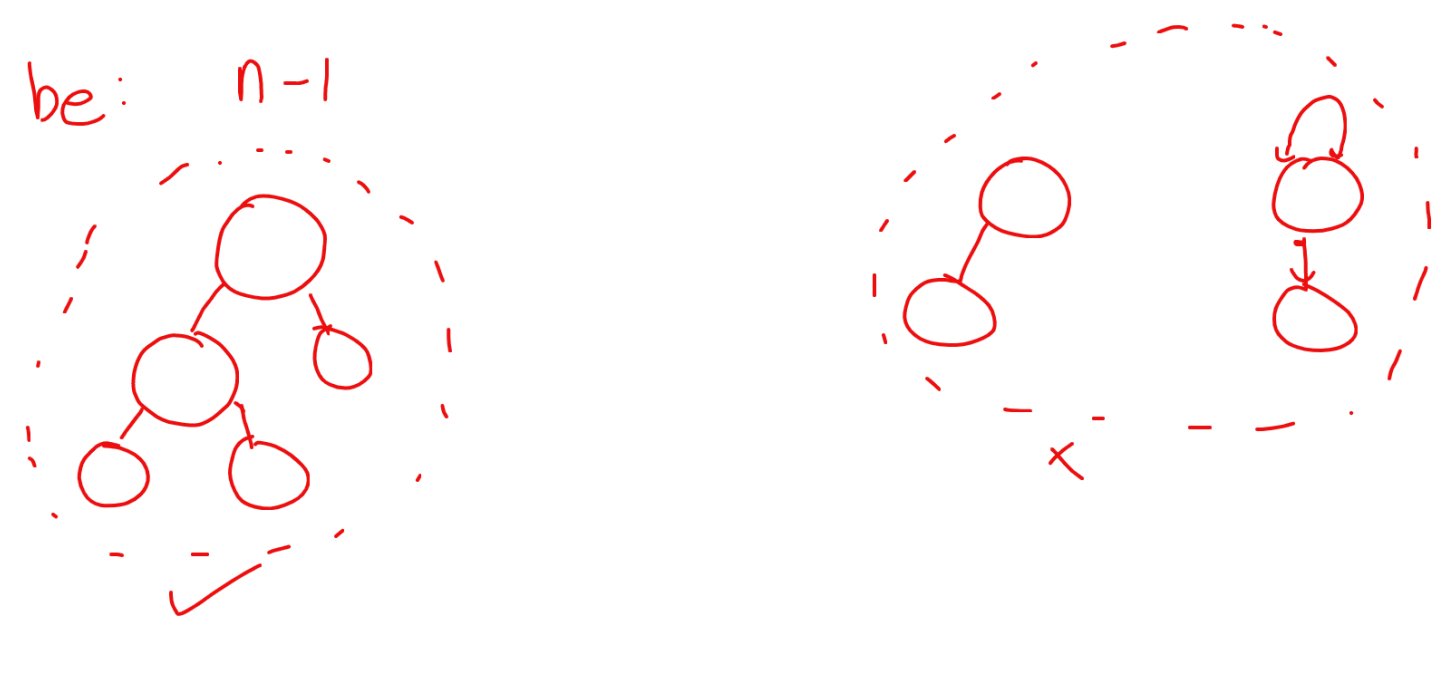
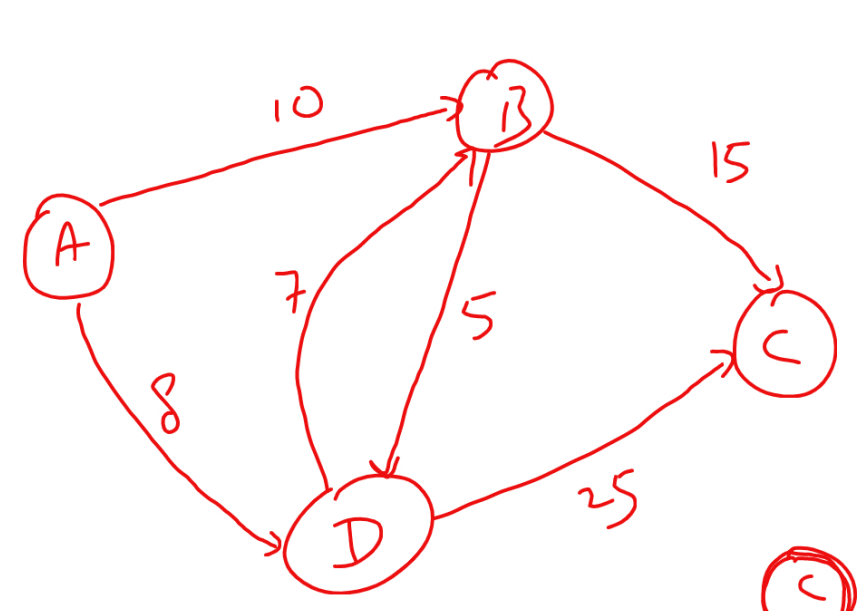
1) k * log(k)
2) (n-k) * log(k)
+
n * log(k)

