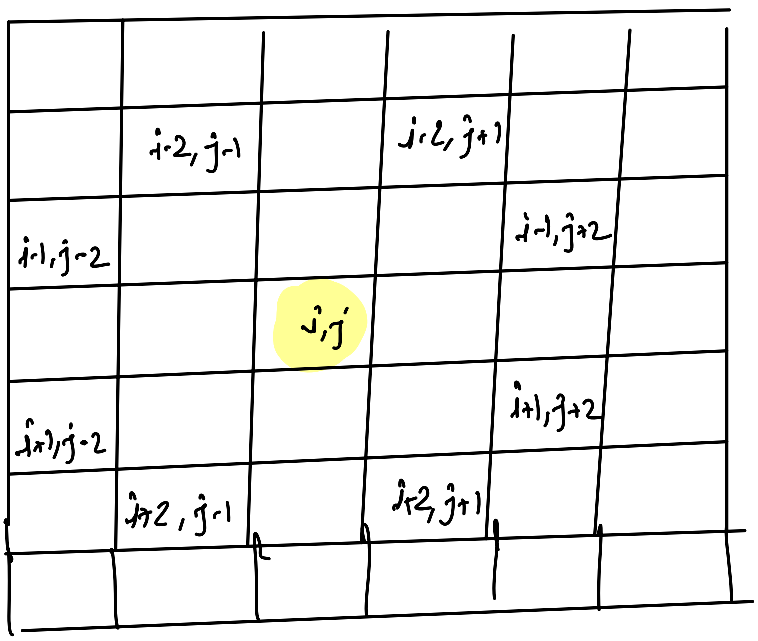
①

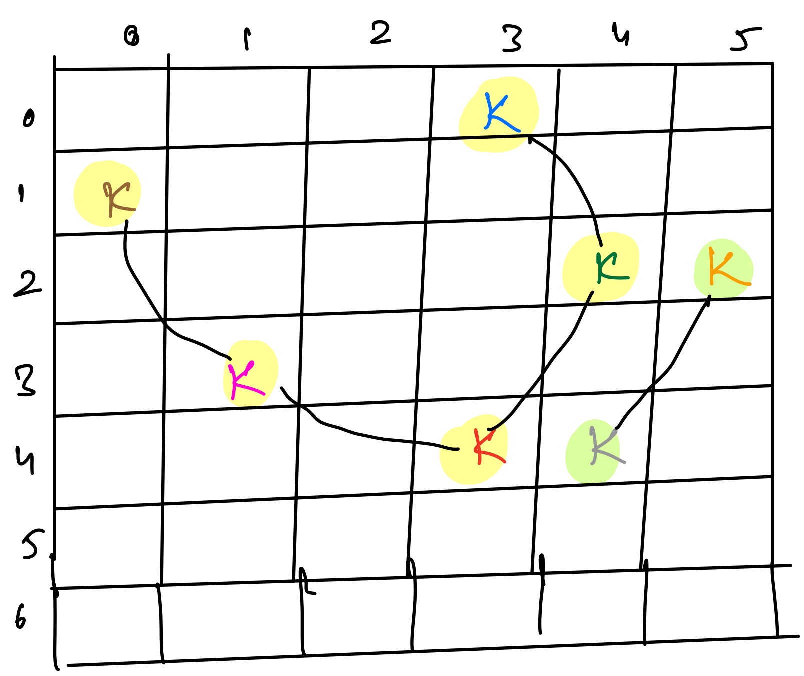
Given a N * N chessboard with K Knights placed on it.

If a Knight is reachable from the other Knight, they can swap their positions

Find the number of ways the Knights can rearrange themselves



Eg.-



K - Knights

K = 7.

$5! \times 2! \Rightarrow 240$

Ways = $Count(CC_1)! \times Count(CC_2)! \times Count(CC_3)! \times \ldots$

Q2. Given a N * M Matrix, the cells of the matrix is either marked as R (Residence) or H (Hospital)

For every residence, find the distance to the nearest Hospital.

From a particular cell, you can move to any adjacent cell (diagonal moves not allowed).

R_1	R_2	R_3	H_1
R_4	R_5	H_2	H_3
R_6	H_4	H_5	R_7

$\Rightarrow$

3	2	1	0
2	1	0	0
1	0	0	1

ans

B.F idea. → For every residence, try to find nearest hospital by exploring all possible ways.

T.C → $O(N^2 M^2)$

Idea.2. → Multi-sourced B.F.S.

