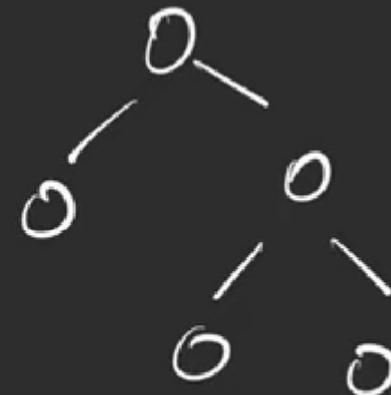
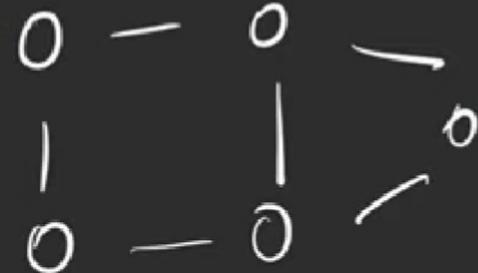


⇒ G-4. What are Connected Components ?

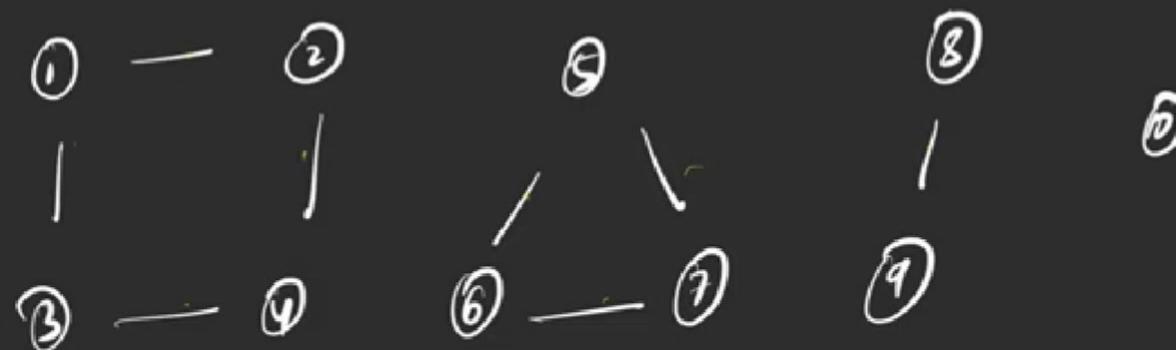
Connected Components



⇒ G-4. What are Connected Components ?

$$N = 10 \quad M = 8$$

1.90



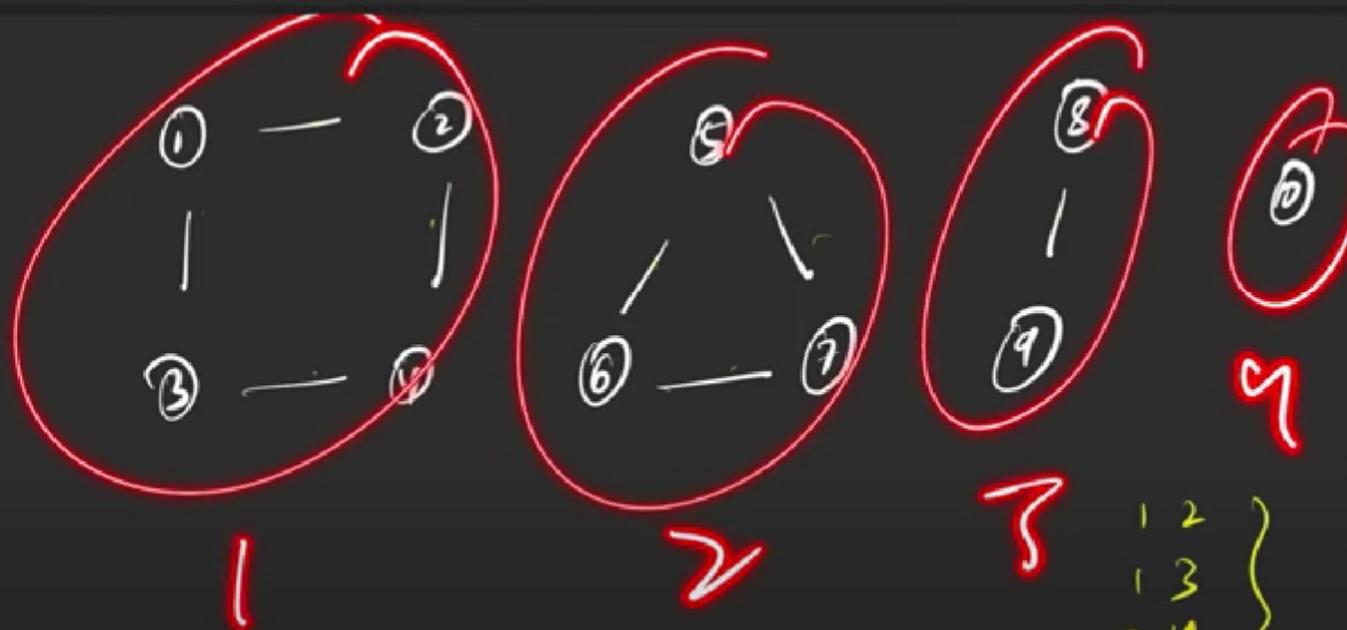
1 2
1 3
2 4
3 4



⇒ G-4. What are Connected Components ?

$$N=10 \quad M=8$$

1.90

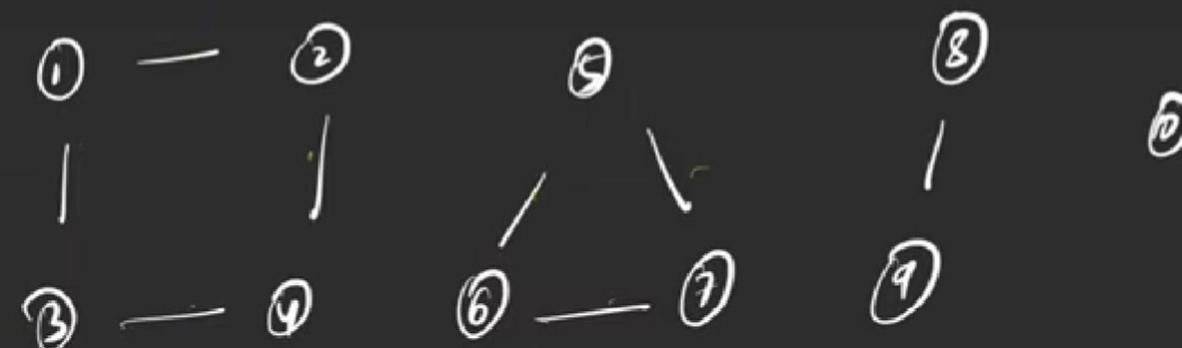


1 2
1 3
2 4
3 4
5 6



► G-4. What are Connected Components ?

1.90



1 components

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



⇒ G-4. What are Connected Components ?

1.90
 $\text{vis} = \boxed{0 | 1 | 1 | } \dots | 10 \quad N=11$

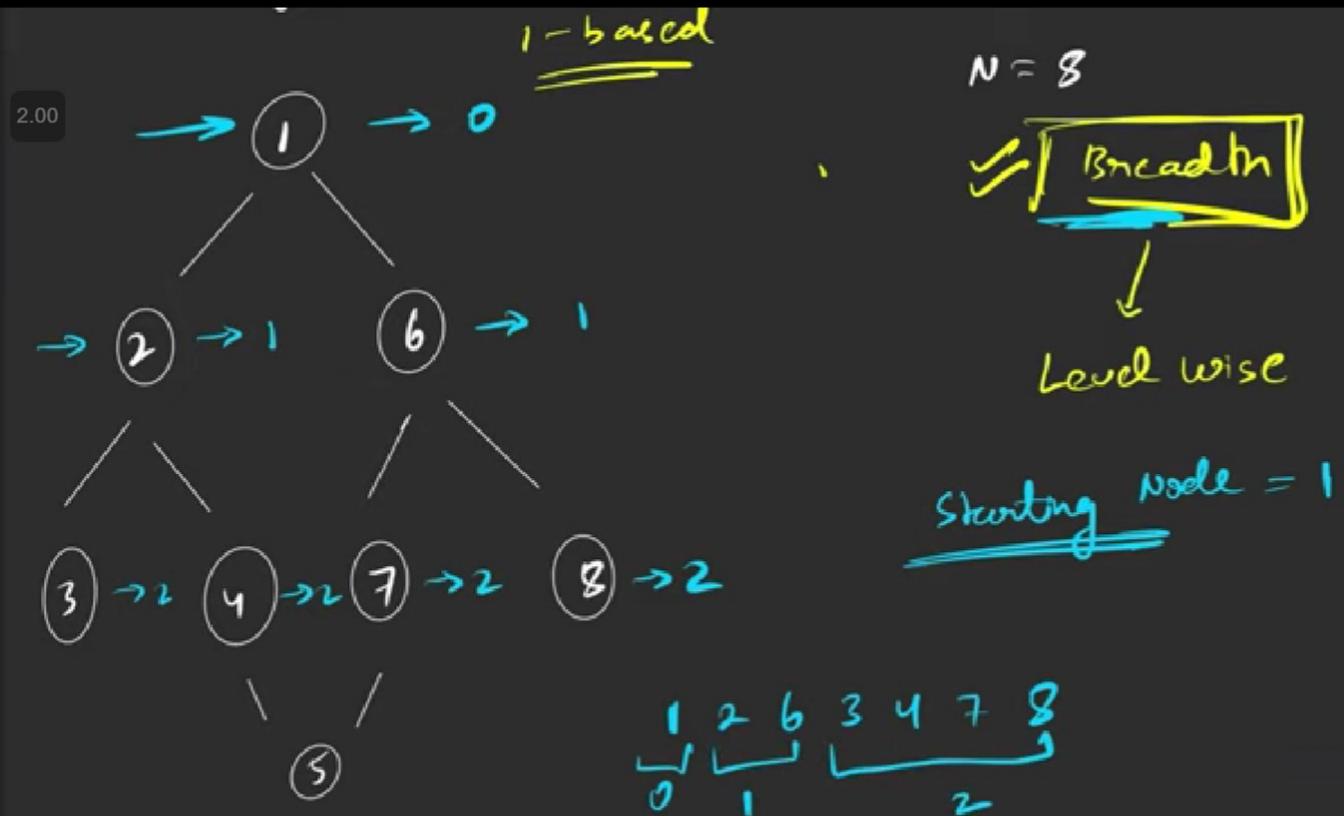
$\text{fun}(i = 1 \rightarrow 10)$
if ($\text{!vis}[i]$)
 $\text{mark}(i);$
 $N=10 \quad M=8$

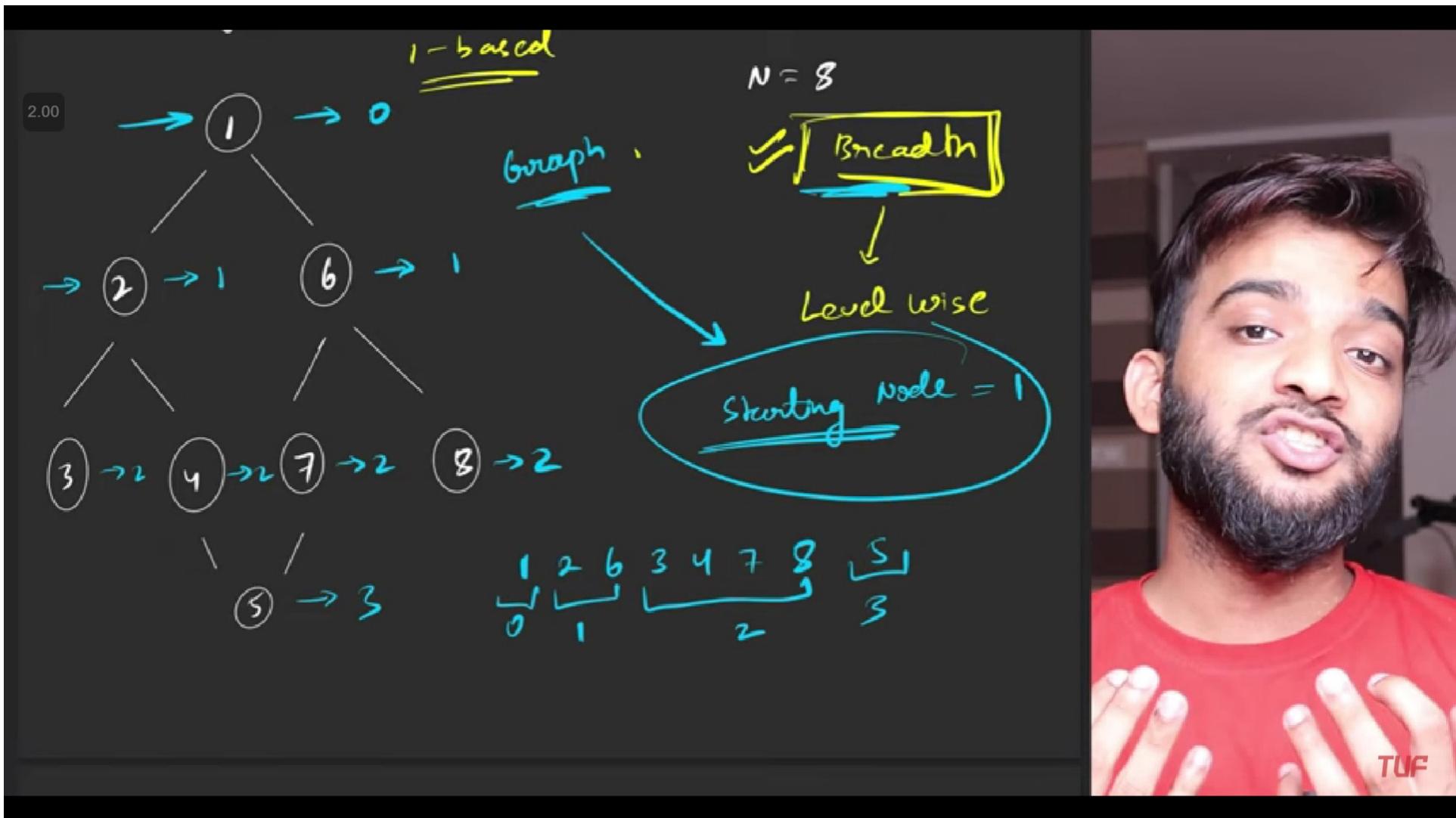
}

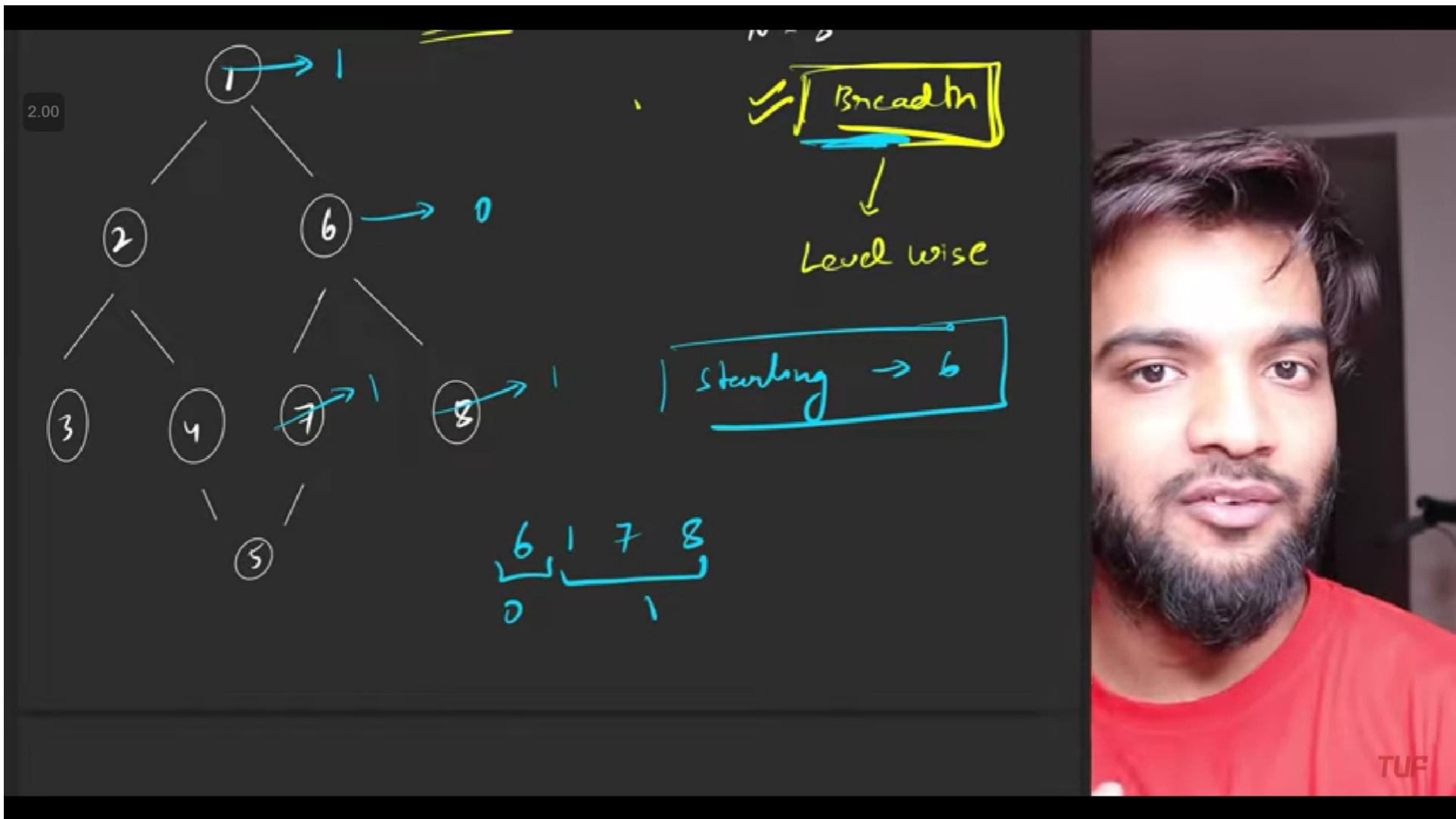


4:10 / 7:06



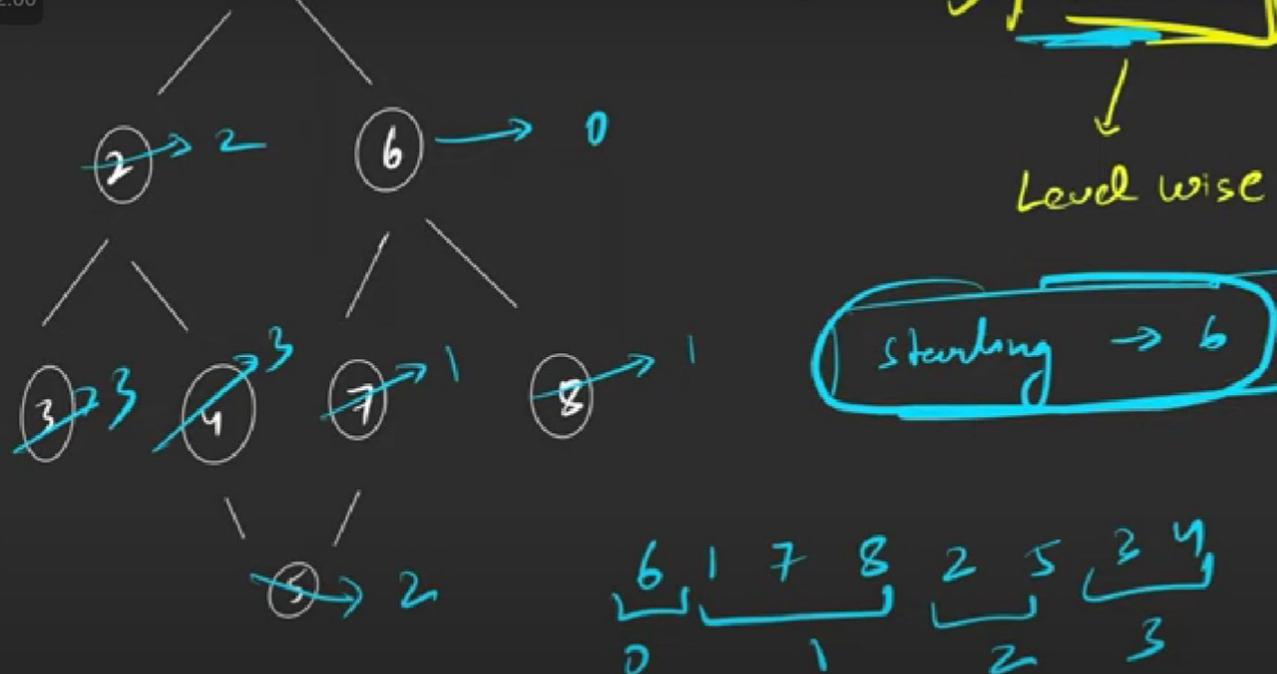






G-5. Breadth-First Search (BFS) | C++ and Java | Traversal Technique in Graphs

2.00



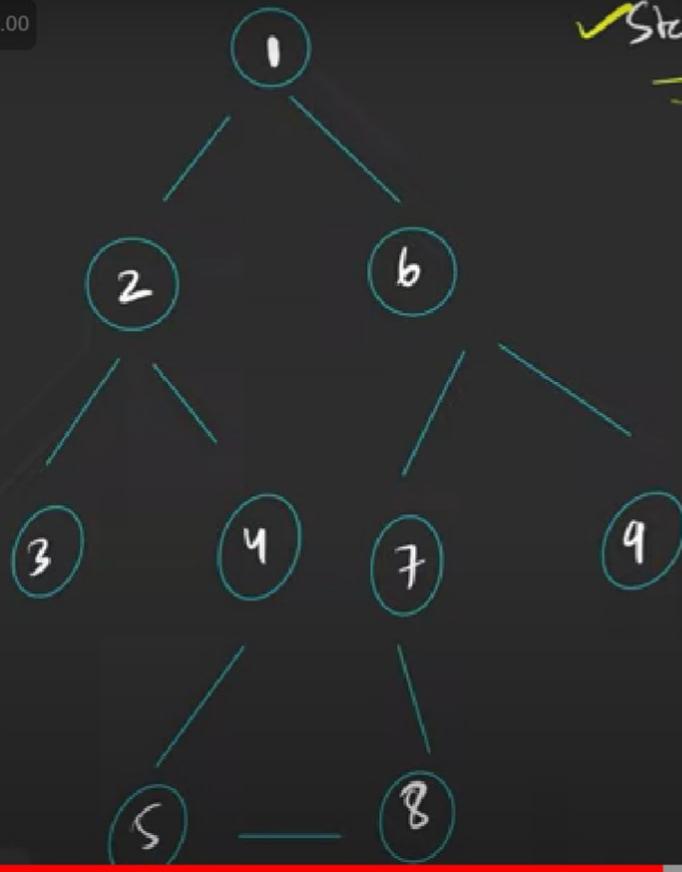
5:38 / 19:38



G-5. Breadth-First Search (BFS) | C++ and Java | Traversal Technique in Graphs

2.00

✓ Starting node = 1 Initial



Queue
↓
FIFO



7:06 / 19:38



► G-5. Breadth-First Search (BFS) | C++ and Java | Traversal Technique in Graphs

2.00

{adjacency
list}

0 → { }
1 → {2, 6}
2 → {1, 3, 4}
3 → {2}
4 → {2, 5}
5 → {4, 8}
6 → {1, 7, 9}
7 → {6, 8}
8 → {5, 7}

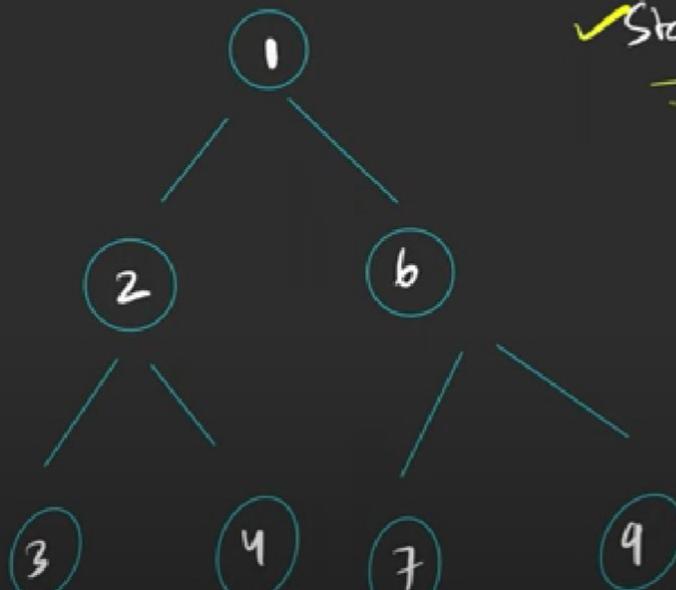


G-5. Breadth-First Search (BFS) | C++ and Java | Traversal Technique in Graphs

2.00

Who are your neighbors

✓ Starting node = 1 Initial



Queue



BFS



10:15 / 19:38



BFS

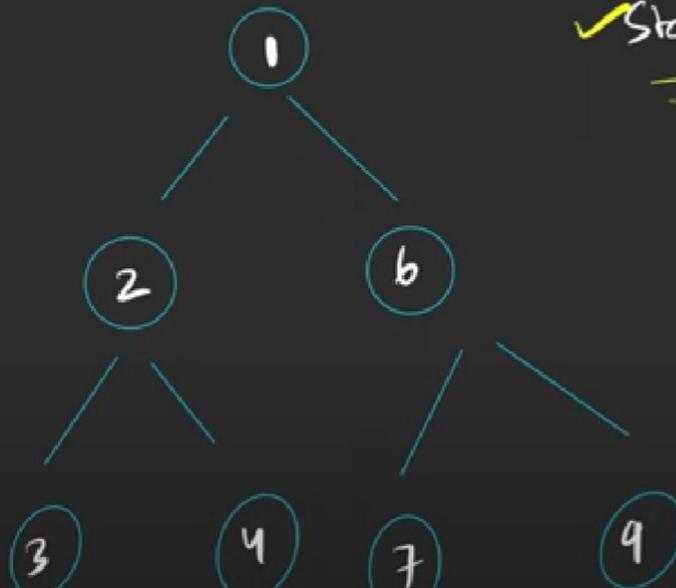


G-5. Breadth-First Search (BFS) | C++ and Java | Traversal Technique in Graphs

2.00

Who are your neighbors

✓ Starting node = 1 Initial



Queue



BFS

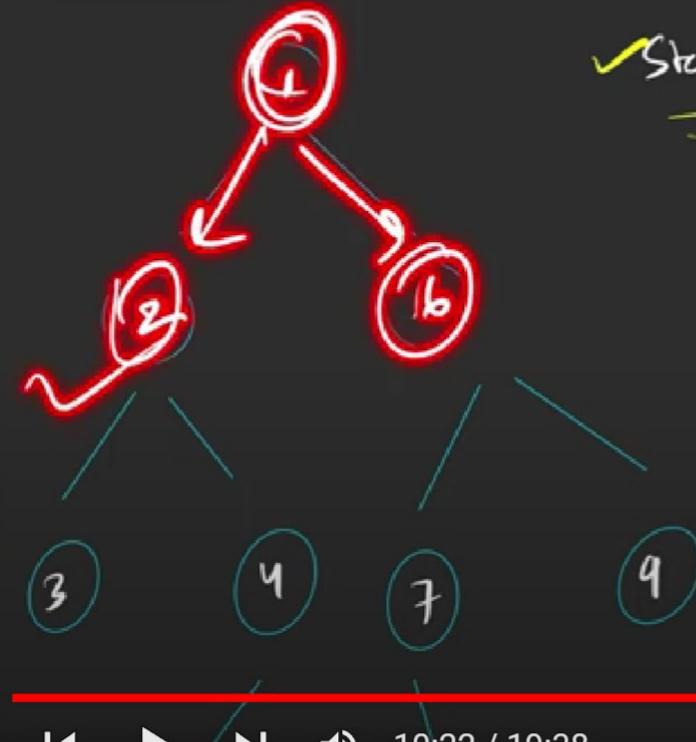


10:15 / 19:38



G-5. Breadth-First Search (BFS) | C++ and Java | Traversal Technique in Graphs

2.00



✓ Starting node = 1 Initial



Queue



G-5. Breadth-First Search (BFS) | C++ and Java | Traversal Technique in Graphs

230

Editorial Submissions Comments C++ (g++ 5.4) ▾

```
class Solution {
    // Function to return Breadth First Traversal of given graph
    public ArrayList<Integer> bfsOfGraph(int V,
    ArrayList<ArrayList<Integer>> adj) {
        ArrayList < Integer > bfs = new ArrayList < > ();
        boolean vis[] = new boolean[V];
        Queue < Integer > q = new LinkedList < > ();

        q.add(0);
        vis[0] = true;

        while (!q.isEmpty()) {
            Integer node = q.poll();
            bfs.add(node);

            // Get all adjacent vertices of the dequeued vertex
            // If a adjacent has not been visited, then
            // mark it as visited and enqueue it
            for (Integer it: adj.get(node)) {
                if (vis[it] == false) {
                    vis[it] = true;
                    q.add(it);
                }
            }
        }

        return bfs;
    }

    // Driver Code Ends
    class Solution {
        public:
            // Function to return Breadth First Traversal of given graph.
            vector<int> bfsOfGraph(int V, vector<int> adj[]) {
                int vis[n] = {0};
                vis[0] = 1;
                queue<int> q;
                q.push(0);
                vector<int> bfs;
                while(!q.empty()) {
                    int node = q.front();
                    q.pop();
                    bfs.push_back(node);

                    for(auto it : adj[node]) {
                        if(!vis[it]) {
                            vis[it] = 1;
                            q.push(it);
                        }
                    }
                }
                return bfs;
            }
    }
}
```

Output Window

Compilation Results Custom Input

Request Queued.

