Introduction to Graph

Sunday, 4 July 2021 10:28 AM

GRAPHS

Data Structure covered :

Linear D.S.

1. Arrays
2. ArrayList
3. Stacks
4. Queues
5. Linked List

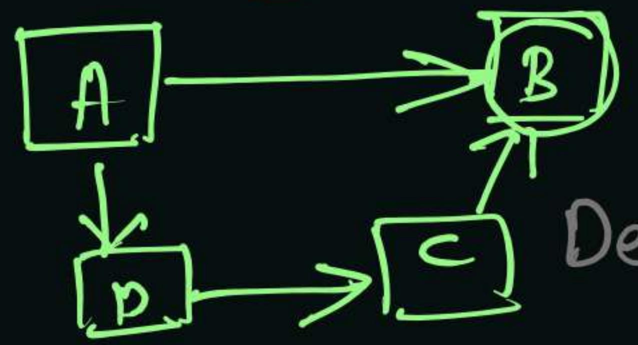
Arrays, ArrayList, Stacks, Queues } continuous Memory Get+Set

Stacks, Queues } fix Rule

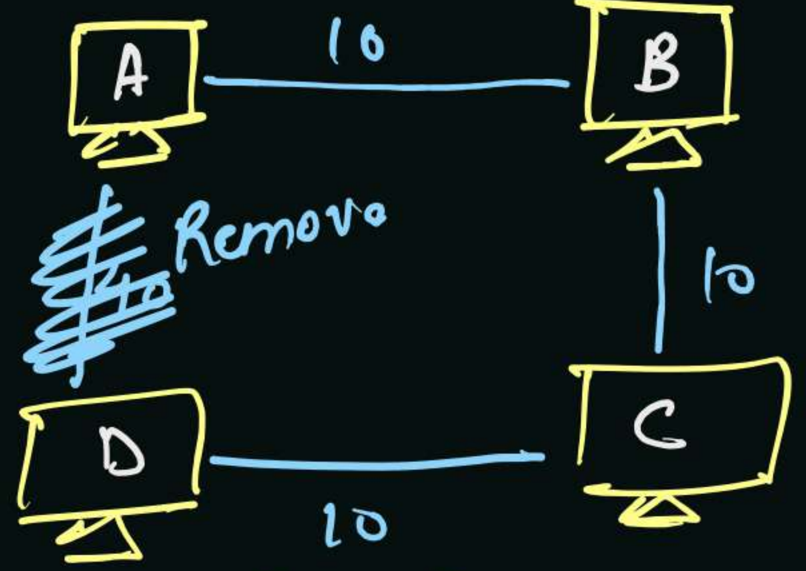
Non-Linear D.S.

6. Trees → Generic Tree → Binary Tree → BST
7. Heap
8. HashMap

Linked List, Trees } Random Memory Allocation connected

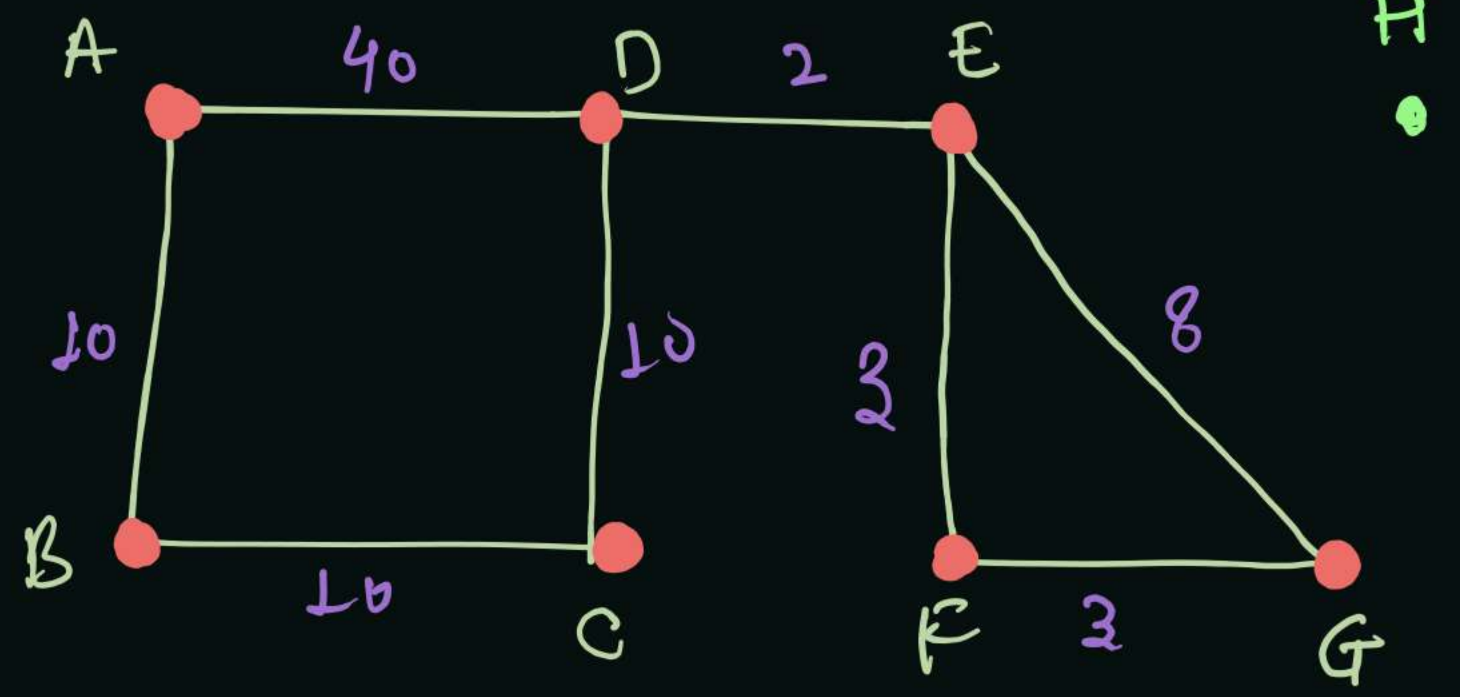


Dependency of Code



B → C → D → A

connections between cities : →



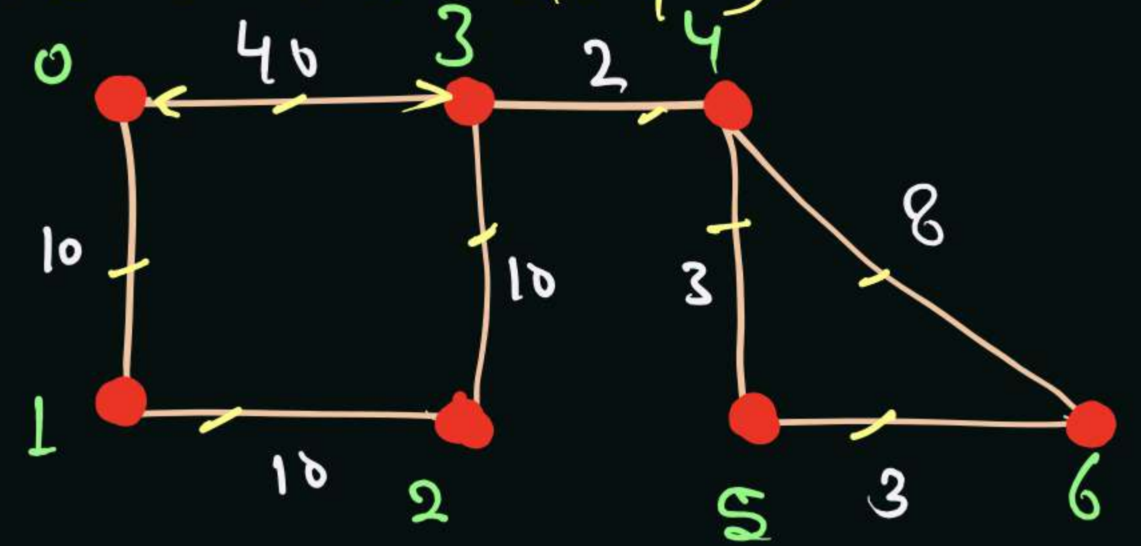
Problem which can be solve using graph →

1. Is there any path from city A to G
2. Min path from a src city to dest. city in terms of dist.
3. Min path from a src city to dest. city in terms of Min Intermediate city
4. ✱ Connect all city with dist
5. Order of Compilation.