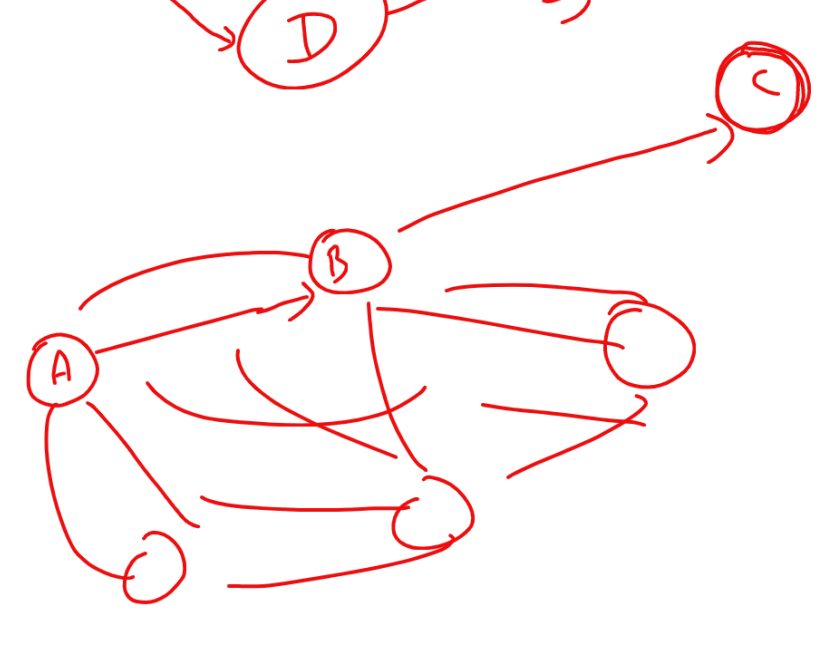
Tree is a graph where all the nodes are connected and also, it does not have a loop.

=> For a tree of size 'n', then the no. of edges will be: n-1




A
B
C
D
10
15
7
5
8
25

A→B→C : 25
A→D→C : 33
A→D→B→C : 30



---

1) Directed graph: Edge(U→V) != Edge(V→U)
2) Undirected graph: Edge(U→V) == Edge(V→U)

Categorization based upon the direction of edges

1) weighted graph: Edges have wts. Eg Google maps
2) Unweighted graph: Edges do not have wts Eg. social network.

Categorization based upon the wts. of edges

---

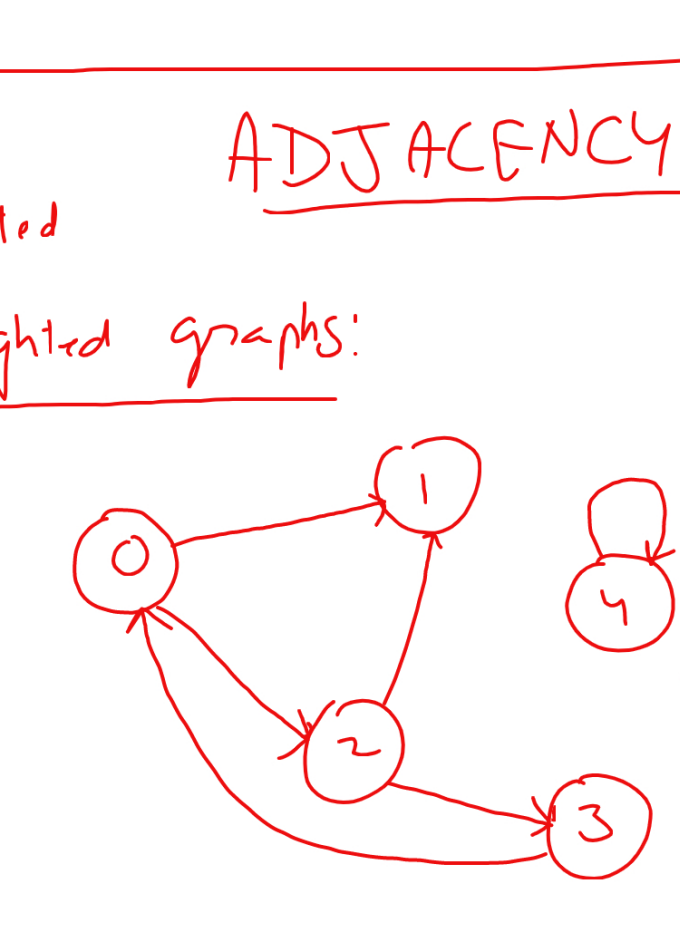
1) Directed + weighted
2) Directed + unweighted
3) Undirected + weighted
4) Undirected + unweighted