Evaluating



G-5 Breadth-First Search (BFS) | C++ and Java | Traversal Technique in Graphs

C++ (g++ 5.4) ▾

```
1 // } Driver Code Ends
2 class Solution {
3 public:
4     // Function to return Breadth First Traversal of given
5     // graph.
6     ArrayList<Integer> bfsOfGraph(int V,
7         ArrayList<ArrayList<Integer>> adj) {
8
9         ArrayList < Integer > bfs = new ArrayList < > ();
10        boolean vis[] = new boolean[V];
11        Queue < Integer > q = new LinkedList < > ();
12
13        q.add(0);
14        vis[0] = true;
15
16        while (!q.isEmpty()) {
17            Integer node = q.poll();
18            bfs.add(node);
19
20            // Get all adjacent vertices of the dequeued vertex
21            // If a adjacent has not been visited, then mark it
22            // visited and enqueue it
23            for (Integer it: adj.get(node)) {
24                if (vis[it] == false) {
25                    vis[it] = true;
26                    q.add(it);
27                }
28            }
29        }
30
31        return bfs;
32    }
33 }
```

then 3. After this 2 to 4, thus bfs will be

0 1 ▶ 4. ▶ 🔍 16:58 / 19:38

The image shows a video player interface. On the left, there is a code editor window titled 'C++ (g++ 5.4)' containing C++ code for Breadth-First Search (BFS). The code uses ArrayList and Queue from the Java standard library. The video player has a progress bar at 16:58 / 19:38. On the right, there is a video feed of a man with a beard and short hair, wearing a red t-shirt with 'TUF' printed on it. He is gesturing with his right hand. The video player includes standard controls like play/pause, volume, and a settings gear icon.

2.00

$\tau \rightarrow \cancel{\sigma} \quad \{ \cancel{\sigma} \}$ while (! \emptyset)

given on all degrees

for all node c



2.00

$$O(N) + O(2E)$$

For queue for loop

for undirected graph
each data appears twice

TUF

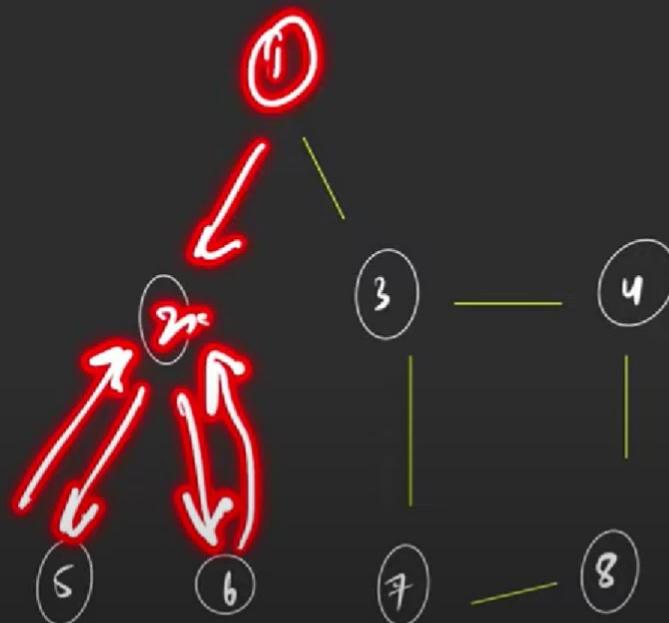
→ G-6. Depth-First Search (DFS) | C++ and Java | Traversal Technique in Graphs

2.00

DFS traversal in a graph

Starting
node = 1

1 2 5 6



1:50 / 20:15



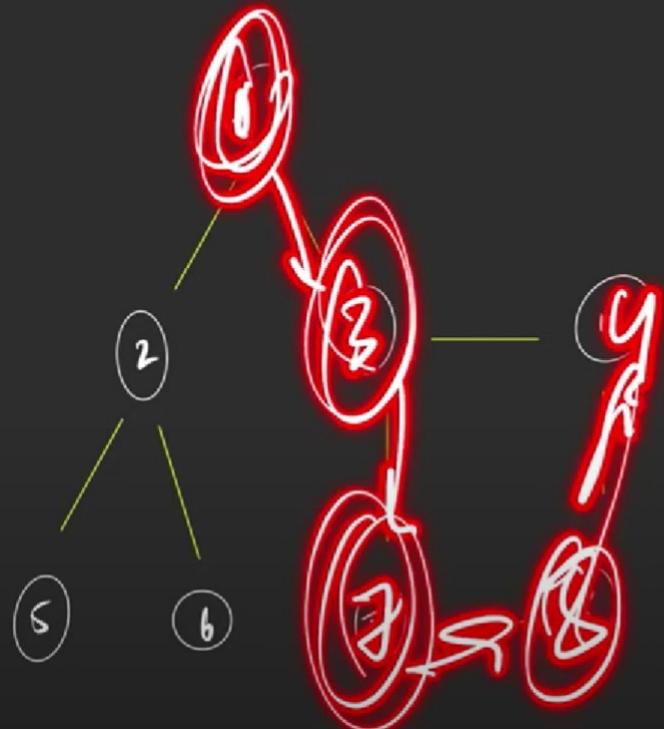
→ G-6. Depth-First Search (DFS) | C++ and Java | Traversal Technique in Graphs

2.00

DFS traversal in a graph

Starting
node = 1

1 2 5 6 3 7 8 4



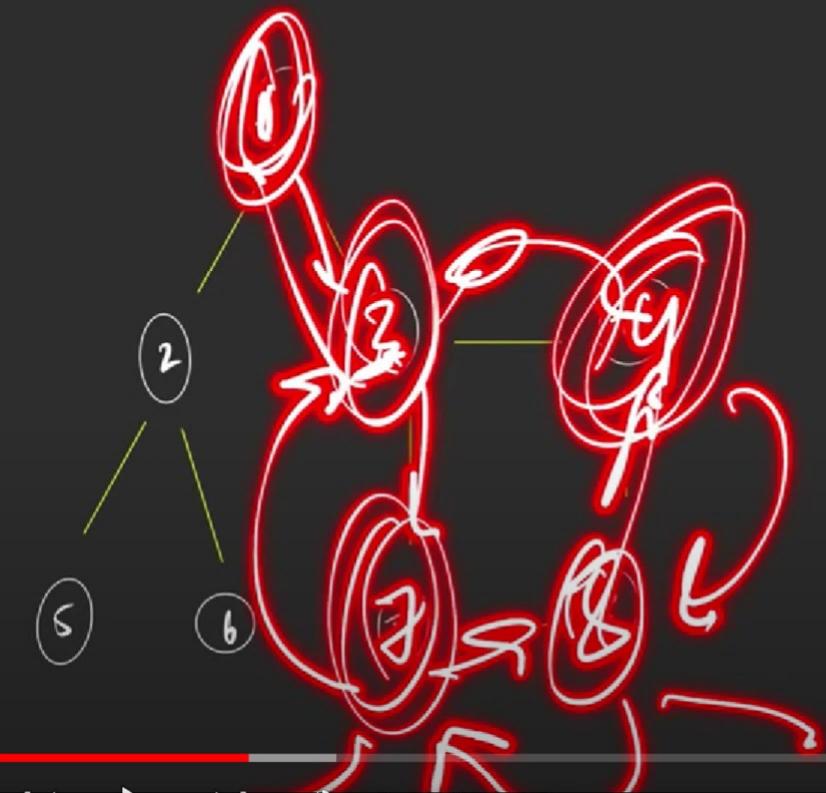
→ G-6. Depth-First Search (DFS) | C++ and Java | Traversal Technique in Graphs

2.00

DFS traversal in a graph

Starting
node = 1

1 2 5 6 3 7 8 4



2:49 / 20:15



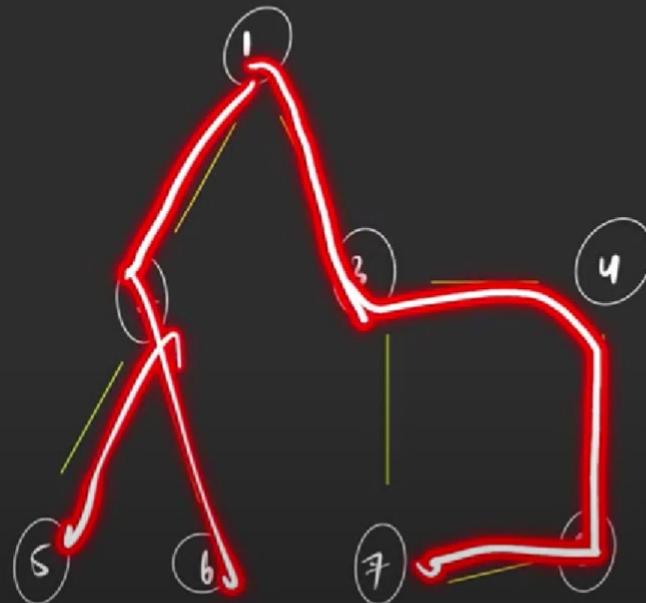
→ G-6. Depth-First Search (DFS) | C++ and Java | Traversal Technique in Graphs

2.00

DFS traversal in a graph

Starting
node = 1

1 2 5 6 3 7 8 4



3:22 / 20:15

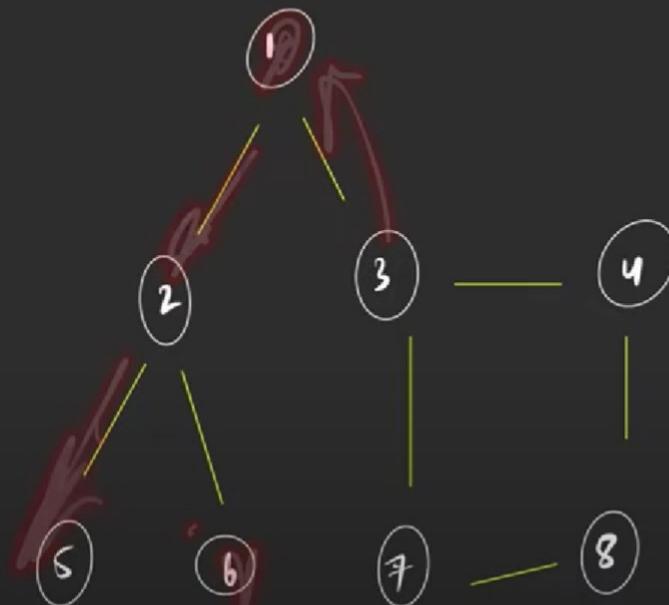


► G-6. Depth-First Search (DFS) | C++ and Java | Traversal Technique in Graphs

2.00 DFS traversal in a graph

Starting node = 3

3 4 8 7



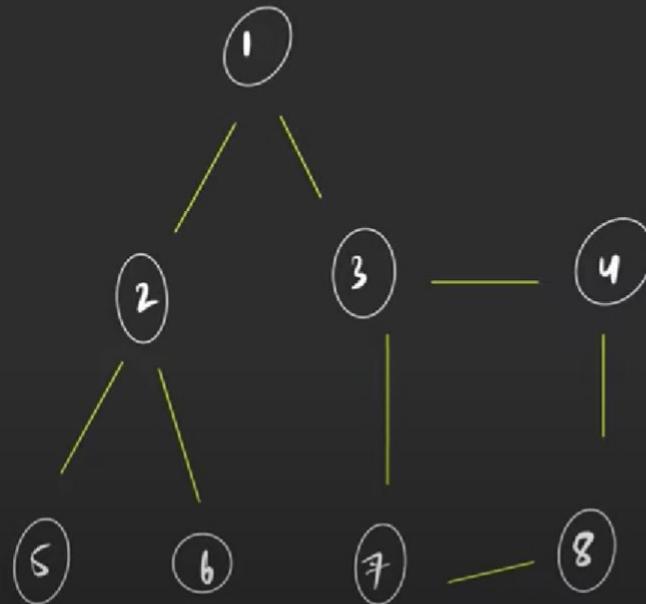
→ G-6. Depth-First Search (DFS) | C++ and Java | Traversal Technique in Graphs

2.00

DFS traversal in a graph

Starting node = 3

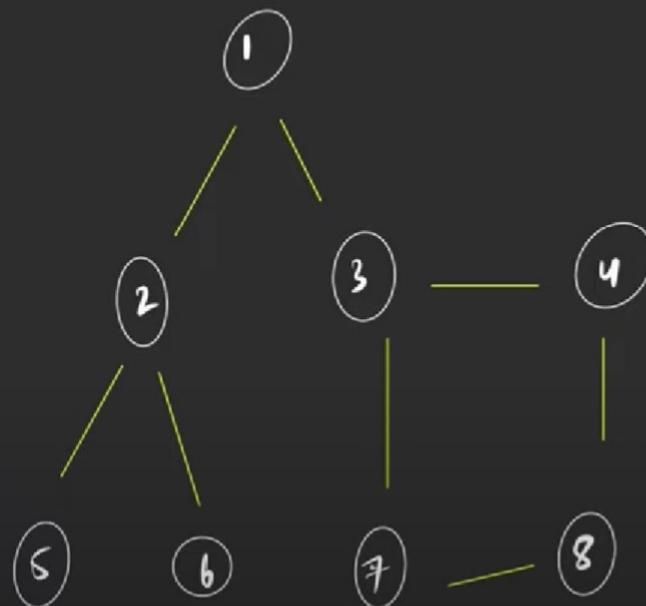
3 4 8 7 1 2 5 6



→ G-6. Depth-First Search (DFS) | C++ and Java | Traversal Technique in Graphs

2.00

DFS traversal in a graph → Recursion



~~adj list~~

1 → {2, 3}
2 → {1, 5, 6}
3 → {1, 4, 7}
4 → {3, 8}
5 → {2}
6 → {2, 3}
7 → {3, 8}
8 → {4, 7}



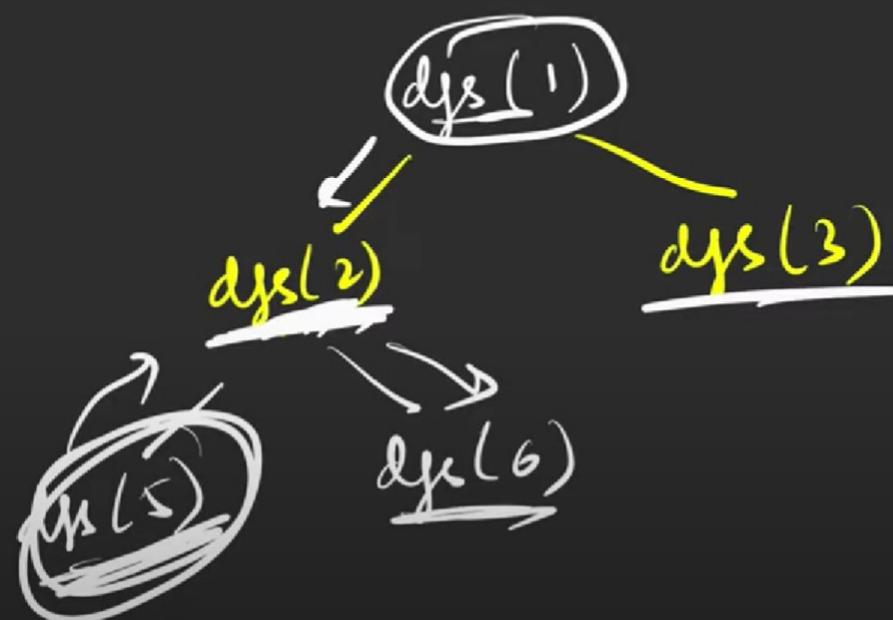
☰ G-6. Depth-First Search (DFS) | C++ and Java | Traversal Technique in Graphs

2.00

$s.N = 1$

\Leftarrow

0	1	2	3	0	0	1	0	0
0	1	2	3	4	5	6	7	8



11:08 / 20:15



G-6 Depth-First Search (DFS) | C++ and Java | Traversal Technique in Graphs

The video player interface includes controls for play/pause, volume, and navigation, along with a timestamp of 16:34 / 20:15 and a progress bar. The video content area displays a code editor with two tabs: 'C++ (g++ 5.4)' and 'Java'. The C++ tab contains the following code:

```
1 // } Driver Code Ends
6 class Solution {
7     private:
8         void dfs(int node, vector<int> adj[], int vis[], vector<int> &ls) {
9             vis[node] = 1;
10            ls.push_back(node);
11            // traverse all its neighbours
12            for(auto it : adj[node]) {
13                if(!vis[it]) {
14                    dfs(it, adj, vis, ls);
15                }
16            }
17        }
18    public:
19        // Function to return a list containing the DFS traversal of the graph
20        vector<int> dfsOfGraph(int V,
21                               ArrayList<ArrayList<Integer>> adj) {
22            // Code here
23            int vis[n] = {0};
24            int start = 0;
25            vector<int> ls;
26            dfs(start, adj, vis, ls);
27        }
28    };
29
30 }
```

The Java tab contains a similar implementation. Below the code editor, the output is shown as 'Output: 0 1 2 4 3' and the explanation states: '0 is connected to 1, 2, 4. 1 is connected to 0. 2 is connected to 0. 3 is connected to 4. 4 is connected to 0, 3. so starting from 0, it will go to 1 then 2'.

G-6 Depth-First Search (DFS) | C++ and Java | Traversal Technique in Graphs



Problem Editorial Submissions Comments C++ (g++ 5.4)

```
1 // } Driver Code Ends
2 class Solution {
3     private:
4         void dfs(int node, vector<int> adj[], int vis[], vector<int> &ls) {
5             vis[node] = 1;
6             ls.push_back(node);
7             // traverse all its neighbours
8             for(auto it : adj[node]) {
9                 if(!vis[it]) {
10                     dfs(it, adj, vis, ls);
11                 }
12             }
13         }
14     public:
15         // Function to return a list containing the DFS traversal of the graph.
16         vector<int> dfsOfGraph(int V, vector<ArrayList<Integer>> adj) {
17             // Code here
18             int vis[n] = {0};
19             int start = 0;
20             vector<int> ls;
21             dfs(start, adj, vis, ls);
22             return ls;
23         }
24     };
25 };
26 
```

Output: 0 1 2 4 3

Explanation:

- 0 is connected to 1, 2, 4.
- 1 is connected to 0.
- 2 is connected to 0.
- 3 is connected to 4.
- 4 is connected to 0, 3.

so starting from 0, it will go to 1 then 2

16:38 / 20:15

CC

G-6 Depth-First Search (DFS) | C++ and Java | Traversal Technique in Graphs

The video content displays the following C++ code for Depth-First Search (DFS) traversal of a graph:

```
1 // } Driver Code Ends
6 class Solution {
7 private:
8     void dfs(int node, vector<int> adj[], int vis[], vector<int> &ls) {
9         vis[node] = 1;
10        ls.push_back(node);
11        // traverse all its neighbours
12        for(auto it : adj[node]) {
13            if(!vis[it]) {
14                dfs(it, adj, vis, ls);
15            }
16        }
17    }
18 public:
19     // Function to return a list containing the DFS traversal of the graph
20     vector<int> dfsOfGraph(int V, vector<ArrayList<Integer>> adj) {
21         // Code here
22         int vis[V] = {0};
23         int start = 0;
24         vector<int> ls;
25         dfs(start, adj, vis, ls);
26         return ls;
27     }
28 };
29
30 }
```

Output: 0 1 2 4 3

Explanation:

- 0 is connected to 1, 2, 4.
- 1 is connected to 0.
- 2 is connected to 0.
- 3 is connected to 4.
- 4 is connected to 0, 3.

so starting from 0, it will go to 1 then 2

Output Window

Compile & Run

2.00

$$TC \rightarrow \boxed{\underline{O(N)} + (\underline{2 \times E})}$$

ϵ degrees



300 Problem

[Editorial](#)[Submissions](#)[Comments](#)

Given a grid of dimension **nxm** where each cell in the grid can have values 0, 1 or 2 which has the following meaning:

0: Empty cell

1: Cells have fresh oranges

2: Cells have rotten oranges

We have to determine what is the minimum time required to rot all oranges. A rotten orange at index (i,j) can rot other fresh orange at indexes $(i-1,j)$, $(i+1,j)$, $(i,j+1)$, $(i,j-1)$ (**up, down, left and right**) in unit time.

Example 1:

Input: grid = {{0,1,2},{0,1,2},{2,1,1}}

Output: 1

Explanation: The grid is-

0 1 2

0 1 2

2 1 1

Oranges at positions (0,2), (1,2), (2,0)

will rot oranges at (0,1), (1,1), (2,2) and

(2,1) in unit time.

Example 2:

```
1 // } Driver Code Ends
2 class Solution
3 {
4     public:
5         //Function to find minimum time required to rot all orang
6         int orangesRotting(vector<vector<int>>& grid) {
7             // Code here
8         }
9     };
10 // } Driver Code Ends
```



► G-10. Rotten Oranges | C++ | Java

2.00

Rotten Oranges

2	2	2
2	2	0
0	0	1

2 → rot

1 → fresh

0 → fresh

f = 1

f = 2



G-10. Rotten Oranges | C++ | Java

2.00



$$t_m = 4 \text{ sec}$$

$$t=2$$

$$t=3$$

$$t=4$$



2:51 / 22:29



☰ G-10. Rotten Oranges | C++ | Java

2.00

0	1	2
0	1	2
2	1	1



3:02 / 22:29



G-10. Rotten Oranges | C++ | Java

2.00

0	1	2
0	1	2
2	1	1

$$t = 1$$



Volume



3:11 / 22:29



2.00

0	2	2
0	2	2
-	2	2

$$t=1$$



TUF

☰ G-10. Rotten Oranges | C++ | Java

2.00

1 2 1
1 1 0
0 0 1



4:13 / 22:29



G-10. Rotten Oranges | C++ | Java

2.00

2 2 R
R R O
O O 1

$$\boxed{\begin{array}{l} t=1 \text{ s} \\ \hline \\ \theta=2 \end{array}}$$

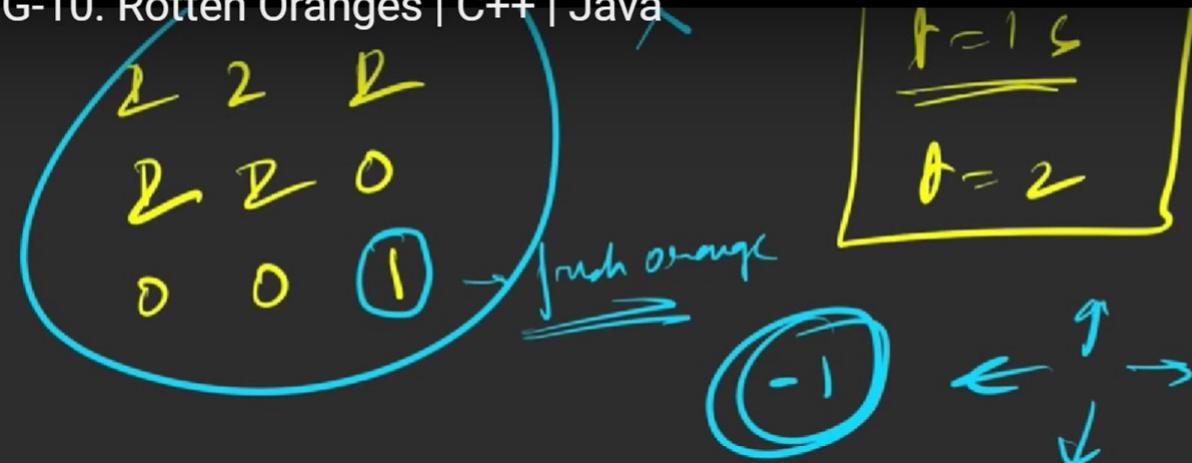


5:07 / 22:29



G-10. Rotten Oranges | C++ | Java

2.00



G-10. Rotten Oranges | C++ | Java

1.70

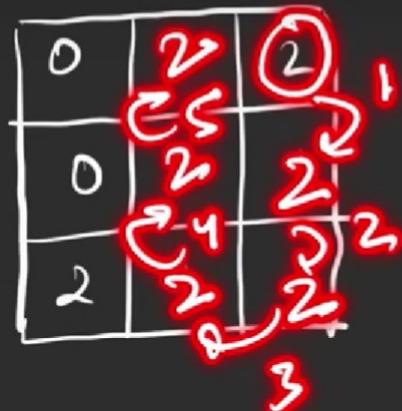
0	2	-
0	1	2
2	2	1

Which algorithm
 $i = 1$



G-10. Rotten Oranges | C++ | Java

1.70



which algorithm

BFS

level wise



G-10. Rotten Oranges | C++ | Java

1.70

	0	1	2
0	0	1	2
1	0	1	1
2	2	1	1

min Time

(2, 0)

$t = 0$

which algorithm

(2, 1), 1
(0, 2), 0
(2, 0), 0

0

	0	1	2
0			
1			
2	2		



► G-10. Rotten Oranges | C++ | Java

1.70

	0	1	2
0	0	1	2
1	0	1	1
2	2	1	1

min Time

which algorithm

(2,0) $t = \underline{\underline{0}}$

	(2,1),1
	(0,2),0
	(2,0),0

0

	0	1	2
0			
1			
2	2	2	



9:14 / 22:29



► G-10. Rotten Oranges | C++ | Java

1.70

	0	1	2
0	0	1	2
1	0	1	1
2	2	1	1

min Time

which algorithm

$$\begin{array}{l} (2,0) \rightarrow 0 \\ (0,2) \rightarrow 0 \end{array}$$

(0,1,1)
(1,2,1)
(2,1),1
(0,2),0
(2,0),0

0

	0	1	2
0		2	2
1			
2	2	2	



G-10. Rotten Oranges | C++ | Java

1.70

time = 1

	0	1	2
0	0	1	2
1	0	1	1
2	2	1	1

mm Time

$$\begin{array}{l} (2+0) + = 0 \\ (0+2) + = 0 \\ (2,1) \quad + = 1 \end{array}$$

which algorithm

(0,1,1)
(1,2,1)
(2,1,1)
(0,2,0)
(2,0,0)

0

	0	1	2
0		2	2
1			2
2			

10:12 / 22:29



G-10. Rotten Oranges | C++ | Java

1.70

0	0	1	2
1	0	0	1
2	2	0	1

0	0	1	2
1	2	2	2
2	2	2	2

$$\begin{array}{l} (2,0) \rightarrow + = 0 \\ (0,2) \rightarrow + = 0 \\ (2,1) \quad + = 1 \end{array}$$

(0,1,1)
(1,2,1)
(2,1,1)
(0,2,0)
(2,0,0)

0



10:28 / 22:29



► G-10. Rotten Oranges | C++ | Java

1.70

	0	1	2
0	0	1	2
1	0	1	1
2	2	1	1

min Time

which algorithm

~~(2,0)~~ → 0
~~(0,2)~~ → 0
(2,1) → 1

(0,1,1)
(1,2,1)
(2,1,1)
(0,2,0)
(2,0,0)

0

	0	1	2
0		2	2
1			
2	2	2	



10:02 / 22:29

▼



1.70

 $\text{time} = 1$

	0	1	2
0	0	1	2
1	0	1	1
2	2	1	1

	0	1	2
0		2	2
1			2

mm Time

$$\begin{array}{l} (2+0) \cancel{+} 0 \\ (0+2) \cancel{+} 0 \\ (2,1) \quad + = 1 \end{array}$$

which algorithm

(0,1,1)
(1,2,1)
(2,1,1)
(0,2,0)
(2,0,0)

0



TUF

1.70

0	0	1	2
1	0	0	1
2	2	1	0

0	0	1	2
1		2	2
2	2	2	

$$\begin{array}{l} (2,0) \xrightarrow{+ = 0} \\ (0,2) \xrightarrow{+ = 0} \\ (2,1) \xrightarrow{- = 1} \end{array}$$

(0,1,1)
(1,2,1)
(2,1,1)
(0,2,0)
(2,0,0)

0



TUF

G-10. Rotten Oranges | C++ | Java

1.70

	0	1	2
0	0	1	2
1	0	1	1
2	2	1	1

	0	1	2
0		2	2
1		2	2
2	2	2	2

mm Time

~~(2,0)~~ + - 0
~~(0,2)~~ + - 0
~~(2,1)~~ + - +

Which algorithm

(2,2,2)
(1,1,2)
(0,1,1)
(1,2,1)
(2,1,1)
(0,2,0)
(2,0,0)

0



1.70

$$\begin{pmatrix} 2 & 2 & 1 \\ 2 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\text{time} = t_2$$

$\overbrace{\quad\quad\quad}^{\text{fresh orange}}$

$$\boxed{t=2}$$

$$\boxed{-1} \quad \leftarrow \begin{matrix} \uparrow \\ \downarrow \end{matrix}$$

	0	1	2
0	0	1	2
1	0	1	1
2	2	1	1

mm Time

$$(2+0) \xrightarrow{+-0}$$

$$(0+2) \xrightarrow{+-0}$$

$$(2+1) \xrightarrow{+-+}$$

Wurch algorithm

$$\left| \begin{array}{c} (2, 2, 1) \\ (1, 2, 2) \\ (0, 1, 2) \\ (1, 2, 1) \\ (2, 1, 1) \\ (0, 2, 0) \\ (2, 0, 0) \end{array} \right|$$



TUF

1.70

$$\begin{array}{ccc}
 2 & 2 & 1 \\
 2 & 2 & 0 \\
 0 & 0 & 1
 \end{array} \xrightarrow{\text{fresh orange}} \boxed{\theta = 2}$$

$\text{time} = \tau_2$

	0	1	2
0	0	1	2
1	0	1	1
2	2	1	1

mm Time

$$\begin{aligned}
 (2, 0) &\xrightarrow{+-0} \\
 (0, 2) &\xrightarrow{+-0} \\
 (2, 1) &\xrightarrow{+-+}
 \end{aligned}$$

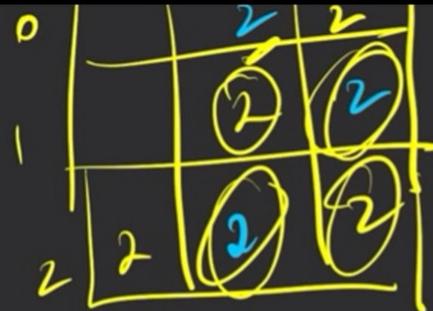
Wunsch algorithm

$$\left| \begin{array}{c}
 (2, 2, 1) \\
 (2, 1, 2) \\
 (1, 2, 1) \\
 (2, 1, 1) \\
 (0, 2, 0) \\
 (2, 0, 2)
 \end{array} \right|$$



TUF

1.70



$\begin{pmatrix} 0, 1 \end{pmatrix}, 1$
 $\begin{pmatrix} 1, 1 \end{pmatrix}, 2$
 $\begin{pmatrix} 2, 2 \end{pmatrix}, 2$

θ
empty

~~$\delta m = 2 \text{ secs}$~~ \rightarrow Answer



G-10. Rotten Oranges | C++ | Java

1.70

row-1, col
↑
 $\text{row}, \text{col}+1 \leftarrow (\underline{\text{row}}, \underline{\text{col}}) \rightarrow \text{row}, \text{col}+1$
↓
row+1, col



G-10. Rotten Oranges | C++ | Java

Courses Get Hired Events POTD Practice ↗ 321

1.70 Problem Editorial Submissions Comments C++ (g++ 5.4) ▾

Task:
You don't need to read or print anything. Your task is to complete the function **orangesRotting()** which takes grid as input parameter and returns the minimum time to rot all the fresh oranges. If not possible returns -1.

