~~N- Queens problem~~

Minimum spanning tree

Trees:- tree তে কোনো cycle থাকবে না.

Graph:- এ cycle থাকে.

Spanning tree: এমন একটি tree যেখানে সবগুলোর vertex কে cover করা হয়।

minimum spanning tree:- minimum weight হবে এবং সব tree vertex কে cover করবে।

Given a graph (V, E) and edge weights W, find the spanning tree of minimum weight

→ find all spanning tree

→ select the one with minimum total weight

→ Complexity : Exponential.

Minimum spanning tree দুই ভাবে solve করা যায়,

i) Krushkal algo

ii) Prim's algo.

Krushkal algo

Krushkal algo (MST):

1) Intialize an empty set mst to store the the minimum spanning tree.

2) Sort all edges of G in non-decreasing order by their weights.

3) Create an empty disjoint - set DS to keep track of connected components

4) Iterate through the sorted edges.

5) i) If adding the edge to mst doesn't create a cycle (i.e. the edge's endpoint are in different connected components.

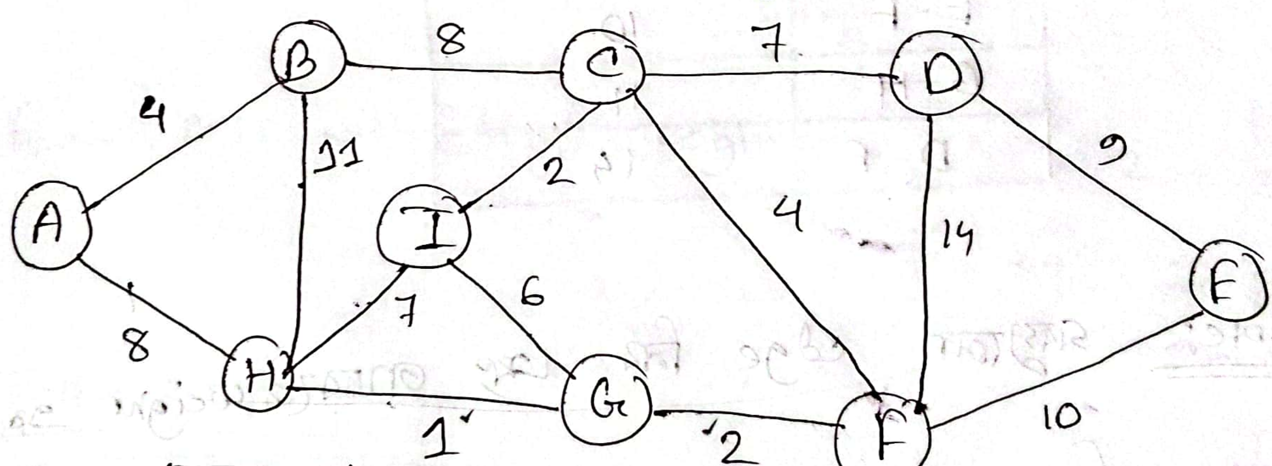
In. DS), add it to MST.

ii] Otherwise, skip the edge.

iii] Update DS to merge the connected components of the end points of the added edge.

5] Return MST.

Krushkal's Algorithm



Now:

Edge	Weight
G-H	1
G-F	2
~~A-B~~ C-I	~~4~~ 2
A-B	4
C-F	4
~~A-H~~ C-D	~~8~~ 7
B-C	8
A-H	8