Implementation of Graph :→ (Undirected Graph)

① Adjacency Matrix

② Adjacency List



vertex → {0, 1, 2, 3, 4, 5, 6}

wts → {10, 40, 2, 3, 8}

Edges → {0-3, 0-1, 1-2, 2-3 ...}

neighbours → 0 → 3
↳ 1

1 → 0
↳ 2

Neighbour of 0 are 1, 3 - -

① Adjacency Matrix →

	0	1	2	3	4	5	6
0	-1	10	-1	40	-1	-1	-1
1	10	-1	10	-1	-1	-1	-1
2	-1	10	-1	10	-1	-1	-1
3	40	-1	10	-1	2	-1	-1
4	-1	-1	-1	2	-1	3	8
5	-1	-1	-1	-1	3	-1	3
6	-1	-1	-1	-1	8	3	-1

Drawbacks of Adjacency Matrix →

① Memory wastage

② No. of nodes can't be handled if it is more than 10^4.

③ Traversal is difficult

Graph → Directed Graph
→ Undirected Graph

② Adjacency List

Display·

vtx·

✓ Helpful to represent the graph?

array

0 → 0-3-40, 0-1-10

1 → 1-0-10, 1-2-10

2 → 2-1-10, 2-3-10

3 → 3-0-40, 3-2-10, 3-4-2

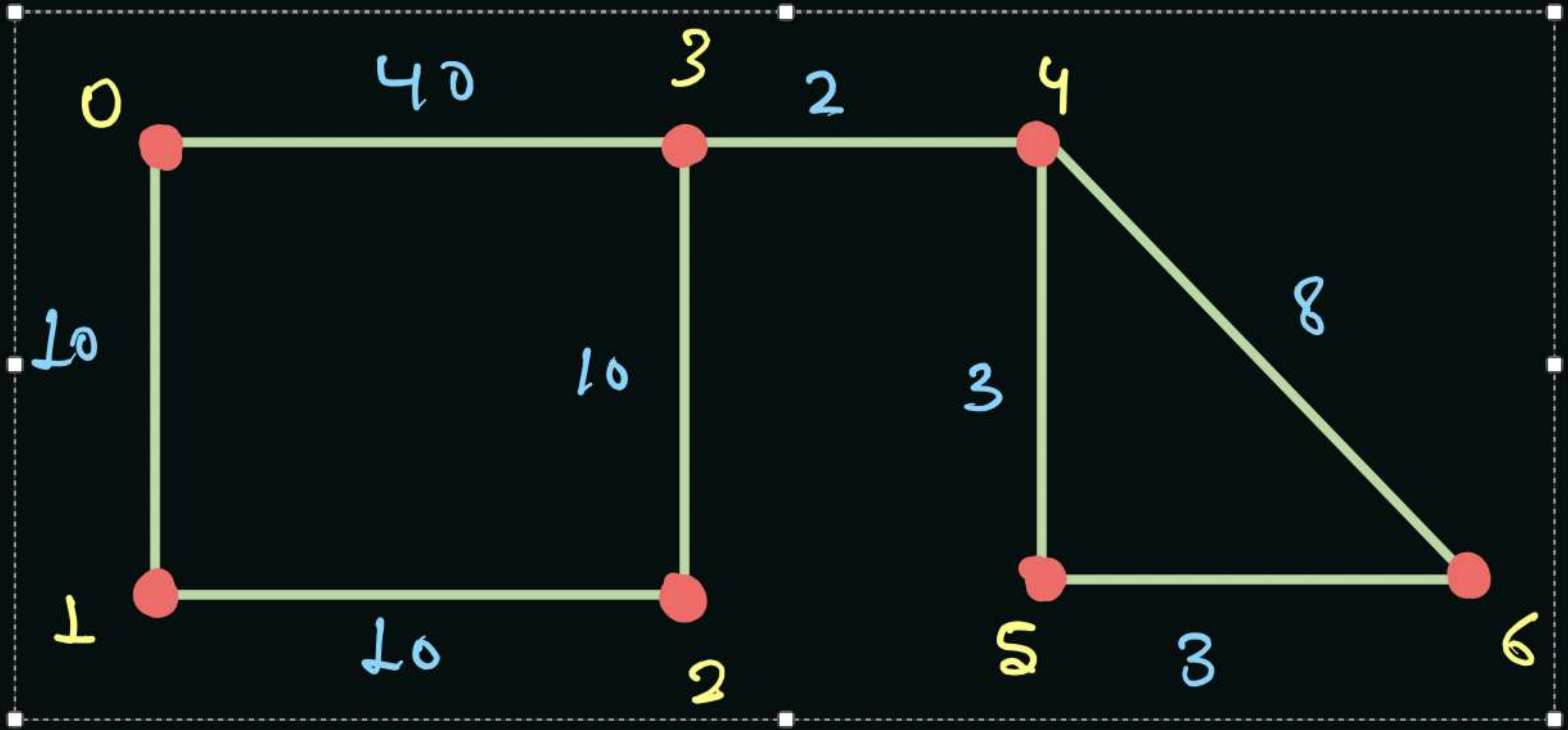
4 → 4-3-2, 4-5-3, 4-6-8

5 → 5-4-3, 5-6-3

6 → 6-4-8, 6-5 3

array List

Represent an Edge;



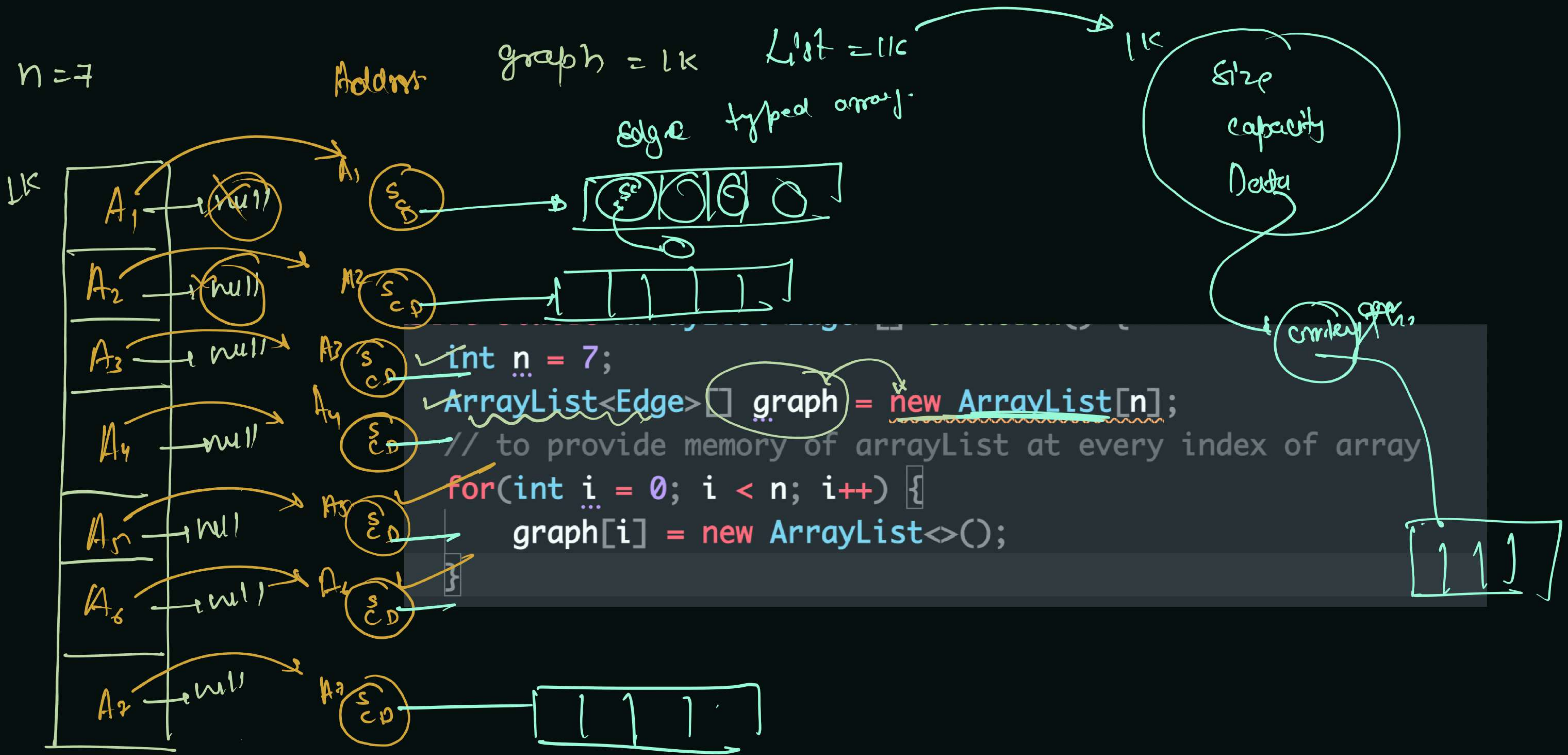
1-2-10

no. of vertex = n

ArrayList<Edge>[] graph = new ArrayList[n];

