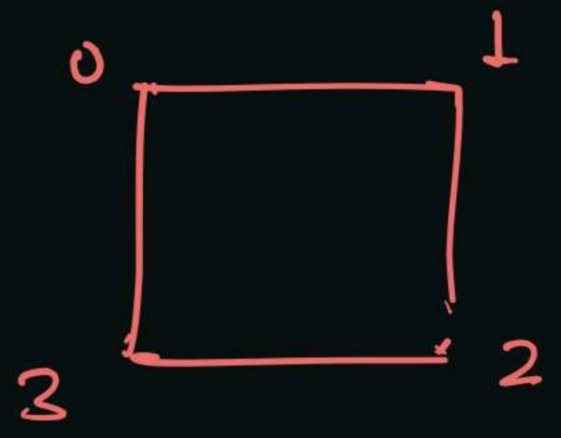
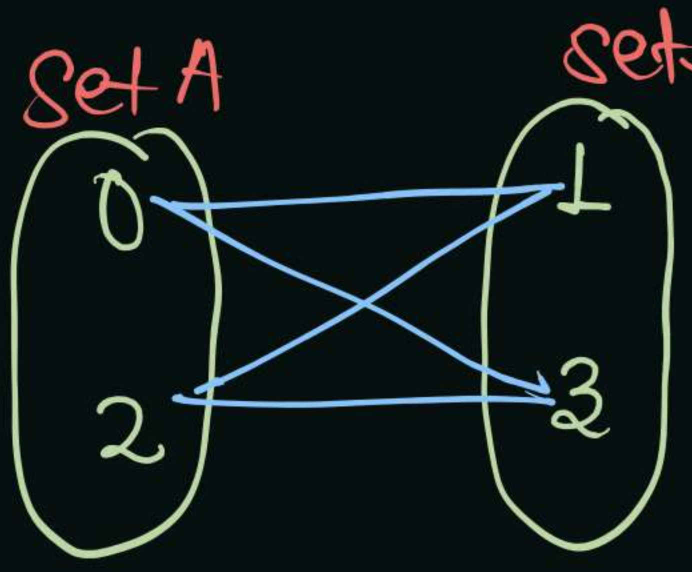
Is Graph Bipartite

Saturday, 10 July 2021 9:54 AM

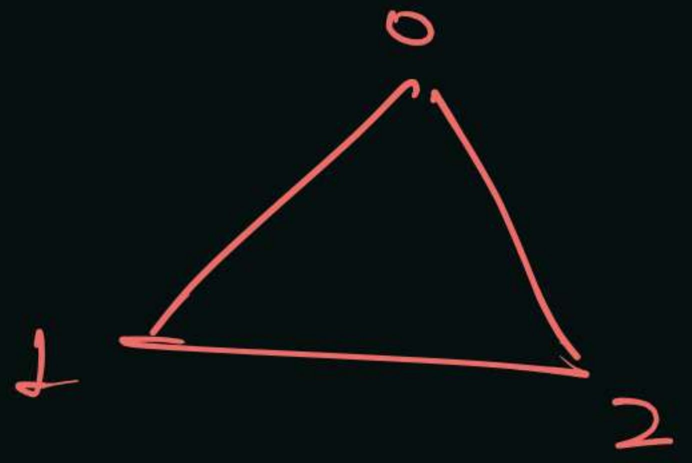
gf we can make two set of verteces so that no edge is present within the same set. then graph is called Bipartite Graph.

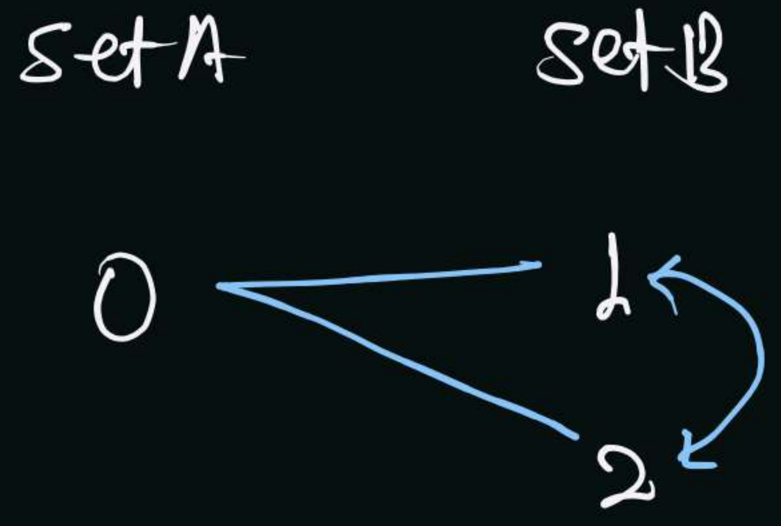
graph (1)





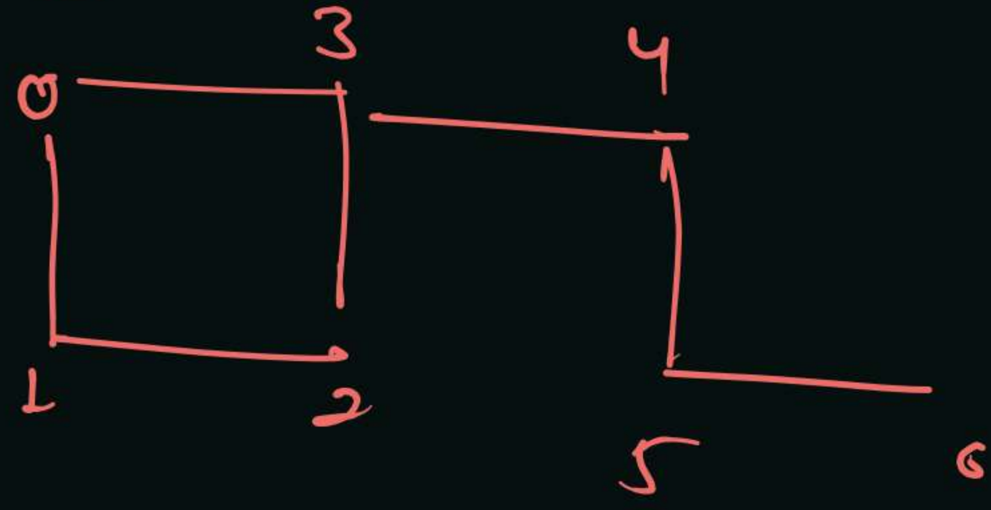
graph (2)



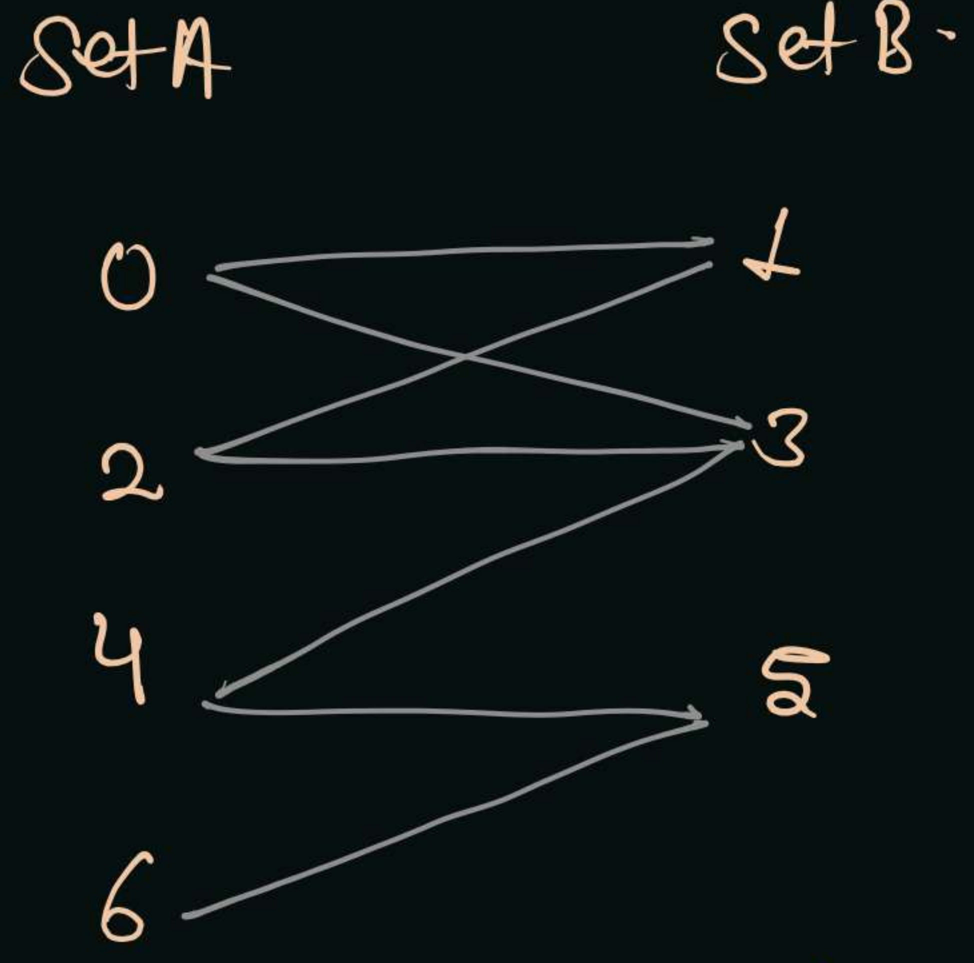


Ls Not a Bipartite graph.

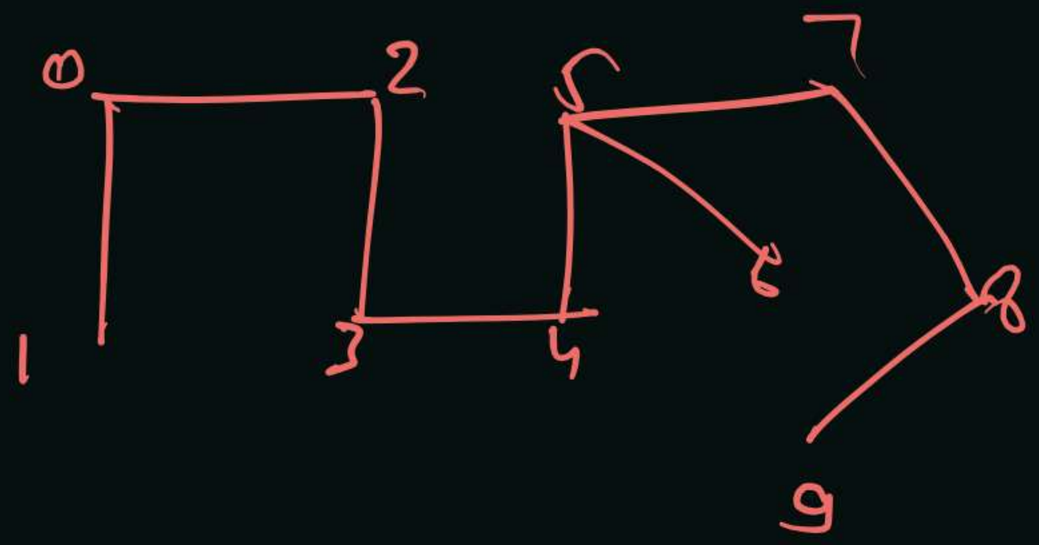
graph (3)



This graph is Bipartite



Hint → Acyclic graph is always Bipartite graph.

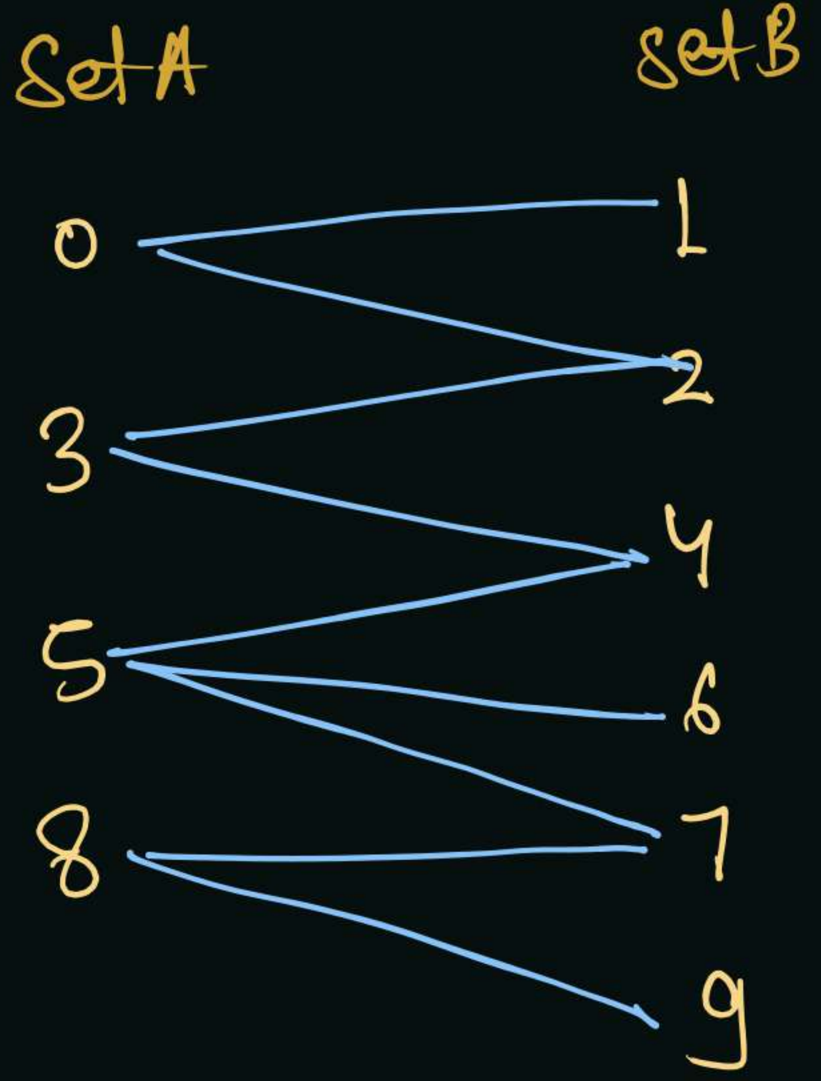


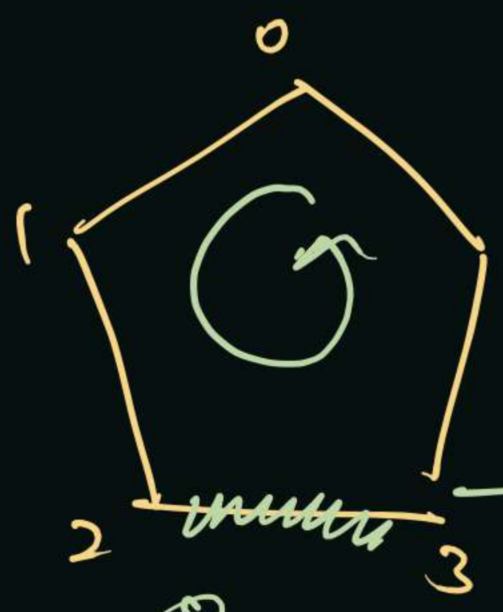
Acyclic graph - always Bipartite.