Edge	weight

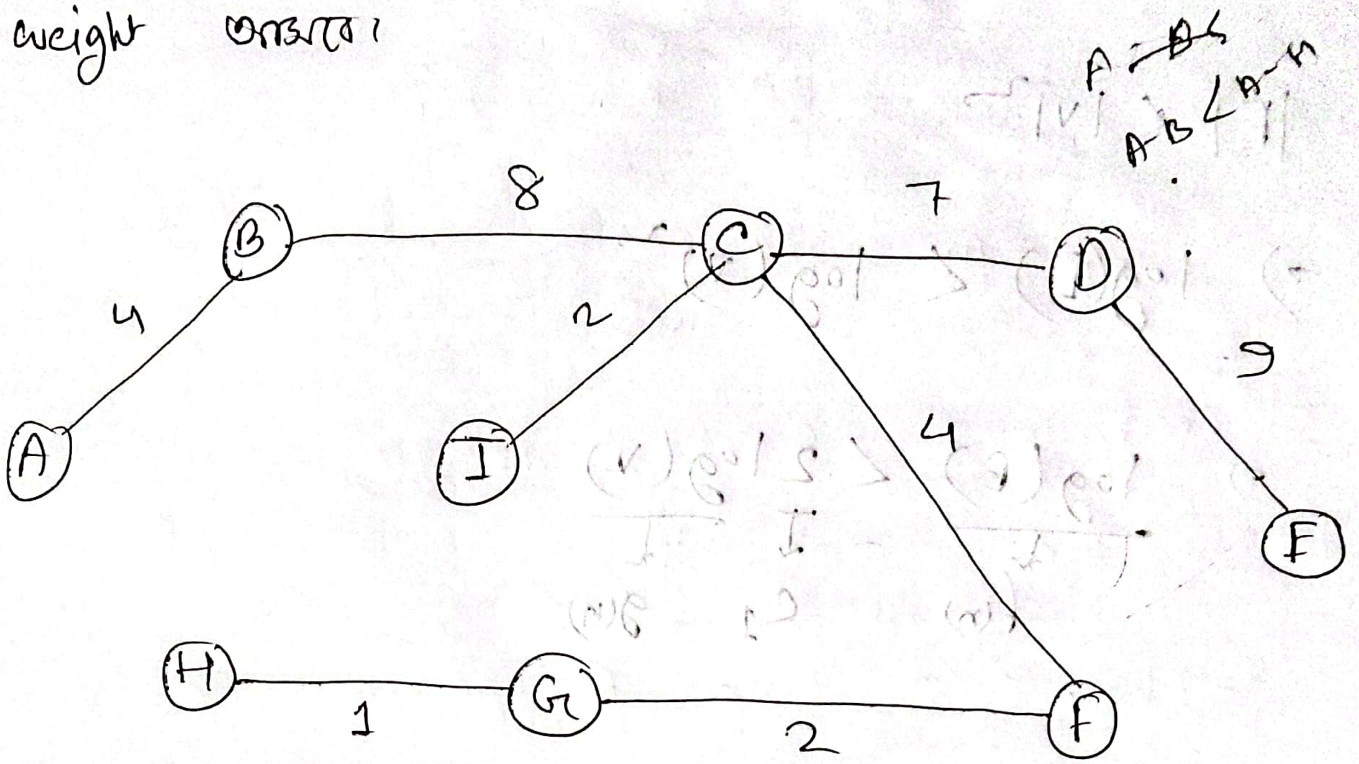
Step-2:-

1, 2, 2, 4, 4, 6, 7, 7, 8, 8, 8, 9, 10, 11, 14

edge	weight
✓ G-H	1
✓ G-F	2
✓ C-I	2
✓ A-B	4
✓ C-F	4
× G-I	~~7~~ 6
✓ C-D	~~8~~ 7
× ~~A-H~~ H-I	~~8~~ 7
✓ B-C	8
× A-H	8
✓ D-E	9
E-F	10
B-H	11
D-F	14

Note: সবগুলো edge নিয়ে এদের ভেতরকে weight এর উ[illegible]পর ভিত্তি করে ascending order এ সাজাবো, একই Group এ নেওয়া যাবে না কারণ নিলে cycle বা loop চলে আসবে. এই tree তে loop বা cycle হয় না.

spanning tree এ সব value বা weight যোগ করে total weight পাবো।



found MST with total weight = 1 + 2 + 2 + 4 + 4 + 7 + 8 + 9 = 37

Time complexity:

Sorting E edges → $O(E \log E)$

Selecting E edges → $O(E)$

Checking V nodes if it is in the same component (Disjoint-set) → $O(V)$

Total T.C = $O(E \log E) + O(E) + O(V) \rightarrow O(E \log E)$

$f(n) < c_1 \cdot g(n)$

$|E| < |V|^2$

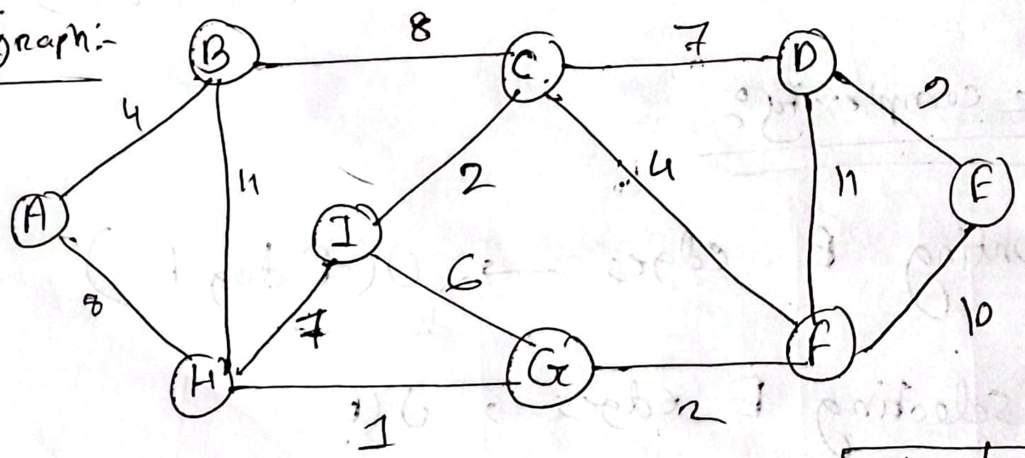
$\Rightarrow \log(E) < \log(V)^2$

$\Rightarrow \underbrace{\log(E)}_{f(n)} < \underbrace{2}_{c_1} \underbrace{\log(V)}_{g(n)}$

$f(n) = O(g(n))$,

$\Rightarrow \log(E) = O(\log V)$

Given graph:-



edge	weight
G-H	1
G-F	2
C-I	2
A-B	4

edge	weight
C-F	4
G-I	6
C-D	7
H-I	7

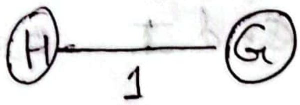
edge	weight
A-H	8
B-C	8
D-E	9
E-F	10
B-H	11
D-F	14

~~Using Adjacency Matrix - $O(V^2)$~~

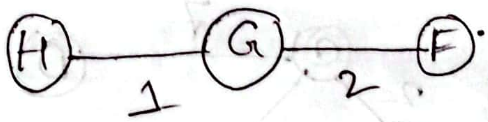
~~Using Adjacency List - $O(E \log V)$~~