no. of vertex

class Edge → [int src
int nbr
int wt]



```
int n = 7;
ArrayList<Edge>[] graph = new ArrayList[n];
// to provide memory of arrayList at every index of array
for(int i = 0; i < n; i++) {
    graph[i] = new ArrayList<>();
}
```

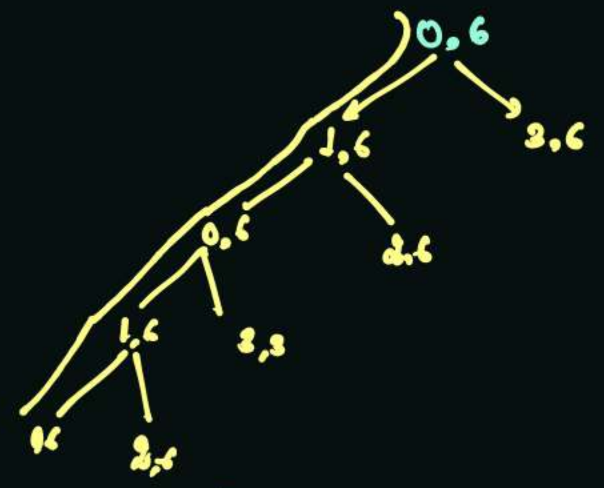
Has Path? / DFS

Sunday, 4 July 2021 10:33 AM

src = 0

dst = 6

marking - ||✓

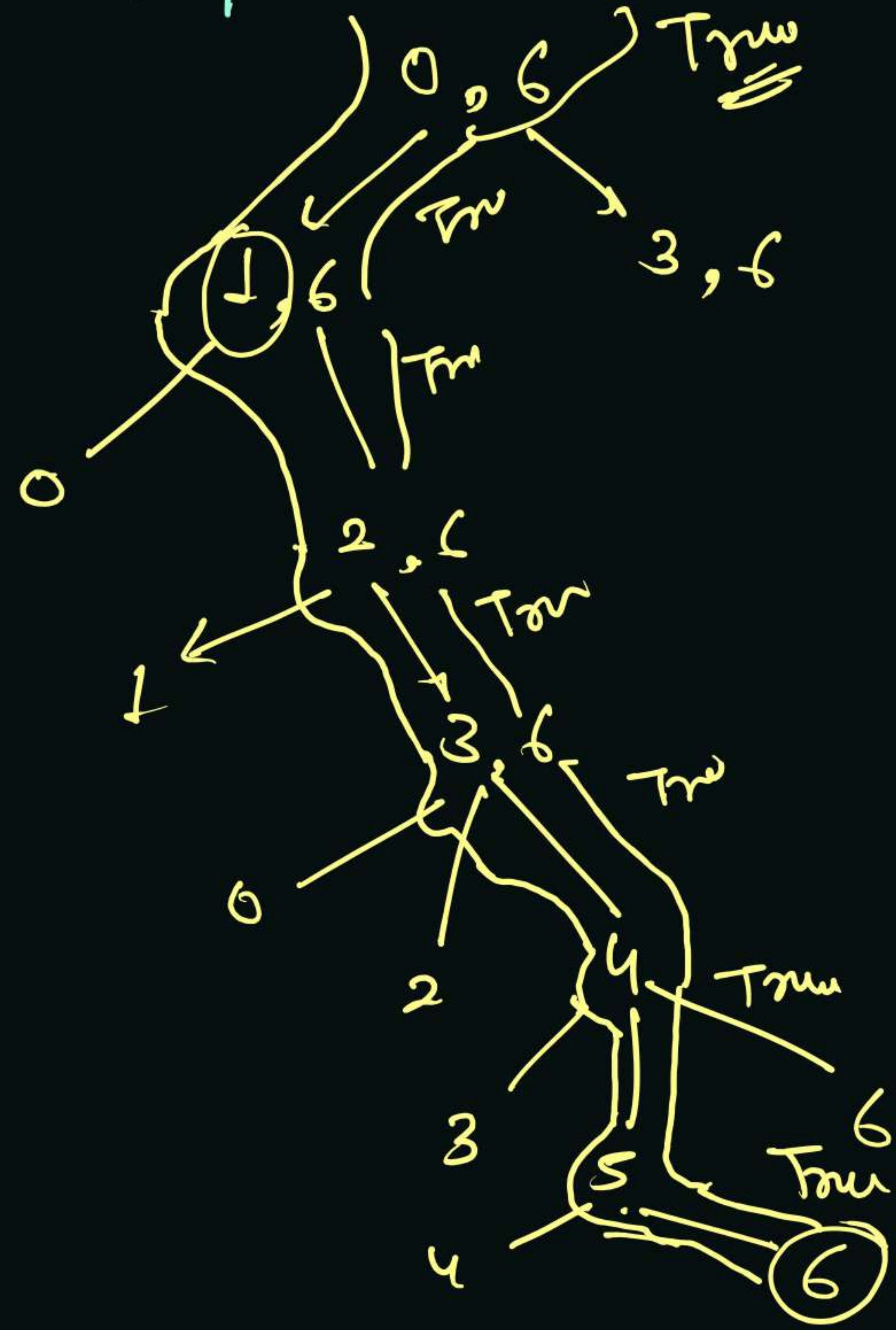


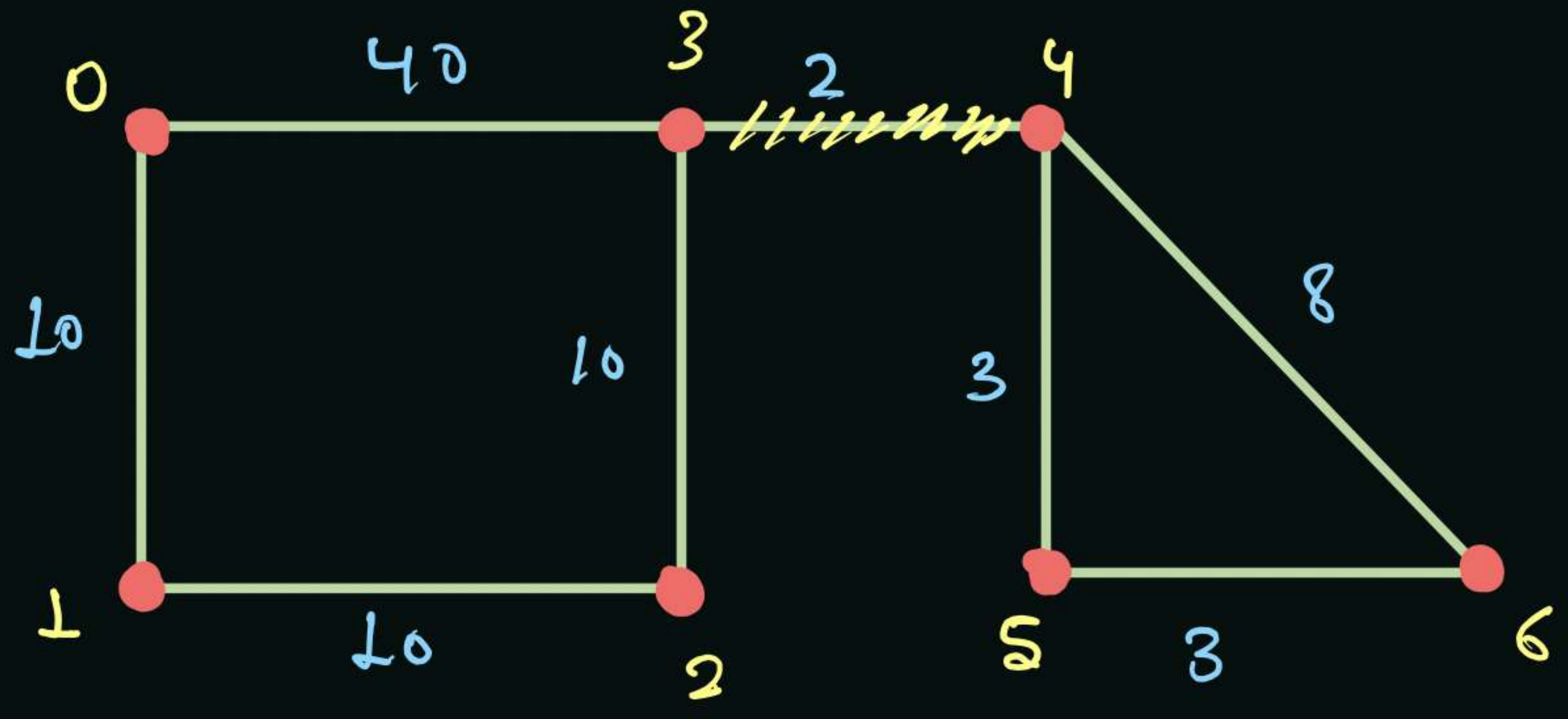
Repitition

Stack overflow

	0	1	2	3	4	5	6
visited	T	T	T	T	T	T	

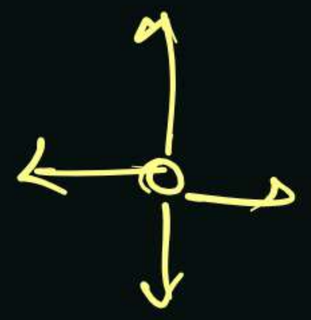
graph, src, dst





Floodfill

tl dr



* visited array.

* Move toward unvisited neighbours.