	0	1	2	3
0	~~R_1~~ 3	~~R_2~~ 2	~~R_3~~ 1	H_1
1	~~R_4~~ 2	~~R_5~~ 1	H_2	H_3
2	~~R_6~~ 1	H_4	H_5	~~R_7~~ 1

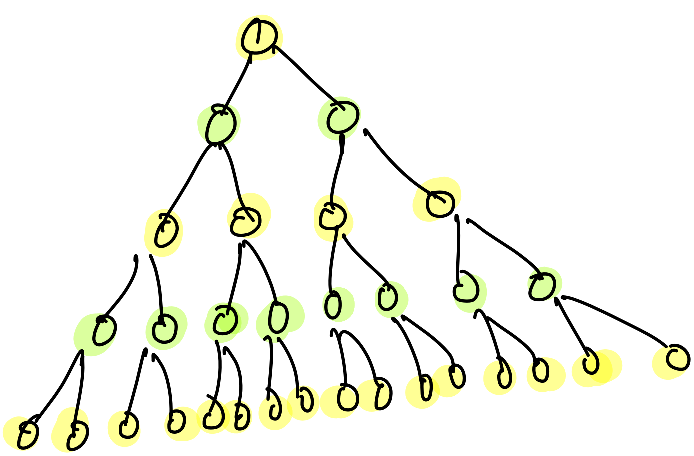
$$\underbrace{\cancel{(0,3)} \mid \cancel{(1,2)} \mid \cancel{(1,3)} \mid \cancel{(2,1)} \mid \cancel{(2,2)}}_{d = \cancel{0}\,1} \mid \underbrace{\cancel{(0,2)} \mid \cancel{(1,1)} \mid \cancel{(2,3)} \mid \cancel{(2,0)}}_{d=2} \mid \underbrace{\cancel{(0,1)} \mid \cancel{(1,0)}}_{d=3} \mid \underbrace{\cancel{(0,0)}}_{d=4}$$

Graph Coloring

Francis Guthrie (1852)

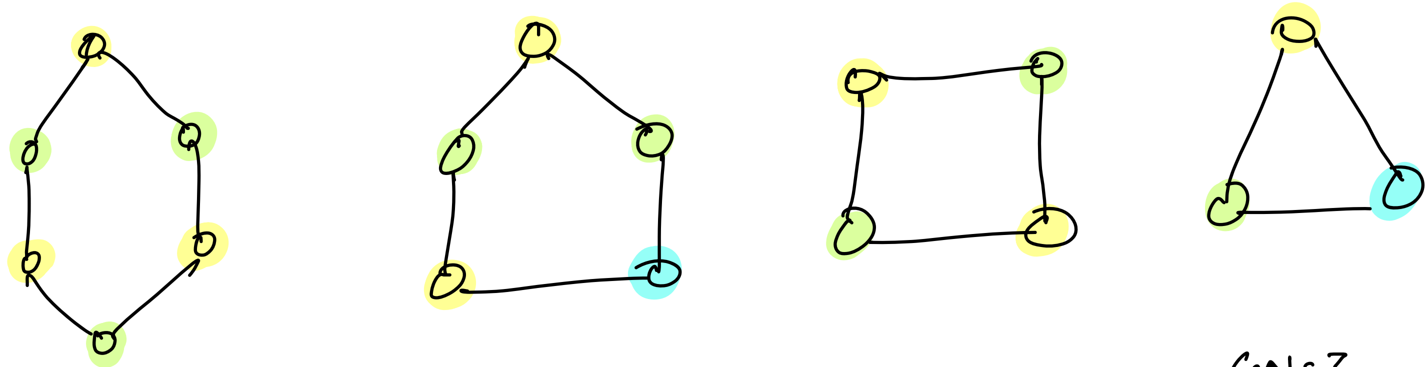
Minimum no. of colors required to paint all the nodes in a graph such that no two adjacent nodes have the same color. ⇒ Chromatic Number

① Tree.



Chromatic no. → 2.