~~N-Qu~~

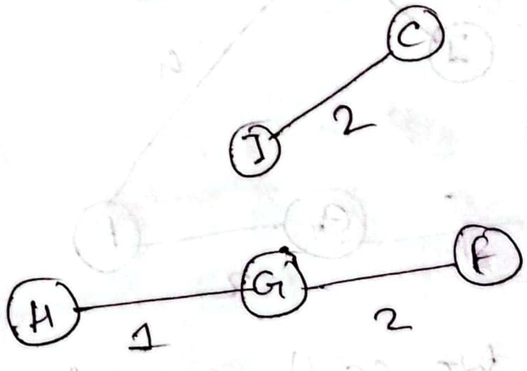
Ite-1|



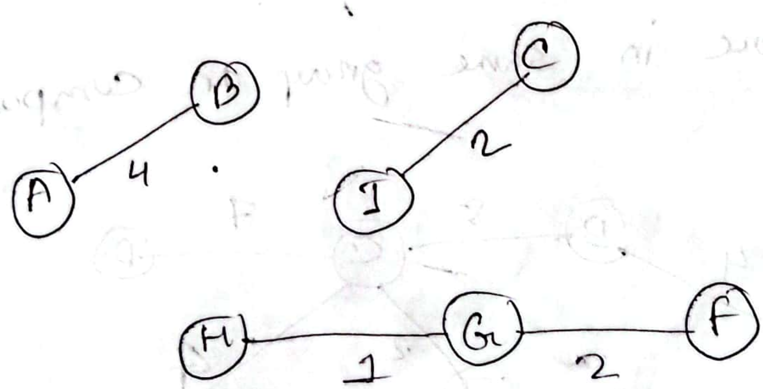
Ite-2|



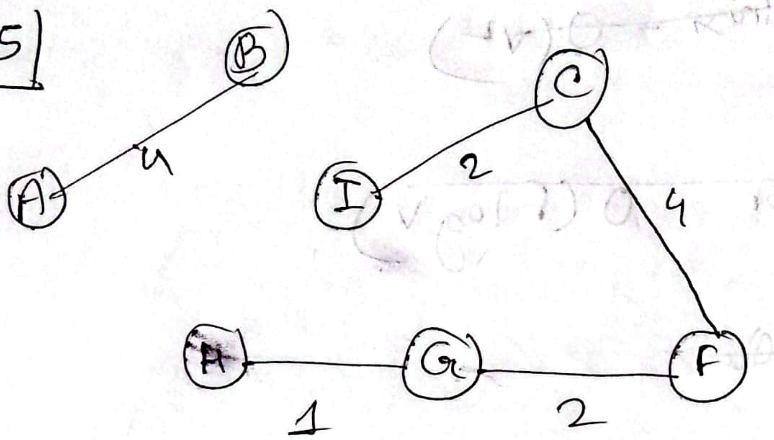
Ite-3|



Ite-4|

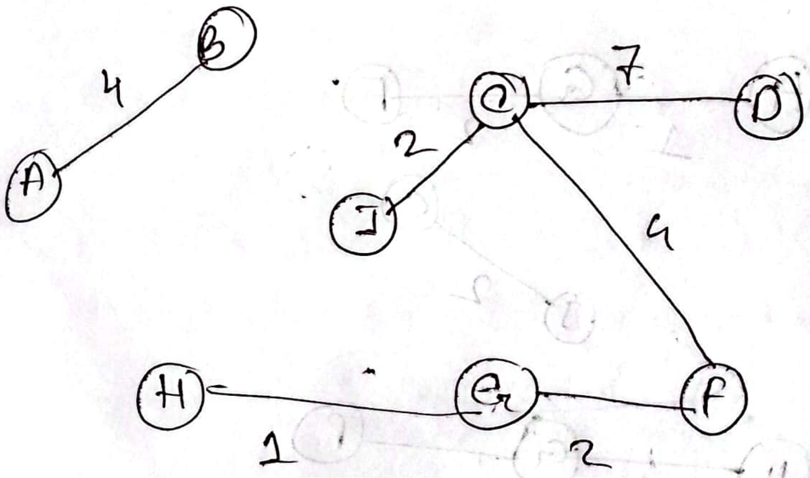


Ite-5|



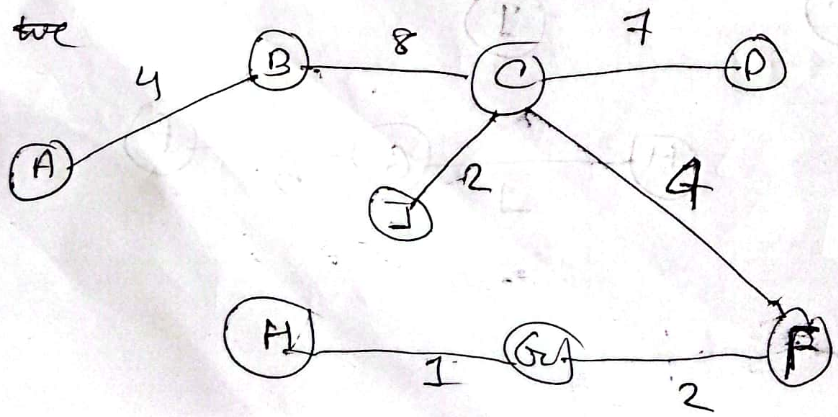
Ite-6| we can't connect G and I as they are in same group.

Ite-7|

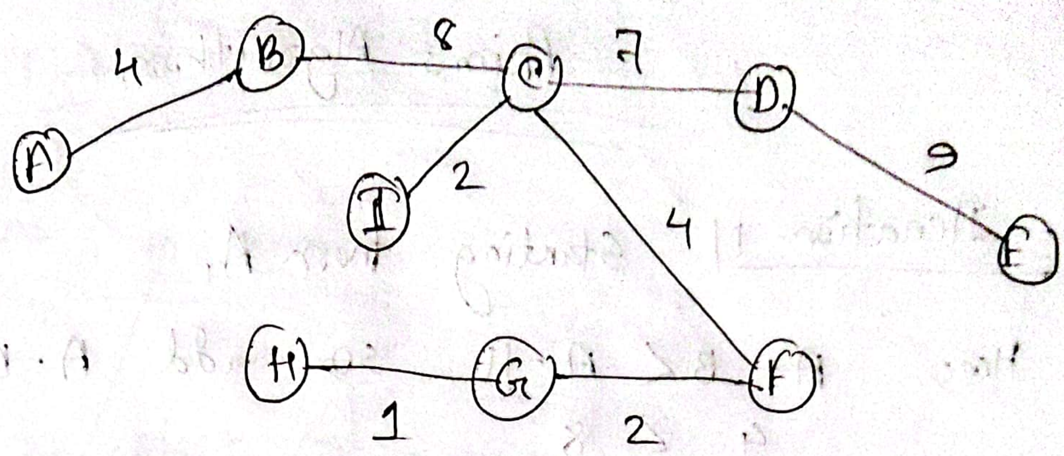


Ite-8| ~~we can't~~ we can't connect H and I, A and H as they are in same group or component.