KINDS OF GRAPHS

ADJACENCY MATRIX

space = O(V²)

For directed + unweighted graphs:



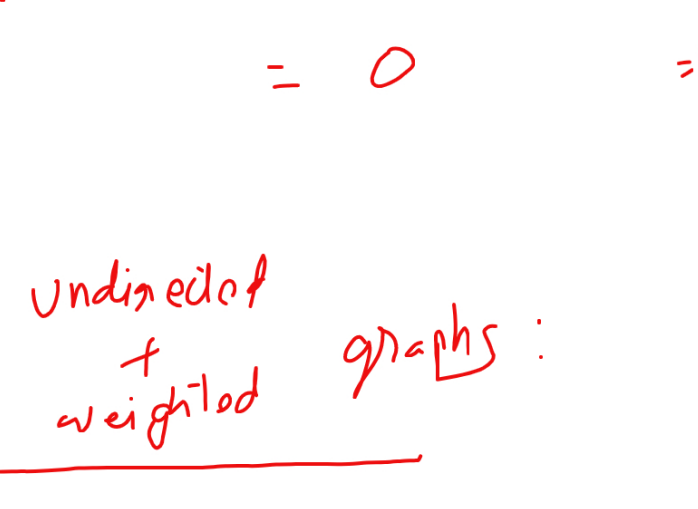
adj[][]

	0	1	2	3	4
0	0	1	1	0	0
1	0	0	0	0	0
2	0	1	0	1	0
3	1	0	0	0	0
4	0	0	0	0	1

V = 5

adj[i][j] = 1 => Edges b/w i to j (i→j)
= 0 => No edge from i to j

For undirected + weighted graphs:


0
1
2
3
4
2
2
5
1
10

	0	1	2	3	4
0	0	7	5	0	0
1	7	0	0	1	2
2	5	0	0	0	0
3	0	1	0	0	10
4	0	2	0	10	0

V = 5

adj[i][j] = w => a directed edge from (i→j) with wt = w.
= 0 => no edge from (i→j)

---

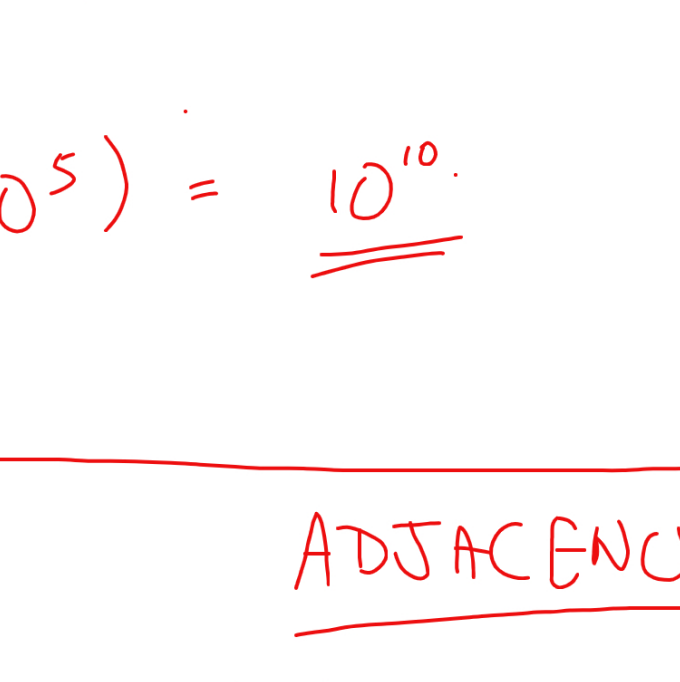
no. of vertices = 10^5 (0, 1, ..., 99999)
no. of edges = 100

$(10^5 \times 10^5) = 10^{10}$

ADJACENCY LIST

Space = O(V+E)