★ cyclic → Bipartite → True.
→ False.

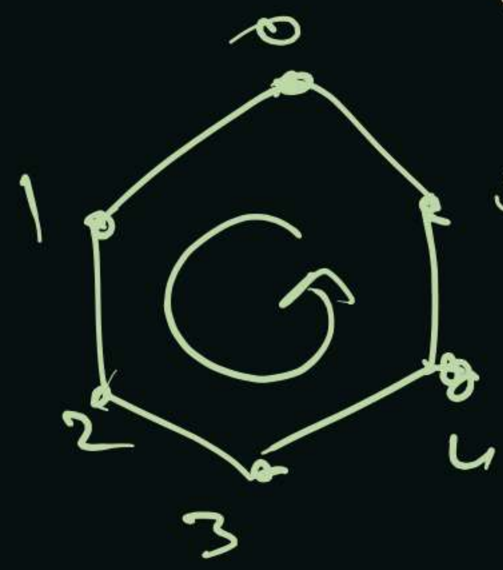
✓Acyclic → Bipartite

✓cyclic → even length Bipartite
→ odd length Not a Bipartite graph

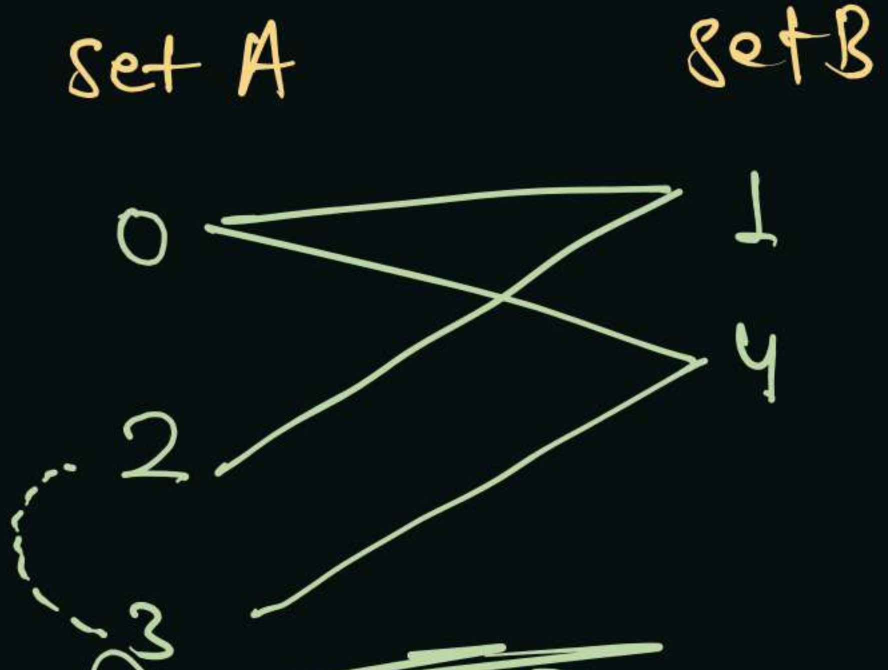




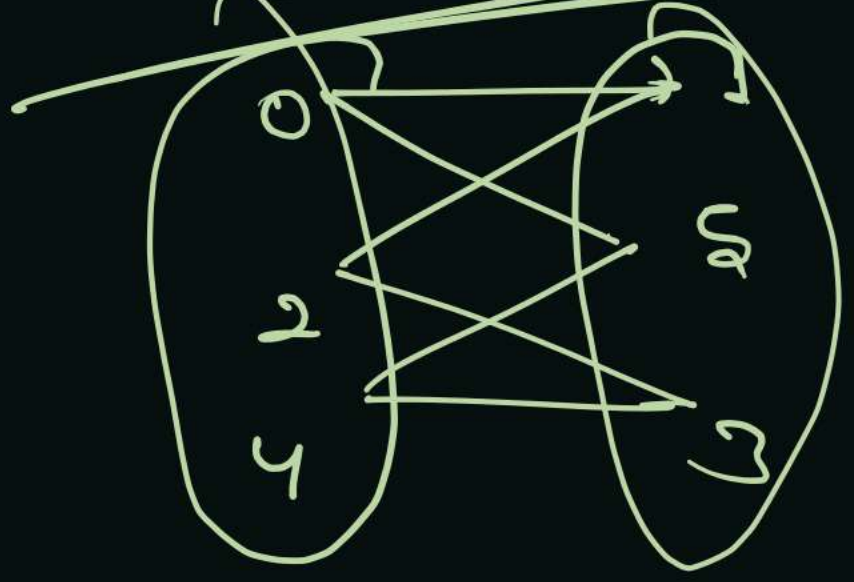
odd length separates

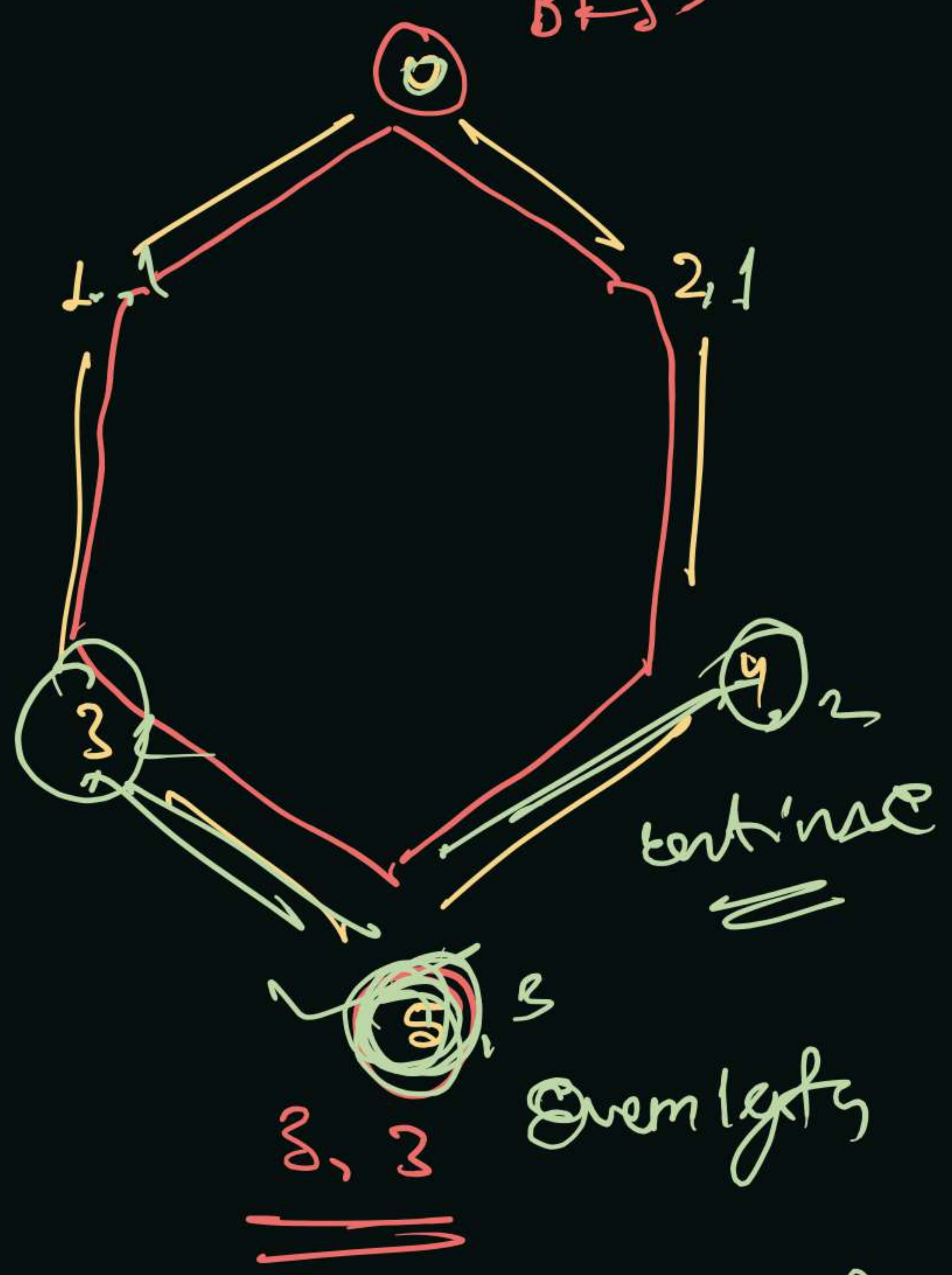


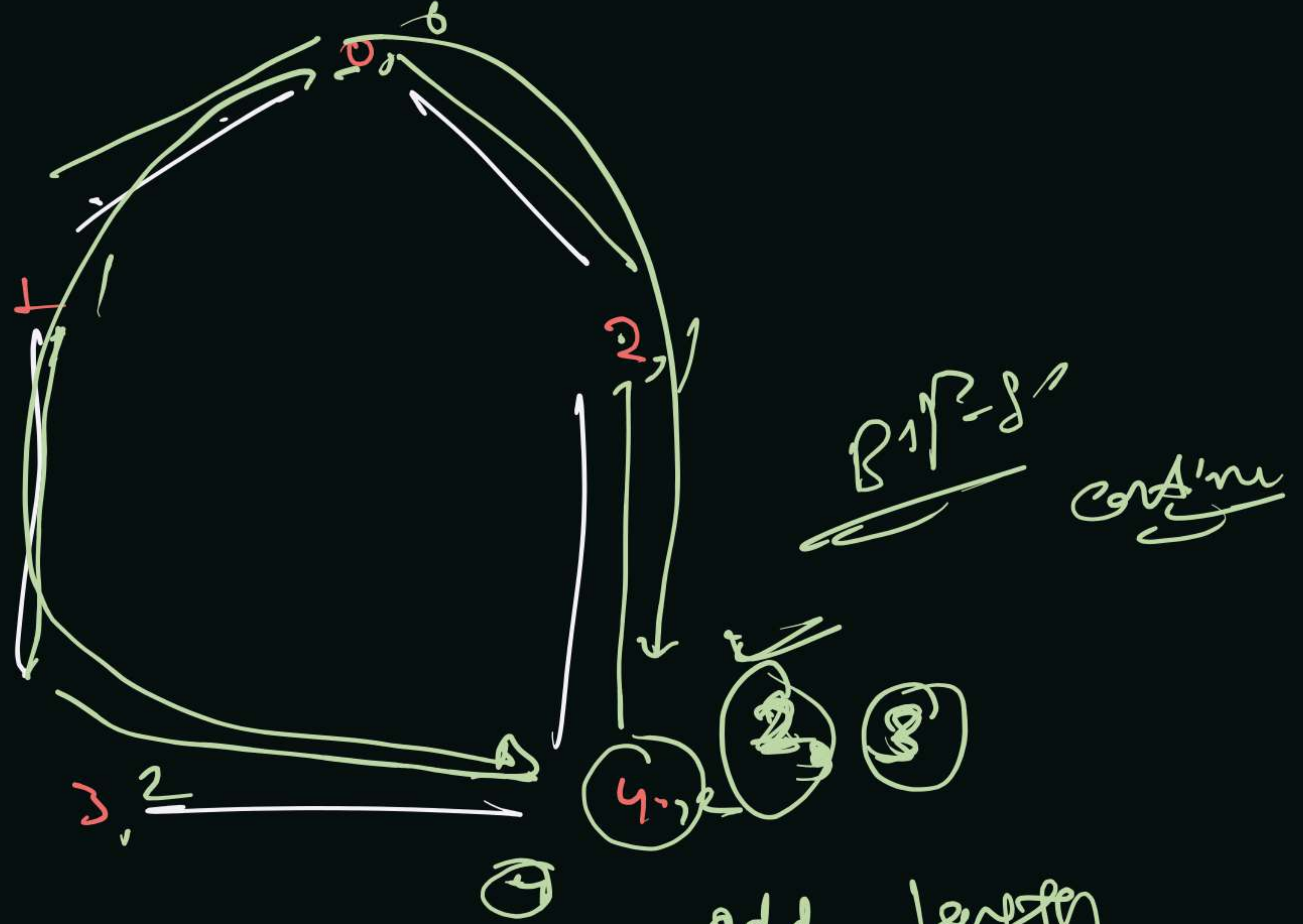
Even length Bipartite.



Not bipartite.