Ite-9| ~~we~~



Ite-10|



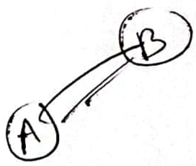
Ite-11| We can't connect E and F, B and H, D and F as they are in same group or component.

~~Prim's algorithm~~

~~Iter-11~~ ~~Start A~~

~~A - B < A - H~~

~~4 < 8~~



T.C

Sorting E edges → $O(E\log E)$

Selecting E edges → $O(E)$

Checking V nodes if it is the same component → $O(V)$

Overall:

$O(E\log E) + O(E) + O(V)$

~~O(Elo~~

$O(E\log E)$

As $|E| < |V^2|$

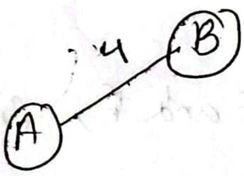
$\log E < \log V^2$

$\log E < 2\log n$ → T.C = So, $\log E \simeq \log V \rightarrow O(E\log V)$

Prim's Algorithom

Iteration-1| Starting from A,

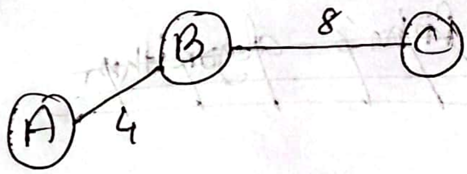
Here A-B < A-H so add A-B,
4 < 8



Now,

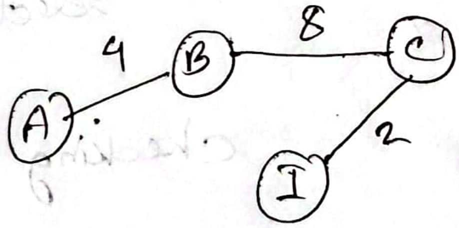
Ite-2| ~~A-B~~ B-C < A-H < B-H
8 8 11

so, add B-C,



Ite-3| Now, C-I < C-F < C-D < A-H < B-H
2 < 4 < 7 < 8 < 11

so, add C-I, ~~to mst~~



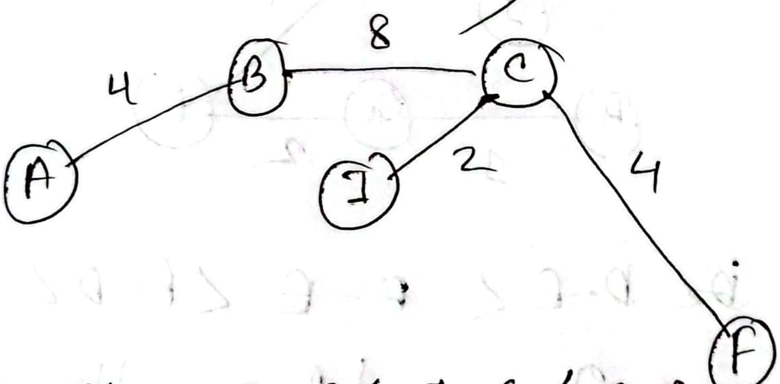
Ite-4| Now ~~C-F < C-D < A-H <~~

C-F < I-G < C-D < I-H < A-H < B-H
4 < 6 < 7 < 7 < 8 < 11

so add C-F,

N- Queens problem:

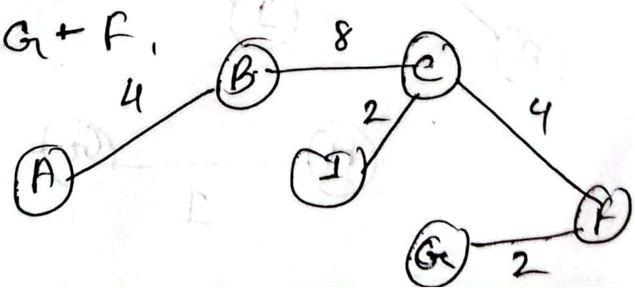
The goals is to place N queens on an N×N chess-board in such a way that no two queens threaten each other (the queen chessboard এর vertically, horizontally and diagonally move করতে পারে কিন্তু এখনে now, column এ diagonally কোন player কেউ মারতে পারে না) ~~এখানে~~ .