② Cycle Graph (whole graph is a cycle)



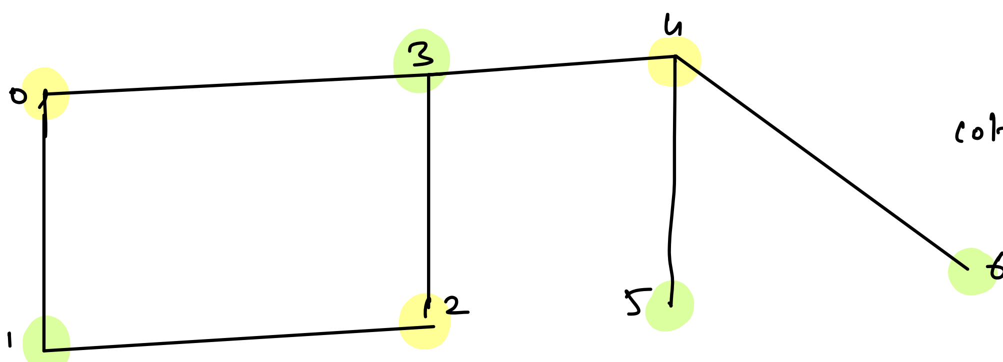
C.N = 2. C.N = 3. C.N = 2. C.N = 3

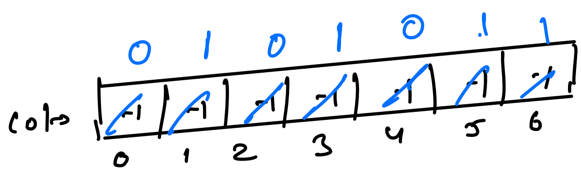
In general, C.N of cycle graph = 2 + (n % 2)

Bi-partite Graph

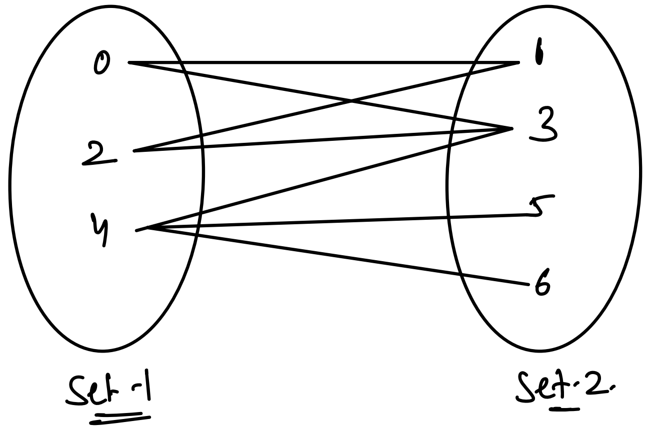
→ Any graph with chromatic no = 2.

→ A graph is called bi-partite if we can divide all the nodes into two sets, such that all the edges are across the sets.





0 → yellow
1 → green



Bi-partite Graph

Q.1) → Check if the given graph is bi-partite or not ?

```
col [N],      ∀ i, col[i] = -1

col[src] = 0;

boolean dfs( graph, src){
        for( int nbr : graph [src] ){
                if( col[nbr] == col[src]) {return false}
                else if( col[nbr] == -1){
                        col[nbr] = 1 - col[src]  //opp. colour of src.
                        if( dfs( graph, nbr) == false){
                            return false;
                        }
                }
        }
        return true;
}
```

main function

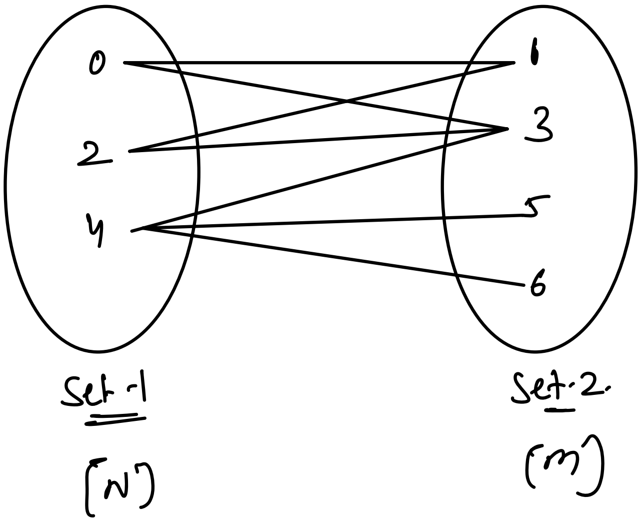
```
for(i=0; i<N; i++){
    if(col[i] == -1){
        if(dfs(graph, i) == false){
            return false;
        }
    }
}
return true;
```

checking for all the components.