Expected Time Complexity: O(n*m)
Expected Auxiliary Space: O(n)

Constraints:
n, m ≤ 500

New Bookmarked Problems

Output Window

Pagination Results Custom Input

Problem Solved Successfully ✓

You get marks only for the first correct submission if you solve the problem without giving the full solution.

Test Cases Passed: 60 / 60 Your Total Score: 321

```
1 // } Driver Code Ends
2 class Solution
3 {
4     public:
5         //Function to find minimum time required to rot all oranges.
6         int orangesRotting(vector<vector<int>>& grid) {
7             // Code here
8             int n = grid.size();
9             int m = grid[0].size();
10            queue<pair< pair<int,int>, int>> q;
11            int vis[n][m];
12            for(int i = 0;i<n;i++) {
13                for(int j = 0;j<m;j++) {
14                    if(grid[i][j] == 2) {
15                        q.push({{i, j}, 0});
16                        vis[i][j] = 2;
17                    }
18                    else {
19                        vis[i][j] = 0;
20                    }
21                }
22            }
23
24            int tm = 0;
25            int drow[] = {-1, 0, +1, 0};
26            int dcol[] = {0, 1, 0, -1};
27            while(!q.empty()) {
28                int r = q.front().first.first;
29                int c = q.front().first.second;
30                int t = q.front().second;
31                tm = max(tm, t);
32                q.pop();
33                for(int i = 0;i<4;i++) {
34                    int nrow = r + drow[i];
35                    int ncol = c + dcol[i];
36                    if(nrow < 0 || nrow >= n || ncol < 0 || ncol >= m || vis[nrow][ncol] != 0) {
37                        continue;
38                    }
39                    if(grid[nrow][ncol] == 1) {
40                        q.push({{nrow, ncol}, t + 1});
41                        vis[nrow][ncol] = 2;
42                    }
43                }
44            }
45            return tm;
46        }
47 }
```

◀ ▶ ⏪ ⏩ 19:24 / 22:29 CC ⚙ #



G-10. Rotten Oranges | C++ | Java

```
Queue<Pair> q = new LinkedList<>();
// n x m
int[][] vis = new int[n][m];
int cntFresh = 0;
for(int i = 0;i<n;i++) {
    for(int j = 0;j<m;j++) {
        if(grid[i][j] == 2) {
            q.add(new Pair(i, j, 0));
            vis[i][j] = 2;
        }
        else {
            vis[i][j] = 0;
        }
        if(grid[i][j] == 1) cntFresh++;
    }
}
int tm = 0;
int drow[] = {-1, 0, +1, 0};
int dcol[] = {0, 1, 0, -1};
int cnt=0;
while(!q.isEmpty()) {
    int r = q.peek().row;
    int c = q.peek().col;
    int t = q.peek().tm;
    tm = Math.max(tm, t);
    q.remove();
    for(int i = 0;i<4;i++) {
        int nrow = r + drow[i];
        int ncol = c + dcol[i];
        if(nrow >= 0 && nrow < n && ncol >= 0 && ncol < m
        && vis[nrow][ncol] == 0 && grid[nrow][ncol] == 1) {
            q.add(new Pair(nrow, ncol, t + 1));
            vis[nrow][ncol] = 2;
            cnt++;
        }
    }
}
if(cnt != cntFresh) return -1;
return tm;
```

Practice

321

```
31     int tm = 0;
32     int drow[] = {-1, 0, +1, 0};
33     int dcol[] = {0, 1, 0, -1};
34     while(!q.empty()) {
35         int r = q.front().first;
36         int c = q.front().first.second;
37         int t = q.front().second;
38         tm = max(tm, t);
39         q.pop();
40         for(int i = 0;i<4;i++) {
41             int nrow = r + drow[i];
42             int ncol = c + dcol[i];
43             if(nrow >= 0 && nrow < n && ncol >= 0 && ncol < m
44             && vis[nrow][ncol] == 0 && grid[nrow][ncol] == 1)
45                 q.push({{nrow, ncol}, t + 1});
46                 vis[nrow][ncol] = 2;
47         }
48     }
49
50
51     for(int i = 0;i<n;i++) {
52         for(int j = 0;j<m;j++) {
53             if(vis[i][j] != 2 && grid[i][j] == 1) return -1;
54         }
55     }
56
57     return tm;
58
59
60
61 }
62 };
63
64
```



G-10. Rotten Oranges | C++ | Java

1.70

$$\text{SL} \approx \frac{N \times M}{\text{TC}} \rightarrow \text{TC} = \frac{(N \times M) \times Y}{\text{SL}}$$

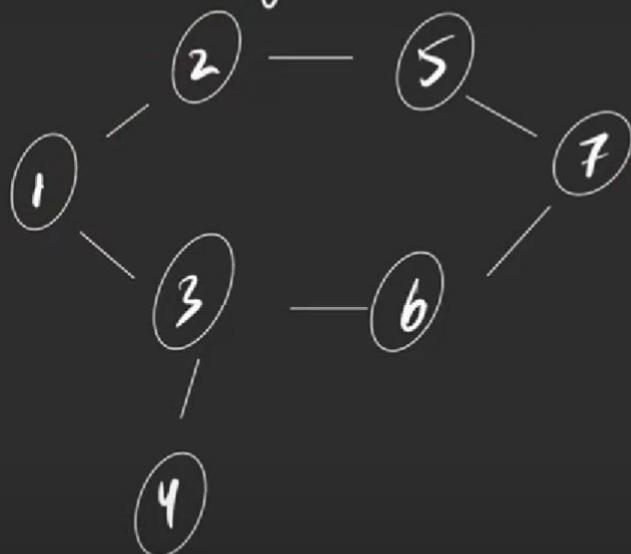


21:52 / 22:29



► G-11. Detect a Cycle in an Undirected Graph using BFS | C++ | Java

2.00



Press Esc to exit full screen

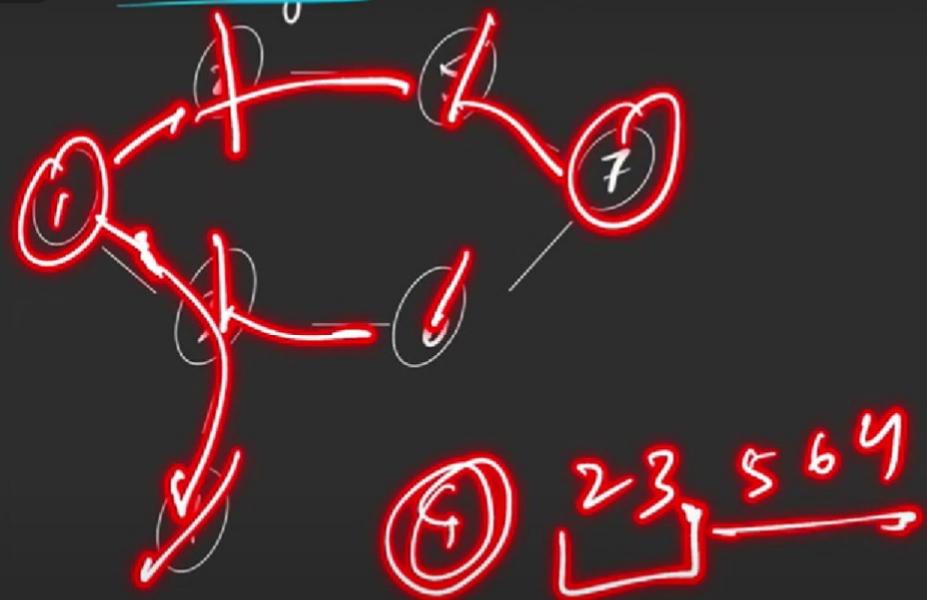
Detect Cycle in an Undirected Graph → BFS
adj. list

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



→ G-11. Detect a Cycle in an Undirected Graph using BFS | C++ | Java

Detect Cycle in an Undirected Graph → BFS
adj. list



adj. list

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



1:39 / 20:18



→ G-11. Detect a Cycle in an Undirected Graph using BFS | C++ | Java

Detect Cycle in an Undirected Graph → BFS
adj. list



adj. list

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



G-11. Detect a Cycle in an Undirected Graph using BFS | C++ | Java

Courses Get Hired Events ↗ PTD Practice 321 🔍 🔔

2.00 Problem Editorial Submissions Comments C++ (g++ 5.4) ↗

```
1 // } Driver Code Ends
2 class Solution {
3     private:
4         bool detect(int src, vector<int> adj[], int vis[]) {
5             vis[src] = 1;
6             queue<pair<int,int>> q;
7             q.push({src, -1});
8             while(!q.empty()) {
9                 int node = q.front().first;
10                int parent = q.front().second;
11                q.pop();
12
13                for(auto adjacentNode: adj[node]) {
14                    if(vis[adjacentNode] == false) {
15                        vis[adjacentNode] = true;
16                        q.push({adjacentNode, node});
17                    }
18                    else if(parent != adjacentNode) {
19                        return true;
20                    }
21                }
22            }
23        }
24
25    }
26
27
28 public:
29     // Function to detect cycle in an undirected graph.
30     bool isCycle(int V, vector<int> adj[]) {
31         // Code here
32     }
33 }
34
35 }
```

adj = {}, {2}, {1, 3}, {2}

Output: 0

12:25 / 20:18



→ G-11. Detect a Cycle in an Undirected Graph using BFS | C++ | Java

2.00

```
for( i = 1; i < N; i++)  
{  
    if (!vis[i])  
        if (detectCycle(i) == true)  
            return true;  
}  
return false;
```



15:14 / 20:18



$$\sum \text{adjacent} = \sum \text{degree} = \boxed{2E}$$

$$\textcircled{\text{TC}} \rightarrow O(\underline{N} + 2E)$$



→ G-11. Detect a Cycle in an Undirected Graph using BFS | C++ | Java

2.00

}
suche fahn

$$\sum \text{adjacent} = \sum \text{degree} = 2E$$

$$TC \rightarrow O(N + 2E) + O(w)$$

$$SC \rightarrow O(N), + O(w) \approx O(w)$$



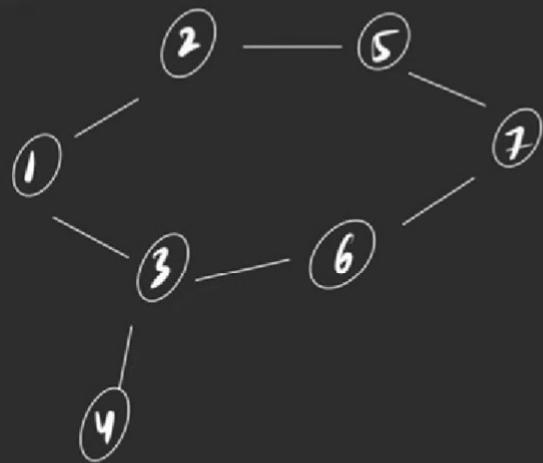
of 132



19:27 / 20:18



Cycle Detection in Undirected Graph → DFS



adj List

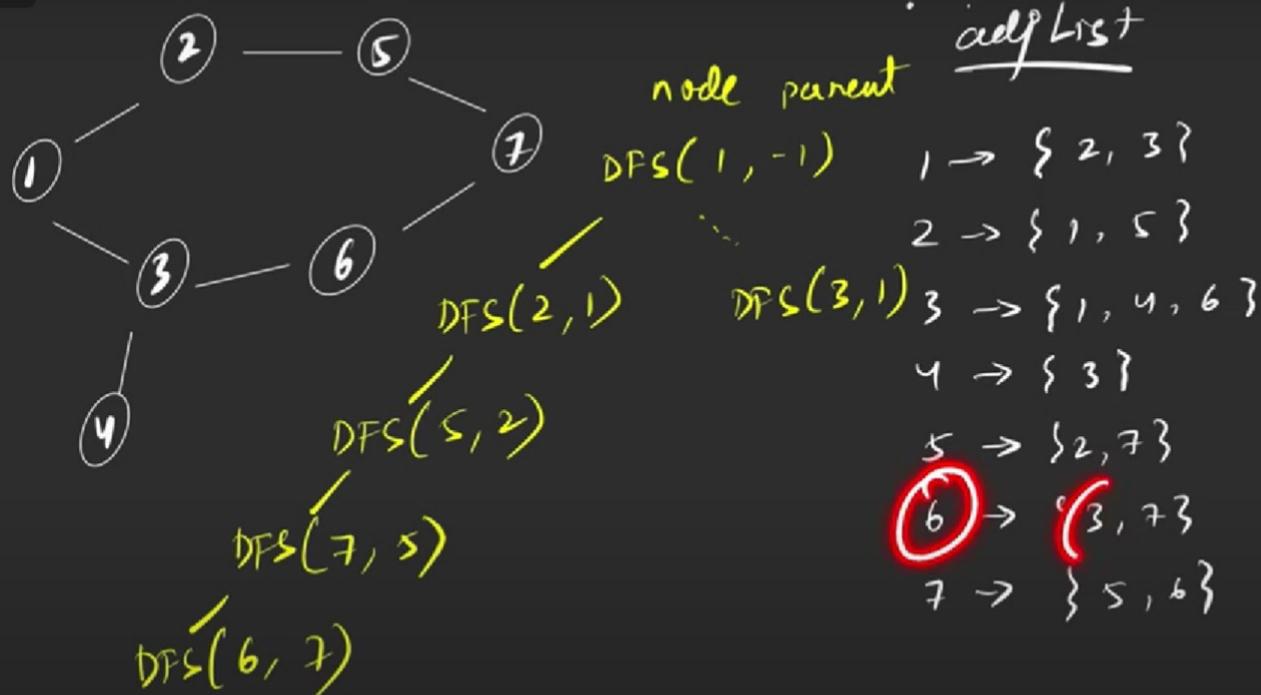
1 → {2, 3}
2 → {1, 5}
3 → {1, 4, 6}
4 → {3}
5 → {2, 7}
6 → {3, 7}
7 → {5, 6}

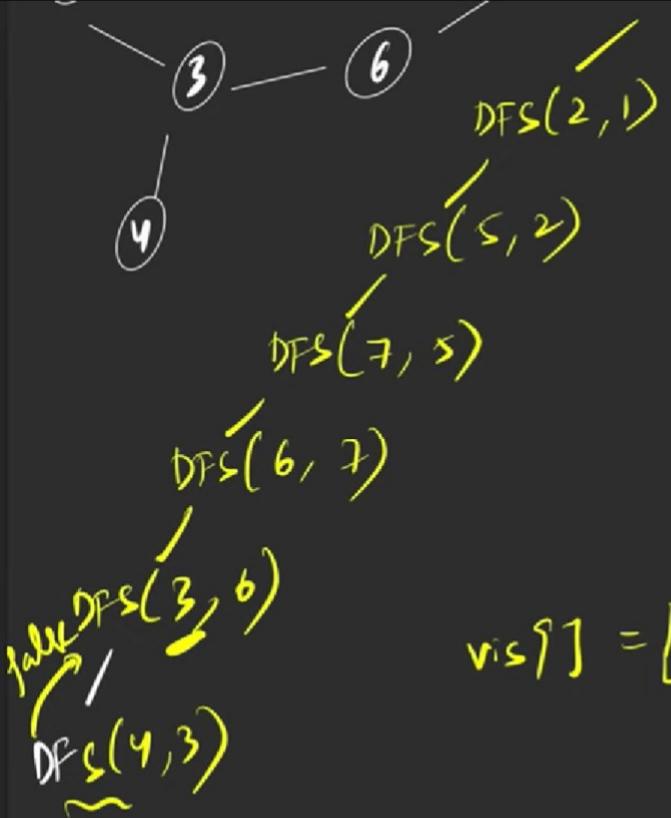


TUF

⇒ G-12. Detect a Cycle in an Undirected Graph using DFS | C++ | Java
Cycle Detection in Undirected Graph → DFS

1.85

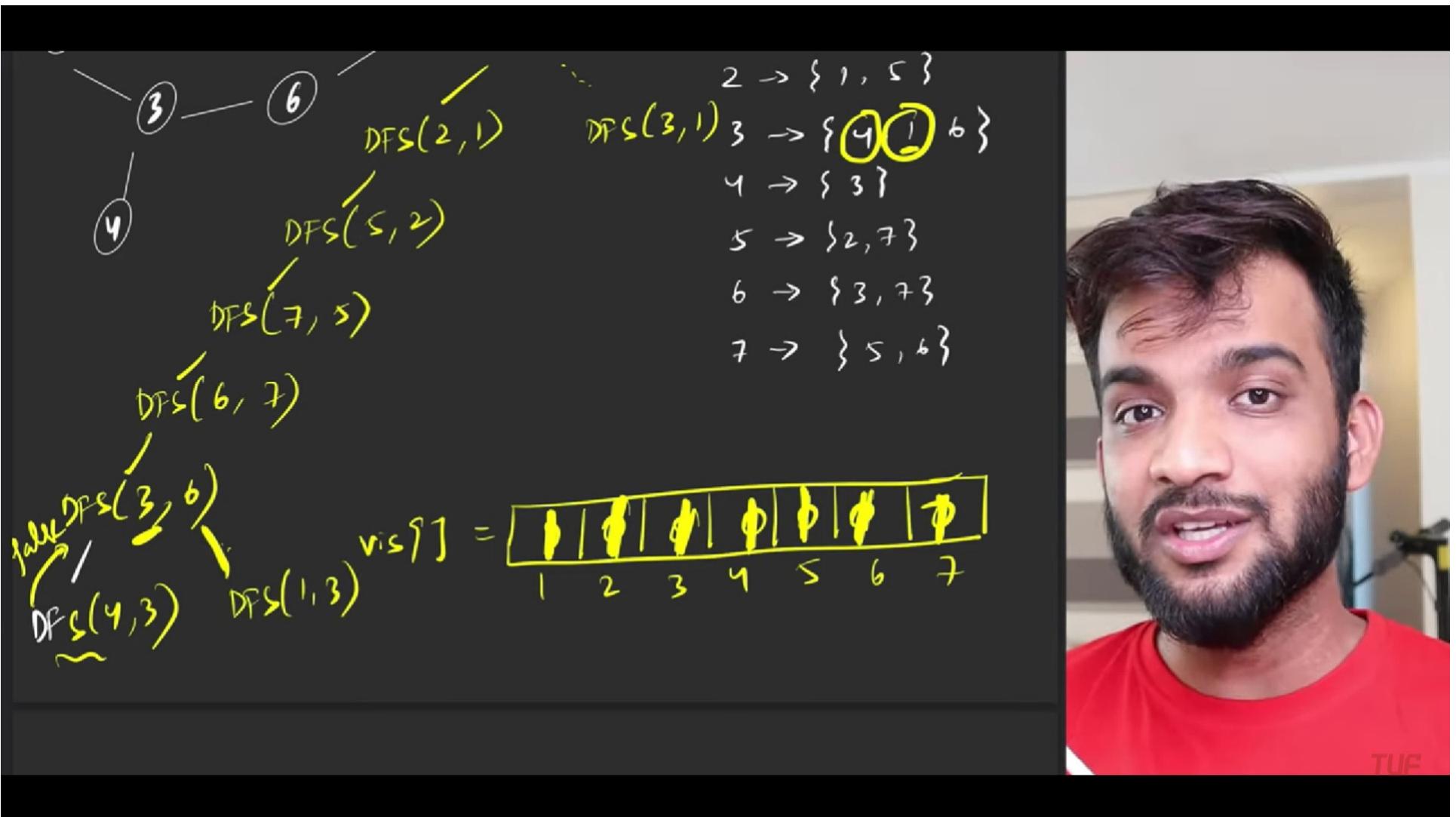


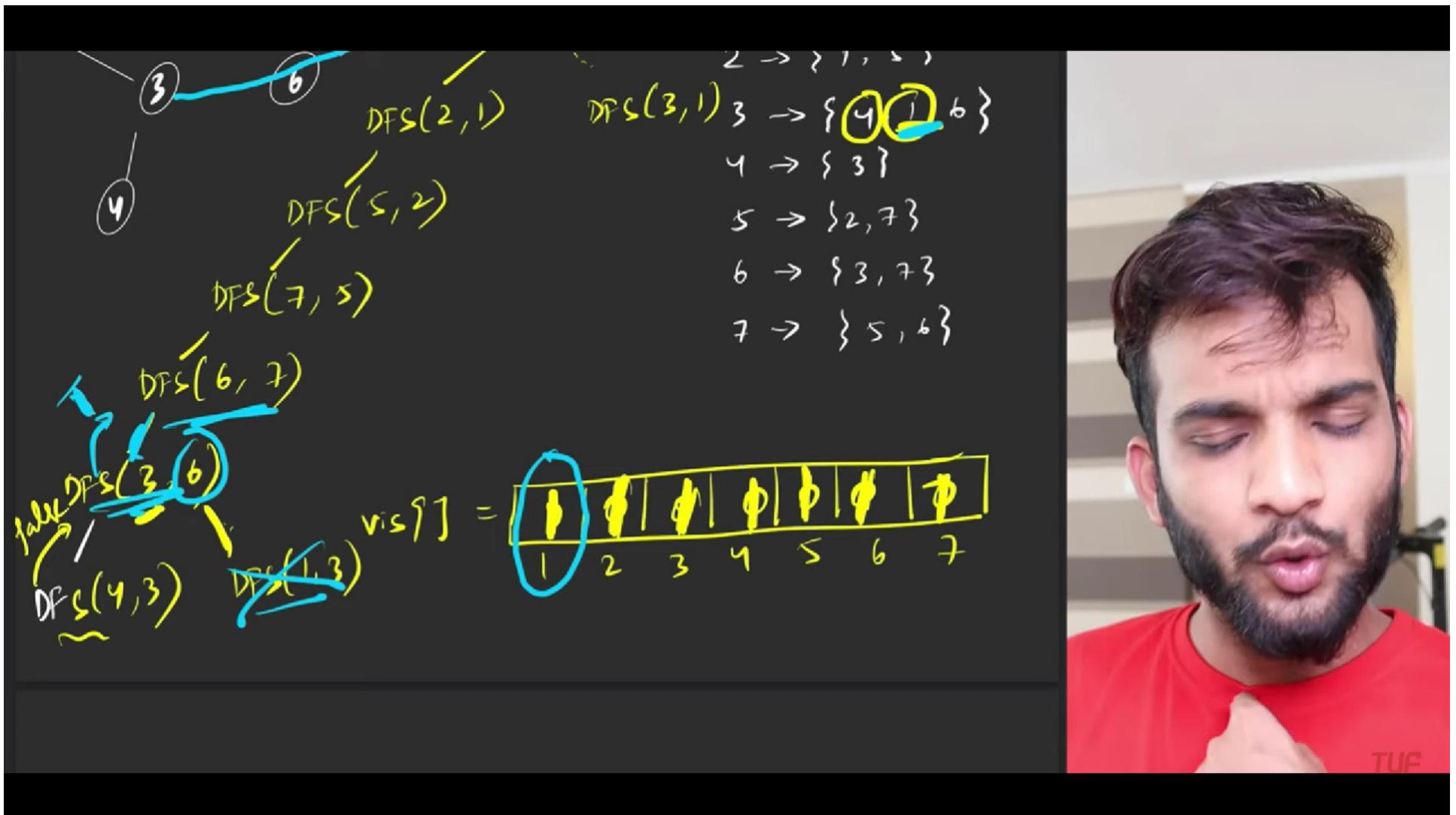


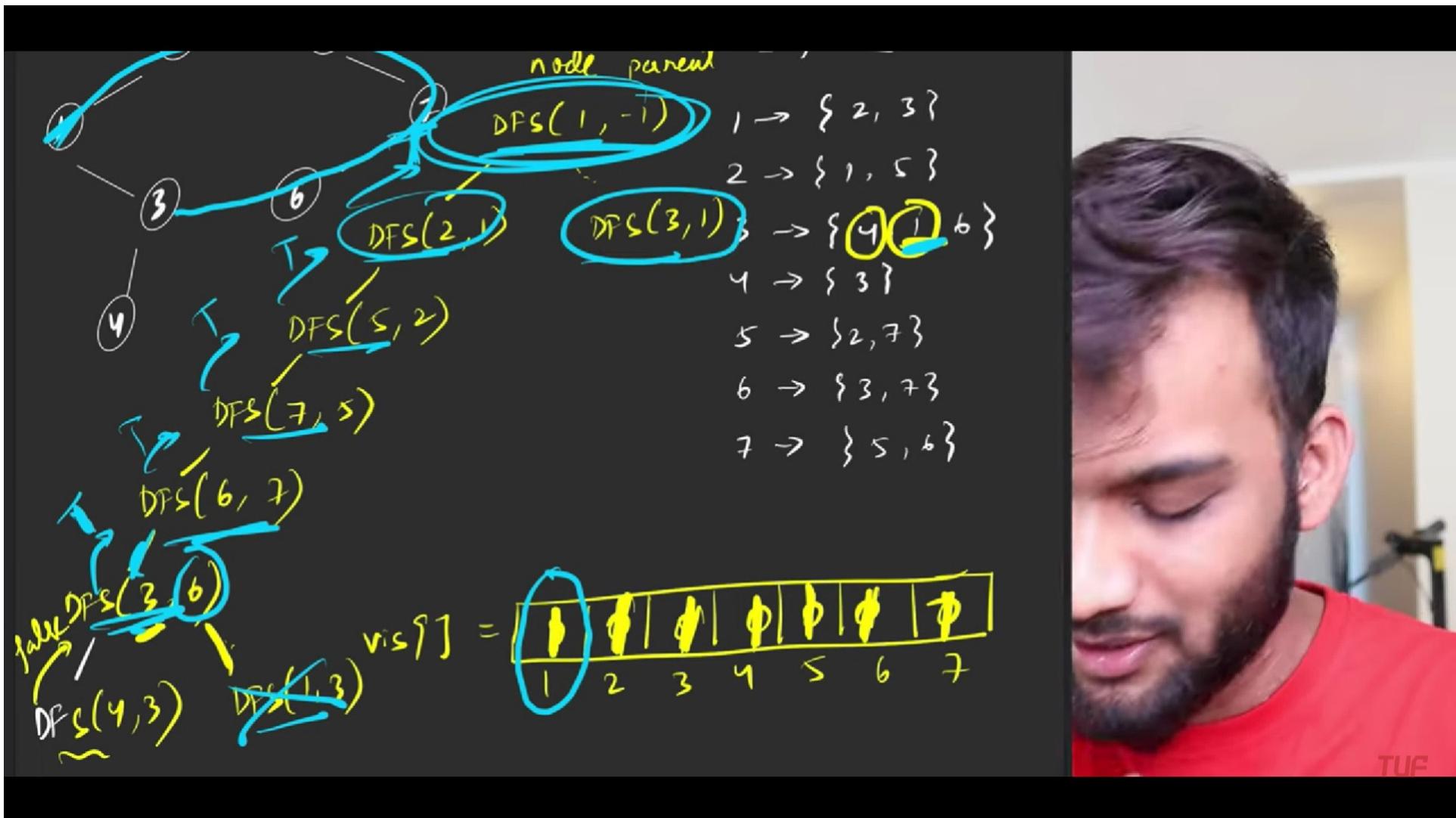
$2 \rightarrow \{1, 5\}$
 $\text{DFS}(3, 1) \quad 3 \rightarrow \{4, 6\}$
 $4 \rightarrow \{3\}$
 $5 \rightarrow \{2, 7\}$
 $6 \rightarrow \{3, 7\}$
 $7 \rightarrow \{5, 6\}$

$\text{vis}[] = [$









$\text{dfs}(\text{node}, \text{parent})$

{
 $\text{vis}[\text{node}] = 1$

 for (auto it : $\text{adj}[\text{node}]$)

 if ($\text{visit}[\text{it}] == 0$)
 if ($\text{dfs}(\text{it}, \text{node}) == \text{true}$)
 return Tree;

 → else if ($\text{it} != \text{parent}$)

 return Tree;

 → }

root_val;



TUF

Courses Get Hired Events ↗ POTD

Practice

321 🔍

problem Editorial Submissions Comments

```
1 // } Driver Code Ends
2 class Solution {
3     private:
4         bool dfs(int node, int parent, vector<int> vis, vector<vector<int>> adj) {
5             vis[node] = 1;
6             for(int adjacentNode: adj[node]) {
7                 if(vis[adjacentNode]==0) {
8                     if(dfs(adjacentNode, node, vis, adj) == true)
9                         return true;
10                }
11            else if(adjacentNode != parent) return true;
12        }
13        return false;
14    }
15    // Function to detect cycle in an undirected graph.
16    public:
17        // Function to detect cycle in an undirected graph.
18        bool isCycle(int V, vector<vector<int>> adj) {
19            int vis[] = new int[V];
20            for(int i = 0;i<V;i++) {
21                if(vis[i] == 0) {
22                    if(dfs(i, -1, vis, adj) == true) return true;
23                }
24            }
25            return false;
26        }
27    }
28 }
```

cycle in the graph.