Ite-5| Now, G-F < I-G < C-D < I-H < A-H < F-E < F-D < B-H

2 < 6 < 7 < 7 < 8 < 10 < 11 < 11

so, add G+F.

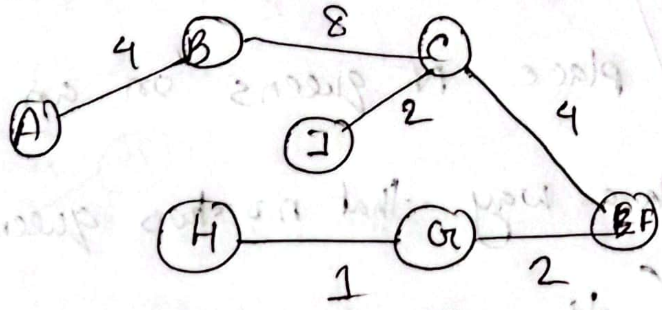


Now,

Ite-6] G-H < I-G < C-D < I-H < A-H < F-E < F-D < B

1 < 6 < 7 < 7 < 8 < 10 < 11 <

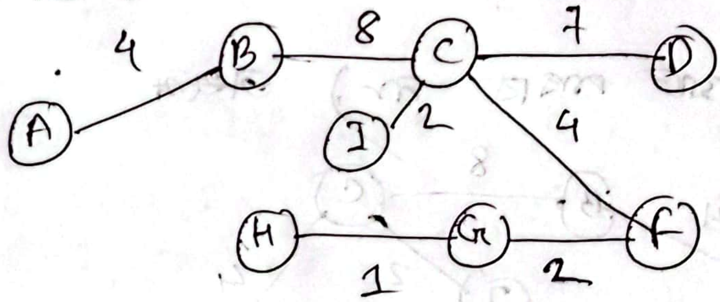
Now add G-H,



Ite-7] Now C-D < I-H < A-H < F-E < F-D < B-H

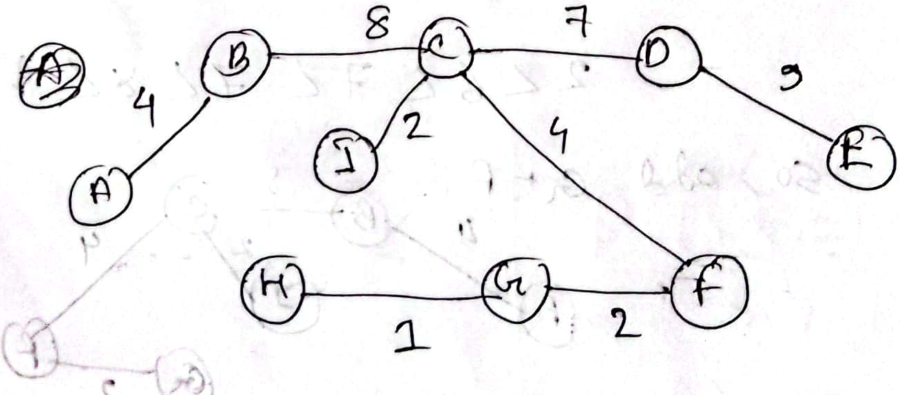
7 < 7 < 8 < 10 < 11 < 11

Now add C-D



Ite-8] Now D-E < F-E < F-D < B-H

9 10 11 11



N-Queen's problem

The goal is to place N-queens on an $N \times N$ chess board in such a way that no two queens threaten each other.

Note: queen কে horizontally, vertically, diagonally move করাতে হবে, এবং এ জন্যে খেয়াল রাখতে হবে, যে row, column বা diagonal এ move করিয়েছি ওই row, column বা diagonal এ কোনো queen আছে কিনা। যদি থাকে তাহলে আগের queen টা new-arrival queen কে জায়গাটি ছেড়ে দিবে।

~~Step: 1~~ **Simulation:**

Solution-1 |

$n = 4$

~~Chase bond~~ :

chess board : 4×4

Step-1 | Start with the first row and place the first Queen (Q_1) in the first column.