Print All Paths

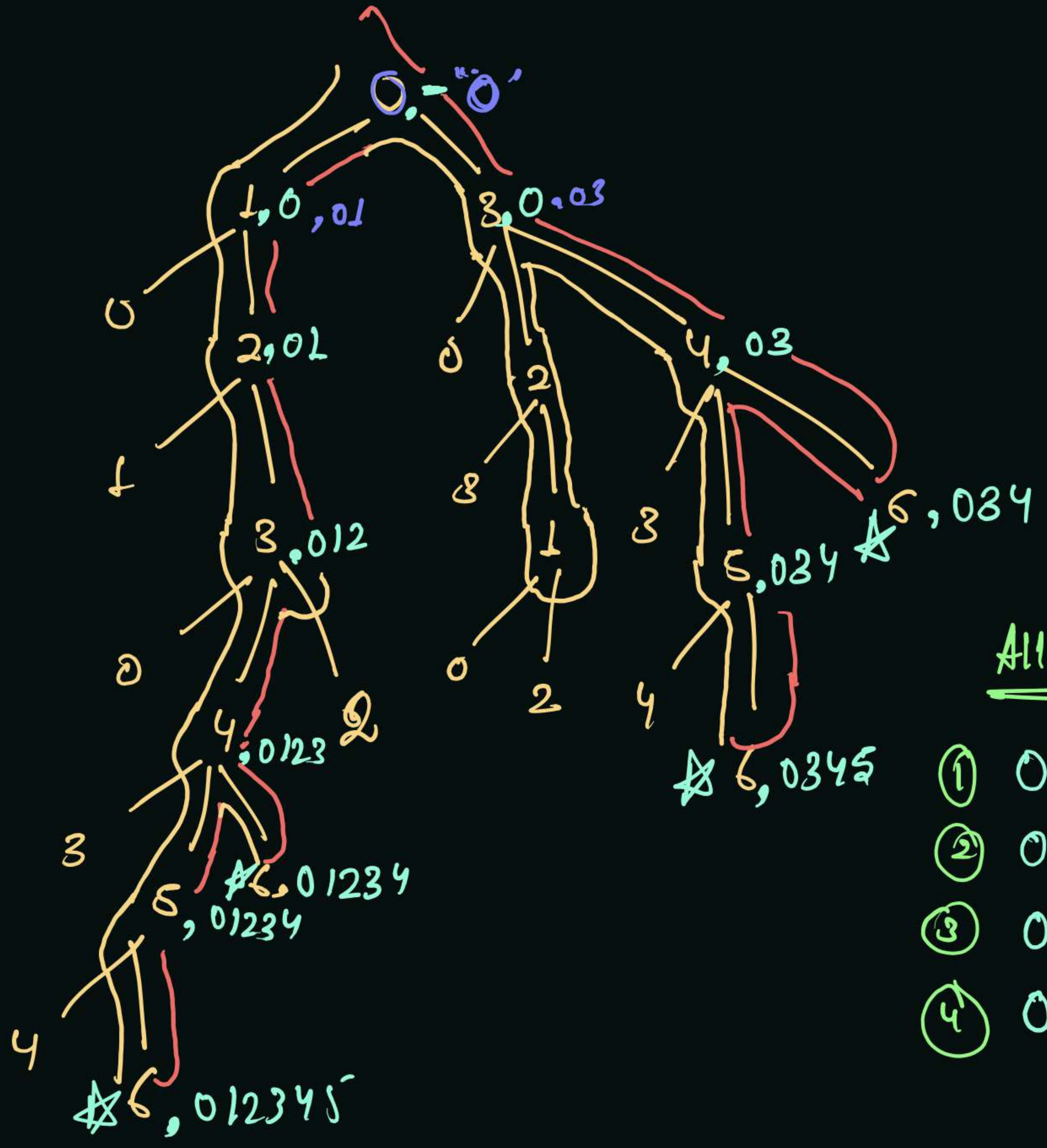
Sunday, 4 July 2021 10:33 AM

0 1 2 3 4 5 6

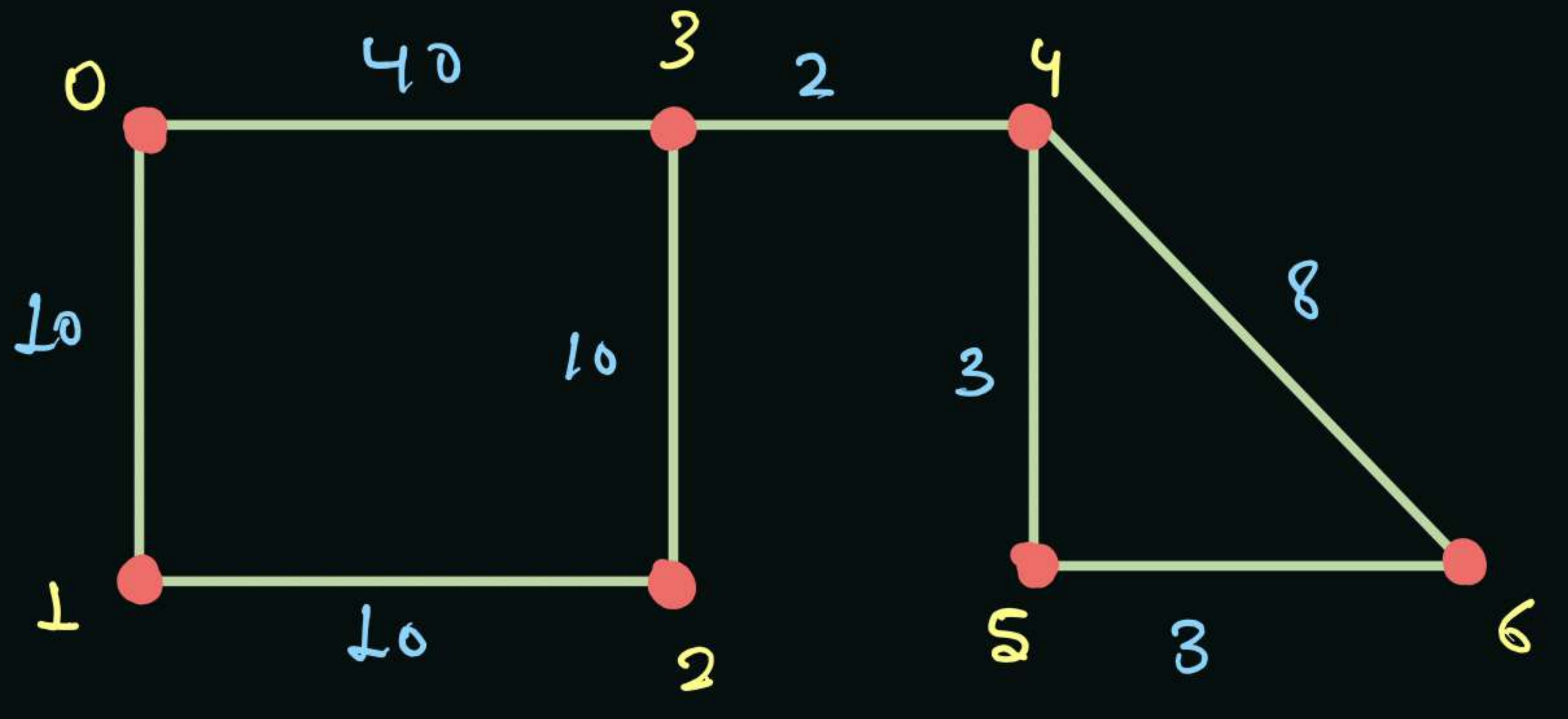
vis →

src = 0

dst = 6



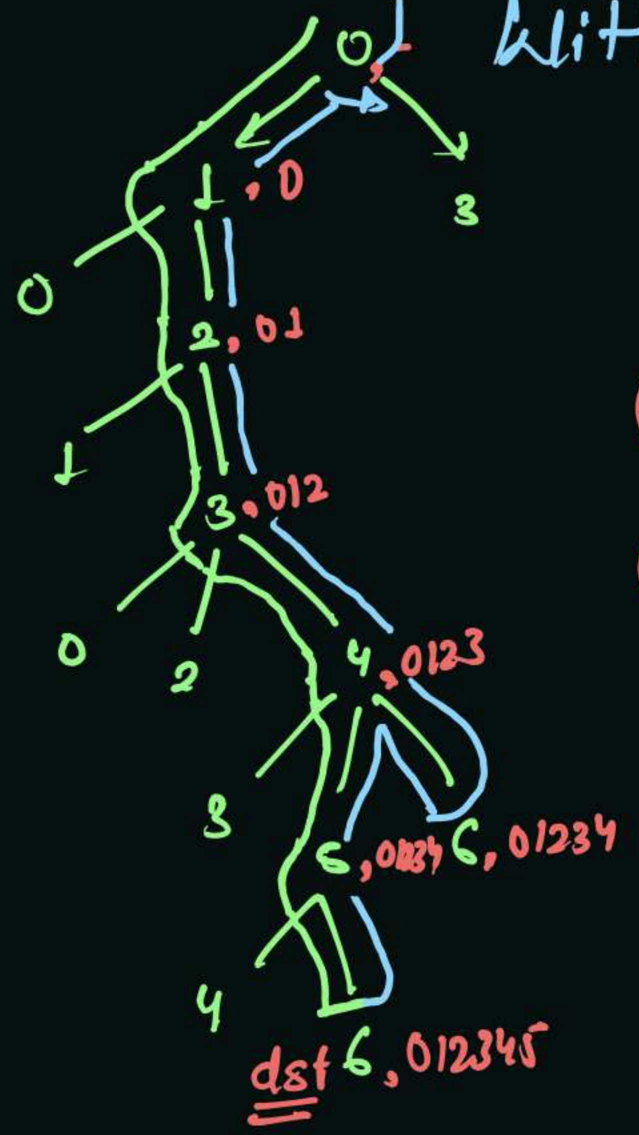
Need of Unmark



All path

1. 0 1 2 3 4 5 6
2. 0 1 2 3 4 6
3. 0 3 4 5 6
4. 0 3 4 6

Without Unmark



0 1 2 3 4 5 6

0 1 2 3 4 6

print all paths from src to dst

with cost sum →