Undirected + Unweighted

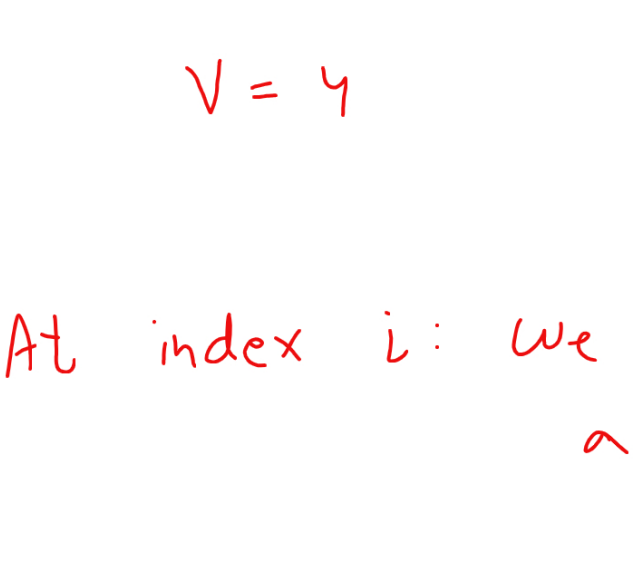


0: 2, 3
1: 2, 3
2: 0, 1, 3
3: 0, 1, 2

V = 4

At index i: we will have a list of values (j) denoting a directed edge (i→j)

Directed + weighted:



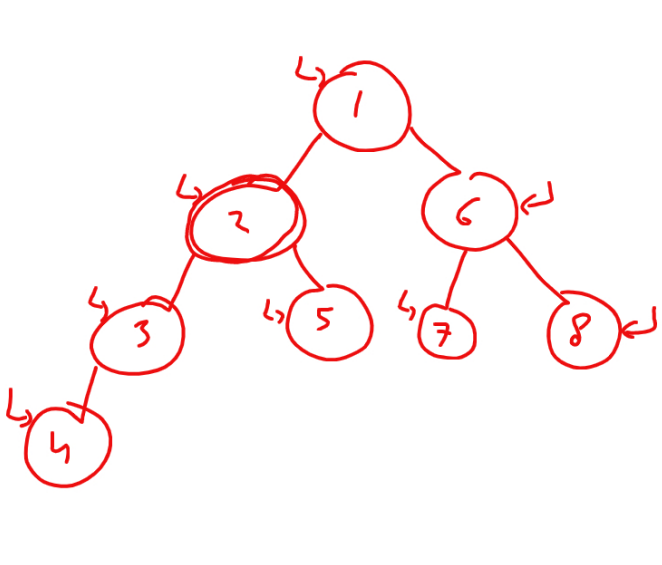
0: (2, 5)
1: (0, 3)
2: (0, 1), (1, 6)
3: (0, 1) (2, 2)

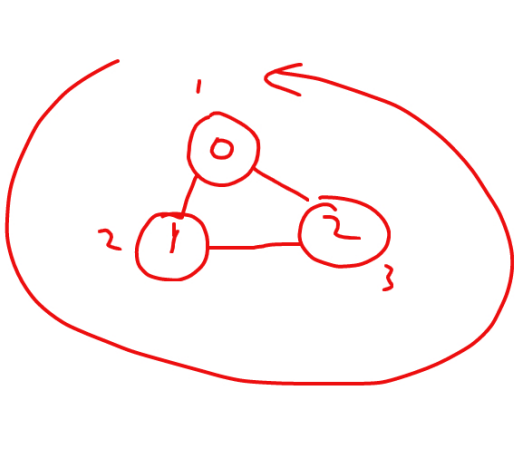
V = 4

At index i: we will have a list of pairs {j, w} denoting a directed edge (i→j) with weight = w