T.C -> O(N+E)
S.C -> O(N)

Q

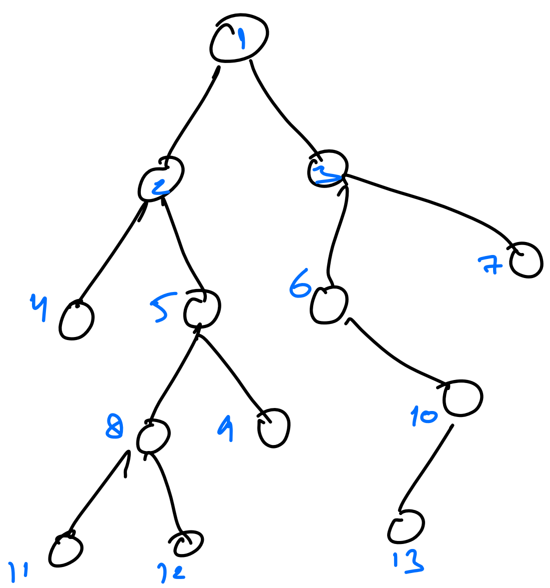
Given a graph which is Bipartite. The Graph is Divided into 2 disjoint and independent sets u and v with N and M nodes respectively. What is the maximum number of edges we can have ?



ans = N × M.

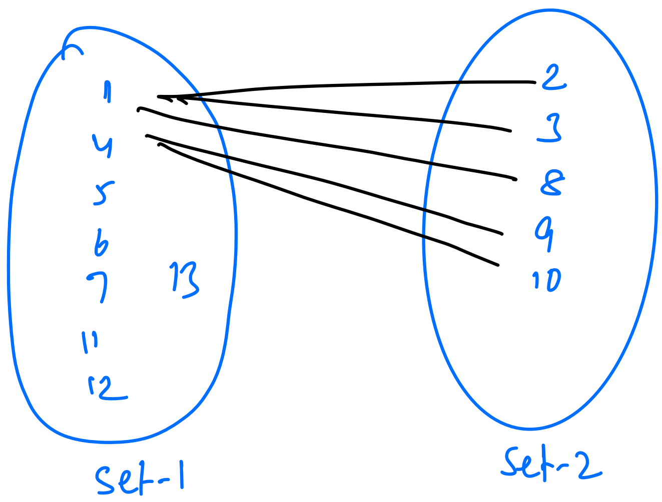
Q

Given a tree with N nodes find the maximum number of edges that can be added to the tree so that it is still bi-partite graph.



no. of nodes → N

no. of Edges → N-1



$8 \times 5 = 40 \Leftarrow$ maximum no. of edges possible.

$n-1 = 12 \Leftarrow$ current no. of edges.

ans. $\rightarrow$ $40 - 12 = \underline{\underline{28}}$

Idea. $\rightarrow$ Apply Level Order traversal & find

$x \rightarrow$ no. of nodes at even level

$y \rightarrow$ no. of nodes at odd level

ans. $= \underline{x \times y - (N-1)}$

$$\begin{bmatrix} T.C \rightarrow O(N) \\ S.C \rightarrow O(N) \end{bmatrix}$$

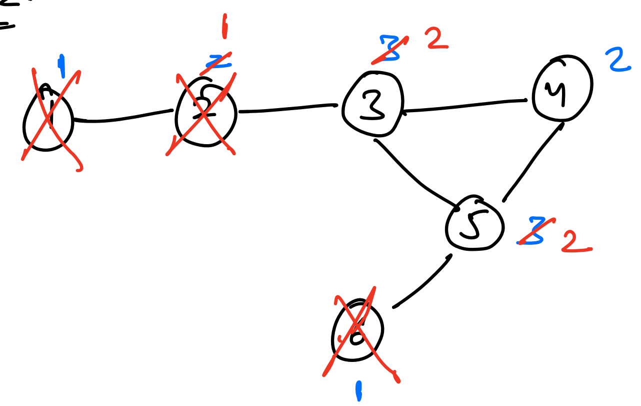
Q:

Given a friendship graph of N persons

You can only attend the party if you have minimum K number of friends attending the party.

Find total number of people that can attend the party.

Eg → K=2.



K=2.