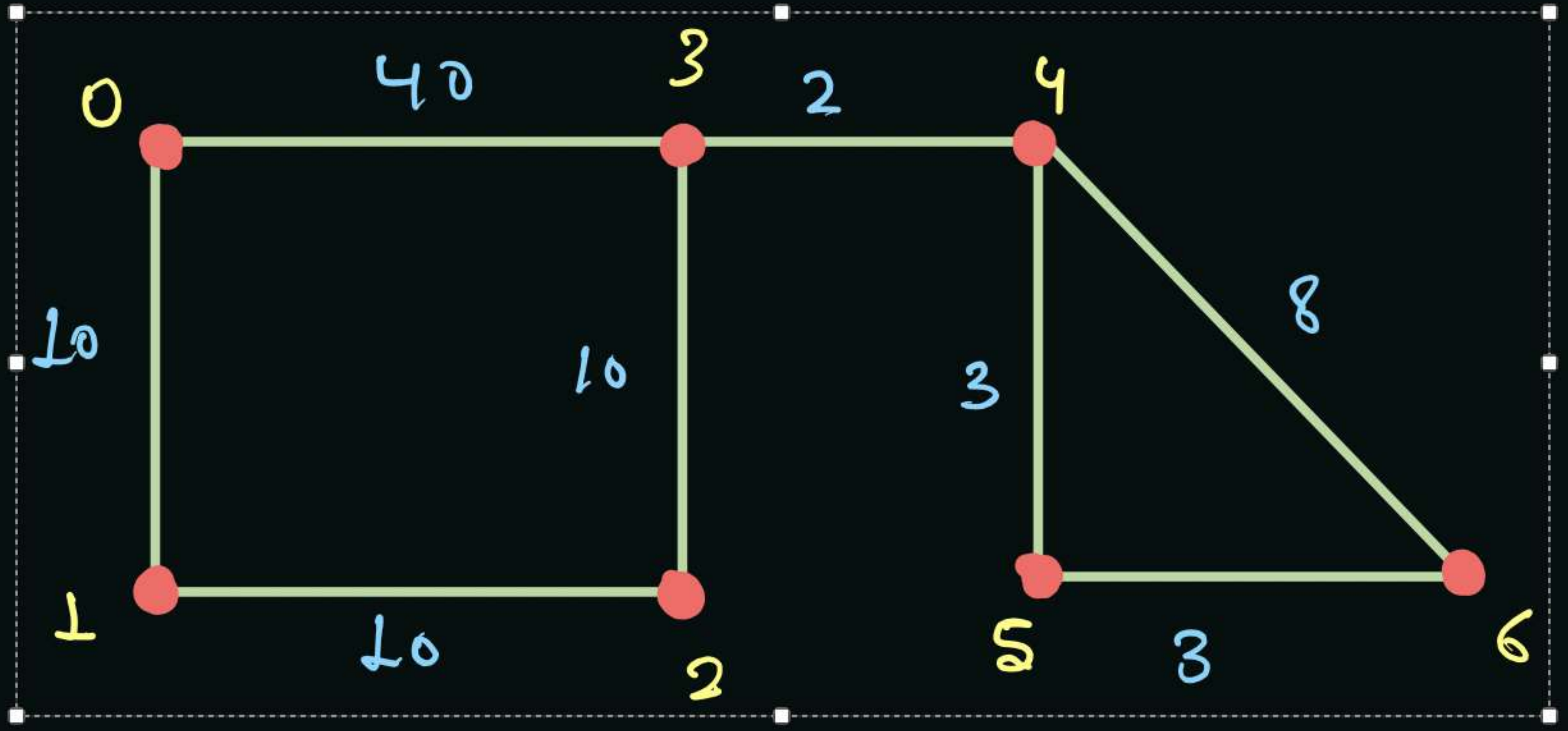
src=0

dst=6

base idea is in all path

- min path
- max path
- ceil path
- floor path

kth largest



paths →

0 1 2 3 4 5 6 @ 38 min cost

0 1 2 3 4 6 @ 40 floor of 48

0 3 4 5 6 @ 48 ceil of path

0 3 4 6 @ 50 max cost

$\boxed{\text{floor, } 48}$

ceil of 48

kth largest

Get Connected Components

Sunday, 4 July 2021 10:34 AM

[Important concept]

connected component.