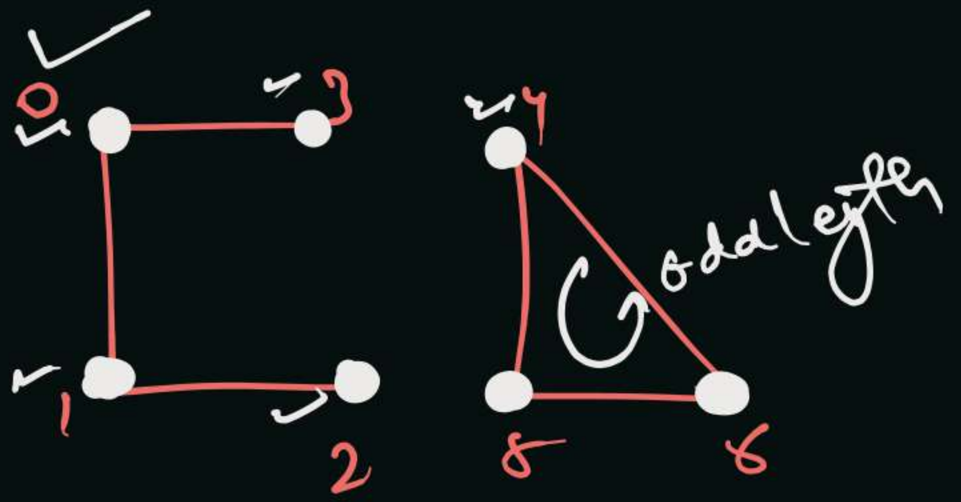




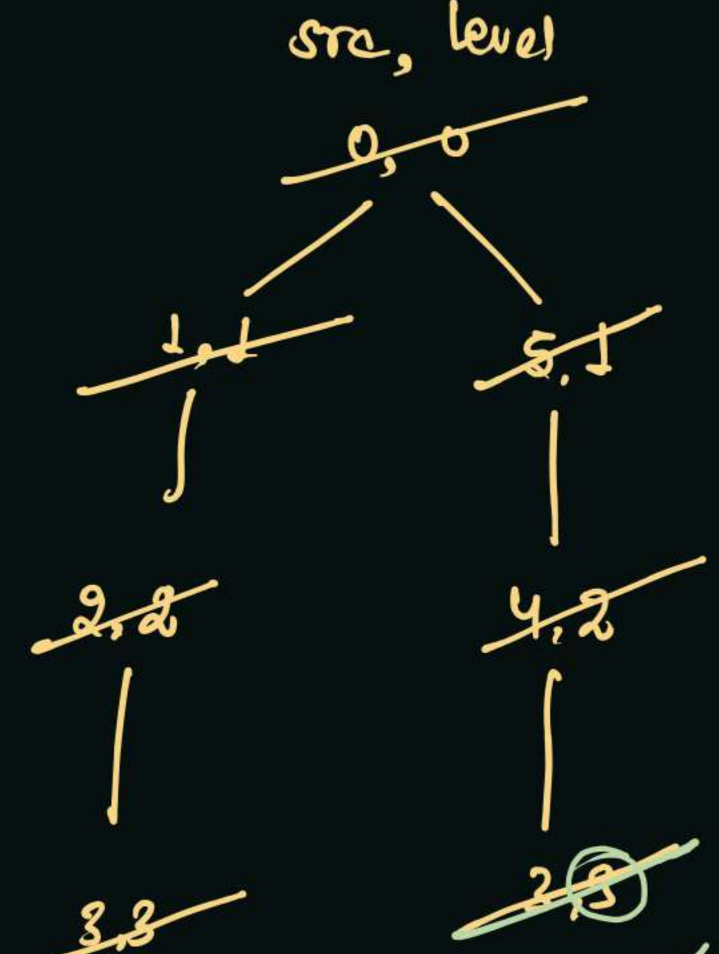


odd length
cyclic] return false

B·F·S· Pair → src, discovery time
level



loop of all the vertex

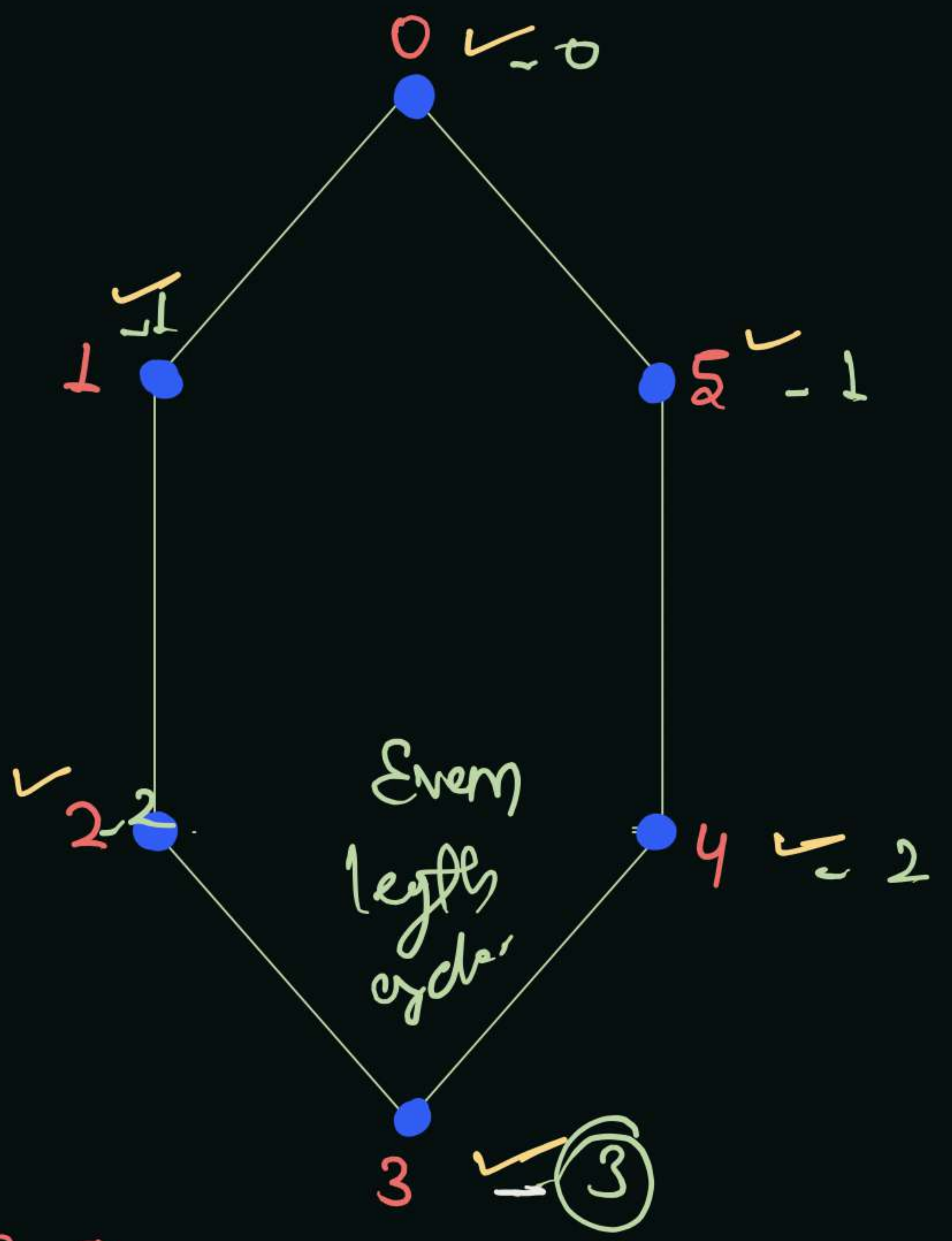


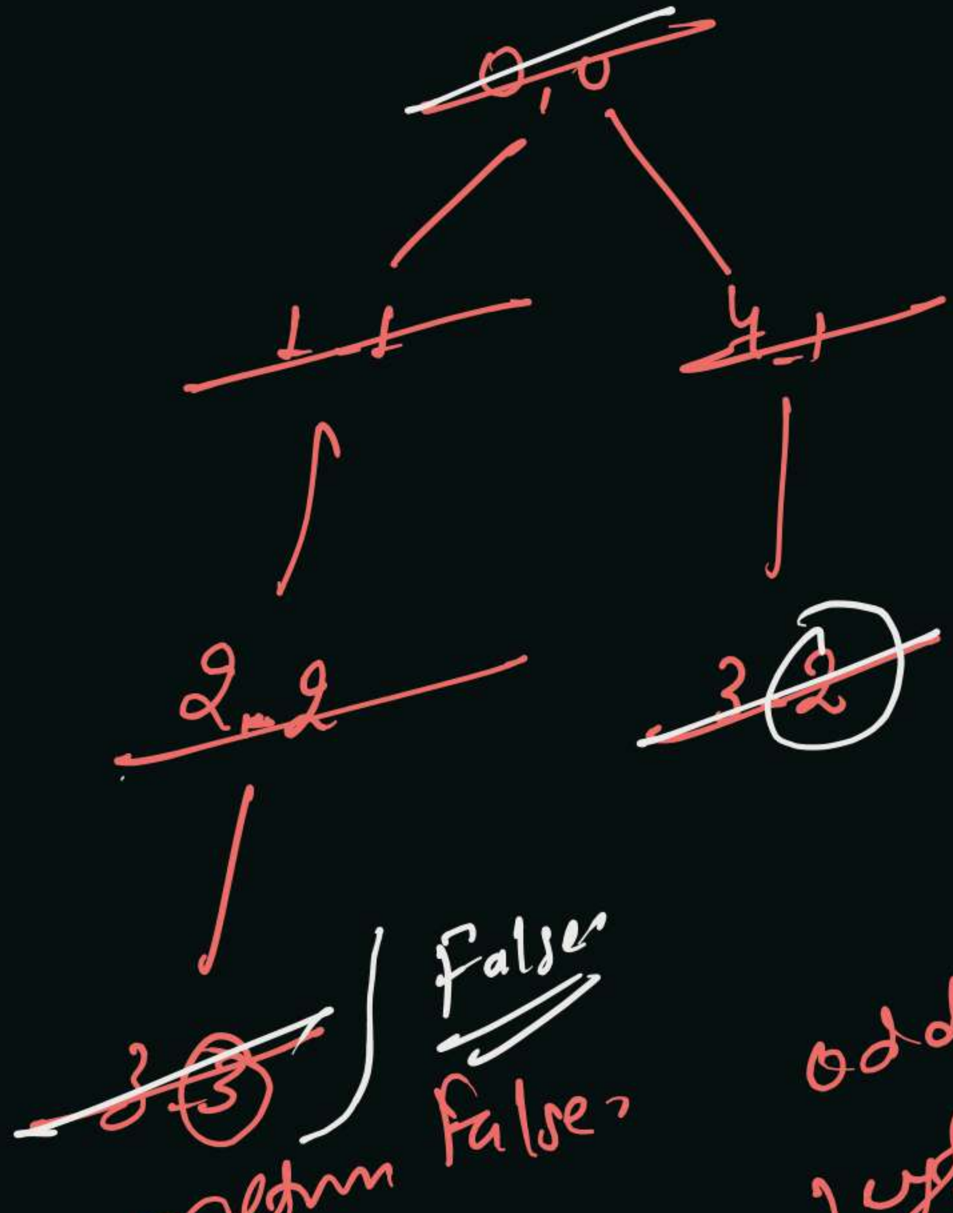
get
remove
mark
add
Neighbour

~~continue~~

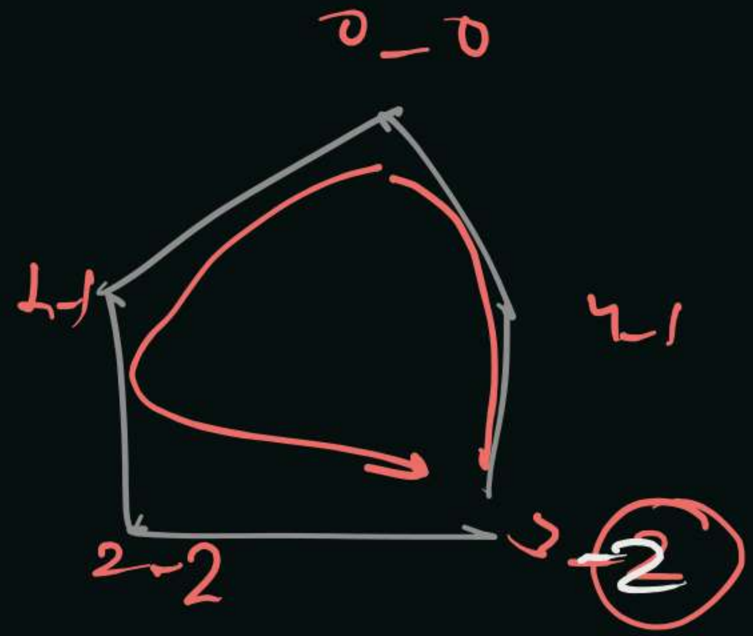
Visited → array of Integer → -1

Discovery level





odd length cycle



Spread of Injection

Saturday, 10 July 2021 9:55 AM

7

8

0 1 10 ✓

1 2 10 ✓

2 3 10 ✓

0 3 10 ✓