~~1 → 1~~

~~2 → 1 2~~

3 → 2 ~~3~~

4 → 2

5 → ~~3~~ 2

~~6 → 1~~

1 → 2

2 → 1, 3

3 → 2, 4, 5

4 → 3, 5

5 → 3, 4, 6

6 → 5

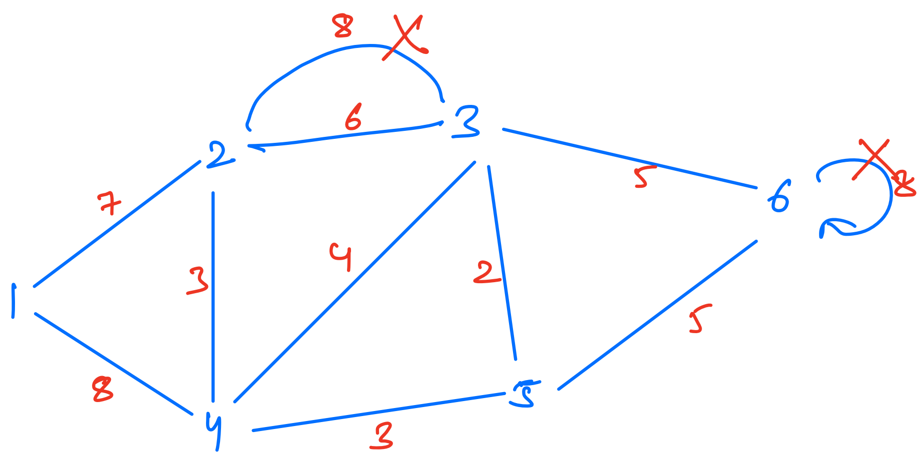
idea → use priority queue (min. Heap)

M.S.T. (Minimum Spanning Tree)

Kruskal's Algo

Prim's Algo

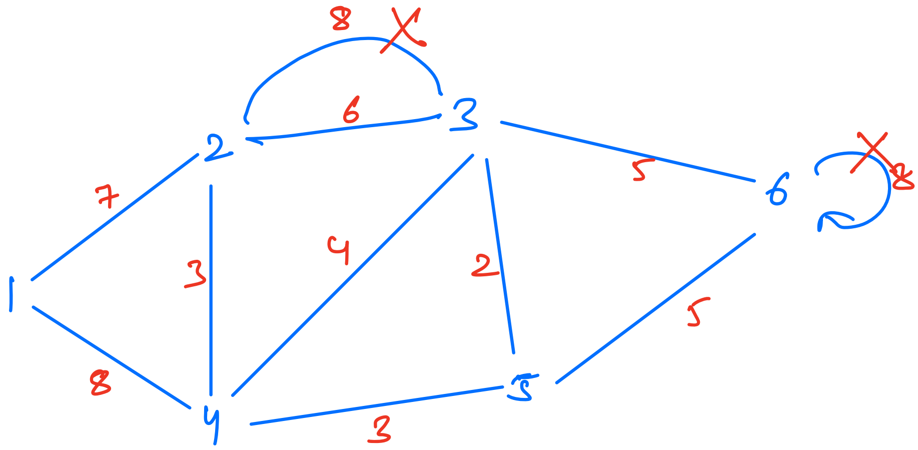
[Tree generated from a connected weighted graph such that all the nodes are connected & sum of weight of all selected edges is minimum.]



① Remove self-loops & multi-edges (with higher ed wts)

② Select edges with minimum weight, if they are not forming the cycle.

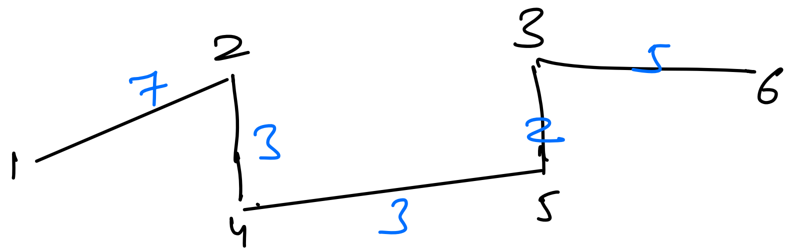
③ As soon as we have n-1 edges, we can stop. [Tree is formed]



<u>input</u>.

N = 6,   E = 11

5.	1	2	7
	~~1~~	~~4~~	~~8~~
	~~2~~	~~3~~	~~6~~
2.	2	4	3
	~~2~~	~~3~~	~~8~~
×	~~3~~	~~4~~	~~4~~
1.	3	5	2
4.	3	6	5
3.	4	5	3
	~~5~~	~~6~~	~~5~~
	~~6~~	~~6~~	~~1~~



ans → <u>20</u>

<u>Minimum</u> <u>spanning</u>
<u>Tree</u>

code :-

① Sort all the edges w.r.t edge weights in ascending order.

② Use D.S.U to check for cycle →

→ All nodes are considered as unique sets initially.

→ select (u,v) if set of u and set of v are disjoint.

→ Take union (u,v)

T.C → $O(E \log E + E \cdot 1)$
S.C → $O(N + E)$

parent array in D.S.U
+
creating triplets.