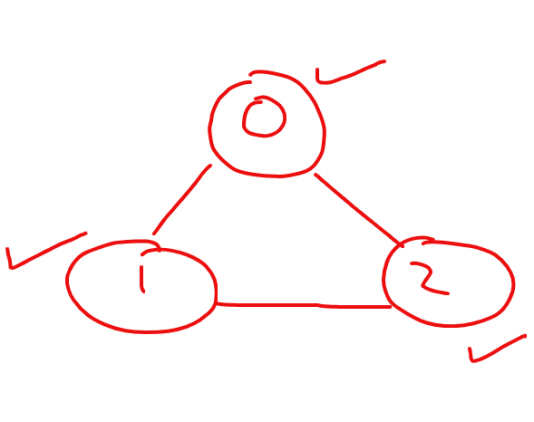
---

DFS

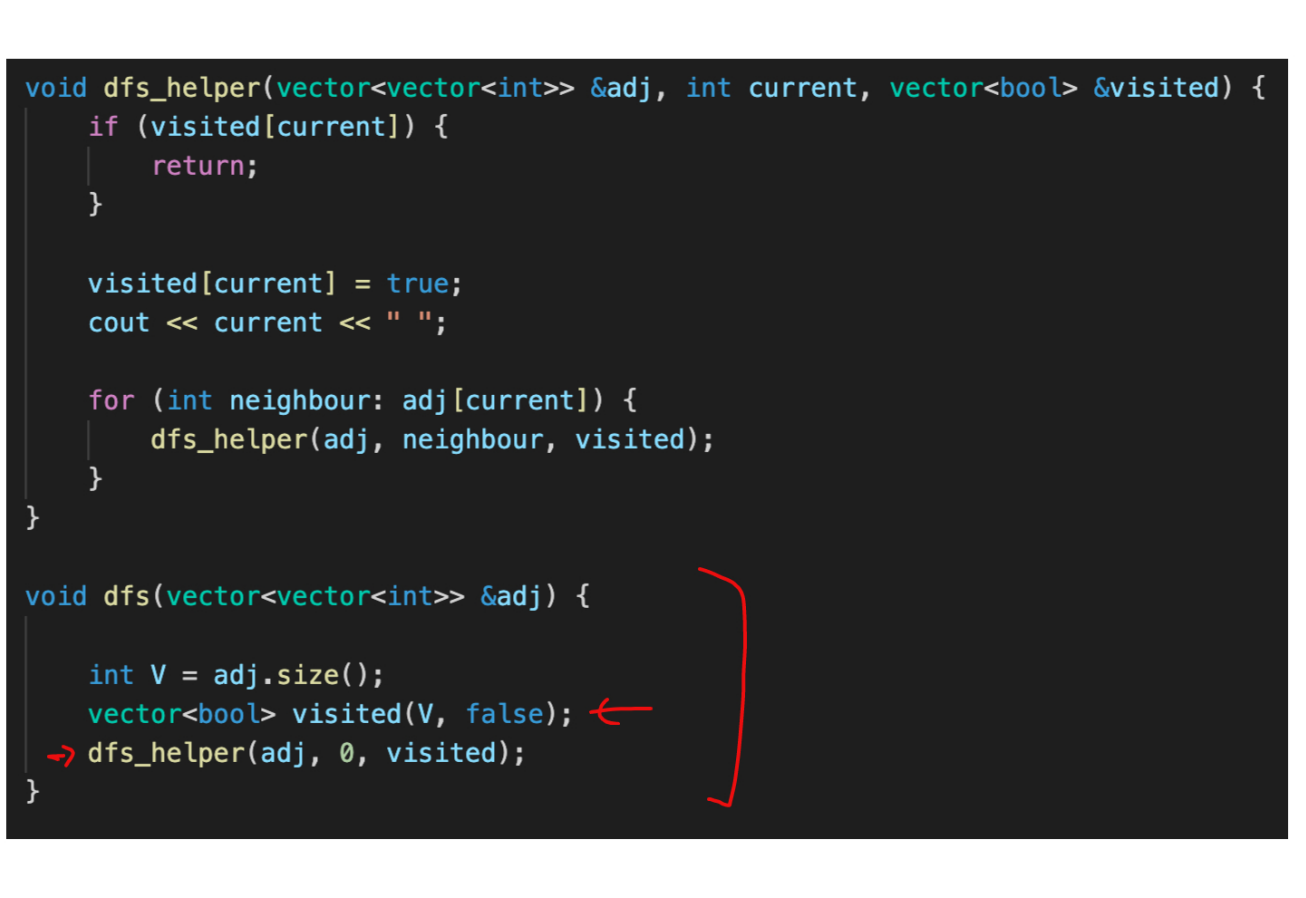




0: 1, 2
1: 0, 2
2: 0, 1



0, 1, 2



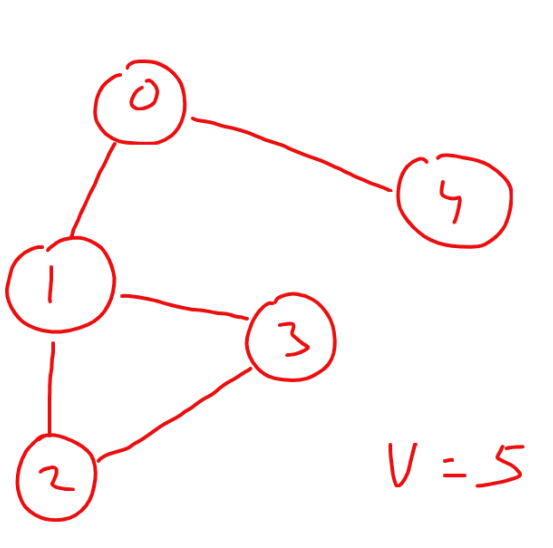
```
void dfs_helper(vector<vector<int>> &adj, int current, vector<bool> &visited) {
    if (visited[current]) {
        return;
    }

    visited[current] = true;
    cout << current << " ";

    for (int neighbour: adj[current]) {
        dfs_helper(adj, neighbour, visited);
    }
}

void dfs(vector<vector<int>> &adj) {

    int V = adj.size();
    vector<bool> visited(V, false);
    dfs_helper(adj, 0, visited);
}
```