3 4 10 ✓

4 5 10 ✓

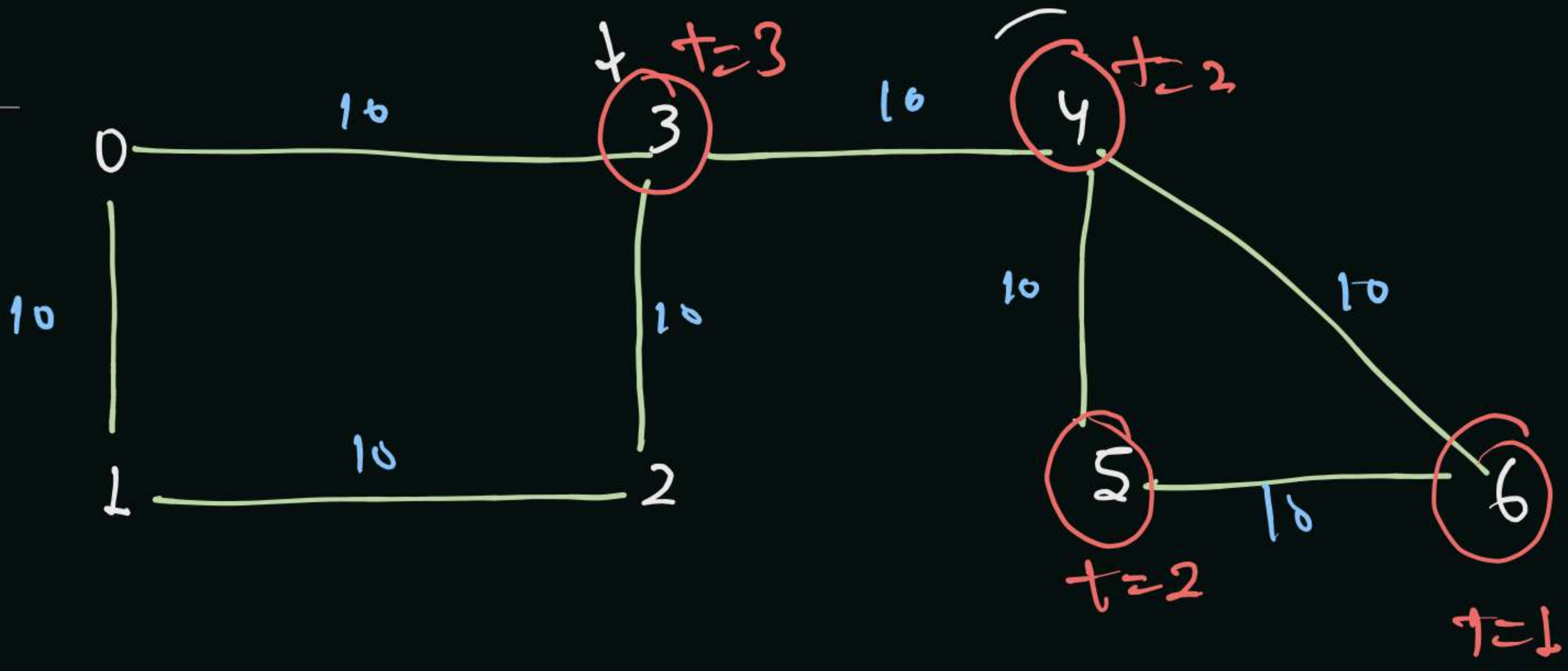
5 6 10 ✓

4 6 10 ✓

6 → Infected

3 Covid

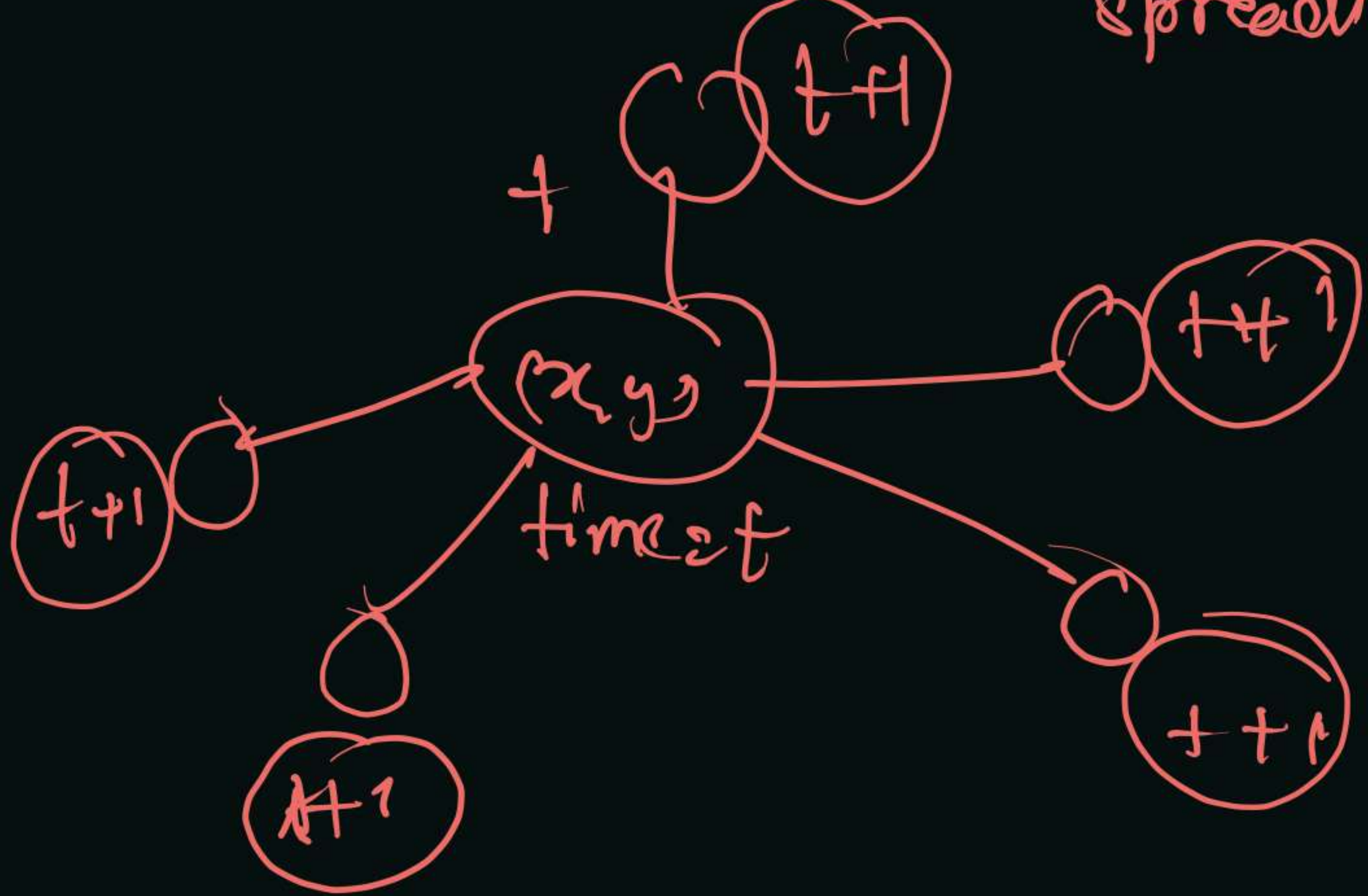
time



Infected,

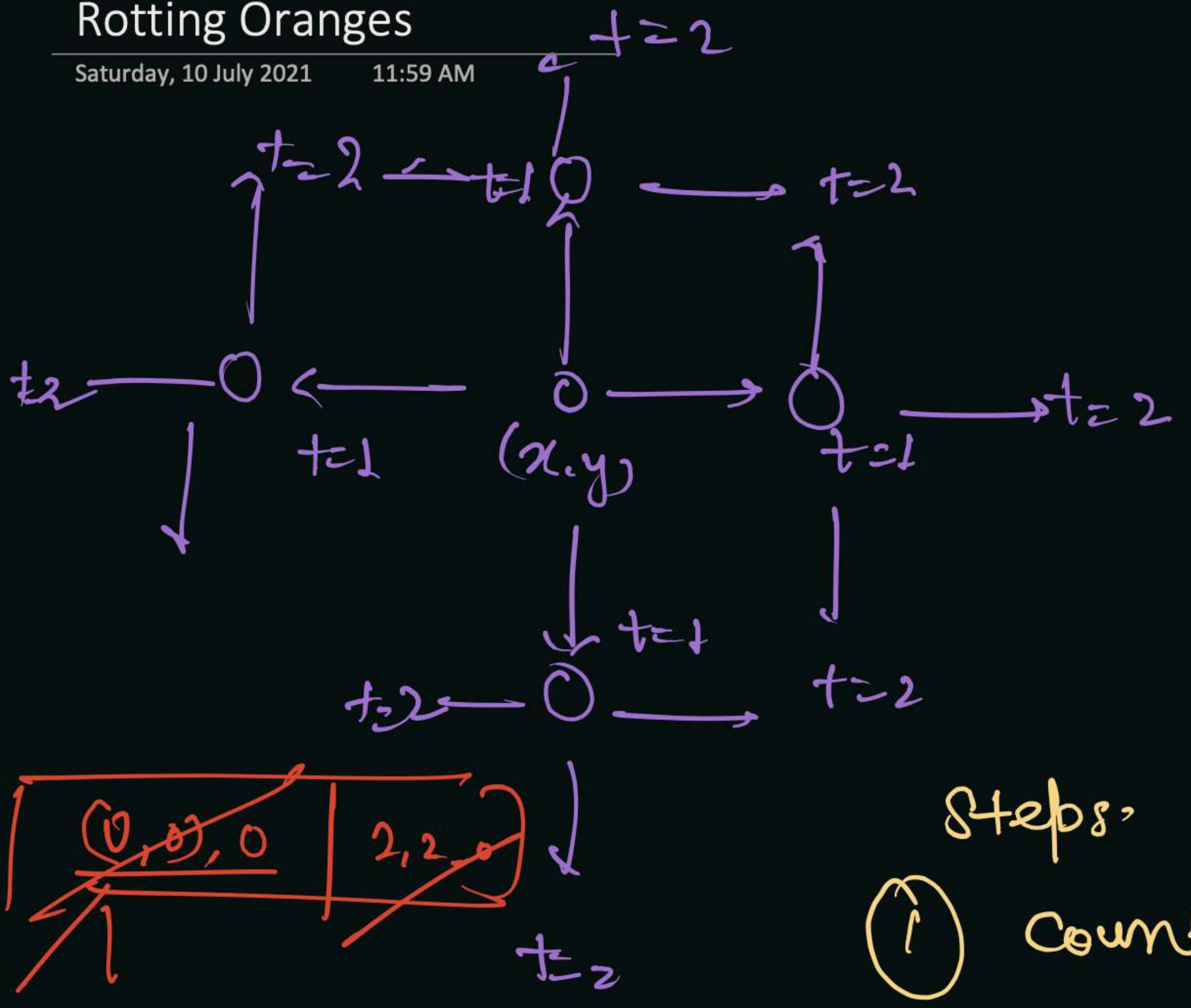
No. of peoples = 4

rate of infection spreading.



Rotting Oranges

Saturday, 10 July 2021   11:59 AM

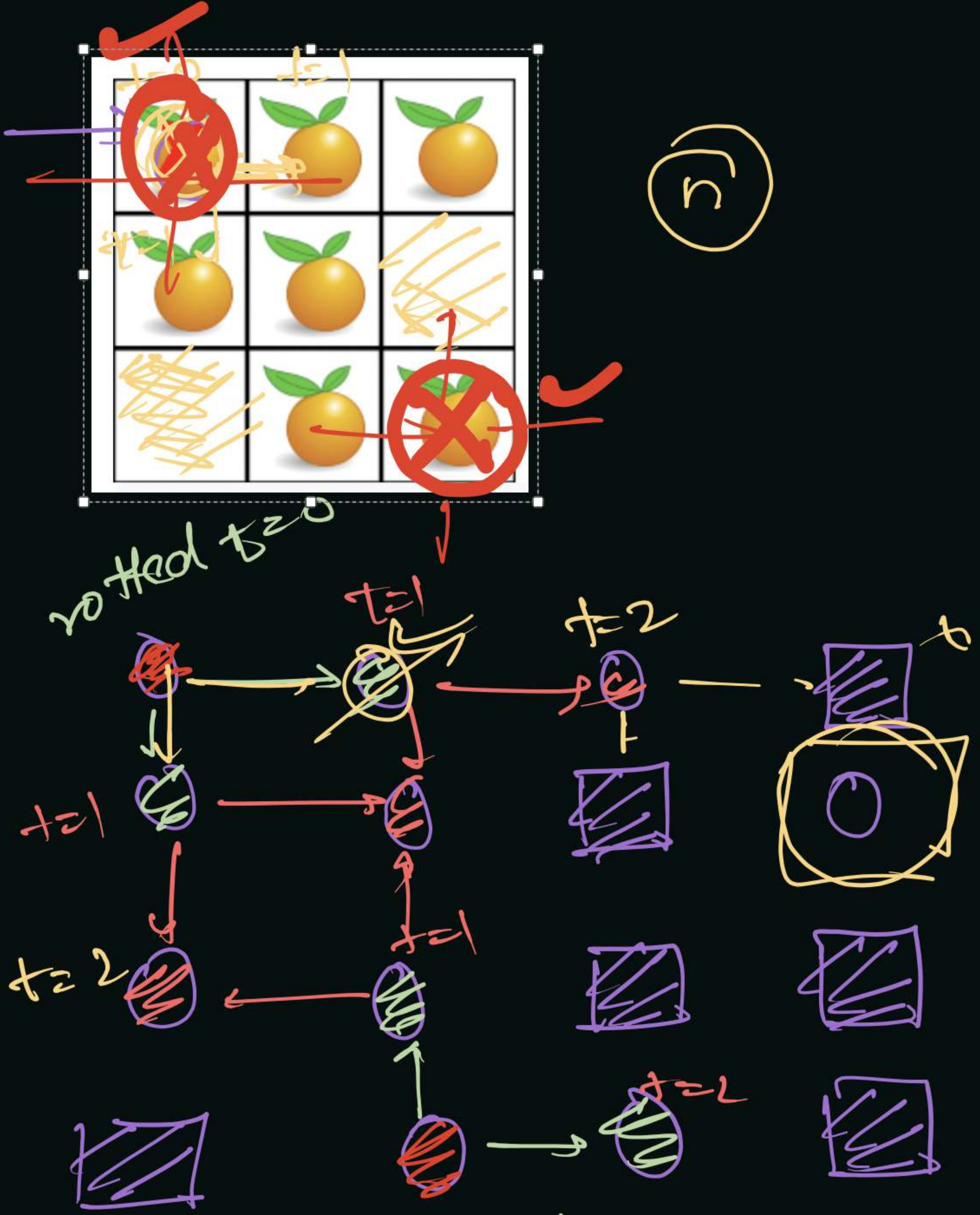


at t=0

rotted oranges = 2

fresh orange = 1

Empty cell = 0



rotted at time = 0

Steps:

① Count of total fresh orange.

② BFS, marking rotted, count --;

③ level (maximise.)