Task:

$$SL \rightarrow O(N) + O(\underline{N}) \asymp O(\omega)$$
$$TL \rightarrow O(\underline{N+2E}) + O(\omega)$$

The diagram illustrates the analysis of time complexities. The first term, $O(N) + O(N)$, is simplified to $O(\omega)$. The second term, $O(N+2E) + O(\omega)$, is also simplified to $O(\omega)$. The simplification is justified by the fact that the first term dominates the overall complexity due to the constant factor 2 in the second term.



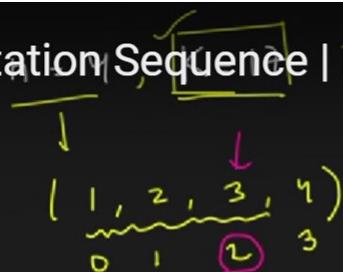
$$\begin{aligned}
 & h = 4, \quad \boxed{K=17} = \underline{\underline{16}}^{+4} \\
 & \boxed{24} \quad \left(\begin{smallmatrix} 1, 2, 3 \\ 0, 1, 2 \end{smallmatrix} \right) \quad 16/6 = 2 \quad \underline{1} + \left[\begin{smallmatrix} 3! \\ (2, 3, 4) \end{smallmatrix} \right] \frac{6}{6} (0-5) \\
 & \underline{2} + \left[\begin{smallmatrix} 3! \\ (1, 3, 4) \end{smallmatrix} \right] \frac{6}{6} (6-11) \\
 & \underline{3} + \left[\begin{smallmatrix} 3! \\ (1, 2, 4) \end{smallmatrix} \right] \frac{6}{6} (12-17) \\
 & \underline{4} + \left[\begin{smallmatrix} 3! \\ (1, 2, 3) \end{smallmatrix} \right] \frac{6}{6} (18-23) \\
 & \overline{24}
 \end{aligned}$$

$$\begin{array}{r}
 3 \\
 - - - \\
 4 3 2 1 - 23^{+4}
 \end{array}$$

L18. K-th Permutation Sequence | Leetcode

1.90

24



$$16/6 = 2$$

$$16 \cdot 6 = 4$$

$$\{1, 2, 3\}, \quad k = 4$$

$$\left[\frac{3!}{6} + (2, 3, 4) \right] 6 \quad (0-5)$$

$$\left[\frac{2!}{6} + (1, 3, 4) \right] 6 \quad (6-11)$$

$$\left[\frac{1!}{6} + (1, 2, 4) \right] 6 \quad (12-17)$$

$$\left[\frac{0!}{6} + (1, 2, 3) \right] 6 \quad (18-23)$$

24

4 3 2 1 — 23th

3
— — —

TUF



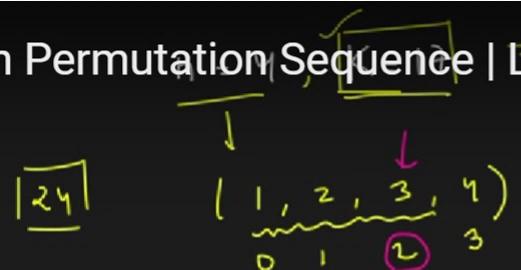
9:34 / 24:40 • Optimal Solution >



L18. K-th Permutation Sequence | Leetcode



1.90



$$\{1, 2, 4\}, \underline{k} = 4$$

4 3 2 1 — 2⁴

3
— — —

TUF



9:45 / 24:40 • Optimal Solution >



L18. K-th Permutation Sequence | Leetcode

2.05

$$\boxed{24} \quad \begin{array}{c} \downarrow \\ (1, \underbrace{2, 3}_{0, 1}, \underbrace{4}_{2}) \end{array}$$

$$\{1, 2, 4\}, \quad \underline{k=4}/2 = 2$$

$$3! = 6$$

$$4 \% 2 = 0$$

$$\{1, \underbrace{2}\} \quad |k=0$$

$$(0) \quad 1 \quad \{2, 4\} \quad \boxed{1} \quad \approx (0-1)$$

$$(1) \quad 2 \quad \{1, 4\} \quad \boxed{2} \quad \approx (2-3)$$

$$\Rightarrow (2) \quad 4 \quad \{1, 2\} \quad \boxed{2} \quad \frac{(4-5)}{6}$$

$$4 \ 3 \ 2 \ 1 \quad \rightarrow \ 2341$$

$$\begin{matrix} 3 & 4 \\ \uparrow & \end{matrix} \quad \boxed{-} \quad \boxed{-}$$

TUF



12:23 / 24:40 • Optimal Solution >



$$\frac{n=4, \sqrt{K=17} = 16^{+4}}{\boxed{24} \quad (1, 2, 3, 4) \quad \downarrow}$$

$$\Rightarrow (0 + \{2\}) \quad \underline{+} \quad (0+0)$$

$$2 + \{1\} \quad \underline{+} \quad (1+1)$$

$$2 + \{ _ \}$$

$$\frac{\{1, 2\} \quad K=0/1 = 0}{2! = 2 \quad K=0 \neq 0 = 0}$$

$$\begin{matrix} 3 & 4 & 1 & 2 \\ \uparrow & & & \end{matrix}$$

L18. K-th Permutation Sequence | Leetcode

2.05

$$(4!) = \boxed{24}$$

$$3! = 6$$

$$2! = 2$$

$$1! = 1$$

$$3! = 6$$

$$\boxed{1}, \boxed{2}, \boxed{3}, \boxed{4}$$

$$(1, 2, \underset{0}{\cancel{3}}, \underset{1}{\cancel{2}}, 3)$$

$$\checkmark_3$$

$$\{1, 2, \cancel{3}\}, \quad k = 4 / 2 = \cancel{2} \checkmark_4$$

$$3! = 6$$

$$4 \% 2 = 0$$

$$\{1, \cancel{2}\}$$

$$2! = 2$$

$$\{2\}$$

$$k = 0$$

$$2$$

$$k = 0 \% 0 = 0$$

$$2 \% 0 = 0$$

$$\{1, 2, \cancel{3}\}$$

$$1! = 1$$

$$1 \% 1 = 0$$

$$\{1, 2, 3\}$$

$$0 \% 1 = 0$$

$$\{1, 2, 3\}$$

$$1 \% 0 = 1$$

$$1 \% 1 = 0$$

$$\{1, 2, 3\}$$

$$0 \% 0 = 0$$

$$\{1, 2, 3\}$$

$$\boxed{1}, \boxed{2}, \boxed{3}, \boxed{4}$$



TUF



16:36 / 24:40 • Optimal Solution >



L18. K-th Permutation Sequence | Leetcode



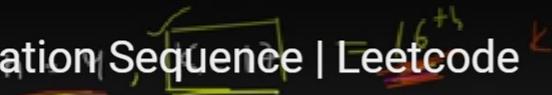
2.05

$$(4) = \boxed{124}$$

$$3! = 6$$

$$2! = 2$$

$$1! = 1$$



$$\checkmark 3$$

$$T_C \rightarrow \mathcal{O}(N) \times \mathcal{O}(P)$$

$$= \mathcal{O}(N^2)$$

$$SC = \mathcal{O}(N)$$

$$, \underline{k=4} / 2 = \underline{\underline{2}} \checkmark 4$$

$$3! = 6$$

$$4 \% 2 = 0$$

$$\{ \cancel{1}, 2 \} \quad \underline{k=0} / 1 = \underline{\underline{0}} \checkmark 1$$

$$2! = 2$$

$$\{ \cancel{0} \} \quad \underline{k=0} / 0 = \underline{\underline{0}}$$

$$\rightarrow (\underline{\underline{3}} \quad \underline{\underline{4}} \quad \underline{\underline{1}} \quad \underline{\underline{2}})$$

TUF



18:02 / 24:40 • Optimal Solution >



G-16. Number of Distinct Islands | Constructive Thinking + DFS | C++ | Java

Courses Get Hired Events </> POTD Practice 333

1.90 Problem Editorial Submissions Comments

Number of Distinct Islands

Medium Accuracy: 39.89% Submissions: 366 Points: 4

Given a boolean 2D matrix `grid` of size $n * m$. You have to find the number of distinct islands where a group of connected 1s (horizontally or vertically) forms an island. Two islands are considered to be distinct if and only if one island is equal to another (not rotated or reflected).

Example 1:

Input:
`grid[][] = {{1, 1, 0, 0, 0},
 {1, 1, 0, 0, 0},
 {0, 0, 0, 1, 1},
 {0, 0, 0, 1, 1}}`

Output:
1

Explanation:
Island 1, 1 at the top left corner is same as island 1, 1 at the bottom right corner.

Example 2:

Input:
`grid[][] = {{1, 1, 0, 1, 1},`

C++ (g++ 5.4) ▾

```
1 // } Driver Code Ends
2 // User function Template for C++
3
4 class Solution {
5 public:
6     int countDistinctIslands(vector<vector<int>>& grid) {
7         // code here
8     }
9 };
10
11 // } Driver Code Ends
```

0:38 / 18:01

CC



G-16. Number of Distinct Islands | Constructive Thinking + DFS | C++ | Java

1.90

Number of distinct islands

1	1	0	1	1
1	0	0	0	0
0	0	0	0	1
1	1	0	1	1



1:23 / 18:01



G-16. Number of Distinct Islands | Constructive Thinking + DFS | C++ | Java

1.90

Number of distinct islands



1:38 / 18:01



G-16. Number of Distinct Islands | Constructive Thinking + DFS | C++ | Java

1.90

Number of distinct islands



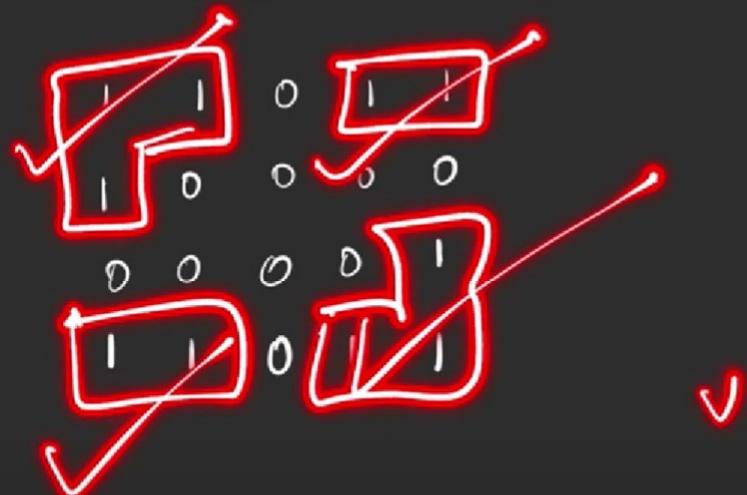
1:41 / 18:01



G-16. Number of Distinct Islands | Constructive Thinking + DFS | C++ | Java

1.90

Number of distinct islands



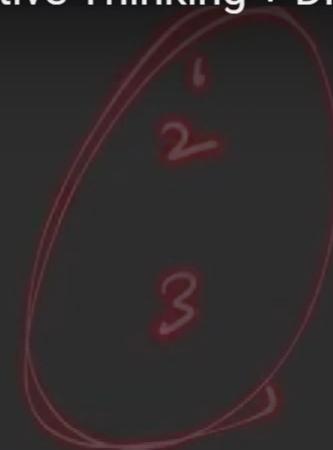
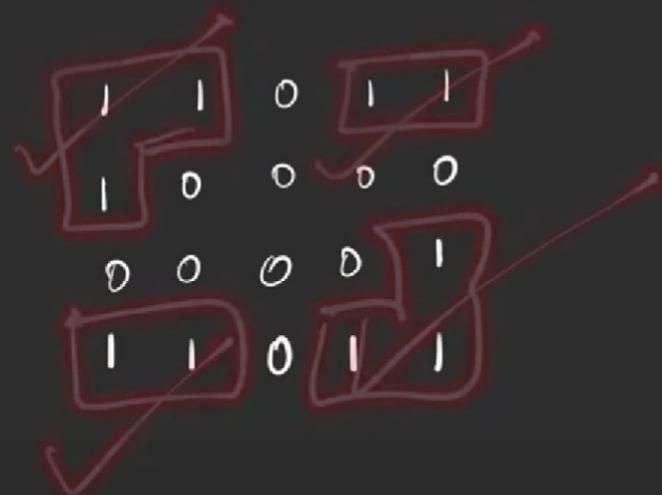
1:47 / 18:01



G-16. Number of Distinct Islands | Constructive Thinking + DFS | C++ | Java

1.90

Number of distinct islands



unique
dust
islands



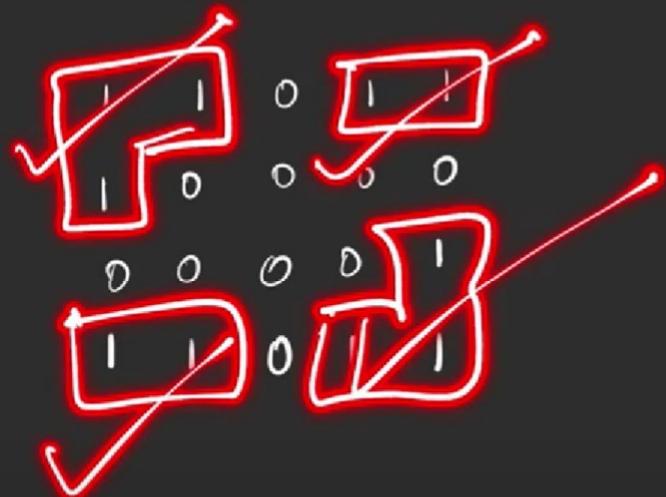
1:52 / 18:01



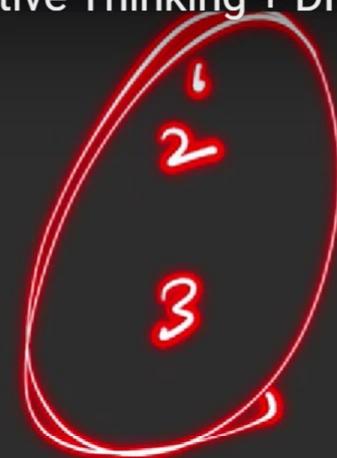
G-16. Number of Distinct Islands | Constructive Thinking + DFS | C++ | Java

1.90

Number of distinct islands



unique
dust
, slender



1:51 / 18:01



G-16. Number of Distinct Islands | Constructive Thinking + DFS | C++ | Java

1.90

Number of distinct islands



How do you

solve

Set



2:34 / 18:01



G-16. Number of Distinct Islands | Constructive Thinking + DFS | C++ | Java

1.90

Number of distinct islands



0

1

0

0

0

1

1

0

0

1

1

0

How do you

solve

Set



How many visit?



2:41 / 18:01



$$\begin{pmatrix} 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 1 \\ 3 & 1 & 1 & 0 & 1 \end{pmatrix}$$

$$\{(0,0), (0,1), (1,0)\}$$

$$\begin{pmatrix} (0,0) \\ | \\ (0,1) \end{pmatrix}$$

$$(1,0)$$

$$B^{\text{out}} \left(\begin{array}{c} 2,3 \\ T \end{array} \right) \left(\begin{array}{c} 2,4 \\ T \end{array} \right)$$

$$\begin{array}{cccc} 2,3 & \xrightarrow{(2,3)} & \xrightarrow{(0,0)} & \xrightarrow{(0,1)} \\ \xrightarrow{(2,4)} & \xrightarrow{(2,3)} & \xrightarrow{(0,1)} & \xrightarrow{(1,0)} \\ \left(\begin{array}{c} 2,3 \\ 3,3 \end{array} \right) & \xrightarrow{\quad} & \left(\begin{array}{c} 0,0 \\ 3,3 \end{array} \right) & \end{array}$$



TUF

α_0 1 1 0 1 1
1 1 0 0 0 0
2 0 0 0 1 1
3 1 1 0 1 0

$$\left\{ \begin{matrix} (0,0) & (0,1) & (1,0) \\ (0,0) & (0,1) & (1,0) \end{matrix} \right\}$$

$$2^3 - \begin{matrix} (2,3) \\ (2,3) \end{matrix} - \begin{matrix} (2,3) \\ (2,3) \end{matrix} - \begin{matrix} (2,3) \\ (2,3) \end{matrix} = \begin{matrix} (0,0) \\ (0,1) \\ (1,0) \end{matrix}$$
$$\begin{matrix} (2,1) \\ (3,3) \end{matrix} - \begin{matrix} (2,3) \\ (2,3) \end{matrix} - \begin{matrix} (2,3) \\ (2,3) \end{matrix} = \underline{\begin{matrix} (3,3) \end{matrix}}$$

$$\begin{matrix} (0,0) & (0,1) \\ | & | \\ (1,0) \end{matrix}$$

$$B^{\text{out}} \begin{matrix} (2,3) \\ T \end{matrix} \begin{matrix} (2,1) \\ T \end{matrix}$$

$$\begin{matrix}) \\ (3,3) \end{matrix}$$



G-16. Number of Distinct Islands | Constructive Thinking + DFS | C++ | Java



1.90

Number of distinct islands

How do you

solve

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

$$\left\{ \begin{array}{l} (0,0), (0,1), (1,0) \\ (0,0), (0,1), (1,0) \end{array} \right\}$$



89 of 132



5:54 / 18:01



► G-16. Number of Distinct Islands | Constructive Thinking + DFS | C++ | Java



1.90

Number of distinct islands

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

$$\overrightarrow{0,3} - \overrightarrow{0,3} = (0,0)$$

$$\overrightarrow{0,4} - \overrightarrow{0,3} = (0,1)$$

How do you

solve

$(0,3)$ $(0,4)$

$\{(0,0), (0,1), (1,0)\}$
 $\{(0,0), (0,1), (1,0)\}$
 $\{(0,0), (0,1)\}$

$(1,1)$
 $(3,0)$ $(3,1)$



6:37 / 18:01



G-16. Number of Distinct Islands | Constructive Thinking + DFS | C++ | Java

2 0 0 0 1
1.90
3 1 1 0 1 0

$\{(0,0), (0,1), (1,0)\}$
 $\{(0,0), (0,1), (1,0)\}$
 $\{(0,0), (0,1), (1,0)\}$
 $\{(0,0), (0,1), (1,0)\}$

1 1
 $(\underline{3}, 0)$ $(\underline{3}, 1)$

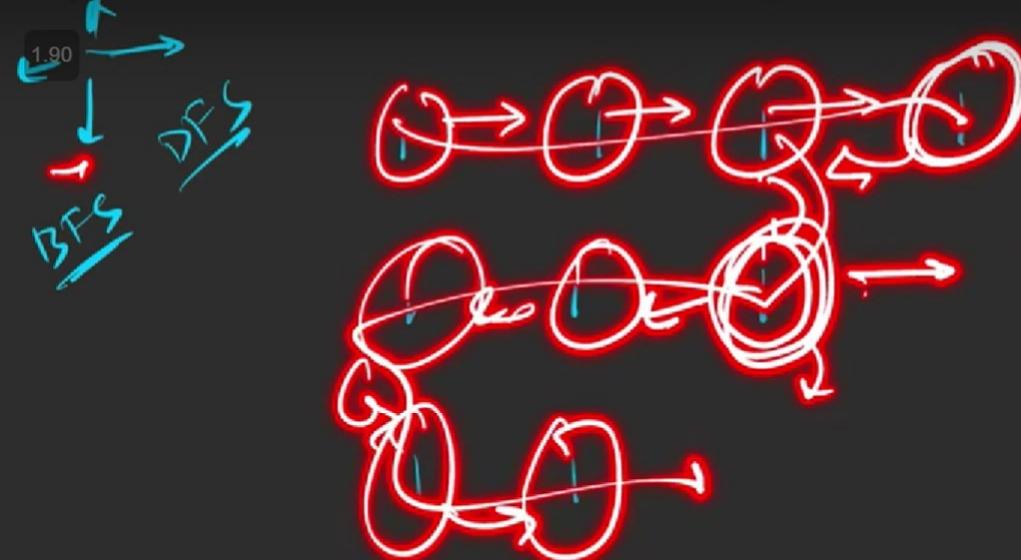
$$3_{1,0} - 3_{1,0} = 0, 0$$
$$3_{1,1} - (\underline{3}_{1,1}) = \underline{0}, 1$$