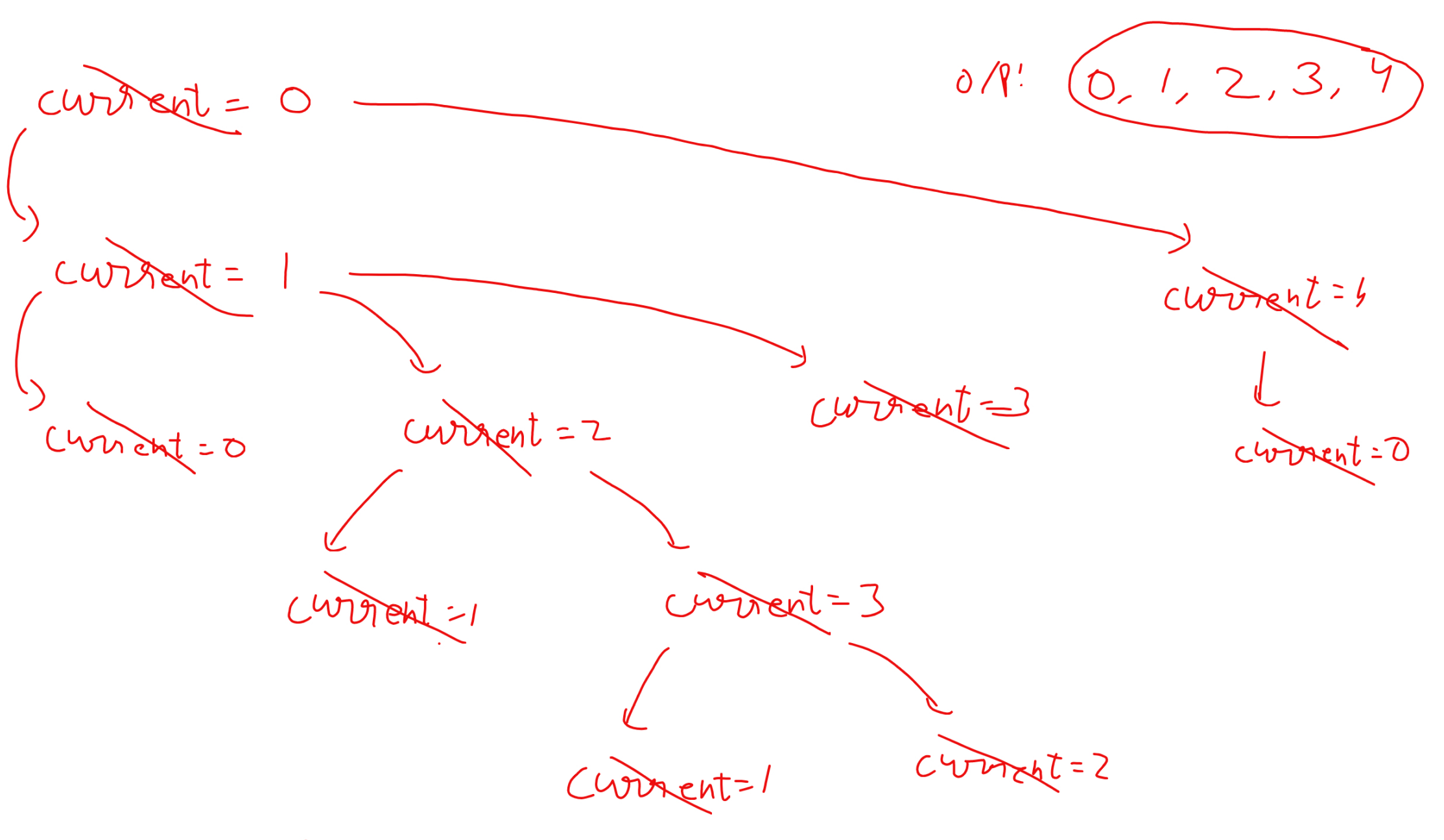


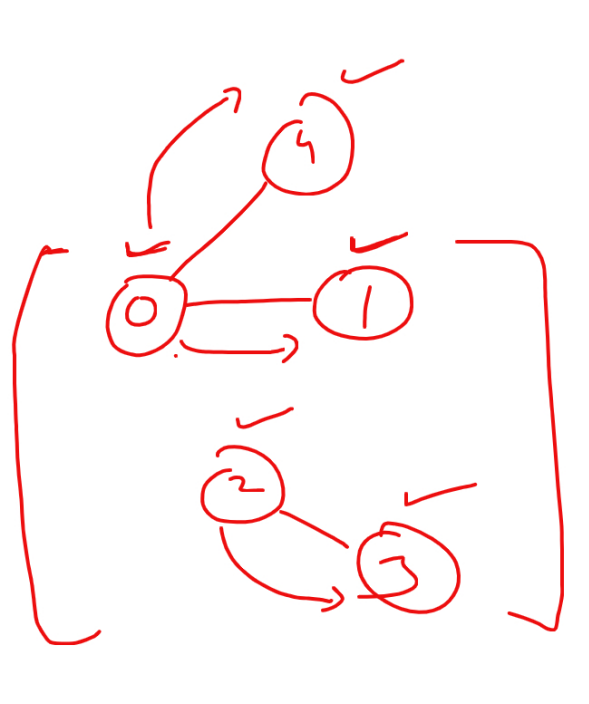
V = 5

0: 1, 4
1: 0, 2, 3
2: 1, 3
3: 1, 2
4: 0

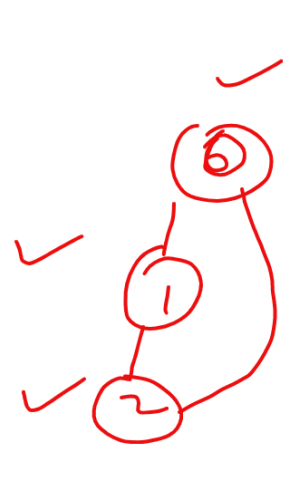
	0	1	2	3	4
vis	~~F~~ T	~~F~~ T	~~F~~ T	~~F~~ T	~~F~~ T