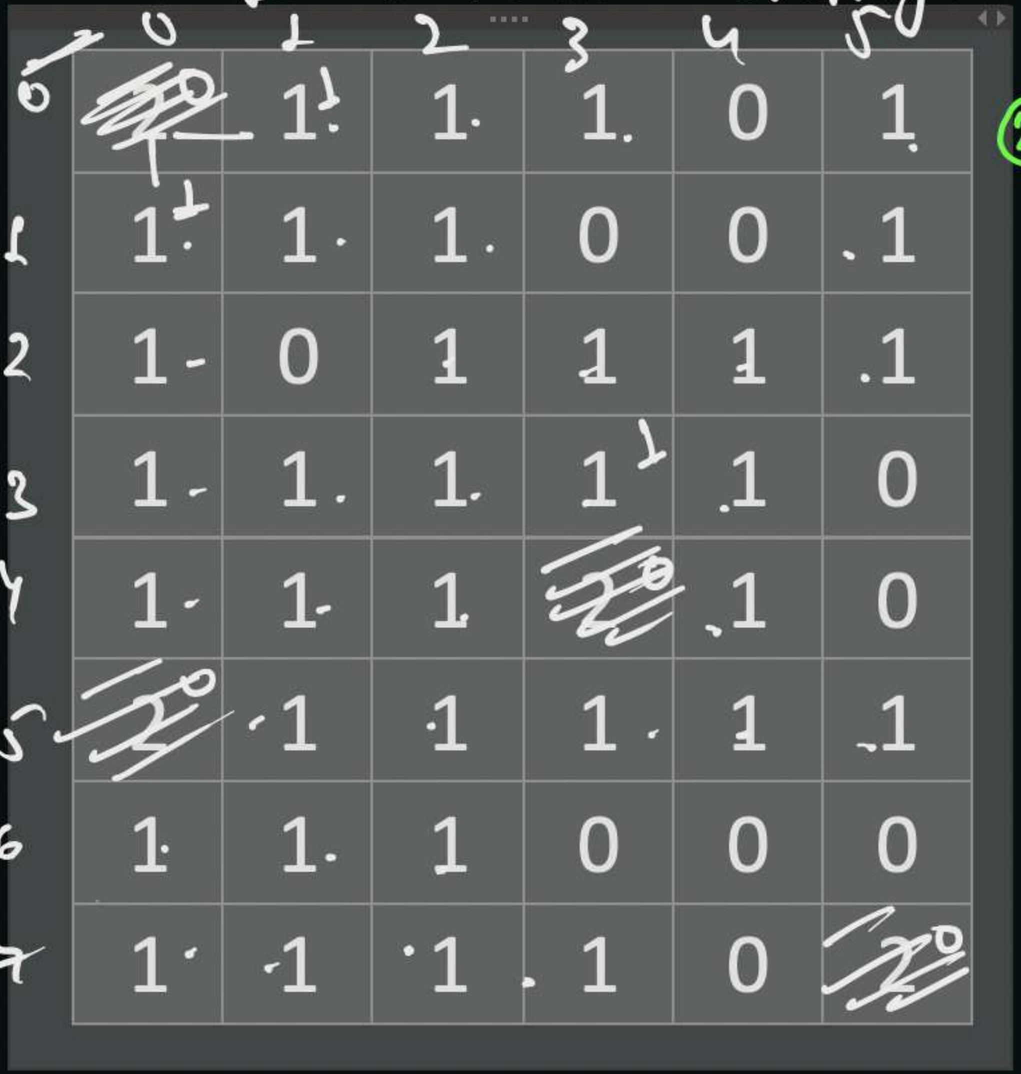
BFS with all rotted oranges ← ②

④ out of while loop ⇒ if count == 0] return level;
↳ return (-1);

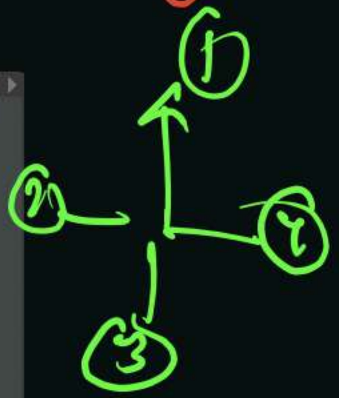
2 → rotted orange

1 → fresh orange

0 → Empty cell

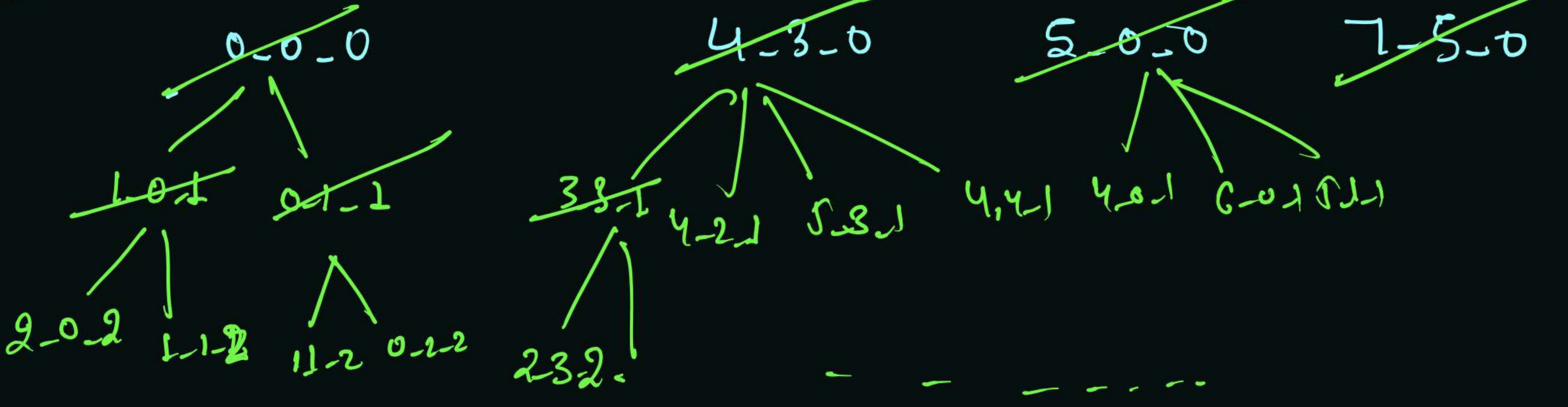


	0	1	2	3	4	5
0	2	1	1	1	0	1
1	1	1	1	0	0	1
2	1	0	1	1	1	1
3	1	1	1	1	1	0
4	1	1	1	2	1	0
5	2	1	1	1	1	1
6	1	1	1	0	0	0
7	1	1	1	1	0	2



(1) Iterate on data, find count of fresh orange and add element in queue which are rotten.

x, y - t



fresh orange = 34 + 4

= ~~38~~

~~37~~

~~36~~

~~35~~ ~~34~~ ~~33~~ ~~32~~ 31 →

(1) get & remove

(2) mark - discover time

(3) work fresh orange --

(4) add neighbours

count → 0] all the oranges are rotted

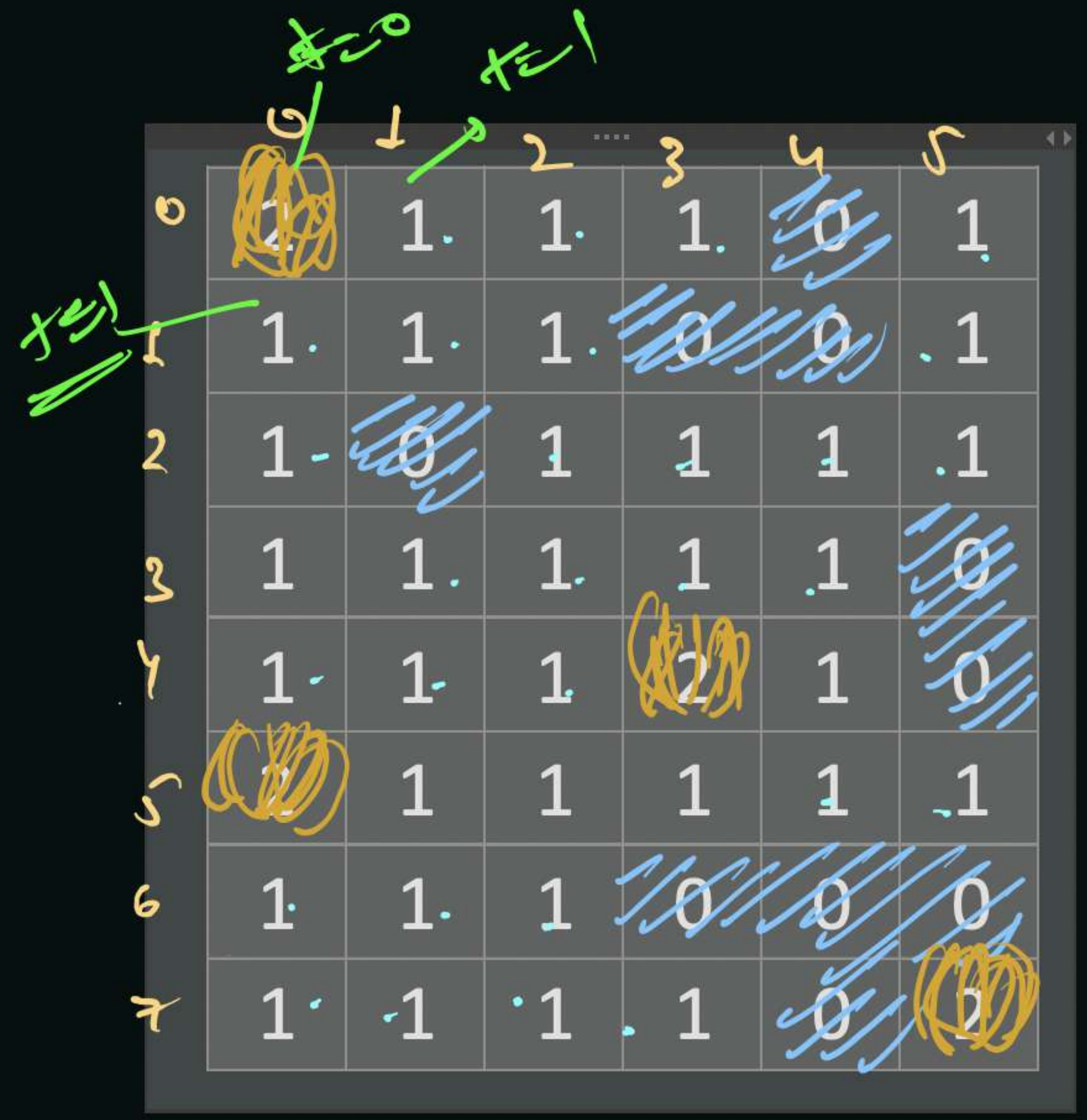
count → >0] → fresh orange is cover with Empty cell

Fire in the City →

0 → water

1 → wooden House

2 → Fired Catched in house



	0	1	2	3	4	5
0	2	1	1	1	0	1
1	1	1	1	0	0	1
2	1	0	1	1	1	1
3	1	1	1	1	1	0
4	1	1	1	2	1	0
5	2	1	1	1	1	1
6	1	1	1	0	0	0
7	1	1	1	1	0	1

✓ Time Required to completely burn the city. if not possible then return $\boxed{-1}$ (rotten orange)

✓ time of burning of every house

✓ t=0 burn at time = t ⇒

Shortest Path in Weights (Dijkstra's Algorithms)

Saturday, 10 July 2021 9:55 AM

B.F.S.

1. get + remove
2. mark
3. work
4. Add neighbour.

Shortest path in terms of Edge → BFS

P.S → ~~Queue~~ Priority Queue (Min)

src, psf, wsf

~~0, 0, 0~~

~~1, 01, 10~~ ~~3, 03, 40~~ continue

~~2, 012, 20~~

~~3, 0123, 30~~

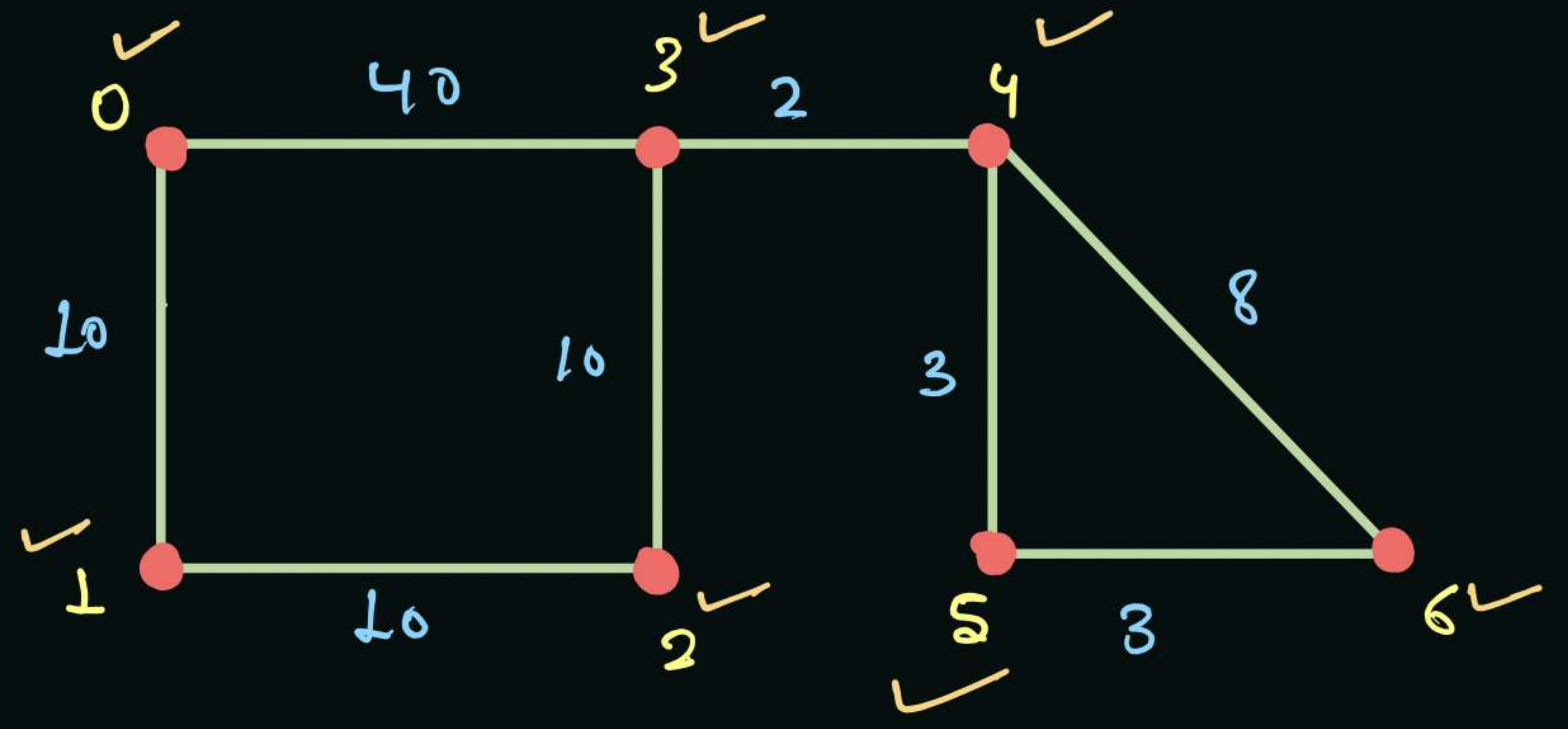
~~4, 01234, 32~~

~~5, 012345, 35~~ ~~6, 012346~~, 40 continue

~~6,~~ 0123456, 38

Src = 0

Shortest path in terms of weight → Dijkstra's



0 ✓	via	0	@ 0
1 ✓	via	01	@ 10
2 ✓	via	012	@ 20
3 ✓	via	0123	@ 30
4 ✓	via	01234	@ 32
5 ✓	via	012345	@ 35
6 ✓	via	0123456	@ 38