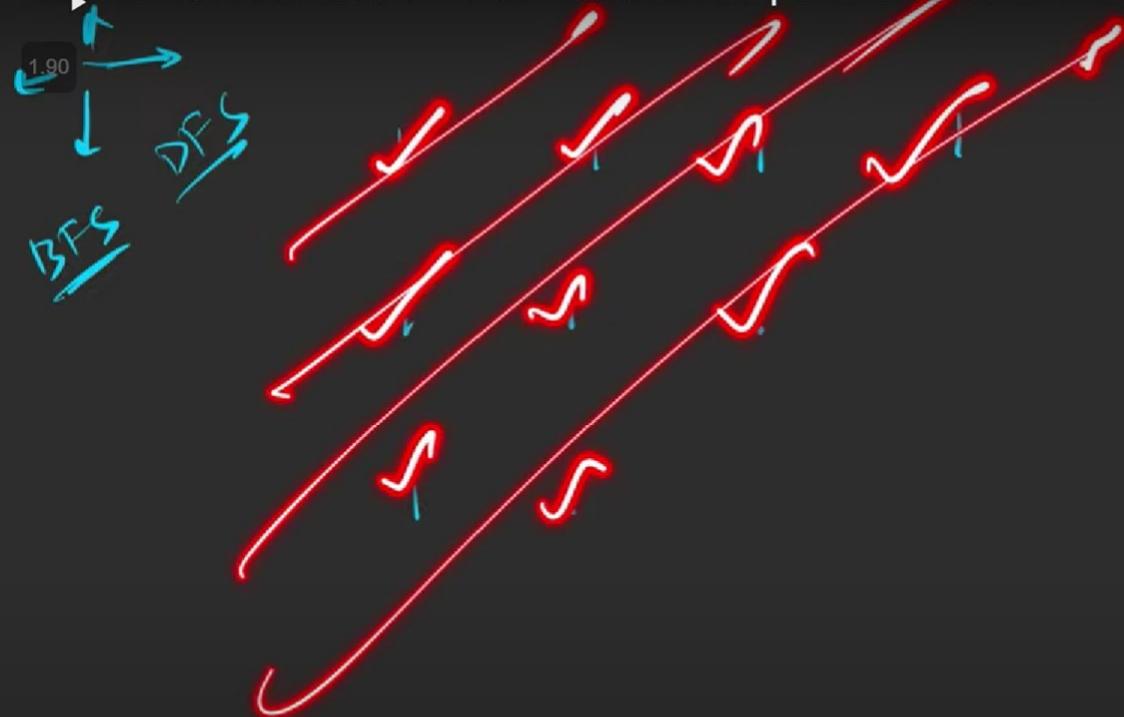
6:53 / 18:01



☰ G-16. Number of Distinct Islands | Constructive Thinking + DFS | C++ | Java

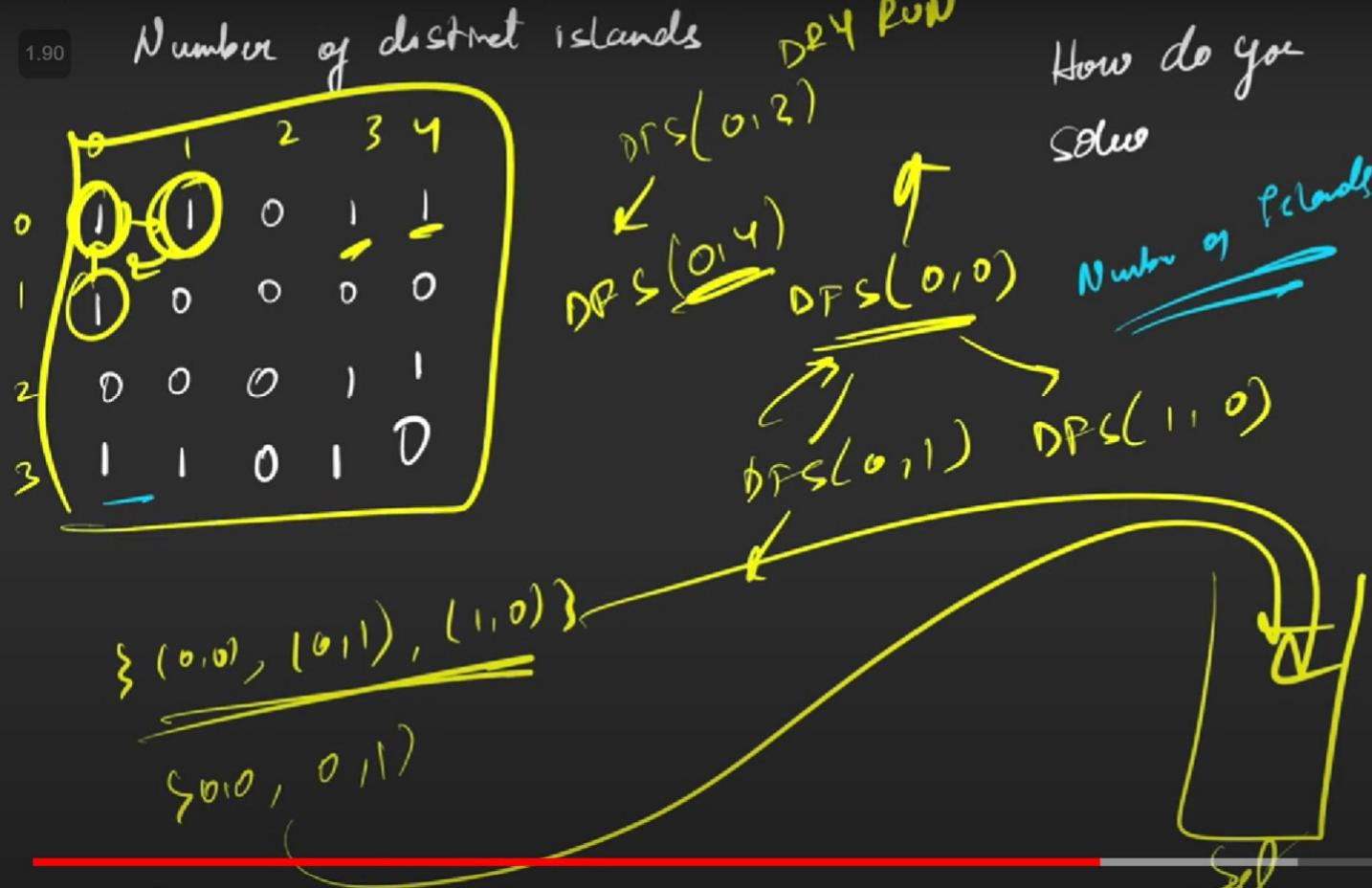


G-16. Number of Distinct Islands | Constructive Thinking + DFS | C++ | Java



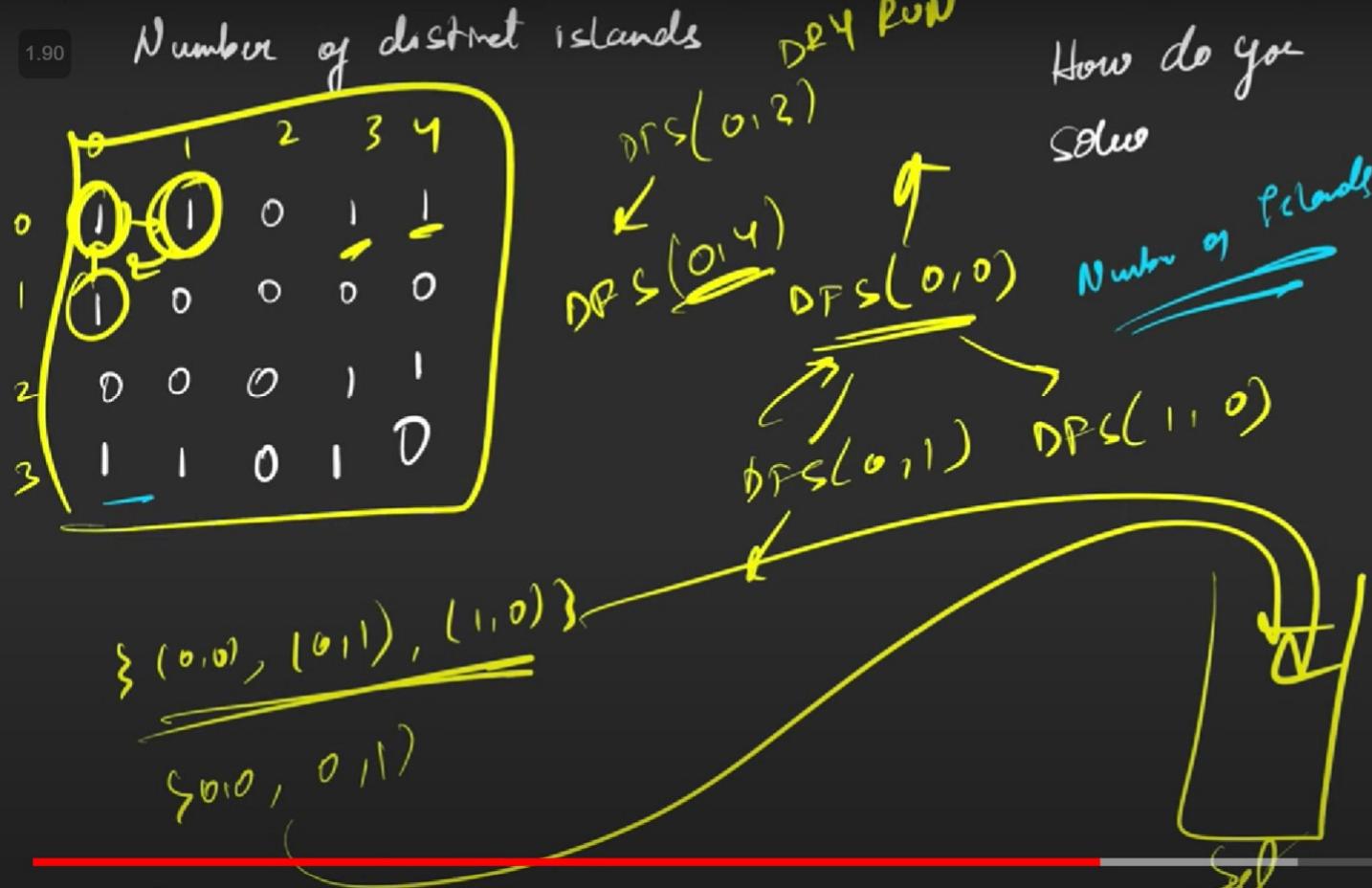
→ G-16. Number of Distinct Islands | Constructive Thinking + DFS | C++ | Java

1.90



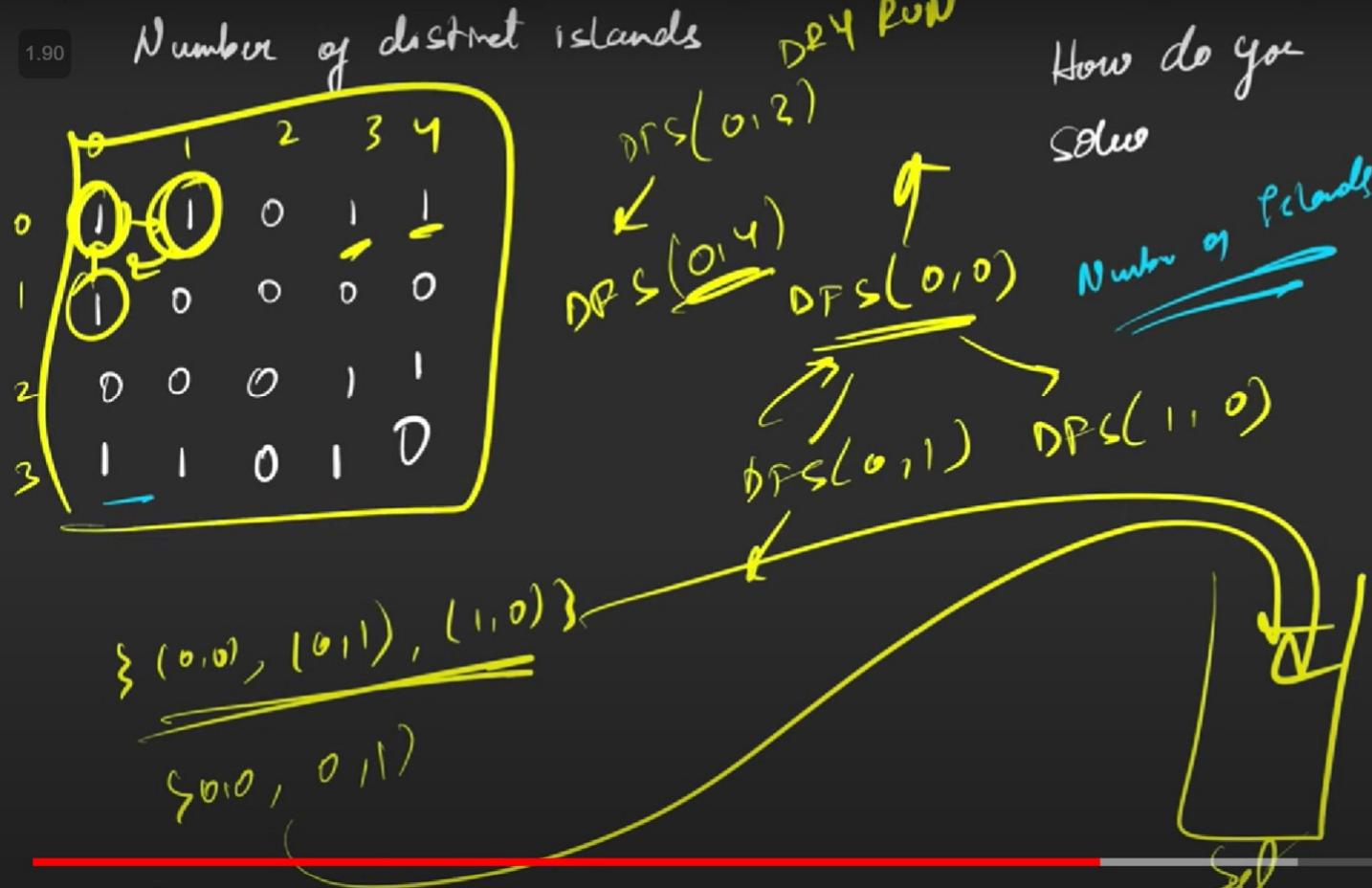
→ G-16. Number of Distinct Islands | Constructive Thinking + DFS | C++ | Java

1.90



→ G-16. Number of Distinct Islands | Constructive Thinking + DFS | C++ | Java

1.90



G-16. Number of Distinct Islands | Constructive Thinking + DFS | C++ | Java

Courses Get Hired Events ↗ POTD Practice 🔍 333

Problem Editorial Submissions Comments

Number of Distinct Islands

Summary Accuracy: 39.89% Submissions: 366 Points: 4

A boolean 2D matrix `grid` of size $n * m$. You have to find the number of distinct islands where a group of connected 1s (horizontally or vertically) forms an island. Two islands are considered to be distinct if and only if one island is not connected to another (not rotated or reflected).

Example 1:

```
Input:
[[0, 0, 1, 1, 0, 0, 0],
 [1, 1, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0, 0, 0]]
```

Output:

Compilation Error

```
prog.cpp: In member function void Solution::dfs(int, int,
std::vector<std::vector<int>> &grid, std::vector<std::vector<int>> &vis, std::set<std::pair<int, int>> &st):
prog.cpp:22:33: error: 'n' was not declared in this scope
        if(nrow >= 0 && nrow < n && ncol >= 0 && ncol < m
                           ^~~~~~
                           n
prog.cpp:22:58: error: 'm' was not declared in this scope
                           &ncol < m
                           ^~~~~~
                           m
```

1.90

C++ (g++ 5.4)

```
11 private:
12 void dfs(int row, int col, vector<vector<int>> &vis,
13           vector<vector<int>> &grid, vector<pair<int,int>> &vec, int row0, int
14           vis[row][col] = 1;
15           vec.push_back({row - row0, col - col0});
```

```
16     }
17     int delrow[] = {-1, 0, +1, 0};
18     int delcol[] = {0, -1, 0, +1};
19     for(int i = 0; i < 4; i++) {
20         int nrow = row + delrow[i];
21         int ncol = col + delcol[i];
22         if(nrow >= 0 && nrow < n && ncol >= 0 && ncol < m
23             && !vis[nrow][ncol] && grid[nrow][ncol] == 1) {
24             dfs(nrow, ncol, vis, grid, vec, row0, col0);
25         }
26     }
27 }
public:
int countDistinctIslands(vector<vector<int>> &grid) {
    int n = grid.size();
    int m = grid[0].size();
    vector<vector<int>> vis(n, vector<int>(m, 0));
    set<vector<pair<int,int>> st;
    for(int i = 0; i < n; i++) {
        for(int j = 0; j < m; j++) {
            if(!vis[i][j] && grid[i][j] == 1) {
                vector<pair<int,int>> vec;
                dfs(i, j, grid, vis, vec, i, j);
                st.insert(vec);
            }
        }
    }
    return st.size();
}
```

Compile & Run Submit

CC Settings

16:04 / 18:01

G-16. Number of Distinct Islands | Constructive Thinking + DFS | C++ | Java

Courses Get Hired Events ↗ PTD Practice 🔍 333

1.90 Problem Editorial Submissions Comments C++ (g++ 5.4)

Number of Distinct Islands

Sum Accuracy: 39.89% Submissions: 366 Points: 4

A boolean 2D matrix `grid` of size $n * m$. You have to find the number of distinct islands where a group of connected 1s (horizontally or vertically) forms an island. Two islands are considered to be distinct if and only if one island is not connected to another (not rotated or reflected).

Example 1:

```
Input:
[[0, 0, 1, 1, 0, 0, 0],
 [1, 1, 0, 0, 0, 0, 0]]
```

Output: 2

Compilation Error

```
prog.cpp: In member function void Solution::dfs(int, int,
std::vector<std::vector<int>> &grid, std::vector<std::vector<int>> &vis, std::set<std::pair<int, int>> &st):
prog.cpp:22:33: error: 'n' was not declared in this scope
        if(nrow >= 0 && nrow < n && ncol >= 0 && ncol < m
                           ^~~~~~
prog.cpp:22:58: error: 'm' was not declared in this scope
        if(nrow >= 0 && nrow < n && ncol >= 0 && ncol < m
                           ^~~~~~
```

16:08 / 18:01



G-16. Number of Distinct Islands | Constructive Thinking + DFS | C++ | Java

Courses Get Hired Events POTD Practice ↗ 337 ↕ 🔍

1/60 Problem Editorial Submissions Comments C++ (g++ 5.4) ↗

Number of Distinct Islands

Medium Accuracy: 39.89% Submissions: 366 Points: 4

Given a boolean 2D matrix `grid` of size $n * m$. You have to find the number of distinct islands where a group of connected 1s (horizontally or vertically) forms an island. Two islands are considered to be distinct if and only if one island is equal to another (not rotated or reflected).

Example 1:

Input:
`grid[][] = {{1, 1, 0, 0, 0},`

Output Window

Compilation Results Custom Input

Problem Solved Successfully ✓

Test Cases Passed: 132 / 132 Total Points Scored: 4 / 4

Your Total Score: 337 Total Time Taken: 0.09

17 int m = grid[0].size();
18 int delrow[] = {-1, 0, +1, 0};
19 int delcol[] = {0, -1, 0, +1};
20 for(int i = 0; i < 4; i++) {
21 int nrow = row + delrow[i];
22 int ncol = col + delcol[i];
23 if(nrow >= 0 && nrow < n && ncol >= 0 && ncol < m
24 && !vis[nrow][ncol] && grid[nrow][ncol] == 1) {
25 dfs(nrow, ncol, vis, grid, vec, row0, col0);
26 }
27 }
28 }
public:
30 int countDistinctIslands(vector<vector<int>>& grid) {
31 int n = grid.size();
32 int m = grid[0].size();
33 vector<vector<int>> vis(n, vector<int>(m, 0));
34 set<vector<pair<int, int>> st;
35 for(int i = 0; i < n; i++) {
36 for(int j = 0; j < m; j++) {
37 if(!vis[i][j] && grid[i][j] == 1) {
38 vector<pair<int, int>> vec;
39 // n x m x 4
40 dfs(i, j, vis, grid, vec, i, j);
41 st.insert(vec);
42 }
43 }
44 }
45 // n x m x log (n X m) + (n x m x 4)
46 return st.size();
47 }
48 };
49 // } Driver Code Ends

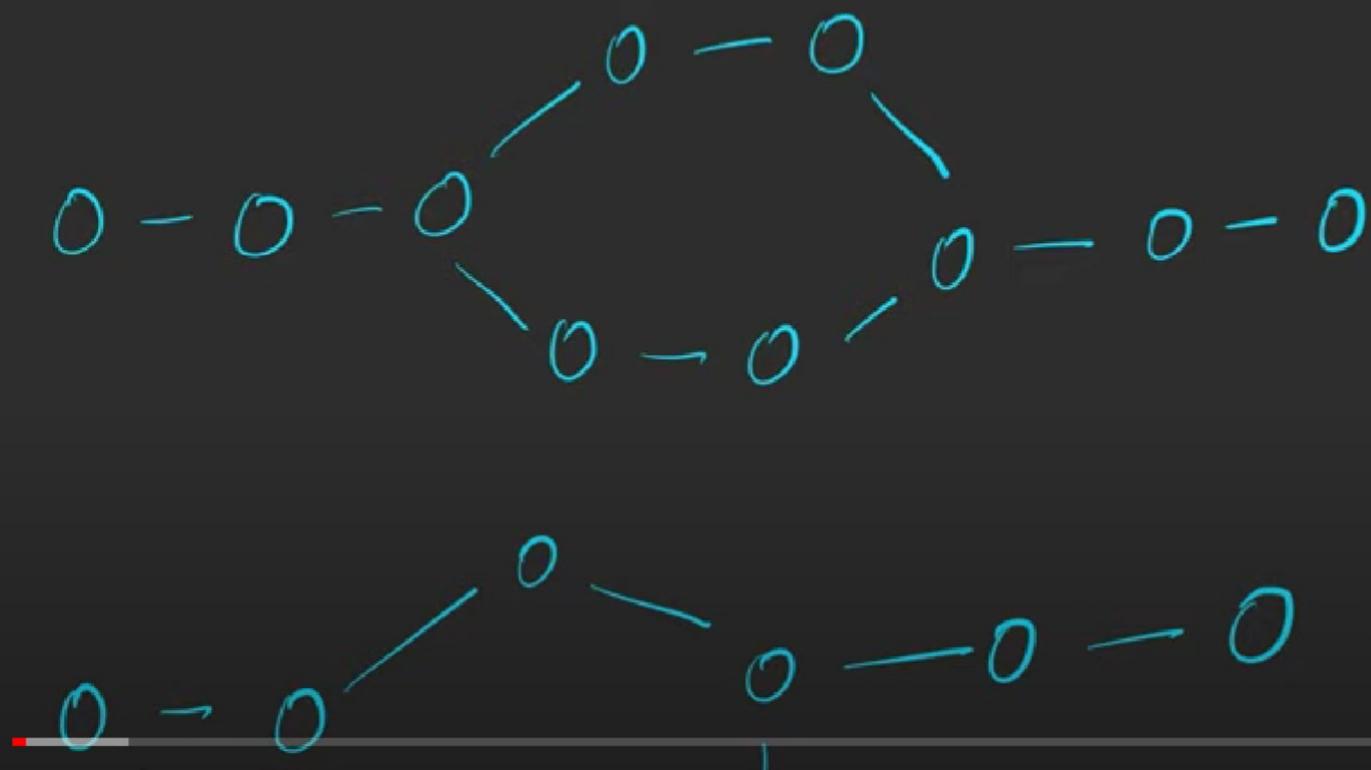
CC ⚙️ 🎧 17:32 / 18:01



G-17. Bipartite Graph | BFS | C++ | Java

1.75

Bipartite graph



0:08 / 18:28

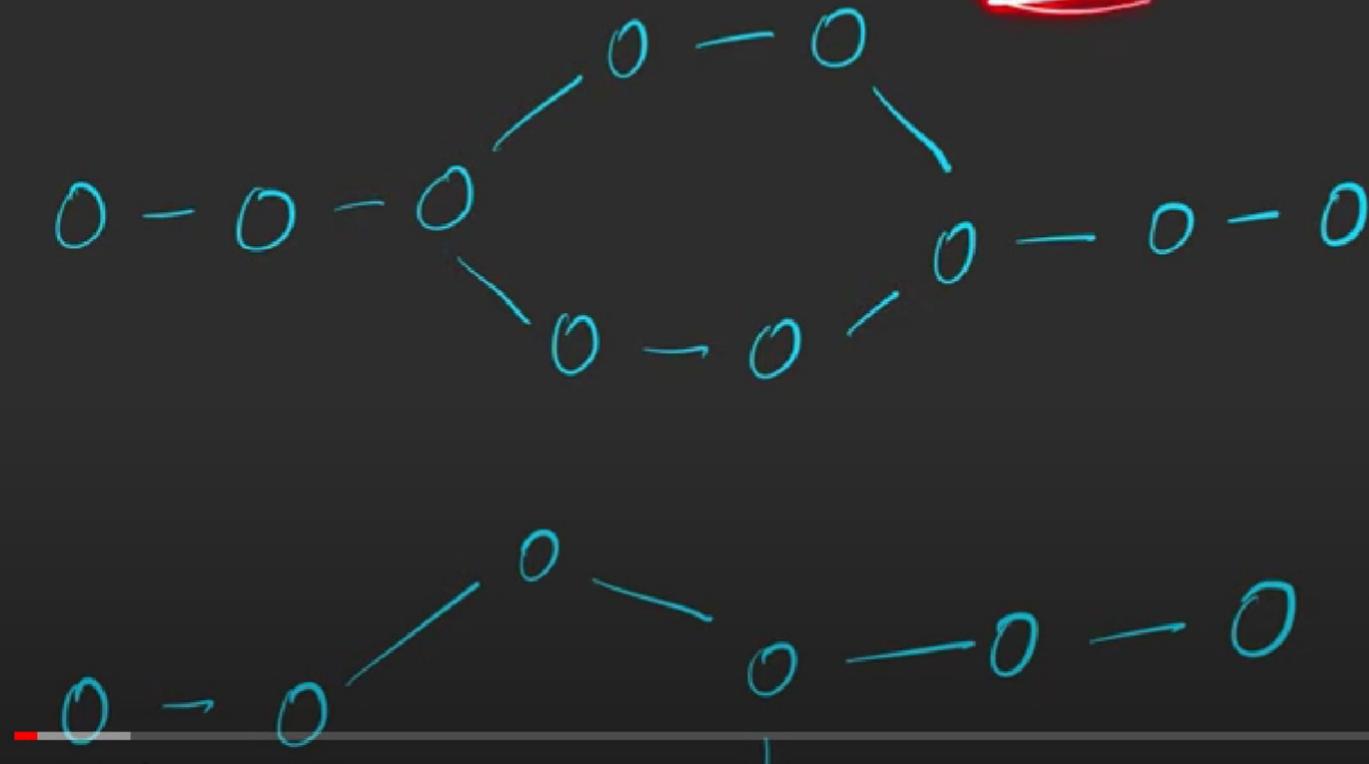


G-17. Bipartite Graph | BFS | C++ | Java

1.75

Bipartite graph

BFS



0:13 / 18:28

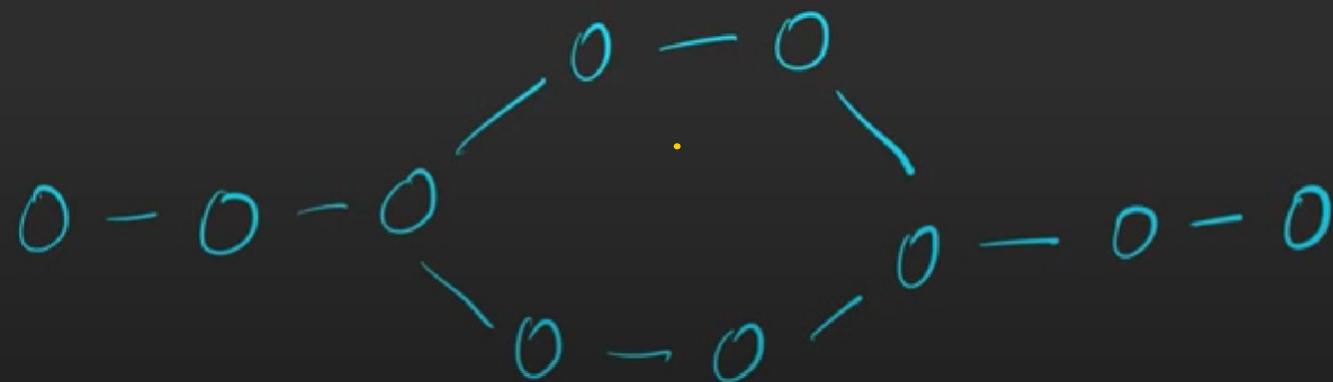


G-17. Bipartite Graph | BFS | C++ | Java

1.75

color the graph
with 2 colors such that
no adjacent nodes have same color.

Bipartite graph



G-17. Bipartite Graph | BFS | C++ | Java

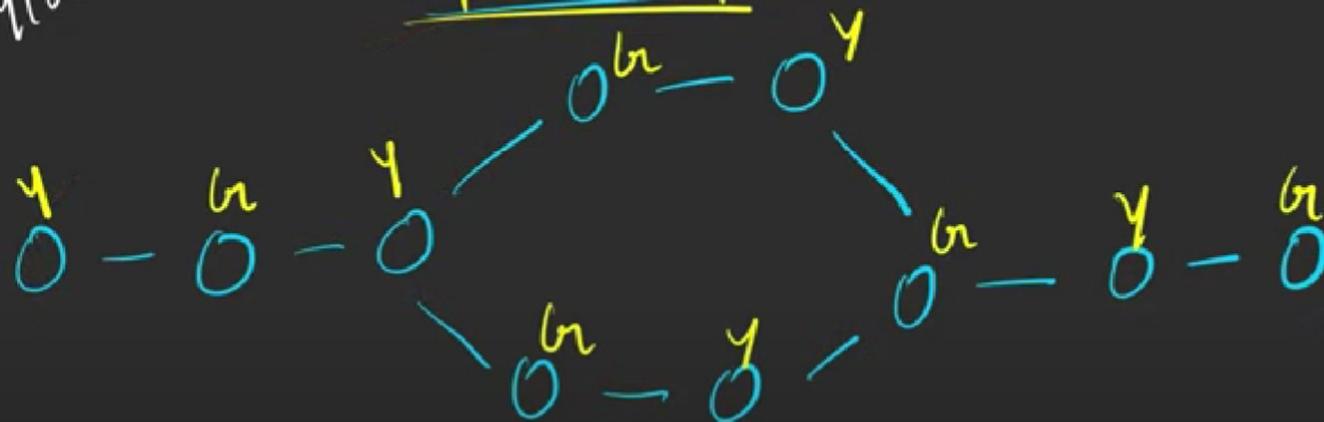
Press Esc to exit full screen

1.75

With 2 where
no adjacent nodes color.

Y/B

Bipartite graph



1.75

with 2 where no adjacent nodes have same color.

Bipartite graph

The diagram illustrates a bipartite graph structure. It consists of two distinct sets of nodes: one set colored blue and the other set colored yellow. The blue nodes are arranged in two horizontal rows. The top row contains three nodes, and the bottom row contains four nodes. The yellow nodes are also arranged in two horizontal rows. The top row contains three nodes, and the bottom row contains four nodes. Edges connect nodes between the two sets: specifically, each node in the top blue row is connected to all nodes in the top yellow row, and each node in the bottom blue row is connected to all nodes in the bottom yellow row. This results in a complete bipartite graph structure where every node in one set is connected to every node in the other set.