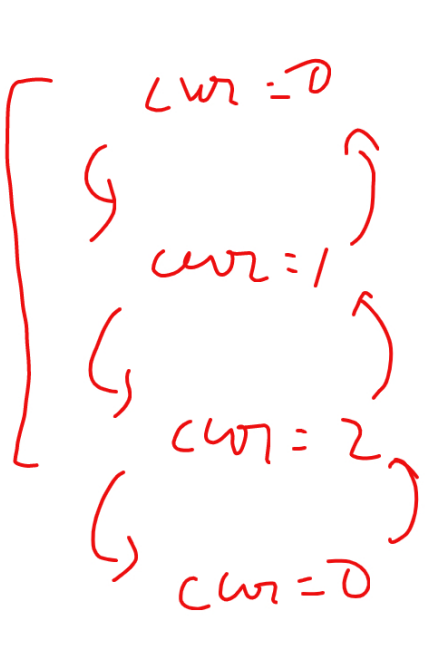
O/P: 0, 1, 2, 3, 4


~~current~~ = 0
~~current~~ = 1
~~current~~ = 0
~~current~~ = 2
~~current~~ = 1
~~current~~ = 3
~~current~~ = 1
~~current~~ = 2
~~current~~ = 3
~~current~~ = 4
~~current~~ = 0