Minimum Wired Required to connect all PCs

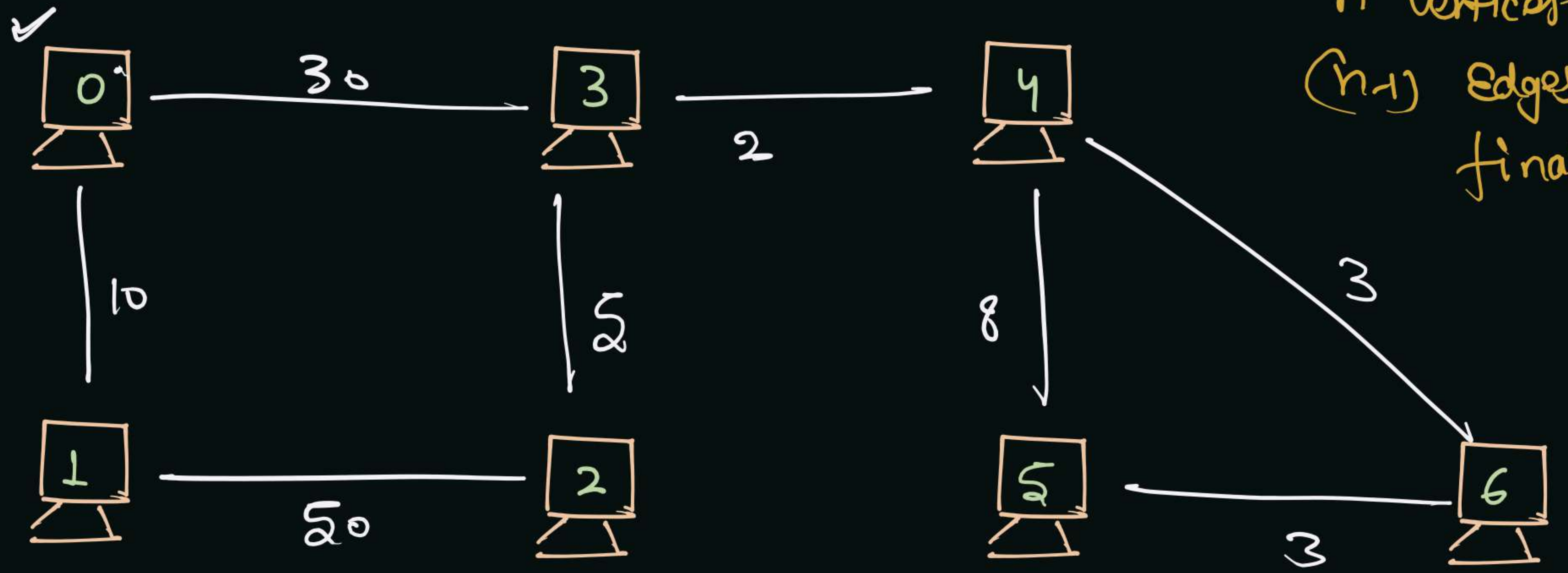
Saturday, 10 July 2021 9:55 AM

graph → connected, → resultant graph is acyclic

Prims Algorithm

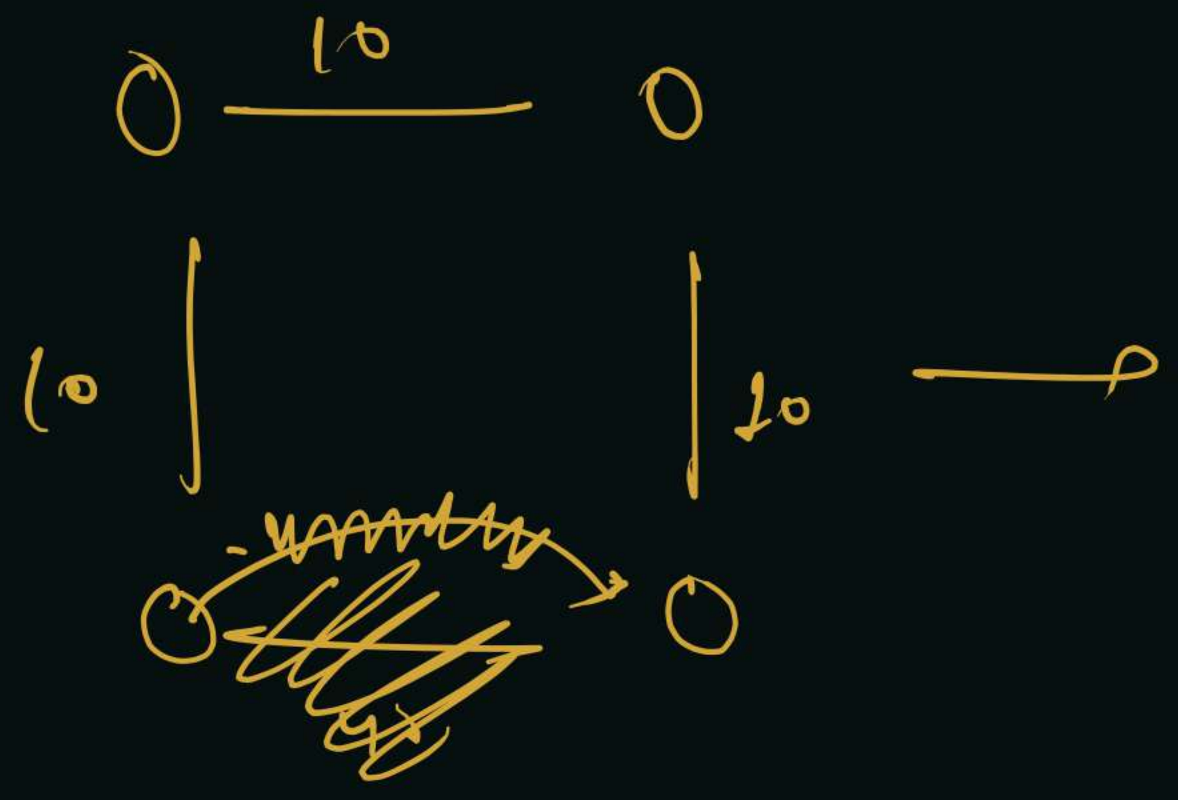
S1	S2	wire required,
0	1	10
1	2	50
0	3	30
3	2	5
3	4	2
4	6	3
6	5	3
4	5	8

↓ Help to find Minimum Spanning Tree



n vertices

(n-1) Edges in final result

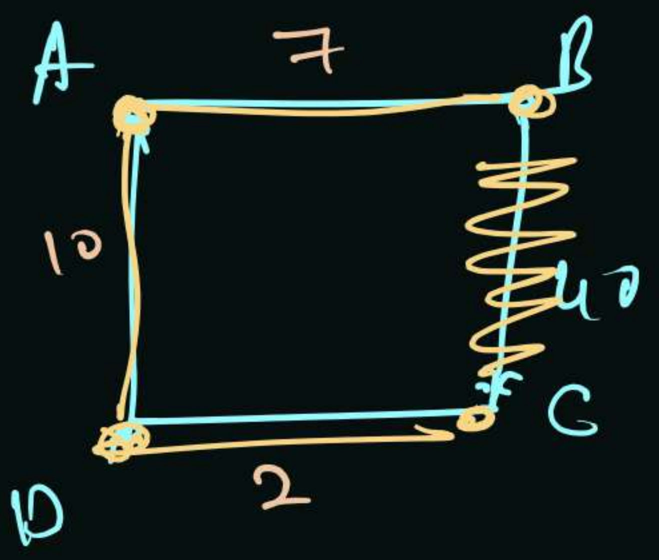


connect all the server in min. wire,

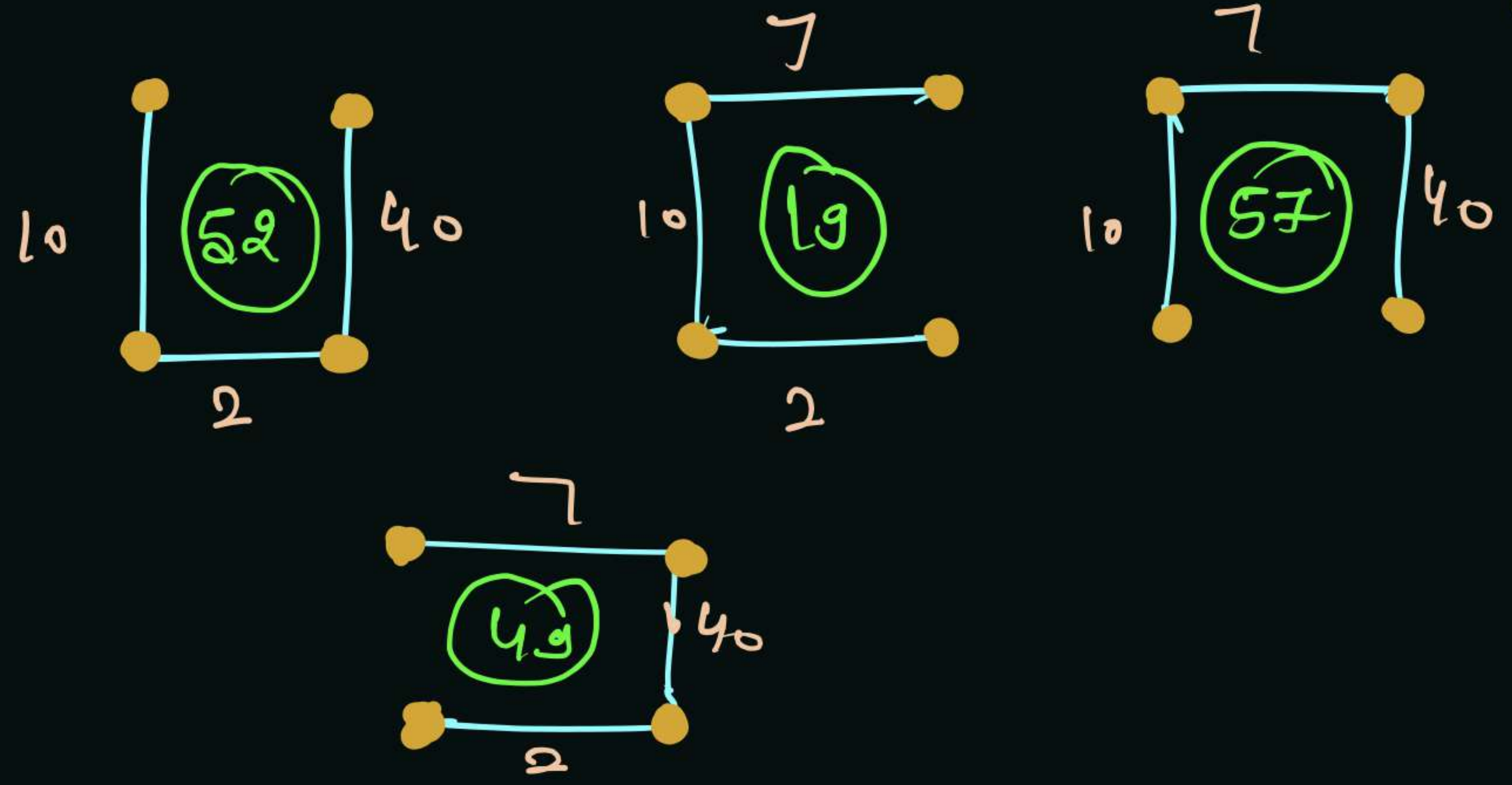
Minimum → Min.