	1	2	3	4
1	Q_1			
2				
3				
4				

Step-2:

	1	2	3	4
1	Q_1	~~Q_1~~		
2	x	x	~~Q_2~~	Q_2
3				
4				

Step-2|

	1	2	3	4
1	Q_1			
2	x	x	Q_2	
3				
4				

Step-3|

	1	2	3	4
1	Q_1			
2			Q_2	
3	x	x	x	x
4				

Step-4: (Backtracking)

	1	2	3	4
1	Q_1			
2				Q_2
3		Q_3		
4				

Step-5:

	1	2	3	4
1	Q_1			
2				Q_2
3		Q_3		
4	x	x	x	x

Step-6: (Backtracking)

	1	2	3	4
1		Q_1		
2				
3				
4				

Step-7:

	1	2	3	4
1		Q_1		
2				Q_2
3				
4				

Step-8:

	1	2	3	4
1		Q_1		
2				Q_2
3	Q_3			
4				

Step-9:

	1	2	3	4
1		Q_1		
2				Q_2
3	Q_3			
4			Q_4	

Solution:
Q_1 Q_2 Q_3 Q_4
[2 4 1 3] → sequences [2413]

~~Soltd~~ Solution – 2]

Step-1:

	1	2	3	4
1			Q_1	
2				
3				
4				

Step-2:

	1	2	3	4
1			Q_1	
2	Q_2			
3				
4				

Step-3]

	1	2	3	4
1			Q_1	
2	Q_2			
3				Q_3
4				

Step-4:

	1	2	3	4
1			Q_1	
2	Q_2			
3				Q_3
4		Q_4		

Solution:

	Q_1	Q_2	Q_3	Q_4
Col:	[3	1	4	2]

Sequence: [3, 1, 4, 2]

Time complexity:-

every ~~posib~~ possible position for 1st queen = n

" " " " 2nd " $= (n-1)$

" " " " 3rd " $= (n-2)$

" " " " 4th " $= (n-3)$

$\therefore n \times (n-1) \times (n-2) \times (n-3) = n!$

$\therefore T.C = O(n!)$

Graph colour

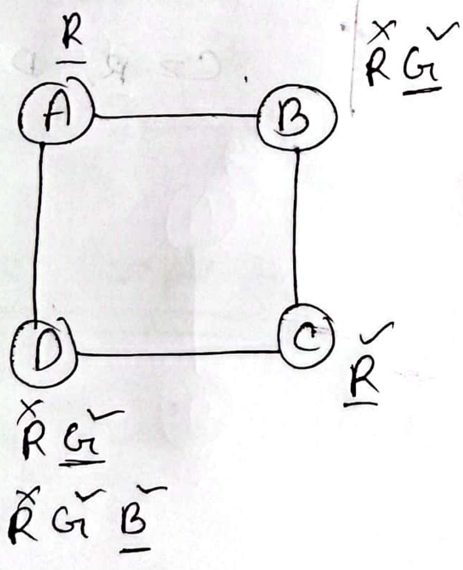
Note: Adjacent node এ same colour হবে না,

Simulation:-

k = number of colour = 3

S.C = R, G, B [R = Red, G = Green, B = Blue]

Step-1) If A = R then,



Sol:-

i) R G R G

ii) R G R B

Here,

A = R B = G

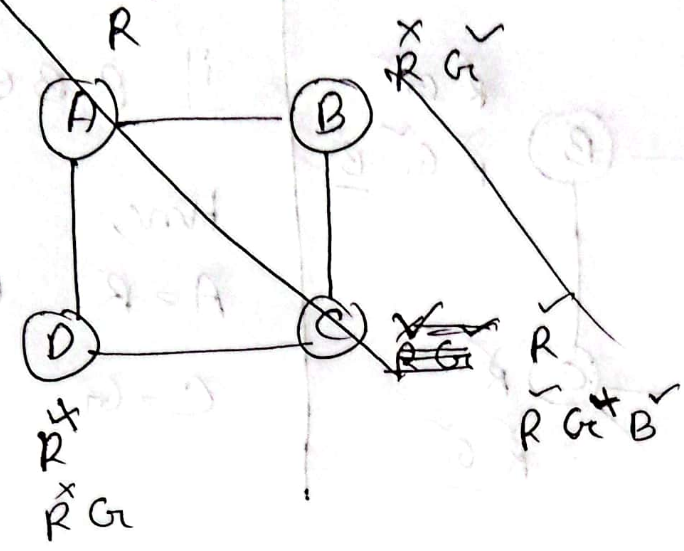
C = R D = G, B

[D completed]

Back track to "C".