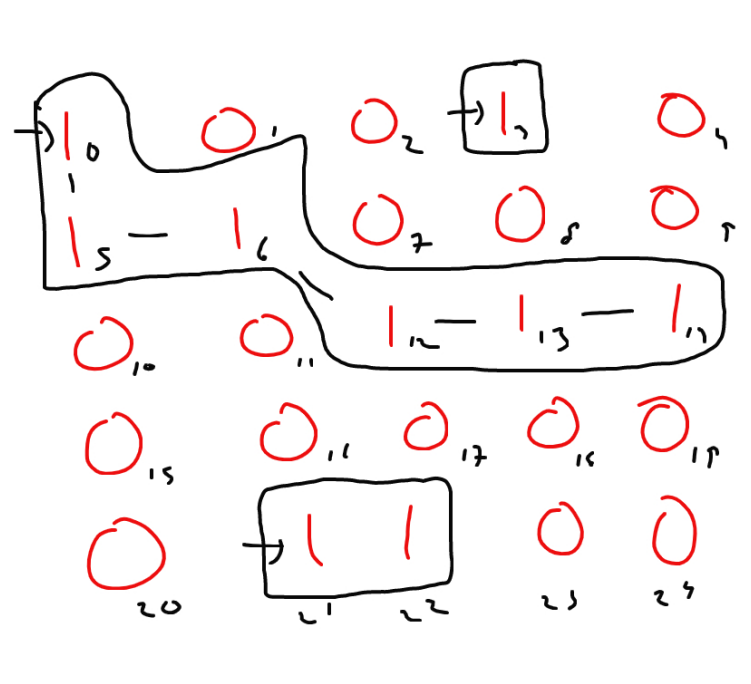


0 : 1, 4
1 : 0
2 : 3
3 : 2
4 : 0

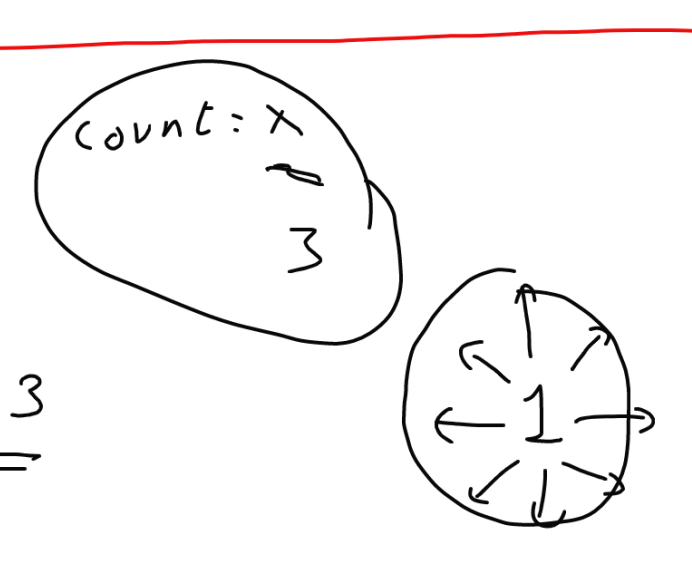


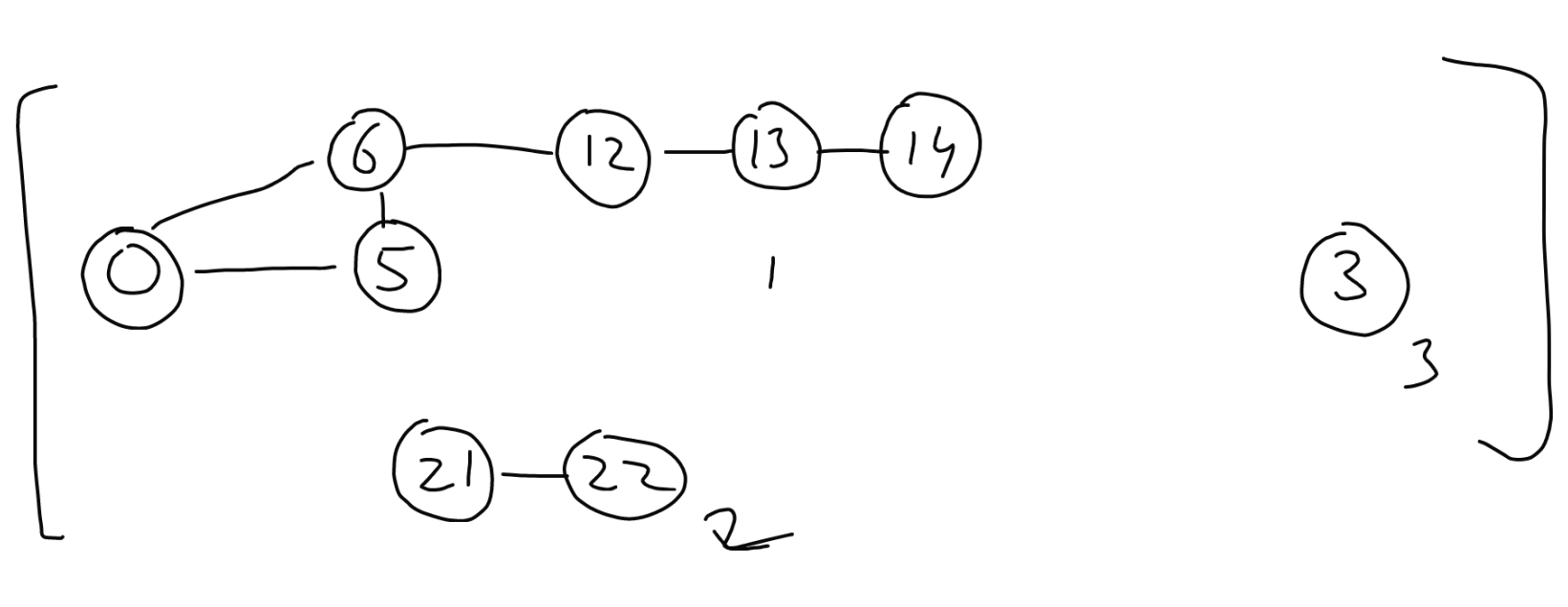

cur = 0
cur = 1
cur = 2
cur = 0

O(V+E)



O/P = 3




0
6
12
13
14
5
21
22
1
2
3
3

i-1, j-1	i-1, j	i-1, j+1
i, j-1	i, j	i, j+1
i+1, j-1	i+1, j	i+1, j+1

⇒

-1, -1	-1, 0	-1, 1
0, -1	0, 0	0, 1
1, -1	1, 0	1, 1