Spanning → possibility

Tree → Connected and Acyclic

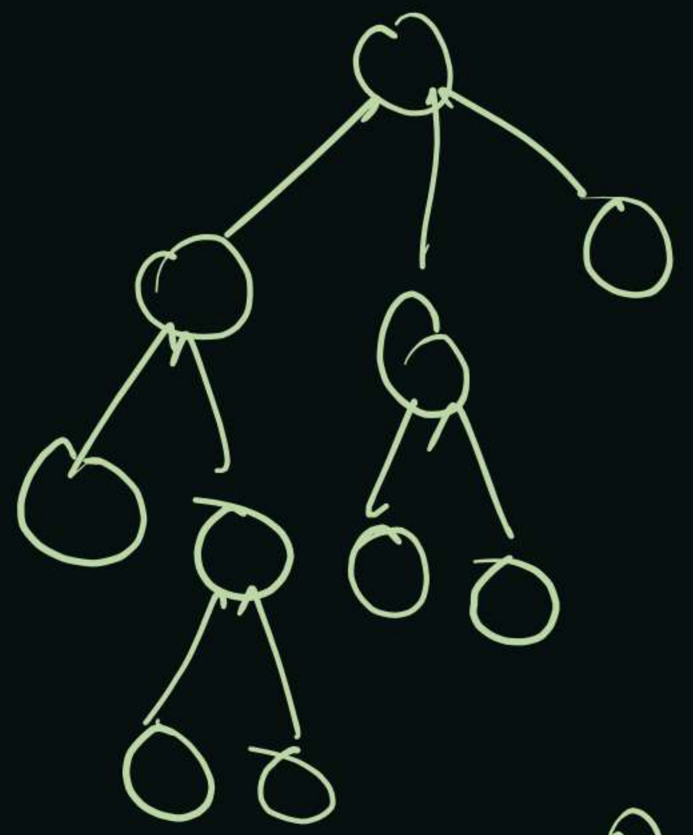


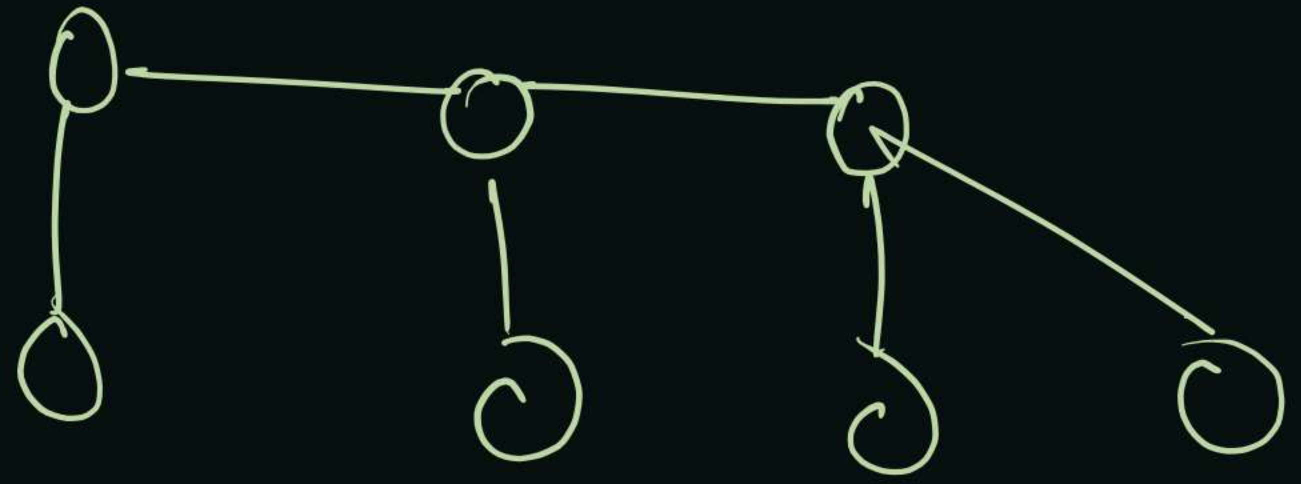
Spanning tree Min = 19



graph is Tree

{ connected, Acyclic } tree

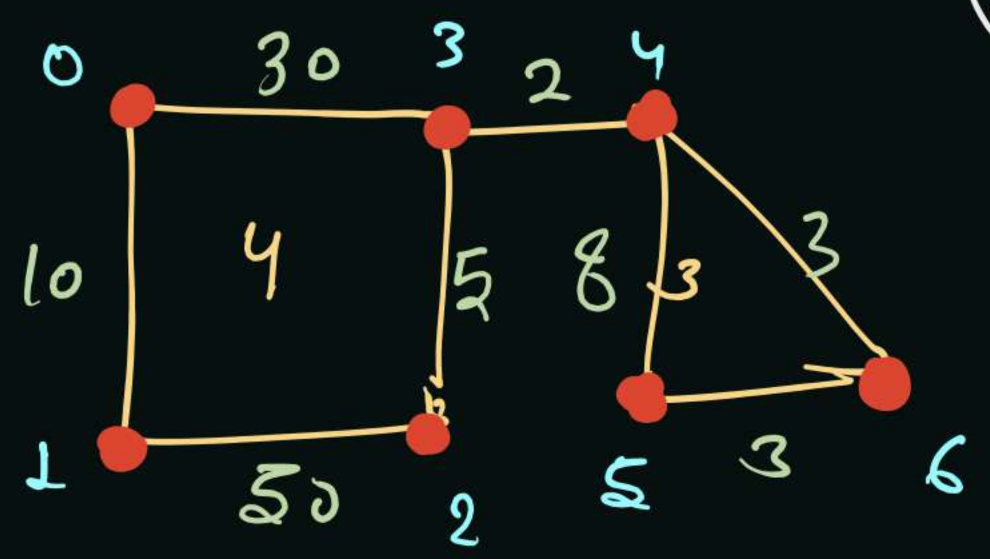




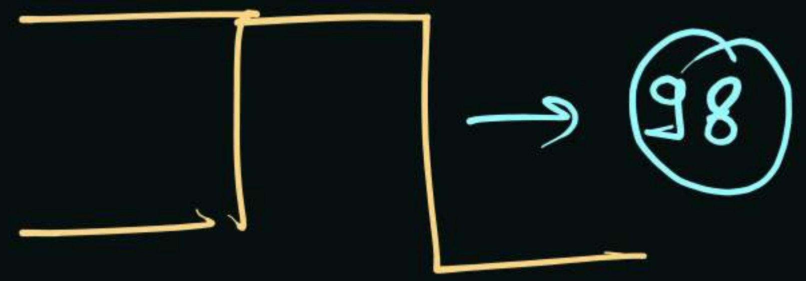
connected & Acyclic

Spanning Tree

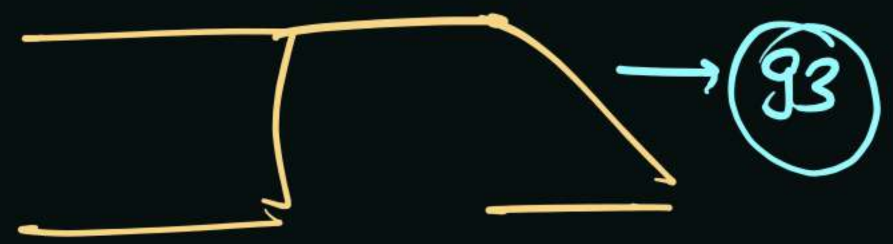
4x3=12



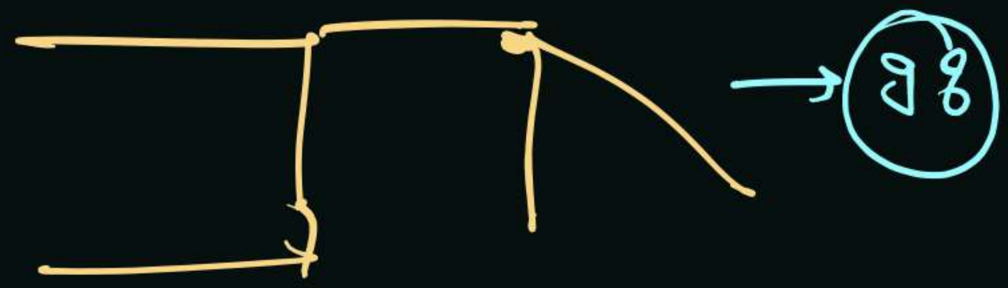
1



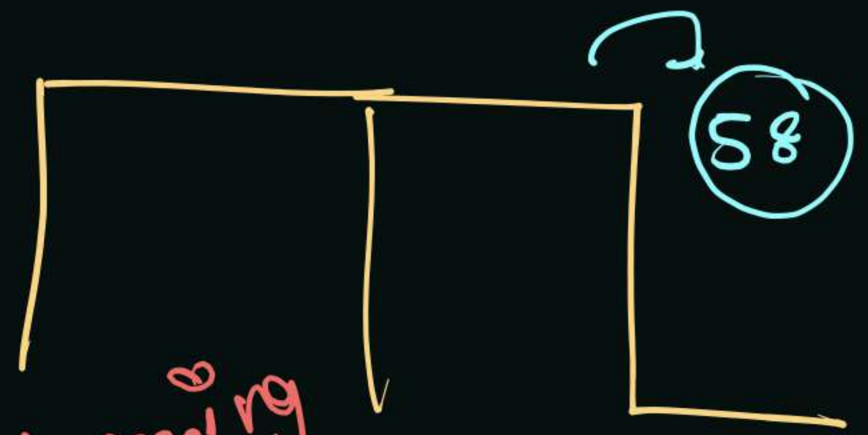
2



3

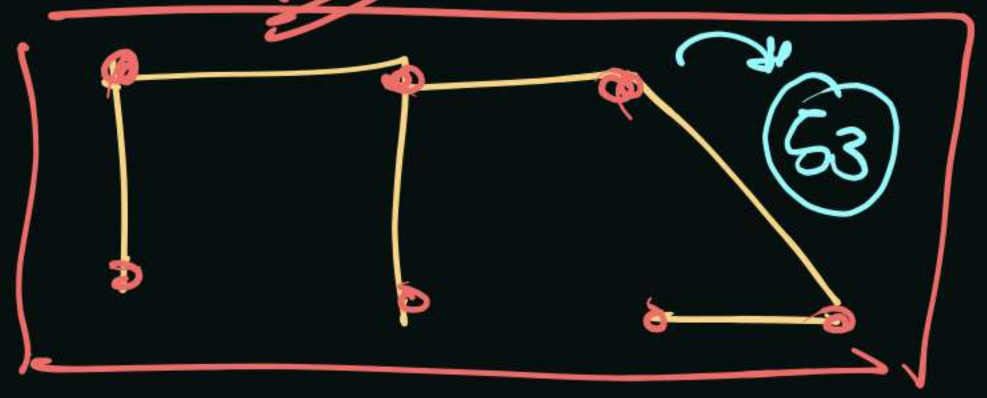


4

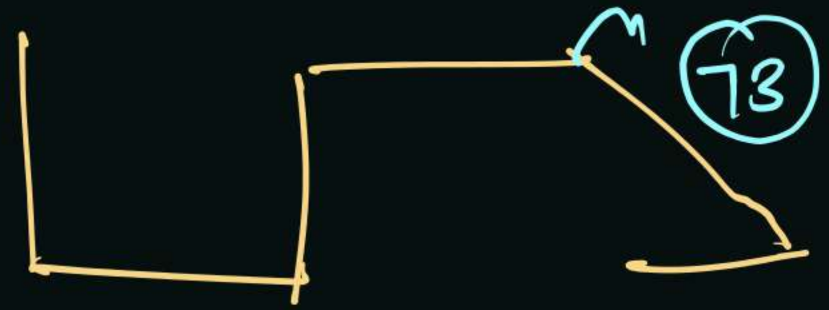


min. spanning tree

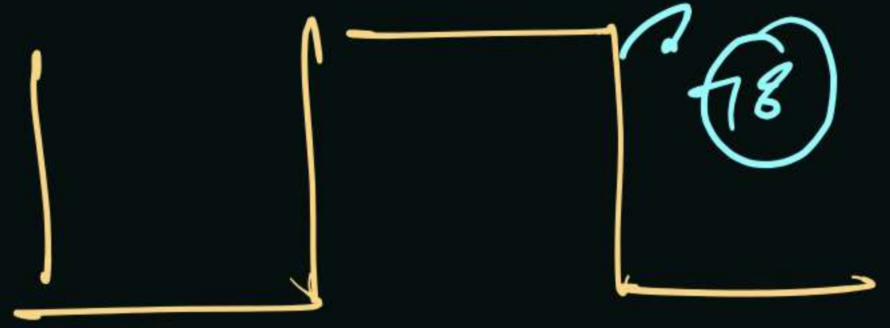
5



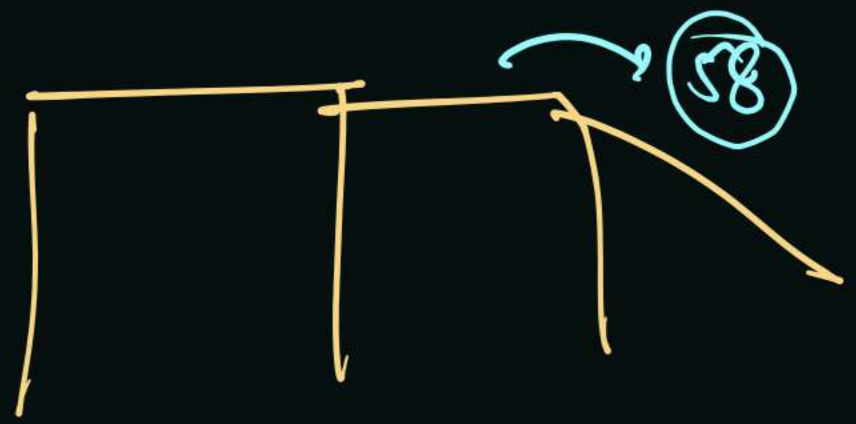
6



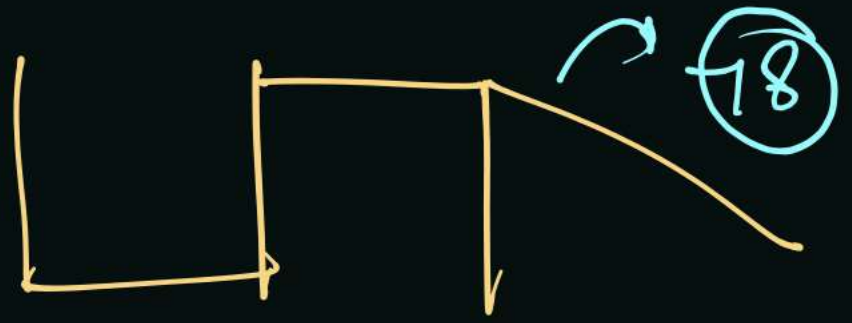
7



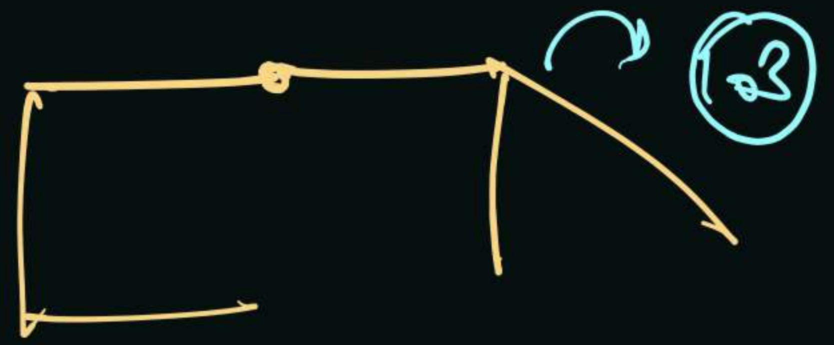
8



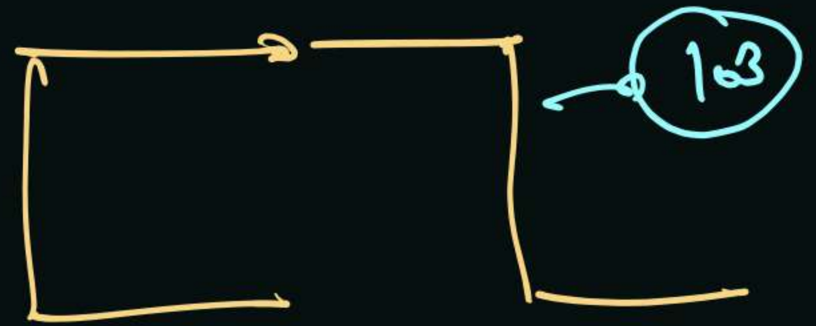
8



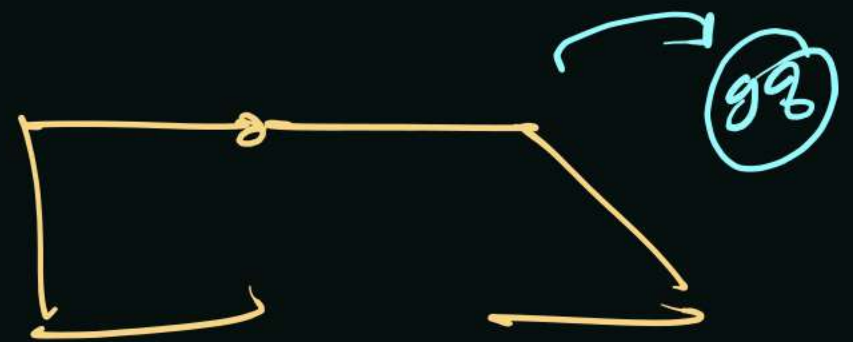
10



11



12



graph→ connected