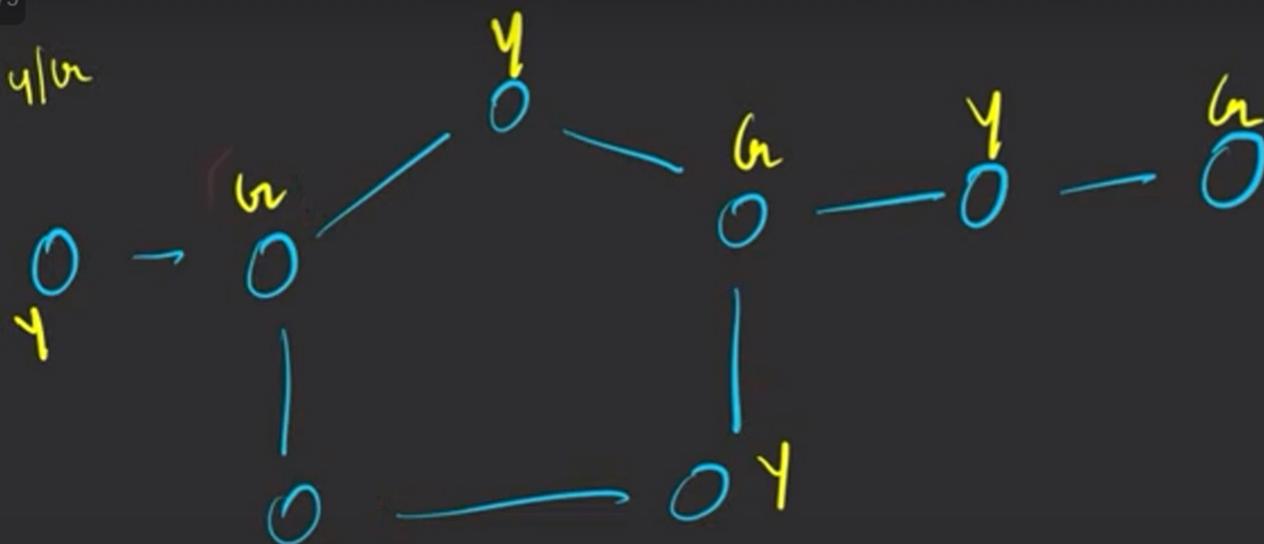
Bipartite

TUG



G-17. Bipartite Graph | BFS | C++ | Java

1.75



2:42 / 18:28



G-17. Bipartite Graph | BFS | C++ | Java

1.75



Bipartite graph

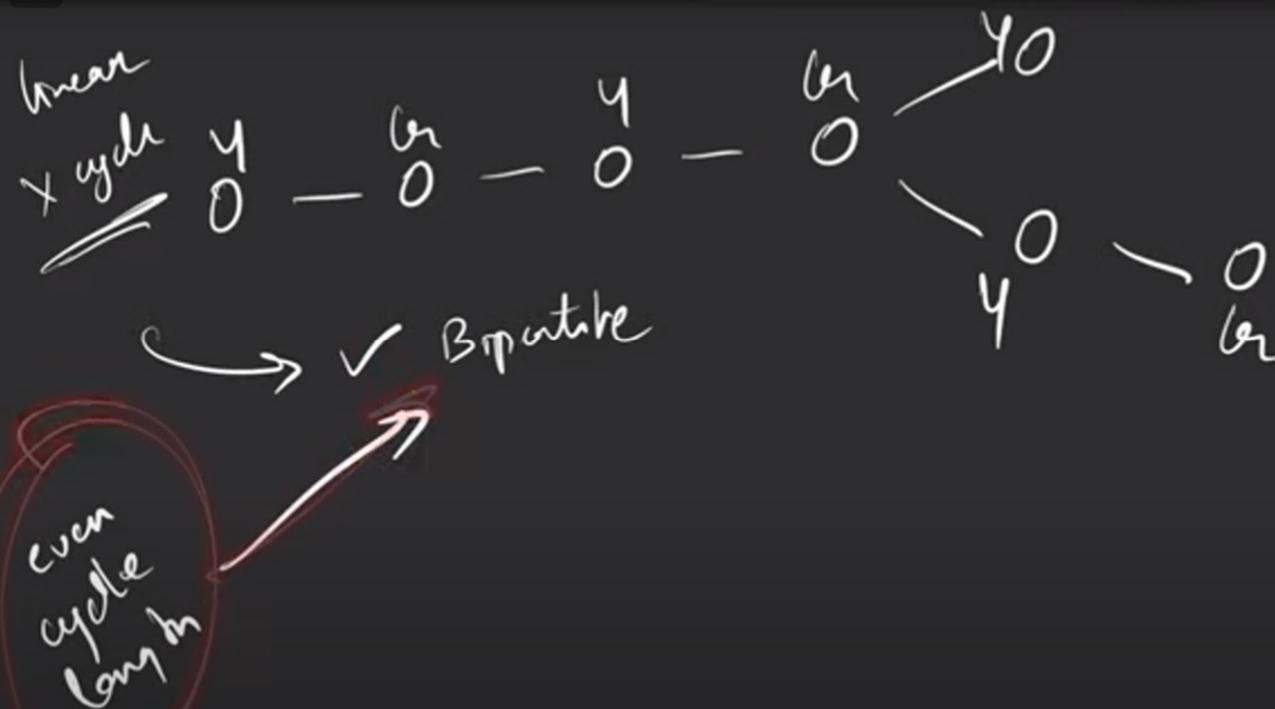


3:44 / 18:28



G-17. Bipartite Graph | BFS | C++ | Java

1.75

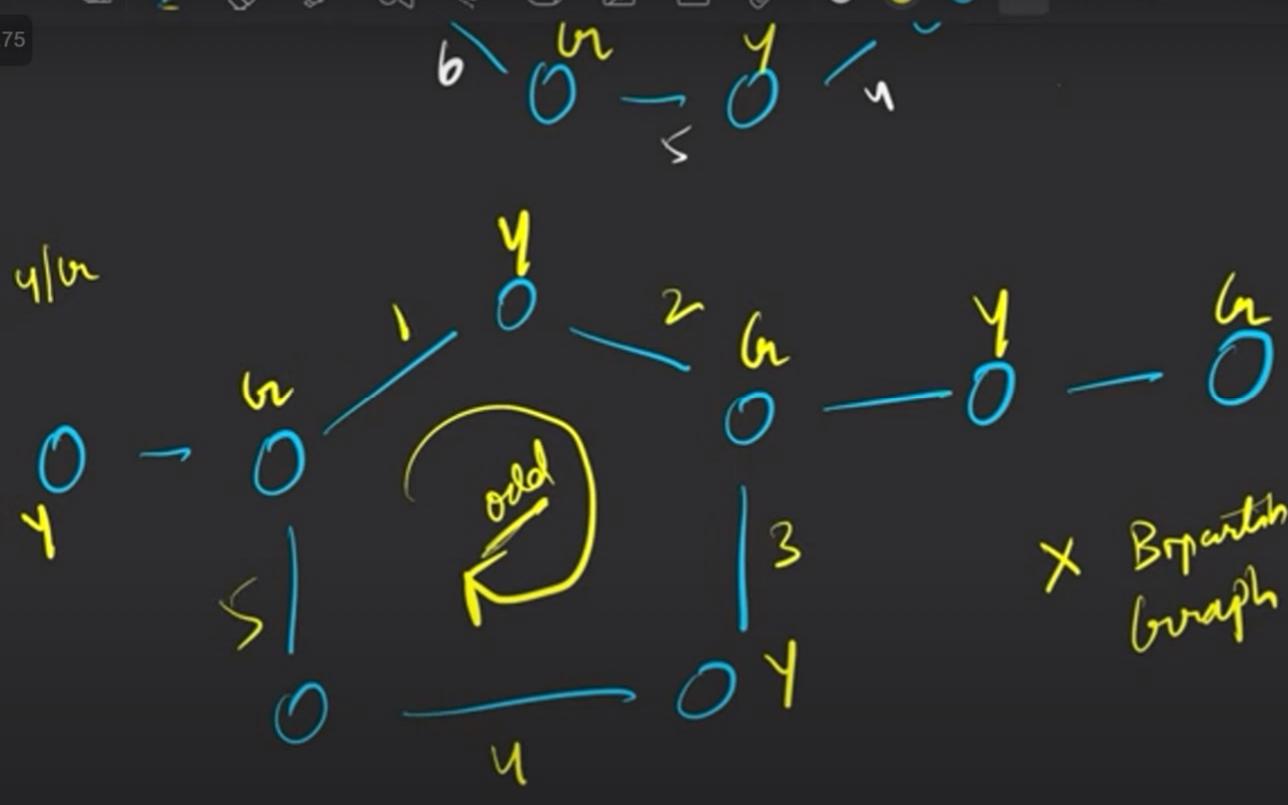


4:01 / 18:28



G-17. Bipartite Graph | BFS | C++ | Java

1.75



4:15 / 18:28



G-17. Bipartite Graph | BFS | C++ | Java

1.75

even
cycle
length

odd length
cycle

X Bipartite



4:37 / 18:28



► G-17. Bipartite Graph | BFS | C++ | Java

1.75

$1 \rightarrow \{2\}$

$2 \rightarrow \{1, 3, 6\}$

$3 \rightarrow \{2, 4\}$

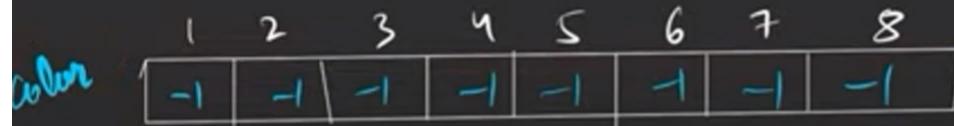
$4 \rightarrow \{3, 5, 7\}$

$5 \rightarrow \{4, 6\}$

$6 \rightarrow \{2, 5\}$

$7 \rightarrow \{4, 8\}$

$8 \rightarrow \{7\}$

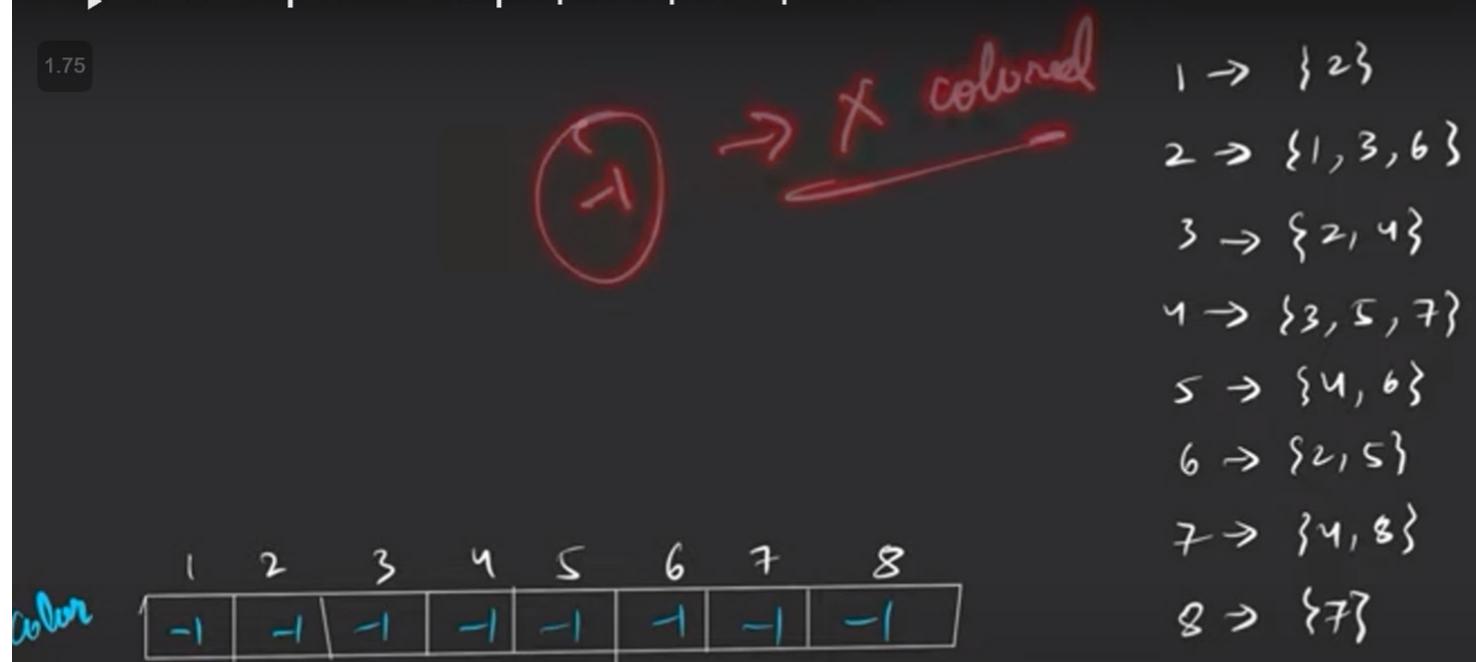


4:59 / 18:28



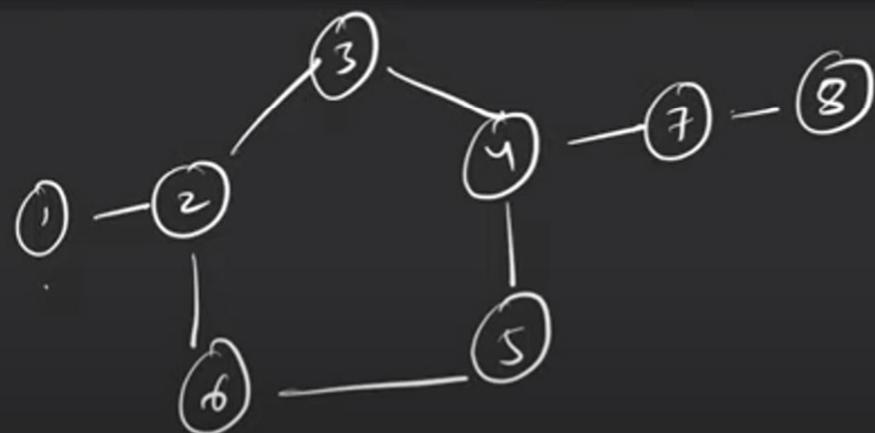
► G-17. Bipartite Graph | BFS | C++ | Java

1.75



G-17. Bipartite Graph | BFS | C++ | Java

1.75



0 8 1



5:14 / 18:28

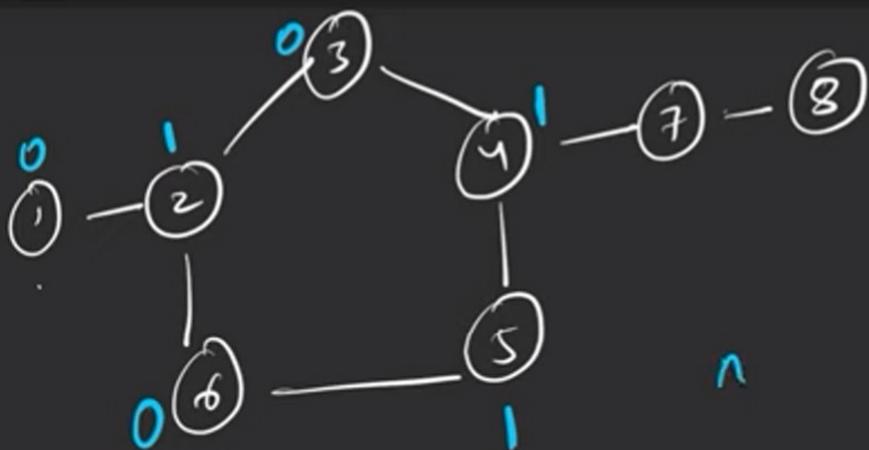


1 → { 2 }



→ G-17. Bipartite Graph | BFS | C++ | Java

0x1



0

1 → {2}

2 → {1, 3, 6}

3 → {2, 4}



8:54 / 18:28

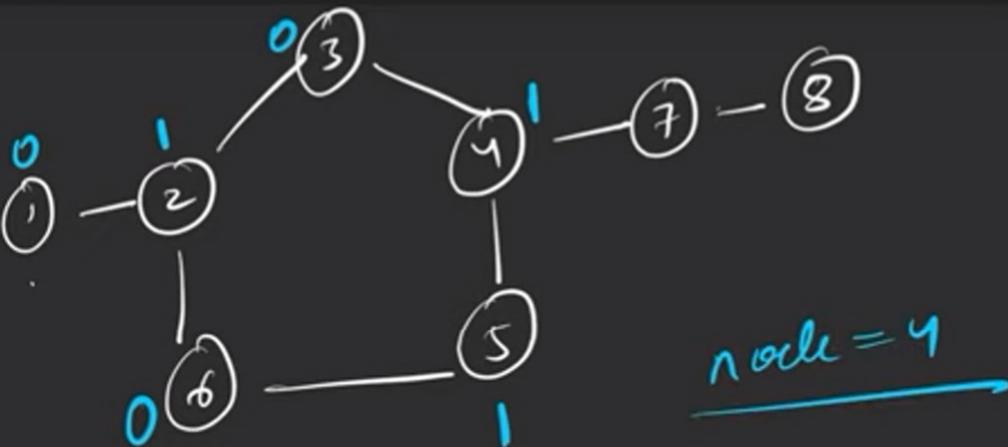


1 → {2, 4}

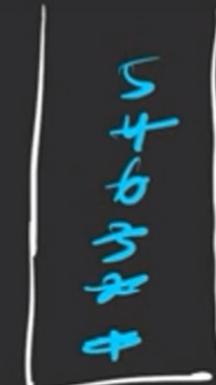


G-17. Bipartite Graph | BFS | C++ | Java

Ques



node = 9



0

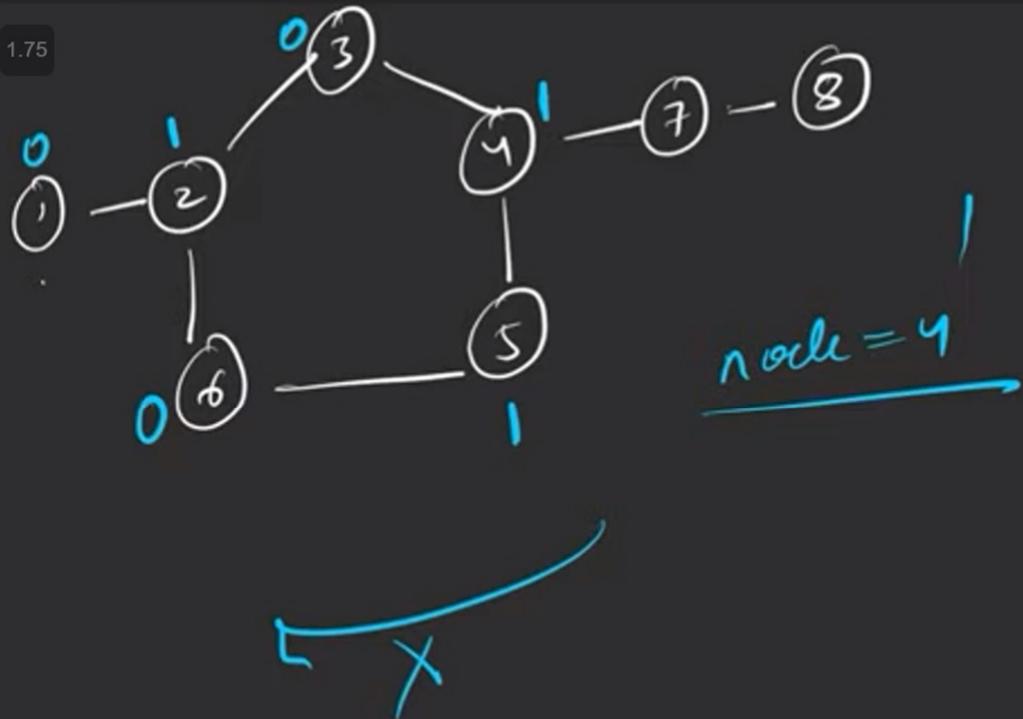
1 → {2}

2 → {1, 3, 6}

3 → {2, 4}



1.75



$$1 \rightarrow \{2\}$$

$$2 \rightarrow \{1, 3, 6\}$$

$$3 \rightarrow \{2, 4\}$$

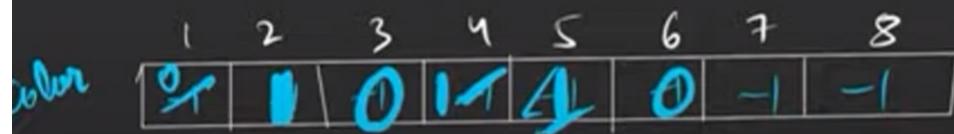
$$4 \rightarrow \{3, 5, 7\}$$

$$5 \rightarrow \{4, 6\}$$



G-17. Bipartite Graph | BFS | C++ | Java

1.75



1 → {2}
2 → {1, 3, 6}
3 → {2, 4}
4 → {3, 5, 7}
5 → {4, 6}
6 → {2, 5}
7 → {4, 8}
8 → {7}



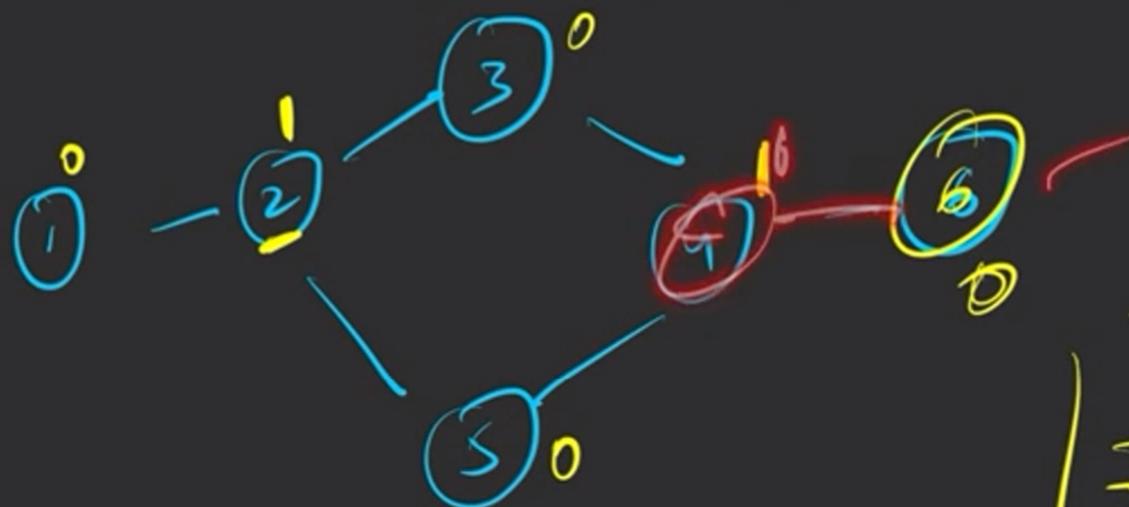
106 of 132



9:59 / 18:28



1.75



hole = 6

107 of 132



G-17. Bipartite Graph | BFS | C++ | Java

Practice

341

175 Problem Editorial Submissions Comments

Input:

Output: 0

Explanation: The given graph cannot be colored in two colors such that color of adjacent vertices differs.

Your Task:

You don't need to read or print anything. Your task is to complete the function `isBipartite()` which takes `V` denoting no. of vertices and `adj` denoting adjacency list of the graph and returns a boolean value true if the graph is bipartite otherwise returns false.

Expected Time Complexity: $O(V + E)$

Expected Space Complexity: $O(V)$

Constraints:

$1 \leq V, E \leq 10^5$

```
C++ (g++ 5.4) *
1 // Driver Code Ends
6 class Solution {
7 public:
8     bool isBipartite(int V, vector<int>adj[]){
9         queue<int> q;
10        q.push(0);
11        int color[V];
12        for(int i = 0;i<V;i++) color[i] = -1;
13        color[0] = 0;
14        while(!q.empty()){
15            int node = q.front();
16            q.pop();
17
18            for(auto it : adj[node]){
19                // if the adjacent node is yet not colored
20                // you will give the opposite color of the node
21                if(color[it] == -1) {
22
23                    color[it] = !color[node];
24                    q.push(it);
25                }
26                // is the adjacent guy having the same color
27                // someone did color it on some other path
28                else if(color[it] == color[node]) {
29                    return false;
30                }
31            }
32        }
33        return true;
34    }
35 };
36 */
37 // Driver Code Ends
```

15:37 / 18:28

Compile & Run

CC

Settings

#

G-17. Bipartite Graph | BFS | C++ | Java

Practice

341

175 Problem Editorial Submissions Comments

```
private boolean check(int start, int V, ArrayList<ArrayList<Integer>>adj, int color[]) {  
    Queue<Integer> q = new LinkedList<Integer>();  
    q.add(start);  
    color[start] = 0;  
    while(!q.isEmpty()) {  
        int node = q.peek();  
        q.remove();  
  
        for(int it : adj.get(node)) {  
            // if the adjacent node is yet not colored  
            // you will give the opposite color of the node  
            if(color[it] == -1) {  
                color[it] = 1 - color[node];  
                q.add(it);  
            }  
            // is the adjacent guy having the same color  
            // someone did color it on some other path  
            else if(color[it] == color[node]) {  
                return false;  
            }  
        }  
    }  
    return true;  
}  
  
public boolean isBipartite(int V, ArrayList<ArrayList<Integer>>adj)  
{  
    int color[] = new int[V];  
    for(int i = 0; i < V; i++) color[i] = -1;  
  
    for(int i = 0; i < V; i++) {  
        if(color[i] == -1) {  
            if(check(i, V, adj, color) == false)  
                return false;  
        }  
    }  
    return true;  
}
```

1 // Driver Code Ends
2 class Solution {
3 // colors a component
4 private:
5 bool check(int start, int V, vector<int>adj[], int col[]) {
6 queue<int> q;
7 q.push(start);
8 color[start] = 0;
9 while(!q.empty()) {
10 int node = q.front();
11 q.pop();

12 for(auto it : adj[node]) {
13 // if the adjacent node is yet not colored
14 // you will give the opposite color of the node
15 if(color[it] == -1) {
16 color[it] = !color[node];
17 q.push(it);
18 }
19 // is the adjacent guy having the same color
20 // someone did color it on some other path
21 else if(color[it] == col[node]) {
22 return false;
23 }
24 }
25 }
26 return true;
27 }
28 public:
29 bool isBipartite(int V, vector<int>adj[]){
30 int color[V];
31 for(int i = 0; i < V; i++) color[i] = -1;
32
33 for(int i = 0; i < V; i++) {
34 if(color[i] == -1) {
35 if(check(i, V, adj, color) == false) {
36 return false;
37 }
38 }
39 }
40 }
41 }
42 }

17:29 / 18:28

Compile & Run

CC

Settings

#



G-17. Bipartite Graph | BFS | C++ | Java

Java Practice

345



```
private boolean check(int start, int V,
ArrayList<ArrayList<Integer>>adj, int color[]) {
    Queue<Integer> q = new LinkedList<Integer>();
    q.add(start);
    color[start] = 0;
    while(!q.isEmpty()) {
        int node = q.peek();
        q.remove();

        for(int it : adj.get(node)) {
            // if the adjacent node is yet not colored
            // you will give the opposite color of the node
            if(color[it] == -1) {

                color[it] = 1 - color[node];
                q.add(it);
            }
            // is the adjacent guy having the same color
            // someone did color it on some other path
            else if(color[it] == color[node]) {
                return false;
            }
        }
    }
    return true;
}

public boolean isBipartite(int V, ArrayList<ArrayList<Integer>>adj) {
    int color[] = new int[V];
    for(int i = 0;i<V;i++) color[i] = -1;

    for(int i = 0;i<V;i++) {
        if(color[i] == -1) {
            if(check(i, V, adj, color) == false) {
                return false;
            }
        }
    }
    return true;
}

private void check(int node, int V,
ArrayList<ArrayList<Integer>>adj, int color[], Queue<Integer> q) {
    q.pop();

    for(auto it : adj[node]) {
        // if the adjacent node is yet not colored
        // you will give the opposite color of the node
        if(color[it] == -1) {

            color[it] = !color[node];
            q.push(it);
        }
        // is the adjacent guy having the same color
        // someone did color it on some other path
        else if(color[it] == color[node]) {
            return false;
        }
    }
    return true;
}

public:
    bool isBipartite(int V, vector<int>adj[]){
        int color[V];
        for(int i = 0;i<V;i++) color[i] = -1;

        for(int i = 0;i<V;i++) {
            if(color[i] == -1) {
                if(check(i, V, adj, color) == false) {
                    return false;
                }
            }
        }
        return true;
    }
};

// } Driver Code Ends
```

|◀ ▶▶| 🔊 17:35 / 18:28

884 ReactJS Tutorial - 6 | Striver's SDE Sheet – Top | (892) G-17. Bipartite Graph | Number of Distinct Islands | YouTube | (892) #kamalsadanah - Y | +

Compile time poly... rohit952513 - LeetC... HTML ASCII Refere... vector - C++ Refere... SDE Off-campus Pl... Striver DP Series : D... Leetcode 75 Questi... Resize image in cm,...

practice.geeksforgeeks.org/problems/number-of-distinct-islands/1?utm_source=youtube&utm_medium=referral&utm_campaign=youtube-geeksforgeeks&utm_content=Number%20of%20Distinct%20Islands

Problems Courses Geek-O-Lympics Events POTD

84 ↑

Problems Courses Geek-O-Lympics Events POTD

Problem Editorial Submissions Comments

Number of Distinct Islands

Medium Accuracy: 62.29% Submissions: 40K+ Points: 4

Sharpen up your programming skills, participate in coding contests & explore high-paying jobs

Given a boolean 2D matrix **grid** of size $n * m$. You have to find the number of distinct islands where a group of connected 1s (horizontally or vertically) forms an island. Two islands are considered to be distinct if and only if one island is not equal to another (not rotated or reflected).