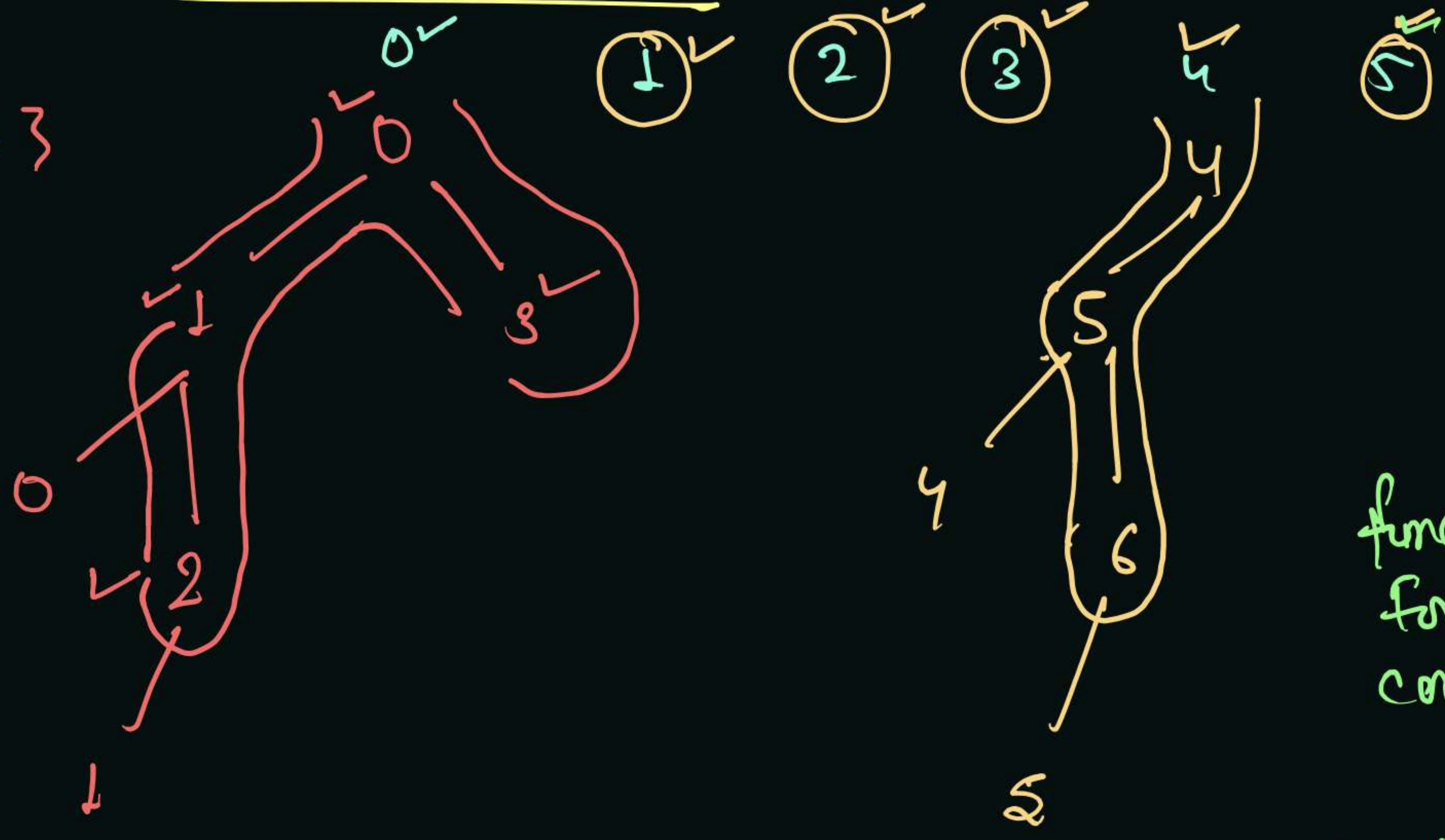
unmark X

{{0,1,2,3}, {4,5,6}, {7}}

{0, 1, 2, 3}

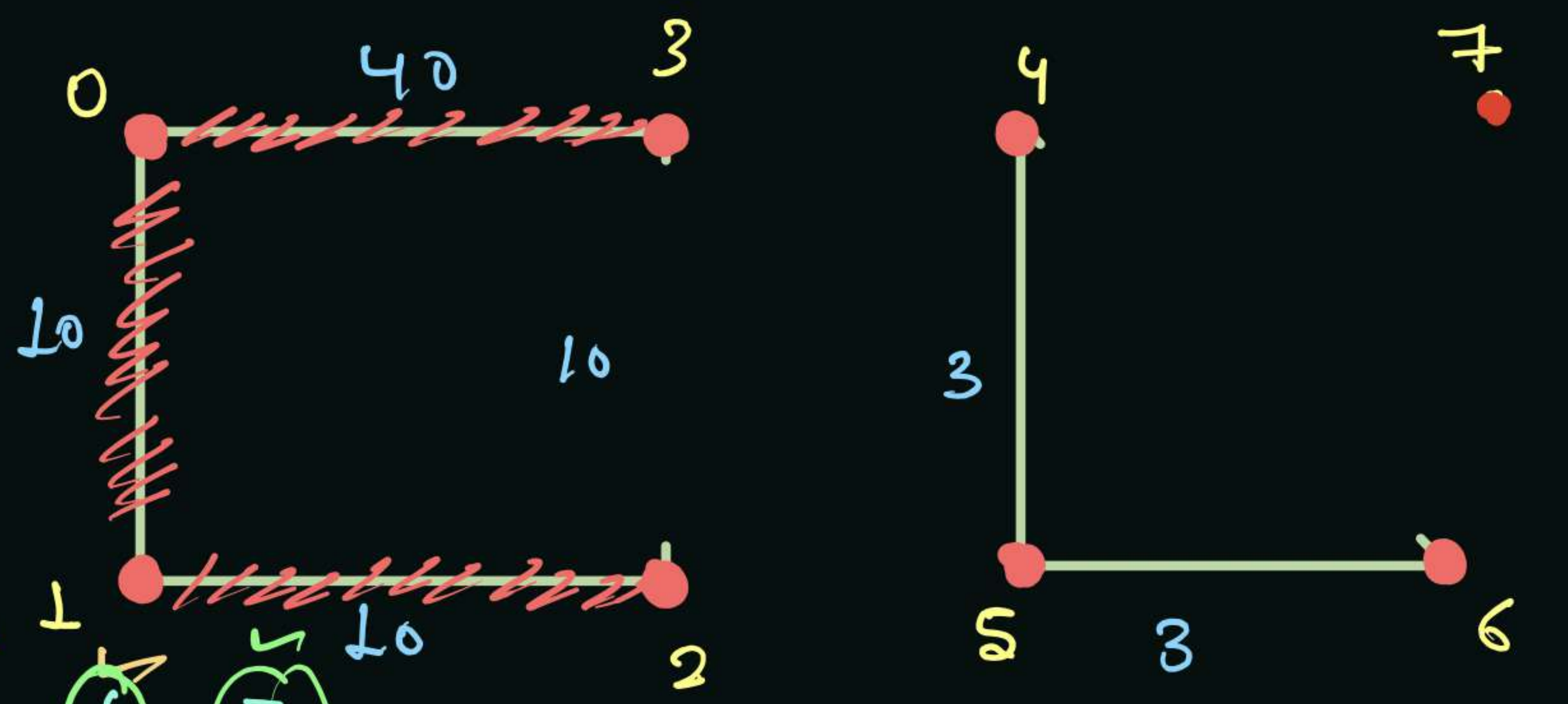
{4, 5, 6}

{7}



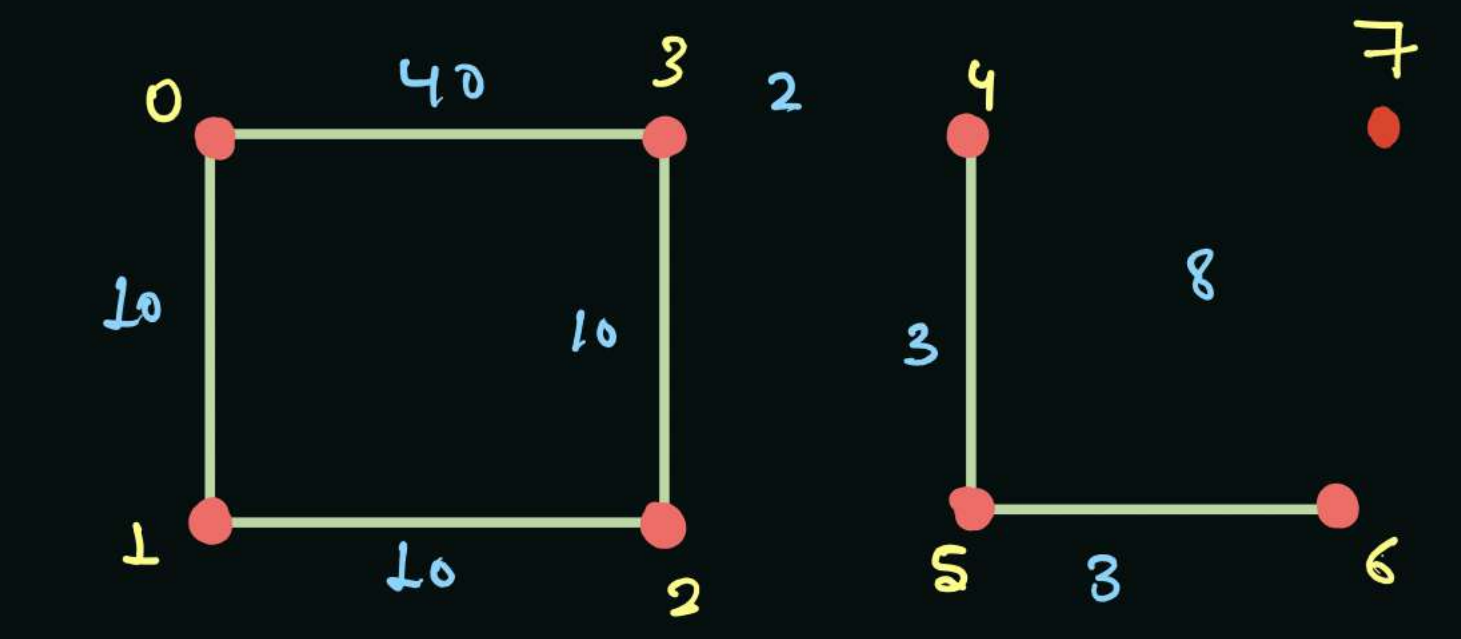
{7}

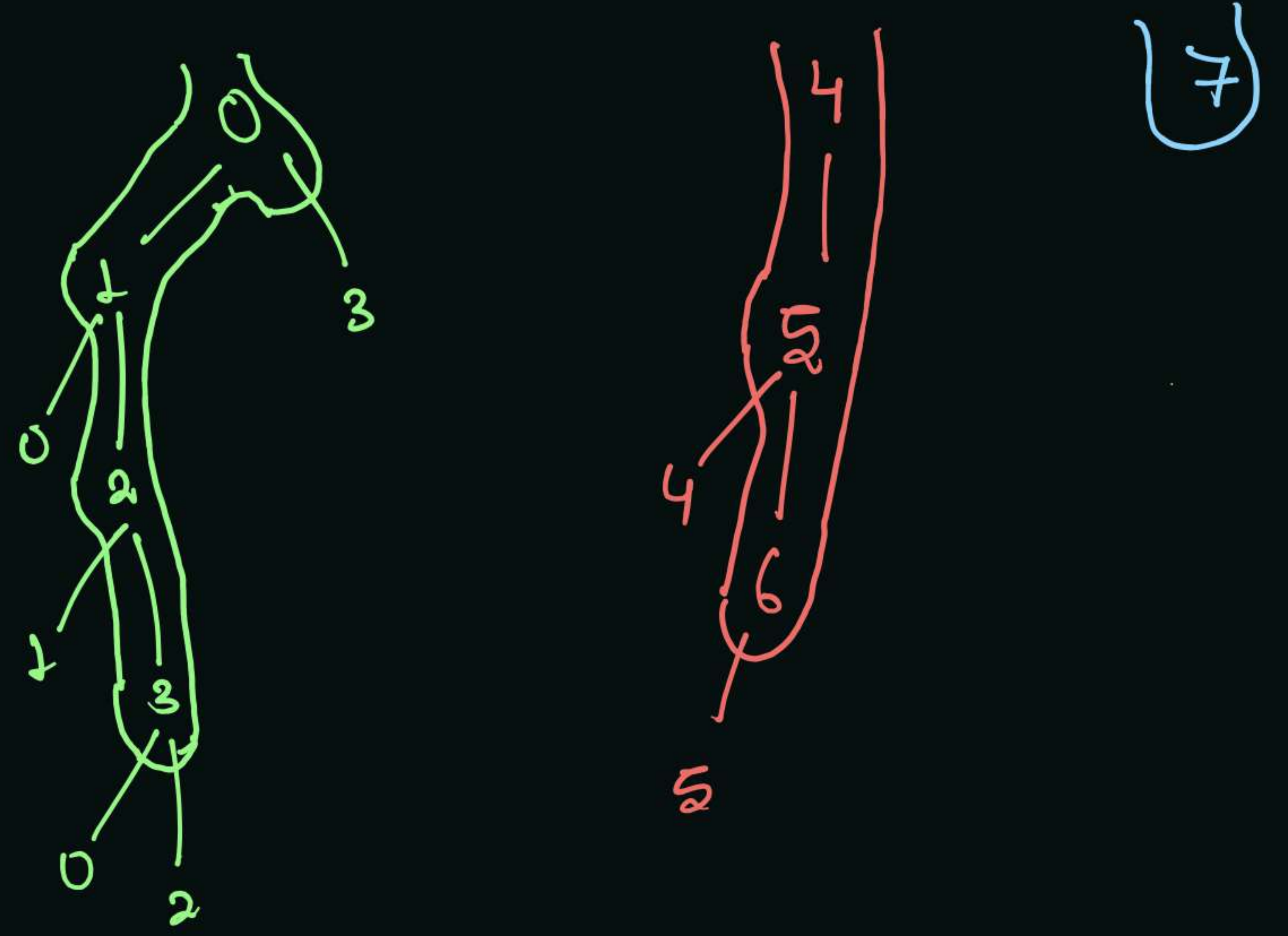
{7}

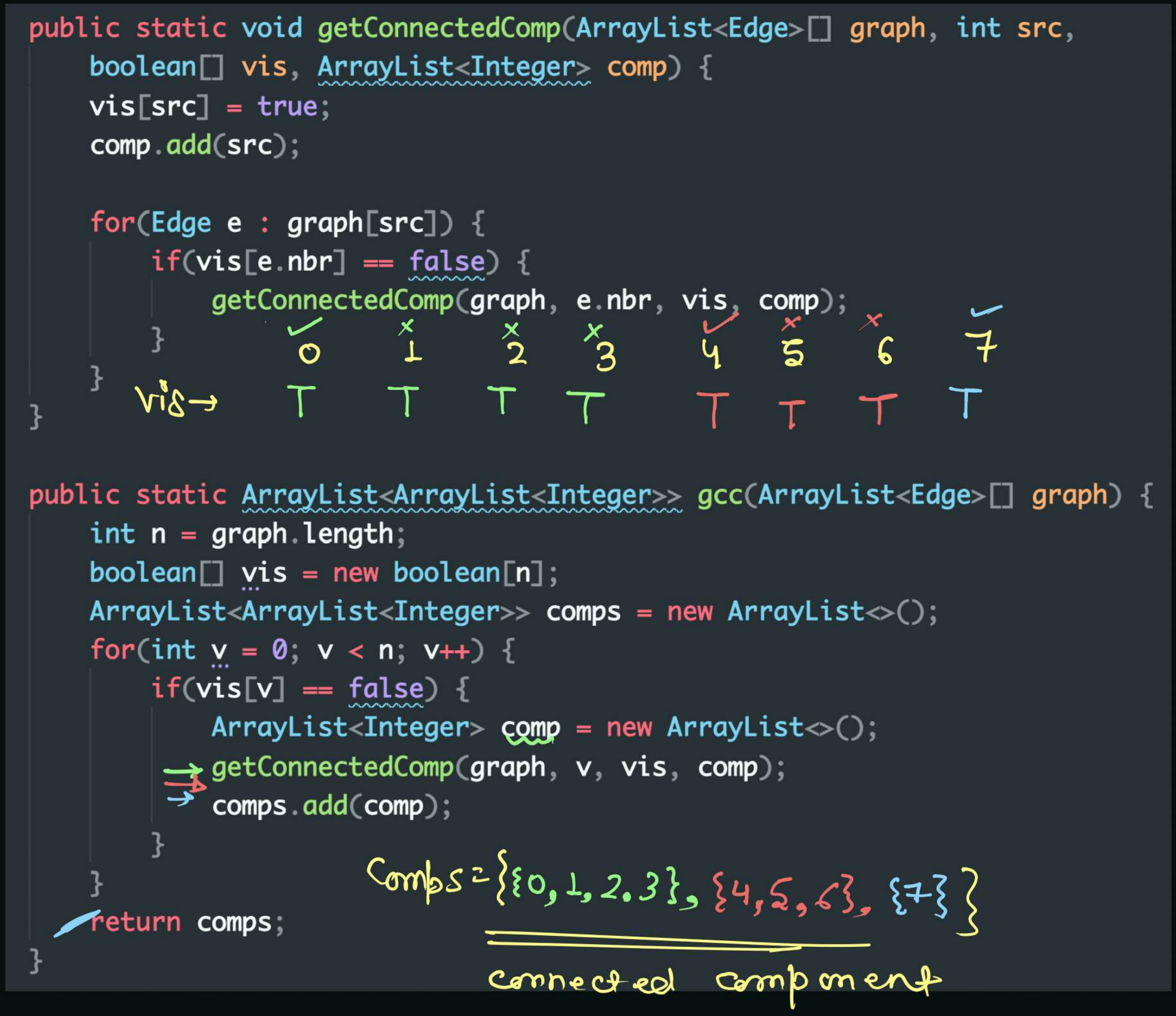


function for comp. [[dfs.

function for comps. [[Make call for every unvisited vertex and get component.







```java
public static void getConnectedComp(ArrayList<Edge>[] graph, int src,