get + remove

mark

work - addEdge

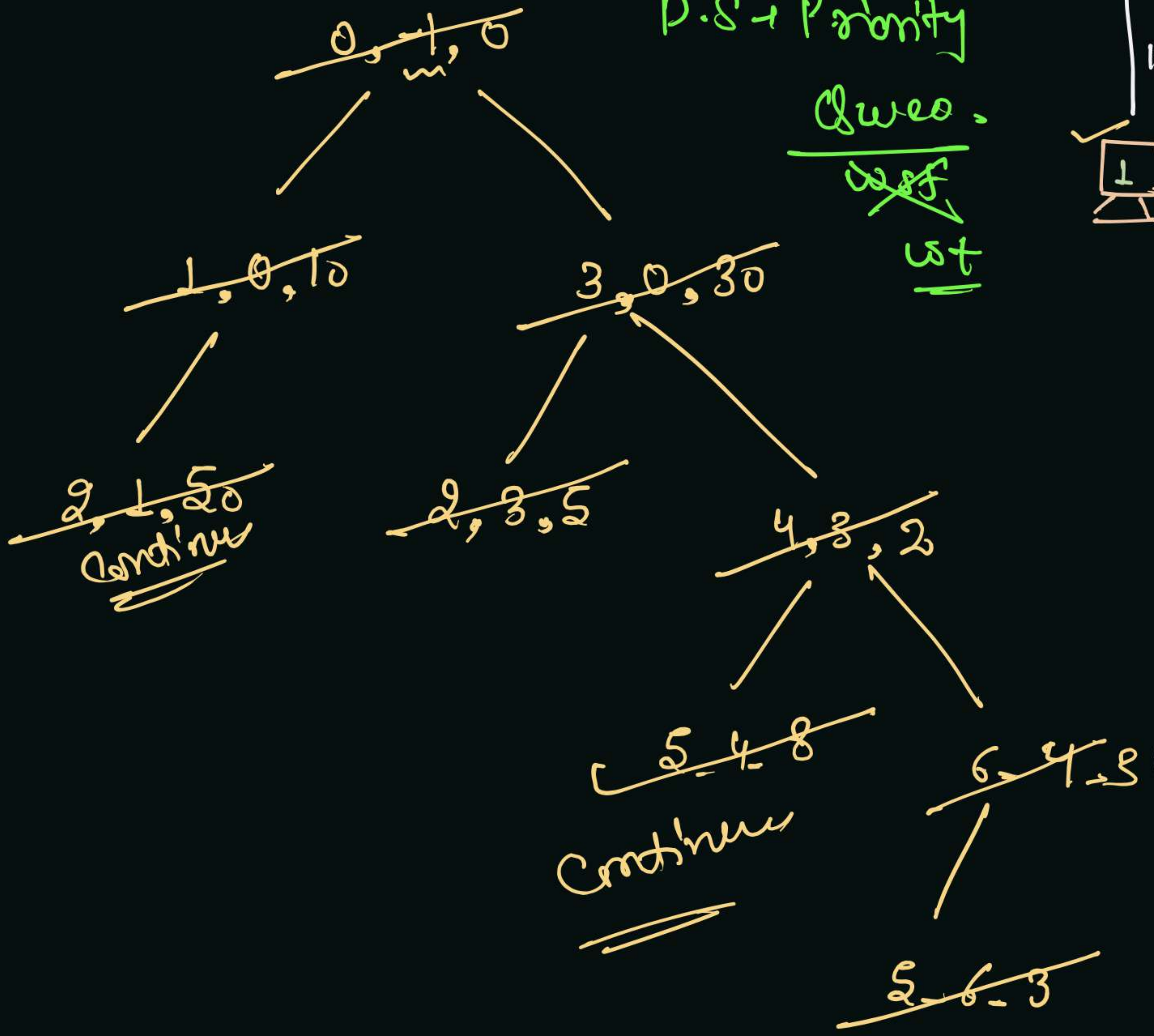
Add neighbors.

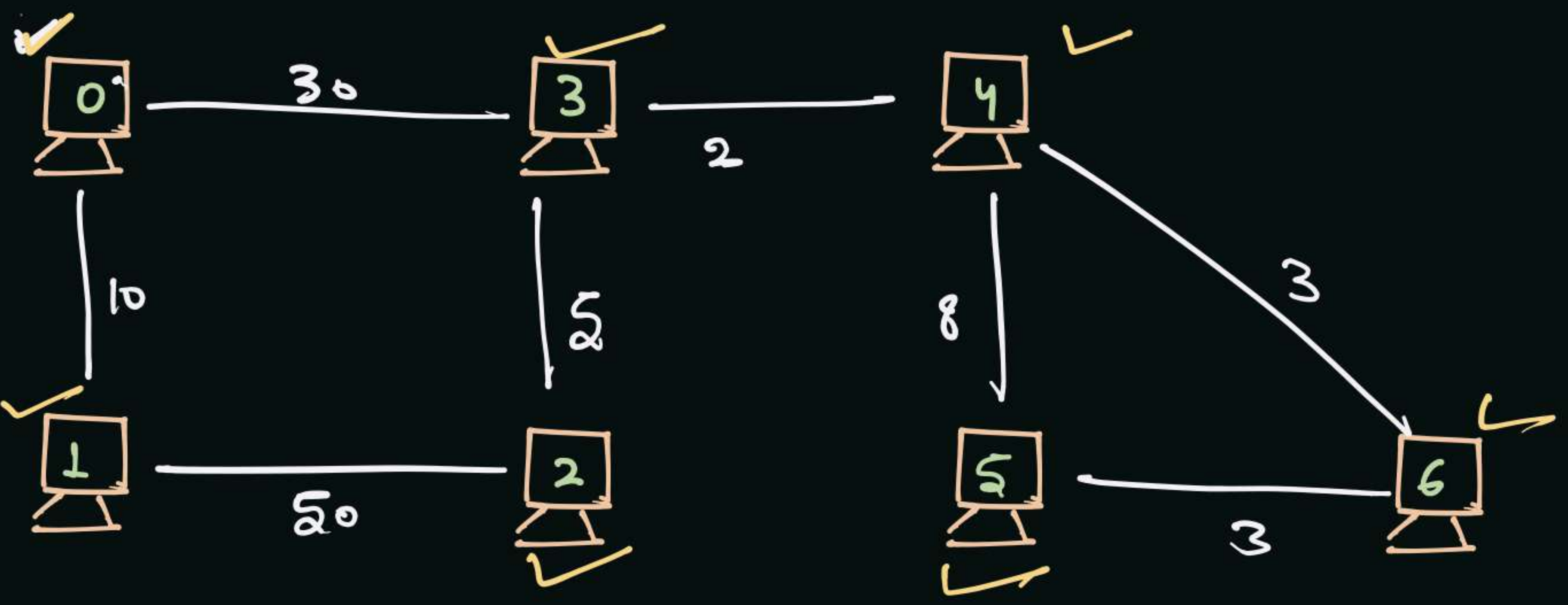
src, parent, wt

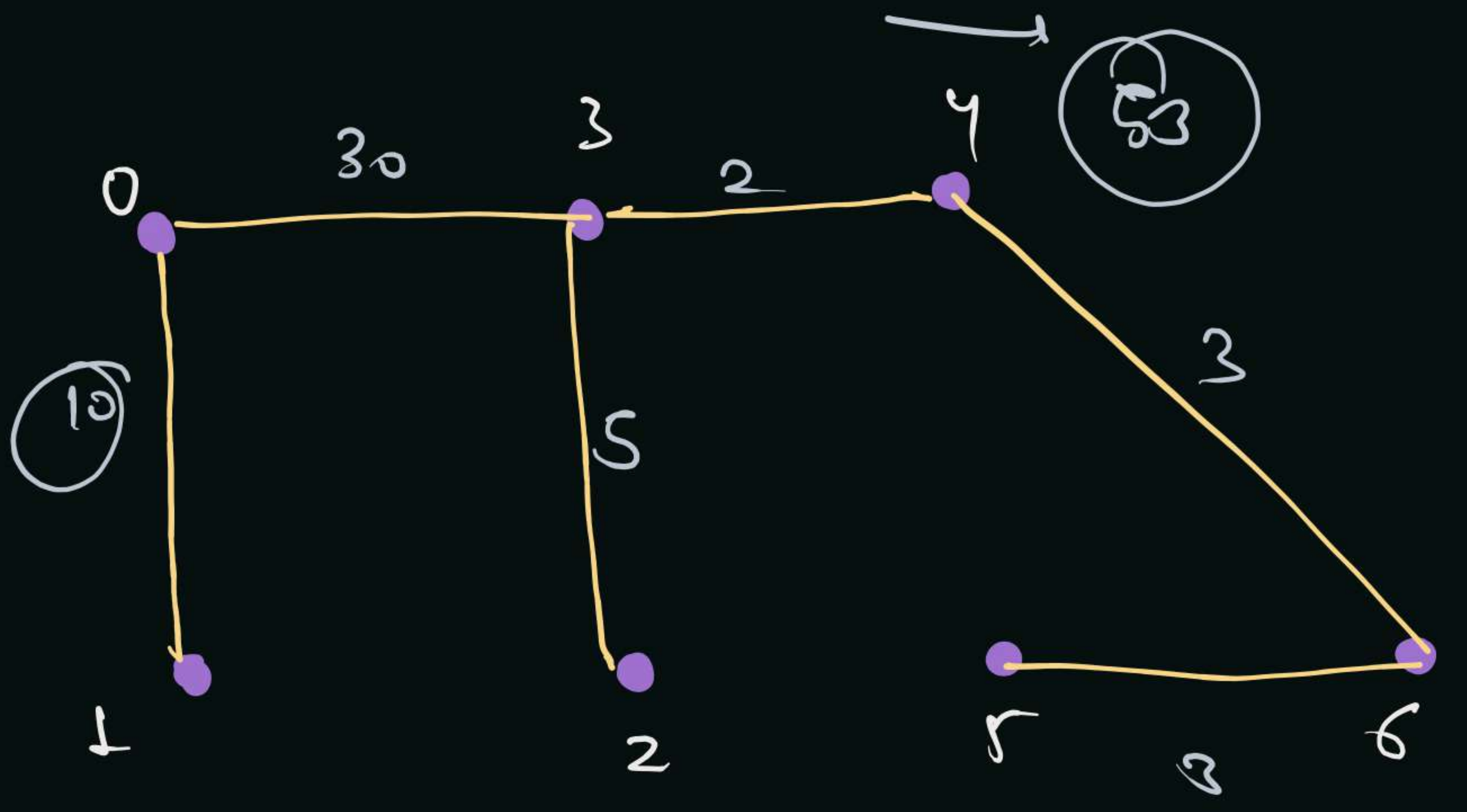
P.S → Priority Queue.

~~wsf~~

wt



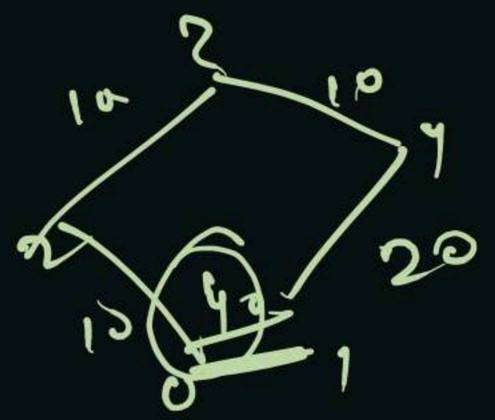




How prims Algorithms (MST. minimum spanning tree) is different fom Dijkshra's Algo.

→ Dijkstra find min path (in terms of weight) from a single source to single dest.

→ Prims find min path sum to connect all all the vertex.



Dijkstra → 0 - 1 → 40

Prims. → 0 - 2 - 3 - 4 - 1]

0 → 1 ✗

Iterative DFS

Saturday, 10 July 2021 2:24 PM

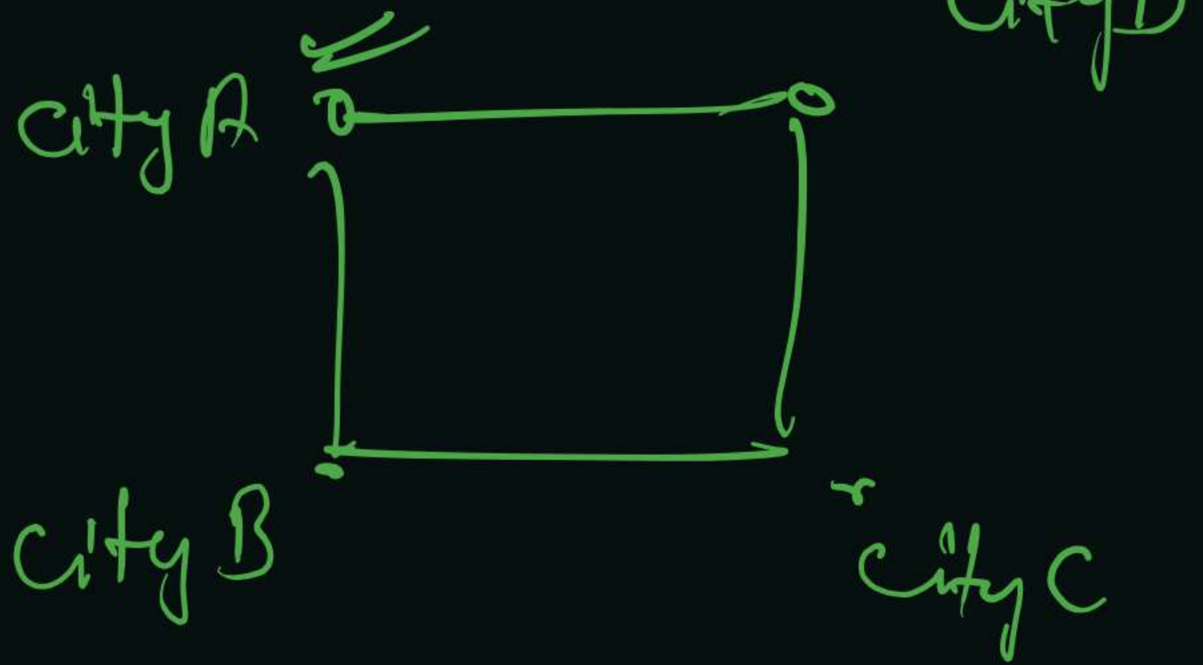
Home work

D.F.S → ① Try to solve by State Method, which is already done in trees.

Reverse order D.F.S ② Try using BFS algorithm, just replace Queue from Stack.

✓ order of compilation

✓ Evaluate Division


City D
City A
City B
City C