Example 1:

Input:

```
grid[][] = {{1, 1, 0, 0, 0},  
           {1, 1, 0, 0, 0},  
           {0, 0, 0, 1, 1},  
           {0, 0, 1, 1, 0},  
           {0, 0, 1, 1, 0}}
```

C++ (g++ 5.4) Average Time: 20m Your Time: 18m 56s

```
class Solution {
public:
    int n,m;
    void dfs(int row,int col,vector<vector<int>>& grid,vector<pair<int,int>>& vec,vector<vector<int>>& vis){
        vis[row][col]=1;
        vec.push_back({row-row0,col-col0});
        int drow[]={-1,0,1,0};
        int dcol[]={0,-1,0,1};
        for(int i=0;i<4;i++){
            int nrow=row+drow[i];
            int ncol=col+dcol[i];
            if(nrow<0 || nrow>n || ncol<0 || ncol>m || !vis[nrow][ncol] && grid[nrow][ncol]==1){
                dfs(nrow,ncol,grid,vec,vis,row0,col0);
            }
        }
    }
    int countDistinctIslands(vector<vector<int>>& grid) {
        n=grid.size();
        m=grid[0].size();
        //int count=0;
        set<vector<pair<int,int>>>st; //for storing unique islands shape
        vector<vector<int>>vis(n,vector<int>(m,0)); //for marking visited
        for(int i=0;i<n;i++){
            for(int j=0;j<m;j++){
                if(grid[i][j]==1 && !vis[i][j]){
                    dfs(i,j,grid,vec,vis,i,j);
                    st.insert(vec);
                    vec.clear();
                }
            }
        }
        return st.size();
    }
};
```

Custom Input Compile & Run Submit

33°C Mostly cloudy

Search

3:32 PM 6/30/2023

884 ReactJS Tutorial - 6 Striver's SDE Sheet – Top (892) G-17. Bipartite Graph Number of Distinct Islands (892) YouTube (892) #kamalsadanah - Y +

Compile time poly... rohit952513 - LeetC... HTML ASCII Refere... vector - C++ Refere... SDE Off-campus Pl... Striver DP Series : D... Leetcode 75 Questi... Resize image in cm,...

practice.geeksforgeeks.org/problems/number-of-distinct-islands/1?utm_source=youtube&utm_medium=referral&utm_campaign=youtube&utm_content=Number%20of%20Distinct%20Islands

Problems Courses Geek-O-Lympics Events POTD 84 ↕ 🔍 📡

☰ </> Problem Editorial Submissions Comments

Number of Distinct Islands

Medium Accuracy: 62.29% Submissions: 40K+ Points: 4

Sharpen up your programming skills, participate in coding contests & explore high-paying jobs

Given a boolean 2D matrix **grid** of size **n * m**. You have to find the number of distinct islands where a group of connected 1s (horizontally or vertically) forms an island. Two islands are considered to be distinct if and only if one island is not equal to another (not rotated or reflected).

Example 1:

Input:

```
grid[][] = {{1, 1, 0, 0, 0},  
           {1, 1, 0, 0, 0},  
           {0, 0, 0, 1, 1},  
           {0, 0, 1, 1, 0},  
           {0, 0, 1, 1, 0}}
```

C++ (g++ 5.4) Average Time: 20m Your Time: 18m 56s

```
24 }  
25 }  
26 }  
27  
28 int countDistinctIslands(vector<vector<int>>& grid) {  
29     n=grid.size();  
30     m=grid[0].size();  
31     //int count=0;  
32     set<vector<pair<int,int>>>st; //for storing unique islands shape  
33     vector<vector<int>>vis(n,vector<int>(m,0)); //for marking visited  
34     for(int i=0;i<n;i++){  
35         for(int j=0;j<m;j++){  
36             vector<pair<int,int>>vec; //for storing all shape of valid islands |  
37             if(vis[i][j]!=1 && grid[i][j]==1){  
38                 dfs(i,j,grid,vec,vis,i,j);  
39                 st.insert(vec);  
40             }  
41         }  
42     }  
43     return st.size();  
44 }  
45 }  
46 };  
47 }  
48 // } Driver Code Ends
```

Custom Input Compile & Run Submit

33°C Mostly cloudy

Search

3:32 PM 6/30/2023

ReactJS Tutorial - 6 | Striver's SDE Sheet - Top | (892) G-17. Bipartite Graph | Practice | (892) YouTube | (892) #kamalsadanah - Y | + | - | X

practice.geeksforgeeks.org/problems/bipartite-graph/1?utm_source=youtube&utm_medium=referral

Compile time poly... rohit952513 - LeetCode HTML ASCII Reference vector - C++ Reference SDE Off-campus Placements Striver DP Series : Dynamic Programming Leetcode 75 Questions Resize image in cm,...

Problems Courses Geek-O-Lympics Events POTD

GeeksforGeeks

Bipartite Graph

Medium Accuracy: 31.25% Submissions: 124K+ Points: 4

Sharpen up your programming skills, participate in coding contests & explore high-paying jobs

Given an adjacency list of a graph `adj` of V no. of vertices having 0 based index. Check whether the graph is bipartite or not.

Example 1:

Input:

```
C++ (g++ 5.4)
1 // } Driver Code Ends
2 class Solution {
3 public:
4     bool bfs(int start, vector<int> adj[], int color[]){
5         queue<int> q;
6         color[start]=0;
7         q.push(start);
8         while(!q.empty()){
9             int node=q.front();
10            q.pop();
11            for(auto &v:adj[node]){
12                if(color[v]==-1){           //if the node is not colored
13                    color[v]=!color[node];
14                    q.push(v);
15                }
16                else if(color[v]==color[node]){
17                    return false;
18                }
19            }
20        }
21        return true;
22    }
23    bool isBipartite(int v, vector<int> adj[]){
24        // Code here
25        int color[v]; //act as visited array , considering two color 0 and 1 , in
26        for(int i=0;i<v;i++){
27            color[i]=-1;
28        }
29        if(bfs(0,adj,color)==true)
30            return true;
31        else
32            return false;
33    }
}
```

Custom Input Compile & Run Submit

33°C Mostly cloudy

Search

b Windows File Explorer Spotify Microsoft Edge VS Code WhatsApp WPS Office Telegram 3:47 PM 6/30/2023 ENG IN

ReactJS Tutorial - 6 | Striver's SDE Sheet - Top | (892) G-17. Bipartite Graph | Practice | (892) YouTube | (892) #kamalsadanah - Y | + | - | X

practice.geeksforgeeks.org/problems/bipartite-graph/1?utm_source=youtube&utm_medium=referral

Compile time poly... rohit952513 - LeetCode HTML ASCII Reference vector - C++ Reference SDE Off-campus Placements Striver DP Series : Dynamic Programming Leetcode 75 Questions Resize image in cm,...

Problems Courses Geek-O-Lympics Events POTD

Bipartite Graph

Medium Accuracy: 31.25% Submissions: 124K+ Points: 4

Sharpen up your programming skills, participate in coding contests & explore high-paying jobs

Given an adjacency list of a graph `adj` of V no. of vertices having 0 based index. Check whether the graph is bipartite or not.

Example 1:

Input:

```
C++ (g++ 5.4)
Average Time: 15m Your Time: 10m 0s
19 } else if(color[v]==color[node]){
20     return false;
21 }
22 }
23 }
24 }
25 }
26 }
27 bool isBipartite(int v, vector<int>adj[]){
28     // Code here
29     int color[v]; //act as visited array , considering two color 0 and 1 , initially -1
30     for(int i=0;i<v;i++){
31         color[i]=-1;
32     }
33     for(int i=0;i<v;i++){ //we are Looping just becoz if we have multiple components
34         if(color[i]==-1){
35             if(bfs(i,adj,color)==false){
36                 return false;
37             }
38         }
39     }
40 }
41 }
42 }
43 };
44 // } Driver Code Ends
```

Custom Input Compile & Run Submit

33°C Mostly cloudy

Search

b F S VS W M 76 ENG IN 3:47 PM 6/30/2023

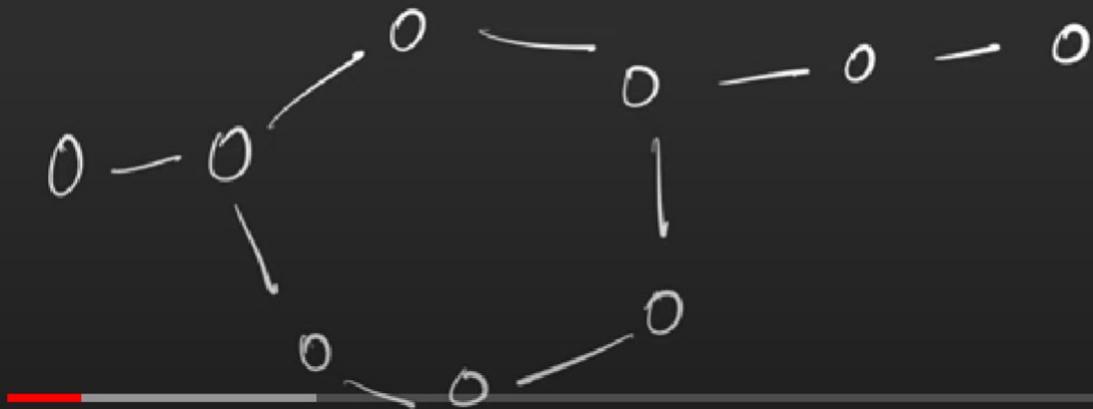
G-18. Bipartite Graph | DFS | C++ | Java

1.75

A graph that can be colored with 2 colors such that no 2 adjacent nodes have the same color.

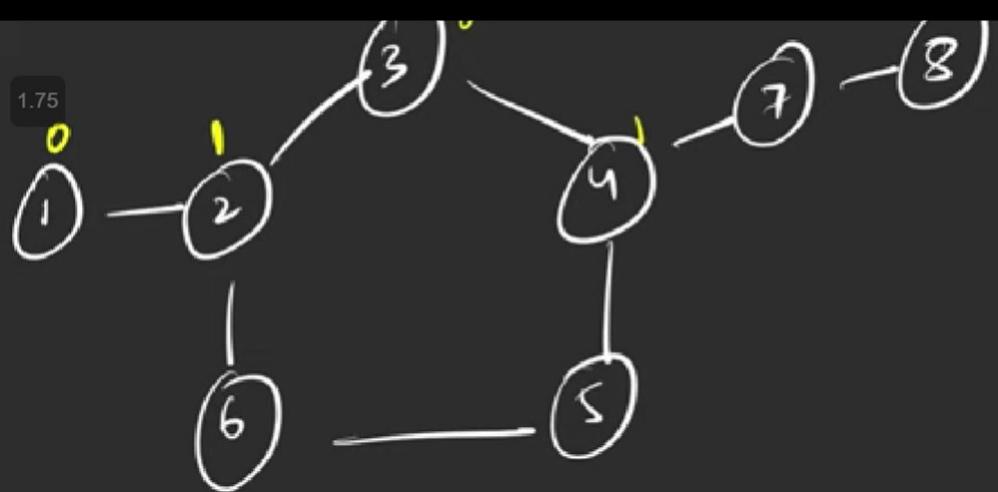


Bipartite Graph



0:35 / 14:53





DFS(1, 0)

DFS(2, 1)

DFS(3, 0)

DFS(4, 1)

$1 \rightarrow \{2\}$
 $2 \rightarrow \{1, 3\}$
 $3 \rightarrow \{2, 4\}$
 $4 \rightarrow \{3, 7, 5\}$
 $5 \rightarrow \{4, 6\}$
 $6 \rightarrow \{2, 5\}$
 $7 \rightarrow \{4, 8\}$
 $8 \rightarrow \{7\}$

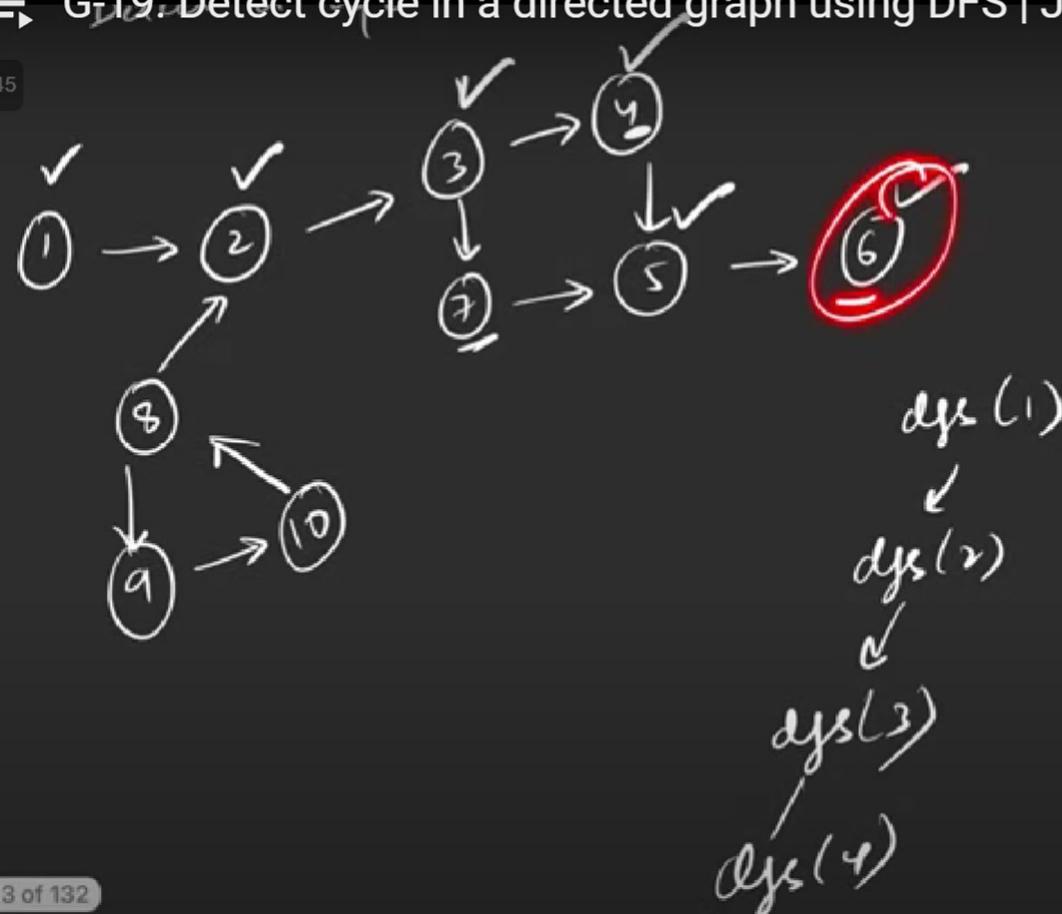
$S_{start} = 1$



TUF

► G19. Detect cycle in a directed graph using DFS | Java | C++

1.45



113 of 132

1:49 / 17:21

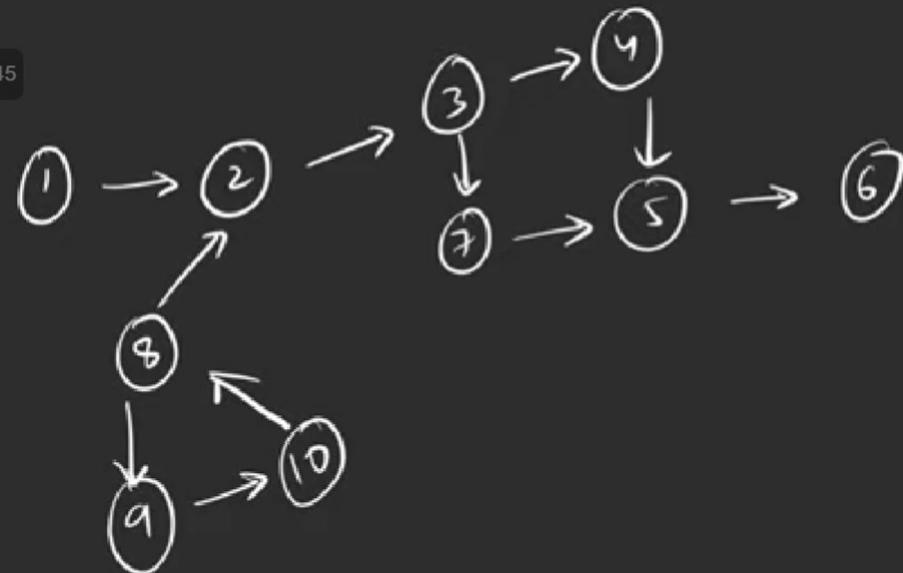
TUP



0.0 / 1.0



1.45



yes

on the same
path, node
has to be
visited again



☰ G-19 Detect cycle in a directed graph using DFS | Java | C++

1.45



6 → {6}
7 → {7}
8 → {8}
9 → {9}
10 → {10}

vis	0	1	2	3	4	5	6	7	8	9	10
vis	0	0	0	0	0	0	0	0	0	0	0

pathvis	0	1	2	3	4	5	6	7	8	9	10
pathvis	0	1	0	0	0	0	0	0	0	0	0



G-19. Detect cycle in a directed graph using DFS | Java | C++

1.45

```
fun( i = 1 → v )  
{  
    if ( !vis[i] )  
        if ( djs(i) == true )  
            return true ;  
  
    }  
}
```



5:41 / 17:21



G-19. Detect cycle in a directed graph using DFS | Java | C++



dfs(1)
dfs(2)

4 → {5}
5 → {6}
6 → {3}
7 → {5}
8 → {9}
9 → {10}
10 → {2}

1.90

vis → [0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10]
[0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0]

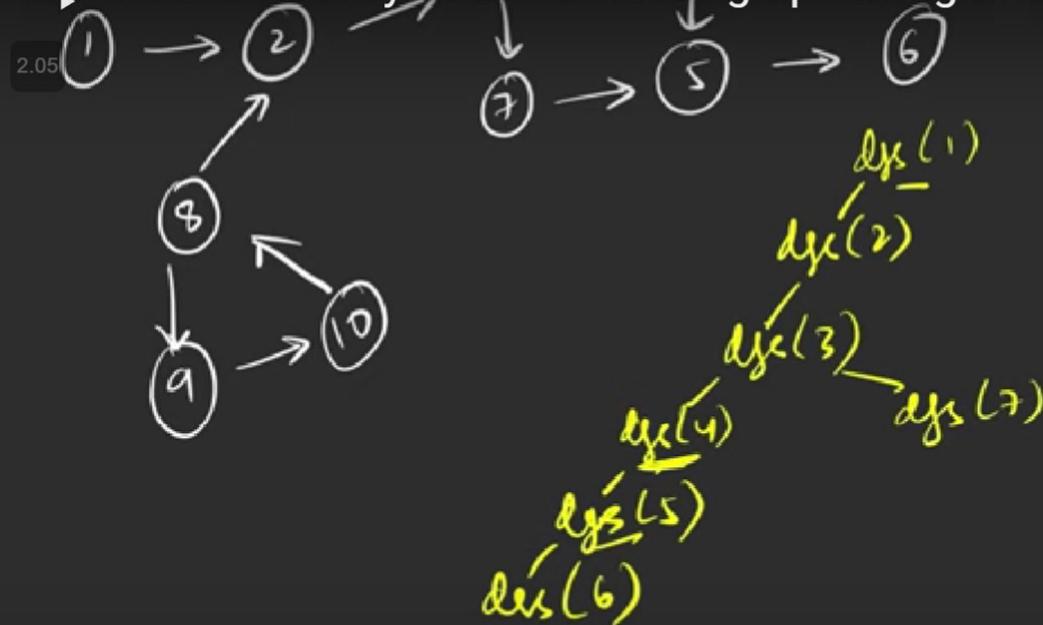
parent → [0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0]



6:17 / 17:21



→ G-19. Detect cycle in a directed graph using DFS | Java | C++



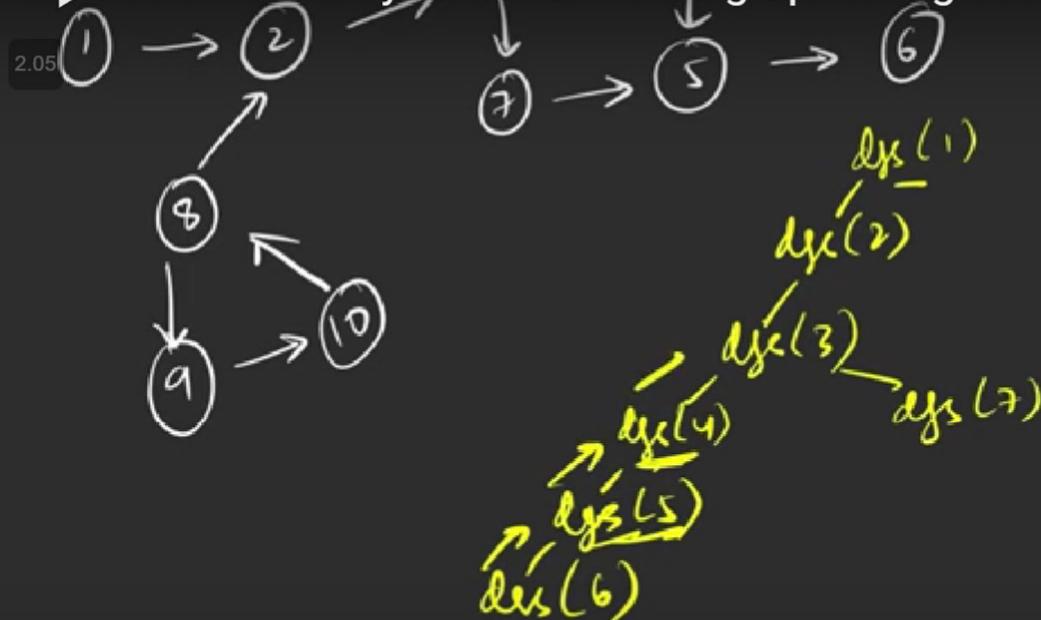
3 → {4, 7}
4 → {5}
5 → {6}
6 → {3}
7 → {5}
8 → {9}
9 → {10}
10 → {2}

vis → [0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0]

pathvis → [0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0]



→ G-19. Detect cycle in a directed graph using DFS | Java | C++



3 → {4, 7}
4 → {5}
5 → {6}
6 → {}
7 → {5}
8 → {9}
9 → {10}
10 → {2}

vis →

0	1	2	3	4	5	6	7	8	9	10
0	1	0	1	0	1	0	0	0	0	0

pathvis →

0	1	0	1	0	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0	0

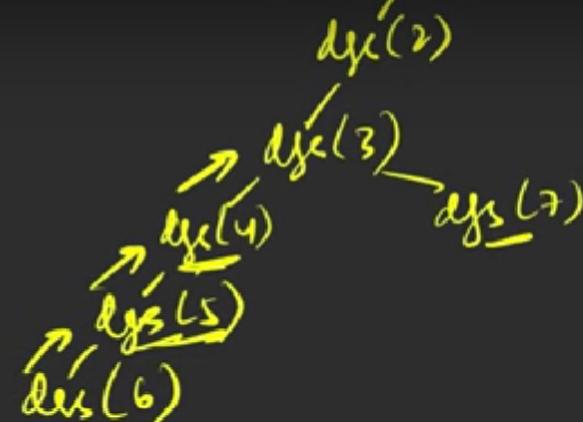
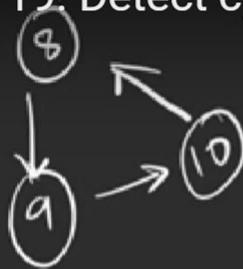


8:08 / 17:21



G-19 Detect cycle in a directed graph using DFS | Java | C++

2.05



6 → {3}
7 → {5}
8 → {93}
9 → {103}
10 → {23}

vis →

0	1	2	3	4	5	6	7	8	9	10
0	1	0	1	0	1	0	1	0	0	0

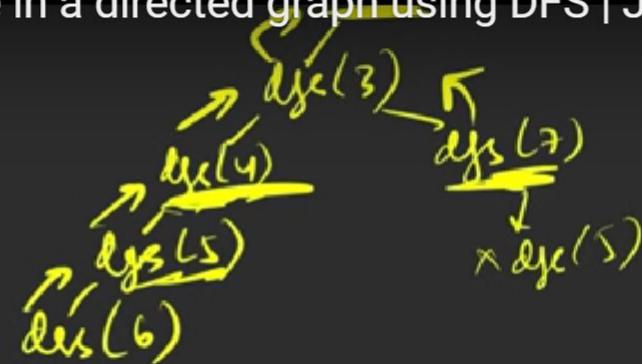
pathvis →

0	1	0	1	0	0	0	0	1	0	0
---	---	---	---	---	---	---	---	---	---	---



→ G-19. Detect cycle in a directed graph using DFS | Java | C++

2.05



7 → {5}
8 → {93
9 → {103
10 → {23

vis →

0	1	2	3	4	5	6	7	8	9	10
0	1	0	1	0	1	0	1	0	0	0

pathvis →

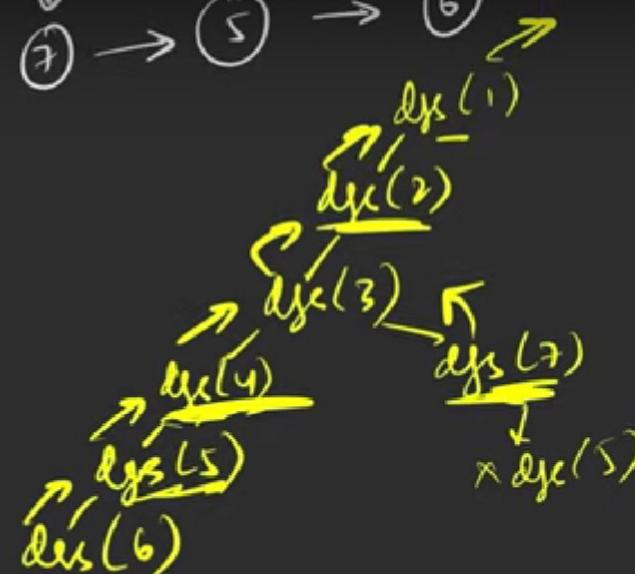
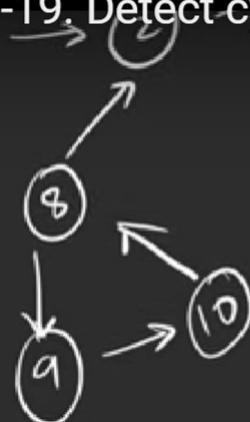
0	1	0	0	0	0	0	1	0	0	0
---	---	---	---	---	---	---	---	---	---	---

for ($i = x \rightarrow v$)



G-19. Detect cycle in a directed graph using DFS | Java | C++

2.05



4 → {5, 6, 7, 8, 9, 10}
5 → {6, 7, 8, 9, 10}
6 → {7, 8, 9, 10}
7 → {8, 9, 10}
8 → {9, 10}
9 → {10}
10 → {0}

vis → [0 1 2 3 4 5 6 7 8 9 10]
0 0 0 0 0 1 0 0 0 0 0

pathvis → [0 1 0 0 0 0 0 0 0 0 0]