    boolean[] vis, ArrayList<Integer> comp) {
    vis[src] = true;
    comp.add(src);

    for(Edge e : graph[src]) {
        if(vis[e.nbr] == false) {
            getConnectedComp(graph, e.nbr, vis, comp);
        }
    }
}

public static ArrayList<ArrayList<Integer>> gcc(ArrayList<Edge>[] graph) {
    int n = graph.length;
    boolean[] vis = new boolean[n];
    ArrayList<ArrayList<Integer>> comps = new ArrayList<>();
    for(int v = 0; v < n; v++) {
        if(vis[v] == false) {
            ArrayList<Integer> comp = new ArrayList<>();
            getConnectedComp(graph, v, vis, comp);
            comps.add(comp);
        }
    }
    return comps;
}
```

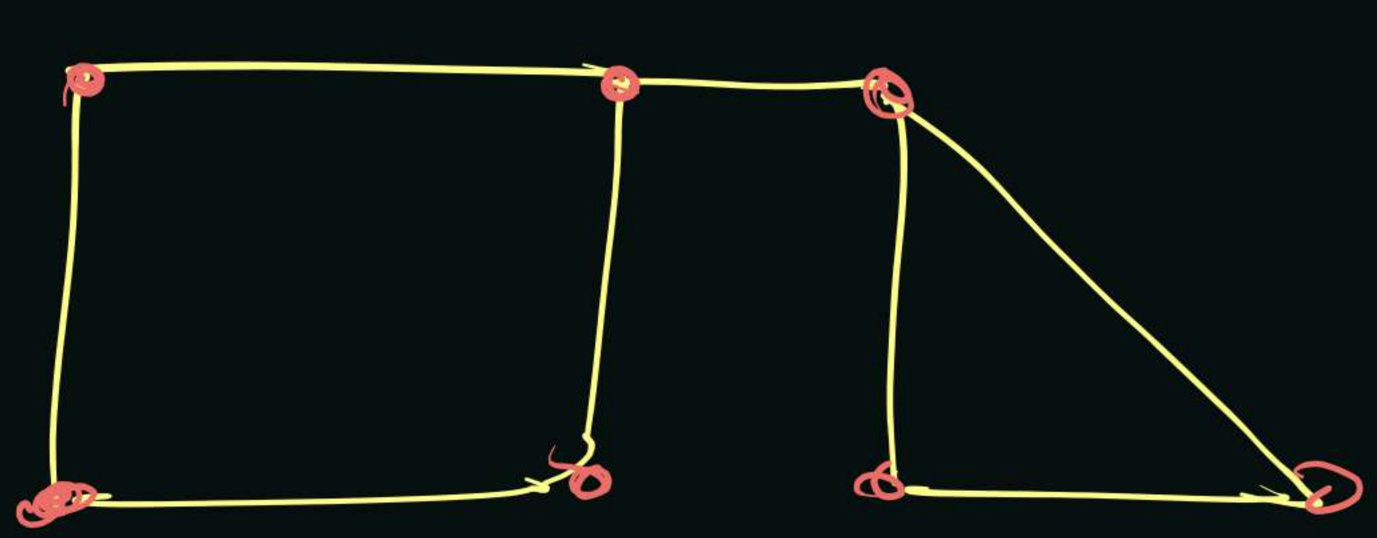
	✓	×	×	×	✓	×	×	✓
	0	1	2	3	4	5	6	7
vis→	T	T	T	T	T	T	T	T

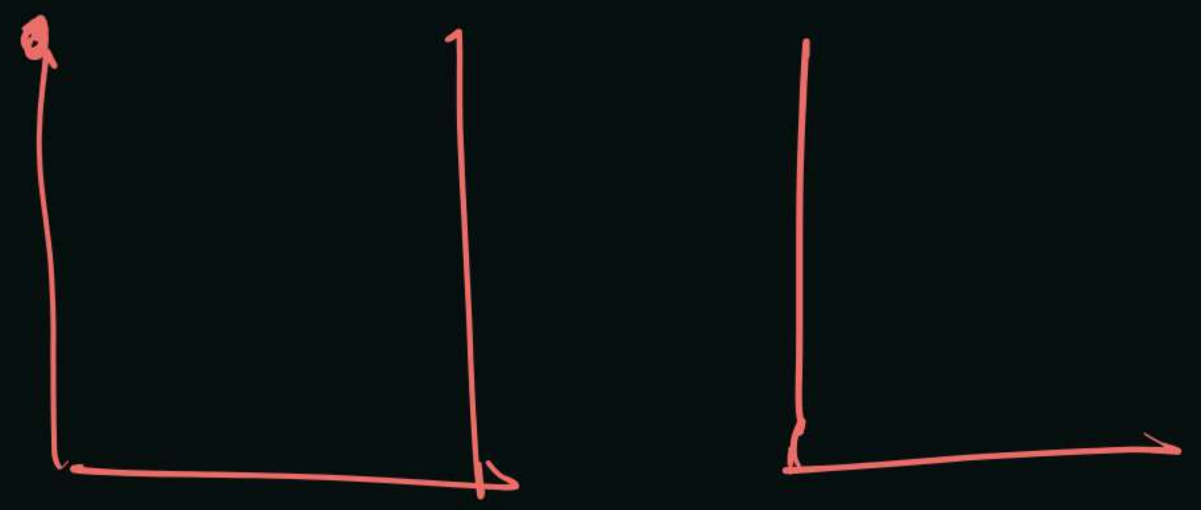
Comps = {{0, 1, 2, 3}, {4, 5, 6}, {7}}

connected component

Is Graph Connected

Sunday, 4 July 2021 2:24 PM





no. of components > 1 } graph is disconnected

== 1 } graph is connected.