Step 2:-

~~Step-2:~~



Sol:-

i) RGBR

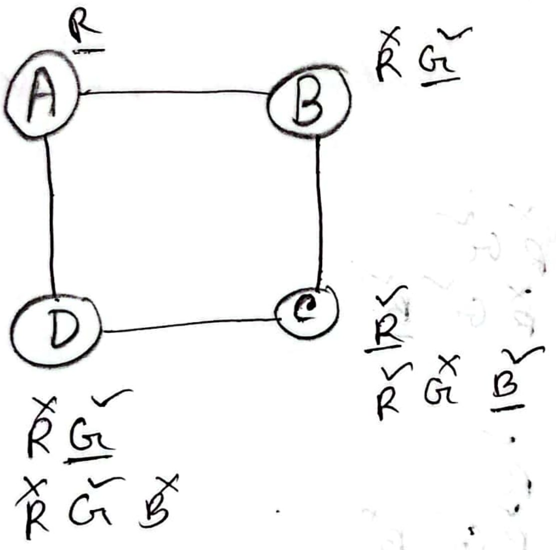
ii)

Here,

A=R B=B

C=G D=

Step-2:-



Sol:-

i) RGBG

ii)

Here,

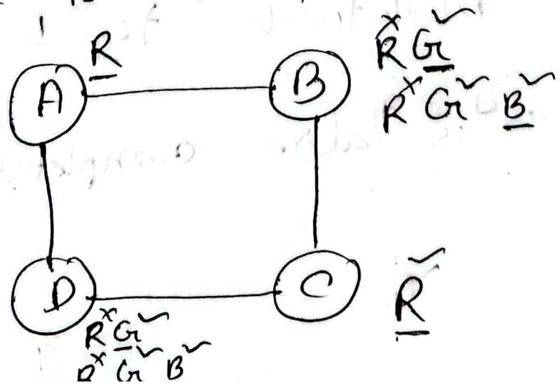
A=R, B=G

C=B, D=G

~~D com~~ ["C" Completed]

∴ Backtrack to "B".

Step-3) :-



Sol:-

i) RBRG

ii) RBRB

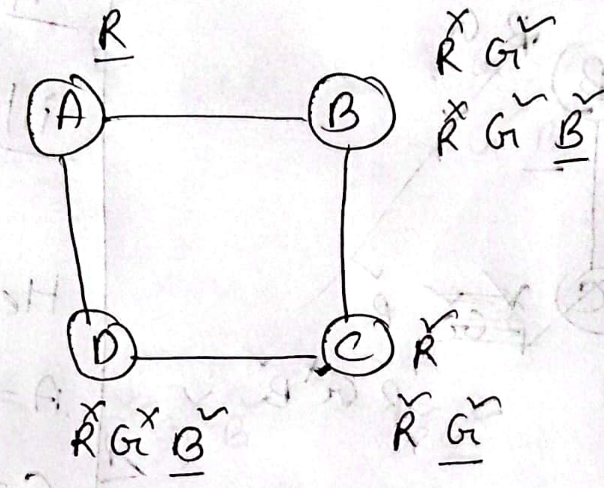
Here

A=R, B=B

C=R, D=G, B

["D" completed] ∴ Backtrack to "C"

Step-4:



Sol:

i) R B G B

Here,

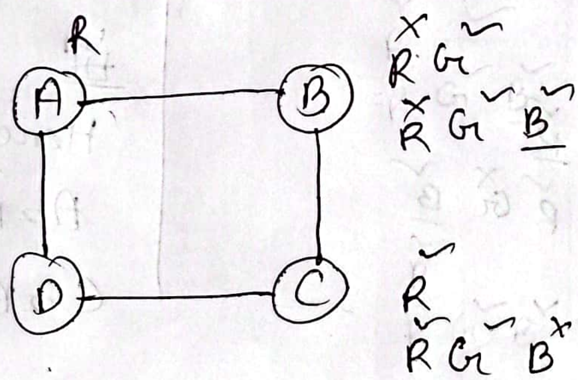
A = R   B = B

C = G,   D = B

∴ [D completed]

∴ Backtrack to "C".

Step-5)



~~so,~~ Here, C completed so, Backtrack to "B"

~~A~~ and After Backtrack to "B", we can see that "B" is also completed.

Sol:

total solution :-

i) RGRG

ii) RGRB

iii) RGBG

iv) RBRG

v) RBRB

vi) RBGB

~~total~~ total number of solution : 6 when A = Red

So, when A = G then t.n.o.s : 6

So, " A = B then " " " : 6

So, total solution : 6+6+6 = 18 ।