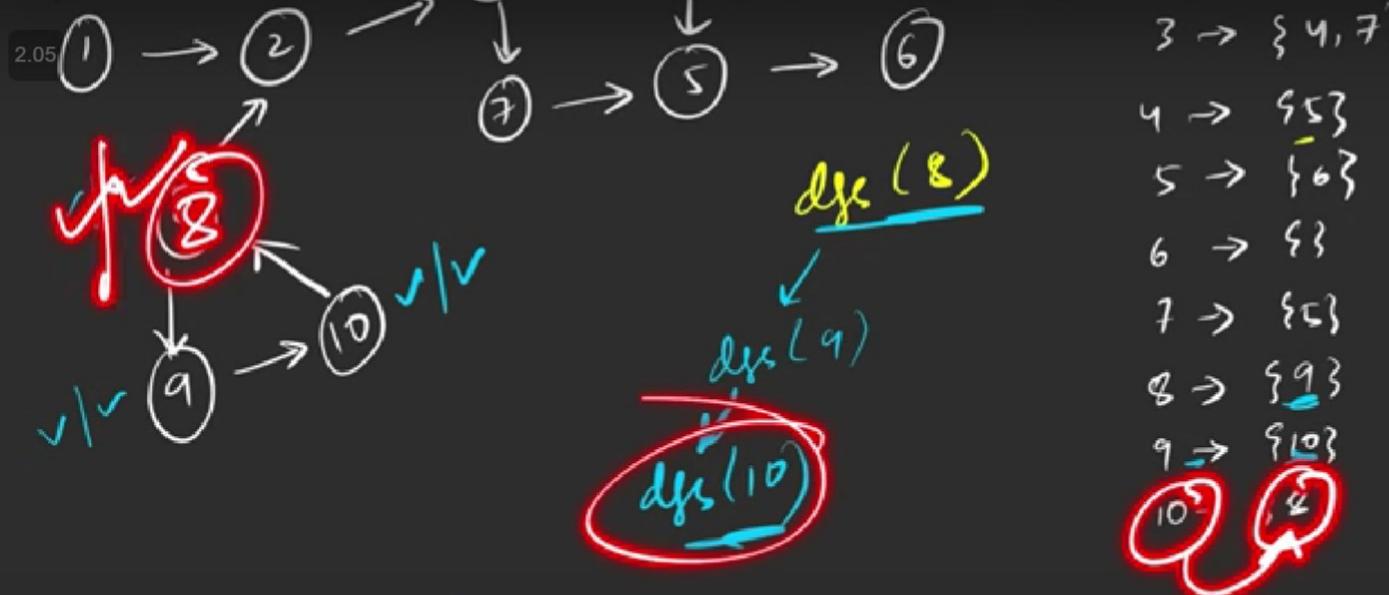
113 of 132



10:04 / 17:21



⇒ G-19. Detect cycle in a directed graph using DFS | Java | C++



$\text{vis} \rightarrow$

0	1	2	3	4	5	6	7	8	9	10
0	1	0	1	0	1	0	1	0	1	2

parent \rightarrow

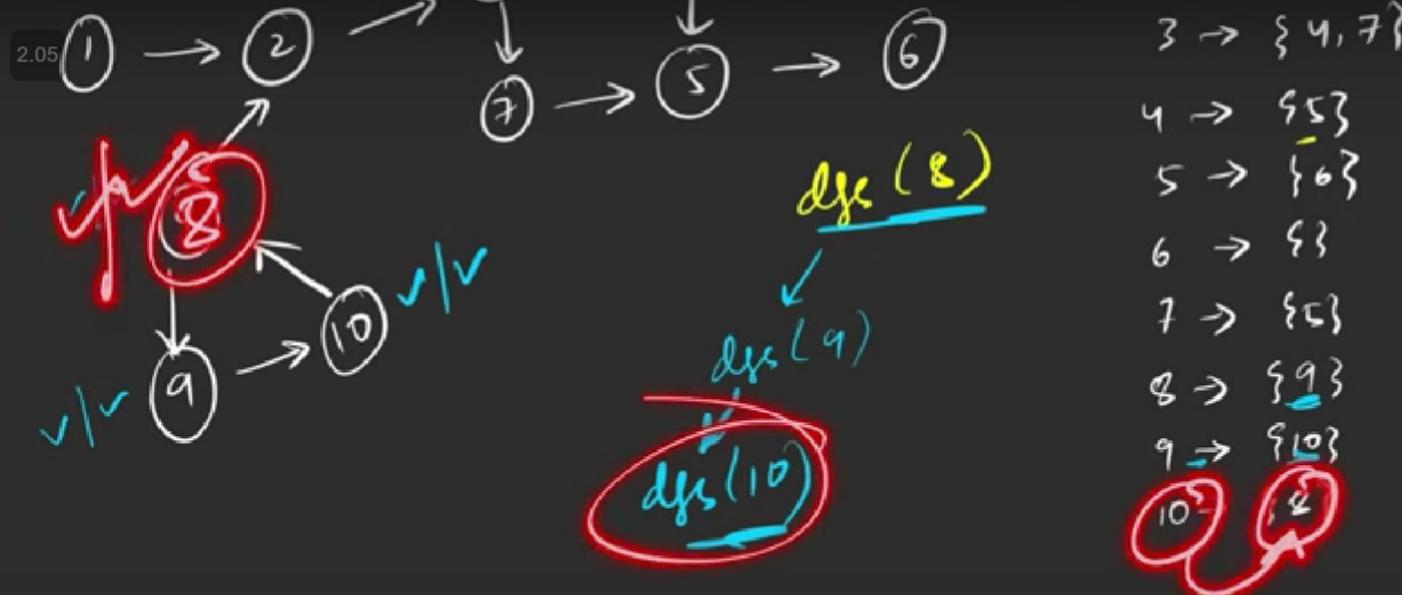
0	1	0	0	0	0	0	0	1	0	0
---	---	---	---	---	---	---	---	---	---	---



11:36 / 17:21



⇒ G-19. Detect cycle in a directed graph using DFS | Java | C++



$\text{vis} \rightarrow$

0	1	2	3	4	5	6	7	8	9	10
0	1	0	1	0	1	0	1	1	0	0

pathvis →

0	1	0	0	0	0	0	0	1	0	0
---	---	---	---	---	---	---	---	---	---	---



11:39 / 17:21



G-19. Detect cycle in a directed graph using DFS | Java | C++



Courses Get Hired Events </> POTD

2.05

Problem Editorial Submissions Comments

```
C++ (g++ 5.4) =
```

```
1- // } Driver Code Ends
2- class Solution {
3-     private:
4-         bool dfsCheck(int node, vector<int> adj[], int vis[], int pathVis[]) {
5-             vis[node] = 1;
6-             pathVis[node] = 1;
7-
8-             // traverse for adjacent nodes
9-             for(auto it : adj[node]) {
10-                 // when the node is not visited
11-                 if(vis[it] == 0) {
12-                     if(dfsCheck(it, adj, vis, pathVis) == true)
13-                         return true;
14-                 }
15-                 // if the node has been previously visited
16-                 // but it has to be visited on the same path
17-                 else if(pathVis[it] == 1) {
18-                     return true;
19-                 }
20-
21-             }
22-         }
23-
24-         // Function to detect cycle in a directed graph.
25-         public boolean isCyclic(int V,
26-                               ArrayList<ArrayList<Integer>> adj) {
27-             int vis[] = new int[V];
28-             int pathVis[] = new int[V];
29-
30-             for(int i = 0;i<V;i++) {
31-                 if(vis[i] == 0) {
32-                     if(dfsCheck(i, adj, vis, pathVis) == true) return true;
33-                 }
34-             }
35-             return false;
36-         }
37-
38-         // Function to detect cycle in a directed graph.
39-         bool isCyclic(int V, vector<int> adj[]) {
40-             int vis[V] = {0};
41-             int pathVis[V] = {0};
42-
43-             for(int i = 0;i<V;i++) {
44-                 if(!vis[i]) {
45-                     if(dfsCheck(i, adj, vis, pathVis) == true) return true;
46-                 }
47-             }
48-             return false;
49-         }
50-     }
```

Example 2:

Input: 15:10 / 17:21

G-19. Detect cycle in a directed graph using DFS | Java | C++

Courses Get Hired Events < POTD

2.05 Problem Editorial Submissions Comments

```
C++ (g++ 5.4) =  
10     pathVis[node] = 1;  
11  
12     // traverse for adjacent nodes  
13     for(auto it : adj[node]) {  
14         // when the node is not visited  
15         if(!vis[it]) {  
16             if(dfsCheck(it, adj, vis, pathVis) == true)  
17                 return true;  
18         }  
19         // if the node has been previously visited  
20         // but it has to be visited on the same path  
21         else if(pathVis[it] == 1) {  
22             return true;  
23         }  
24     }  
25  
26     pathVis[node] = 0;  
27     return false;  
28 }  
29  
public:  
30     // Function to detect cycle in a directed graph.  
31     bool isCyclic(int V, vector<int> adj) {  
32         int vis[V] = {0};  
33         int pathVis[V] = {0};  
34  
35         for(int i = 0;i<V;i++) {  
36             if(vis[i] == 0) {  
37                 if(dfsCheck(i, adj, vis, pathVis) == true) return true;  
38             }  
39         }  
40         return false;  
41     }  
42 };  
43 // } Driver Code Ends
```

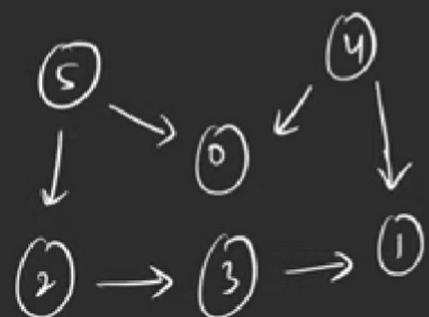
Example 2:

Input: 15:13 / 17:21



1.90

Topological Sorting (DPS)



↳ linear ordering of vertices such that if there is an edge between $u \rightarrow v$, u appears before v in that ordering.

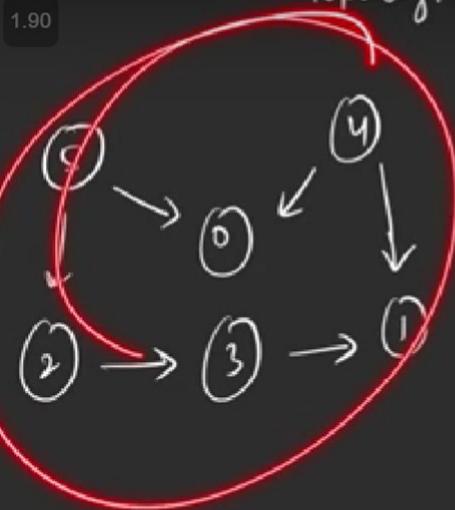
5 4 2 3 1 0

4 5 2 3 1 0



G-21. Topological Sort Algorithm | DFS

Topological Sorting (DPS)



↳ linear ordering of vertices such that if there is an edge between $u \rightarrow v$, u appears before v in that ordering.

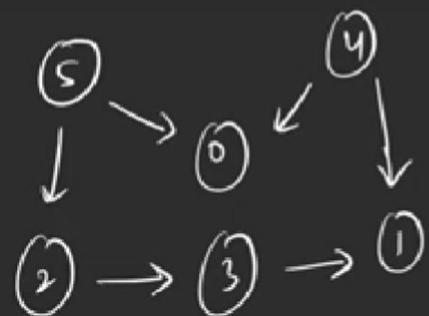
5 4 2 3 1 0

4 5 2 3 1 0



⇒ G-21. Topological Sort Algorithm | DFS Topological Sorting (DFS)

1.90



↳ linear ordering of vertices such that if there is an edge between $u \rightarrow v$, u appears before v in that ordering.

$5 \rightarrow 0$

$4 \rightarrow 0$

$5 \rightarrow 2$

$2 \rightarrow 3$

$3 \rightarrow 1$

$4 \rightarrow 1$

5 4 2 3 1 0

4 5 2 3 1 0

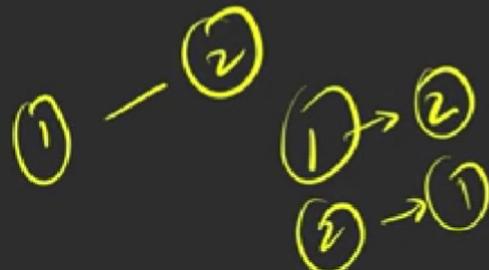


1:34 / 13:29 • Introduction >



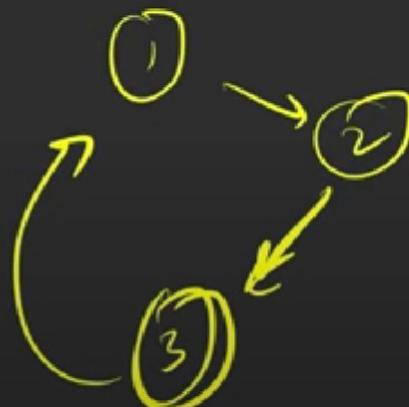
G-21. Topological Sort Algorithm | DFS

$$2 \rightarrow 3 \rightarrow 1$$



$$1 \rightarrow 2$$

$$3 \rightarrow 1$$

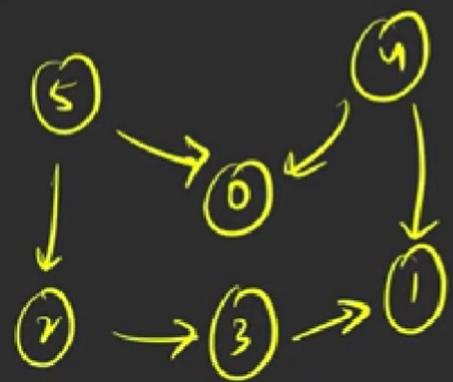


Directed Acyclic Graph DAG

if v , u appears before v in that ordering.



1.90

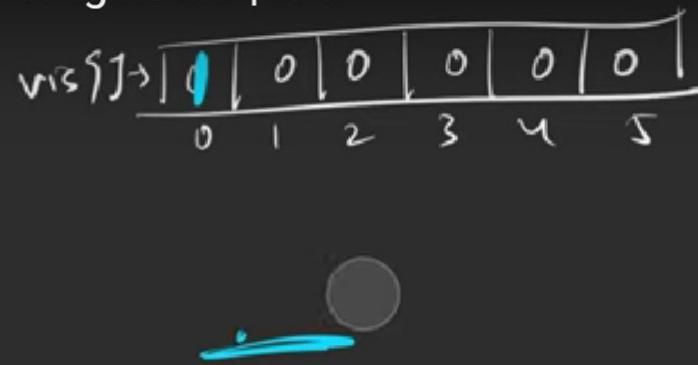
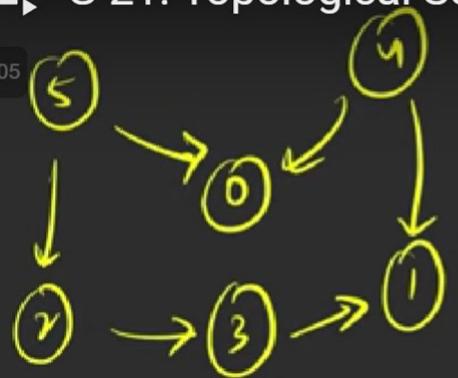
DFS

0 → {}
1 → {}
2 → {3}
3 → {1, 3}
4 → {0, 1, 3}
5 → {0, 2}



⇒ G-21. Topological Sort Algorithm | DFS

2.05



$0 \rightarrow \{\}$
 $1 \rightarrow \{1\}$
 $2 \rightarrow \{2, 3\}$
 $3 \rightarrow \{1, 3\}$
 $4 \rightarrow \{0, 1, 3\}$
 $5 \rightarrow \{0, 2\}$

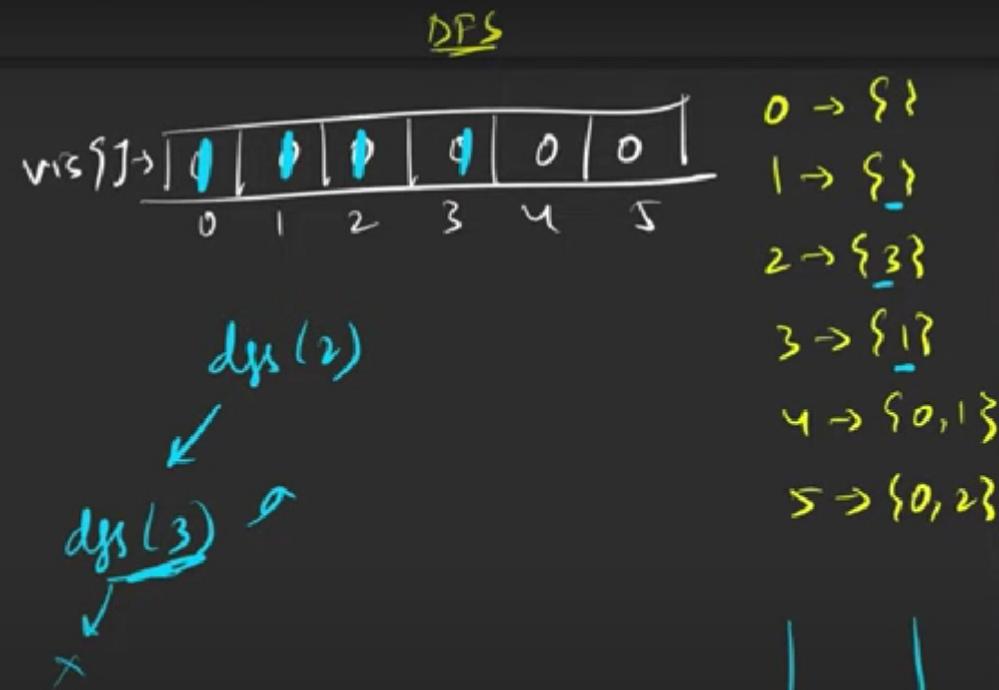
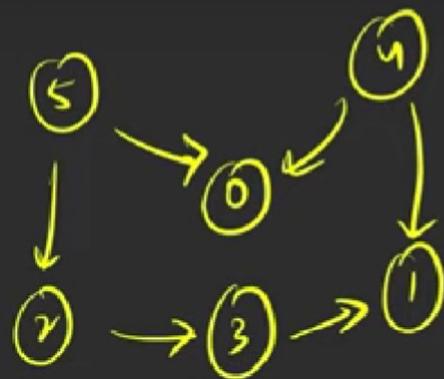


5:41 / 13:29 • Introduction >



⇒ G-21. Topological Sort Algorithm | DFS

2.05

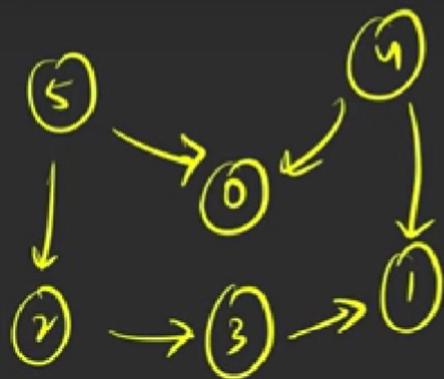


6:44 / 13:29 • Introduction >

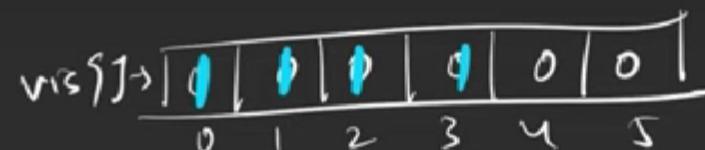


⇒ G-21. Topological Sort Algorithm | DFS

2.05



DFS



dfs(2) ↗
↖ dfs(3) ↗
↖ x

0 → { }
1 → { }
2 → { 3 }
3 → { 1 }
4 → { 0, 1 }
5 → { 0, 2 }

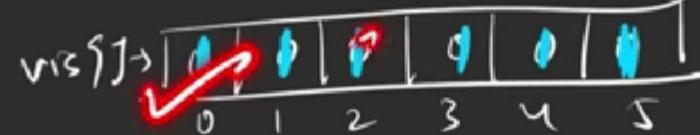
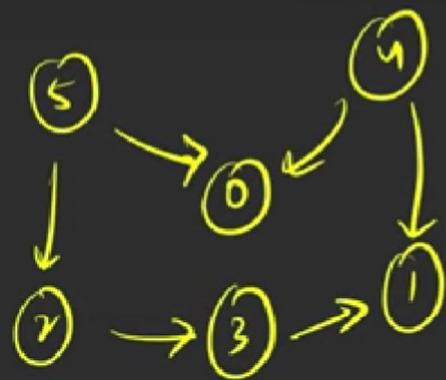
2
3
1



⇒ G-21. Topological Sort Algorithm | DFS

2.05

DFS



dfs(4)

dfs(5)

$0 \rightarrow \{\}$
 $1 \rightarrow \{\}$
 $2 \rightarrow \{3\}$
 $3 \rightarrow \{13\}$
 $4 \rightarrow \{0,13\}$
 $5 \rightarrow \{0,23\}$

↑↑↑

4
2
3
1



7:33 / 13:29 • Introduction >



G-21. Topological Sort Algorithm | DFS

2.05

$y \rightarrow \{0, 1\}$

$s \rightarrow \{0, 2\}$

$\{s, y, 2, 3, 1, 0\}$



TOPICS

7:56 / 13:29 • Introduction >



G-21. Topological Sort Algorithm | DFS

2.05

$y \rightarrow \{0, 1\}$

$s \rightarrow \{0, 2\}$

$\{s, y, 2, 3, 1, 0\} \rightarrow \text{Topo Sort}$

$\rightarrow \left\{ \begin{array}{l} \text{one of the linear} \\ \text{orderings} \end{array} \right\}$



TUF



TOP

8:11 / 13:29 • Introduction >



G-21. Topological Sort Algorithm | DFS

2.05

$\{5, 4, 2, 3, 1, 0\}$ → *Topo Sort*
→ $\{$ one of the linear
 $\}$ orderings

$y \rightarrow \{0, 1\}$
 $s \rightarrow \{0, 2\}$



Firefox browser window showing practice.geeksforgeeks.org/problems/topological-sort/1?utm_source=youtube&utm_medium=referral

Tab titles: (884) ReactJS Tutorial - 6 - Class, Striver's SDE Sheet - Top Codin, (898) G-21. Topological Sort Alg, Topological sort | Practice, (892) YouTube

Address bar: practice.geeksforgeeks.org/problems/topological-sort/1?utm_source=youtube&utm_medium=referral

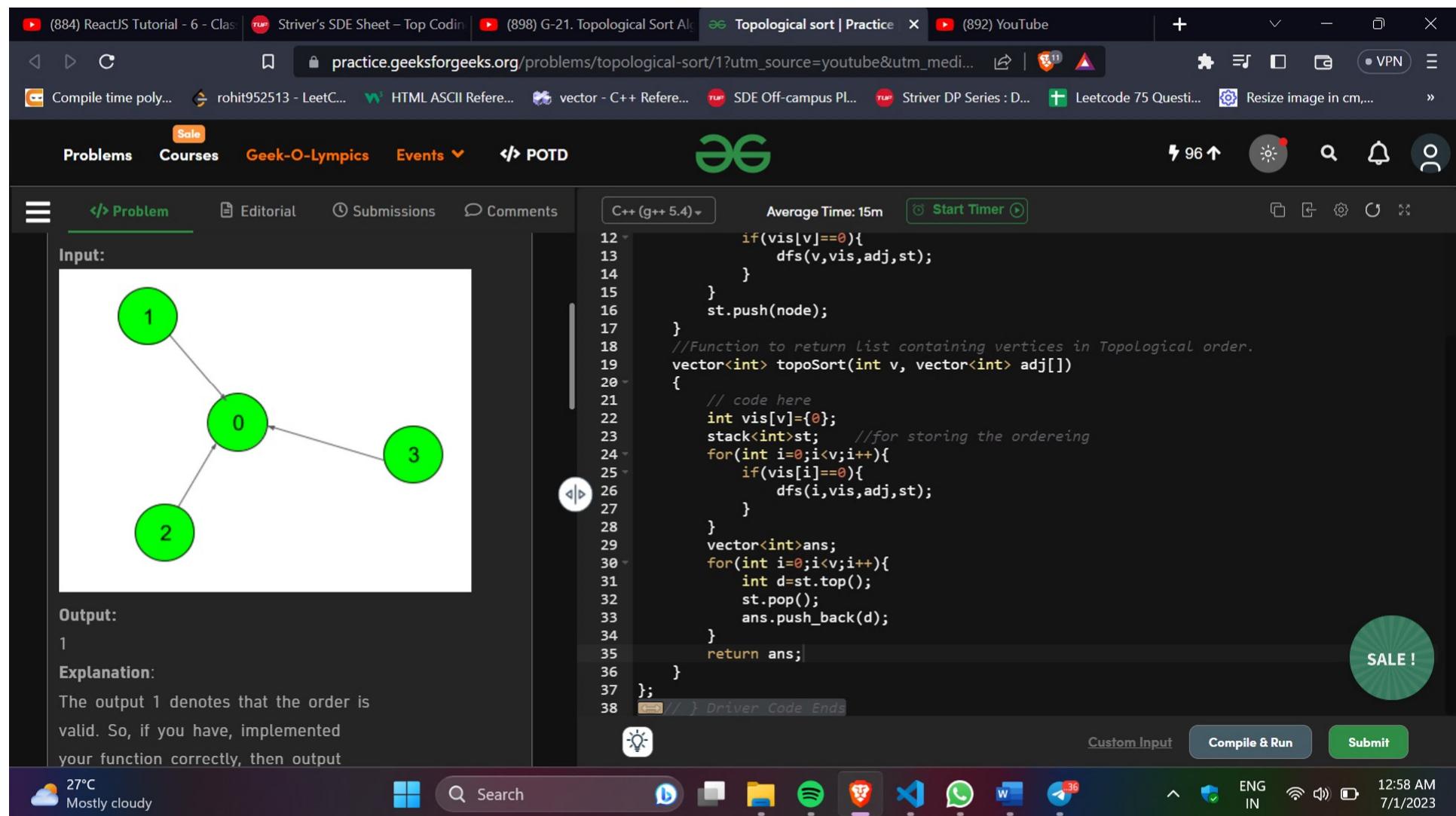
Page navigation: Back, Forward, Stop, Reload, Home, VPN, etc.

Page content:

- Topological sort**
- Medium Accuracy: 56.52% Submissions: 158K+ Points: 4
- Sharpen up your programming skills, participate in coding contests & explore high-paying jobs
- Given a Directed Acyclic Graph (DAG) with V vertices and E edges, Find any Topological Sorting of that Graph.
- Example 1:**
- Input:** A diagram showing a green circle labeled '1' connected by a single directed edge to another green circle.
- Code Editor:** C++ (g++ 5.4) Average Time: 15m Start Timer

```
1 // } Driver Code Ends
6 class Solution
7 {
8     public:
9         void dfs(int node, int vis[], vector<int> adj[], stack<int>&st){
10             vis[node]=1;
11             for(auto &v:adj[node]){
12                 if(vis[v]==0){
13                     dfs(v,vis,adj,st);
14                 }
15             }
16             st.push(node);
17         }
18         //Function to return List containing vertices in Topological order.
19         vector<int> topoSort(int v, vector<int> adj[])
20     {
21         int vis[v]={0};
22         stack<int>st; //for storing the ordering
23         for(int i=0;i<v;i++){
24             if(vis[i]==0){
25                 dfs(i,vis,adj,st);
26             }
27         }
28         vector<int>ans;
29         for(int i=0;i<v;i++){
30             int d=st.top();
31             st.pop();
```

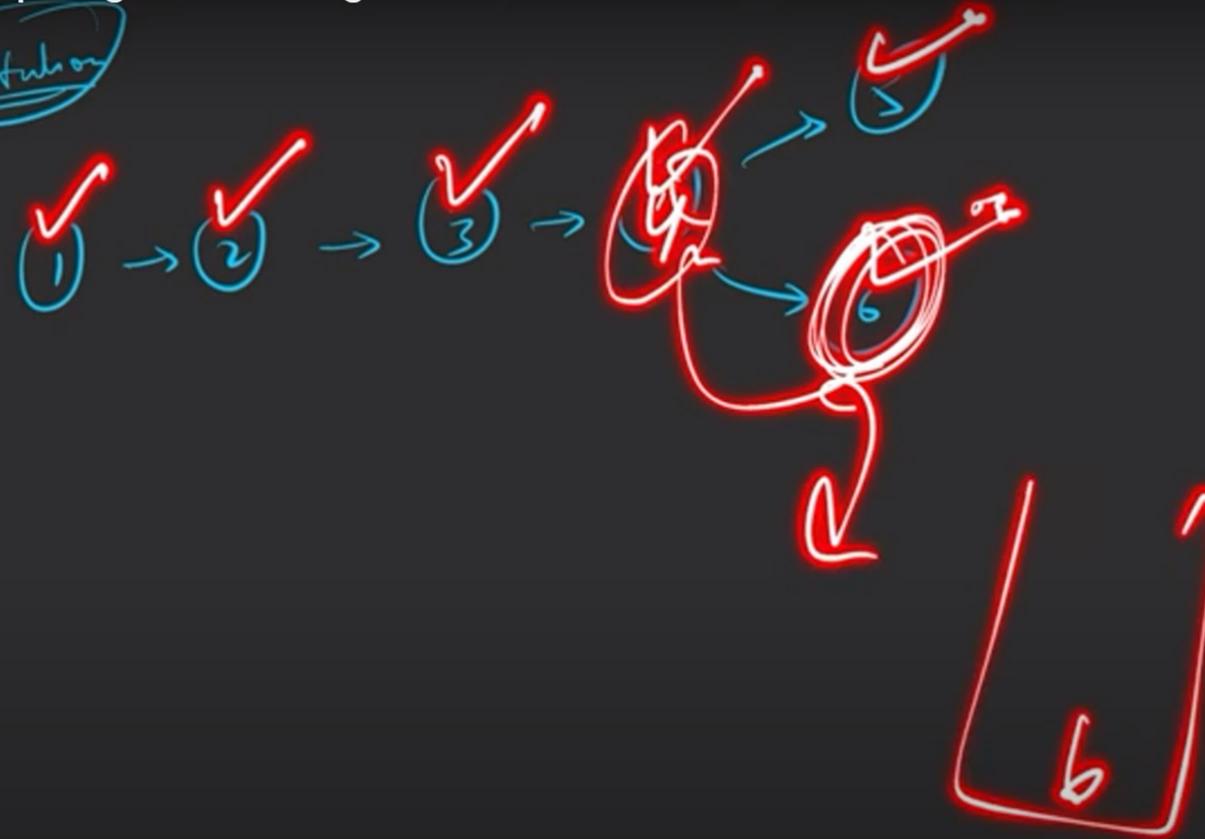
- Custom Input, Compile & Run, Submit buttons
- Weather: 27°C Mostly cloudy
- System tray: Search, File, Spotify, VS Code, WhatsApp, Mail, ENG IN, WiFi, Battery, 12:58 AM, 7/1/2023



⇒ G-21. Topological Sort Algorithm | DFS

2.05

Intuition



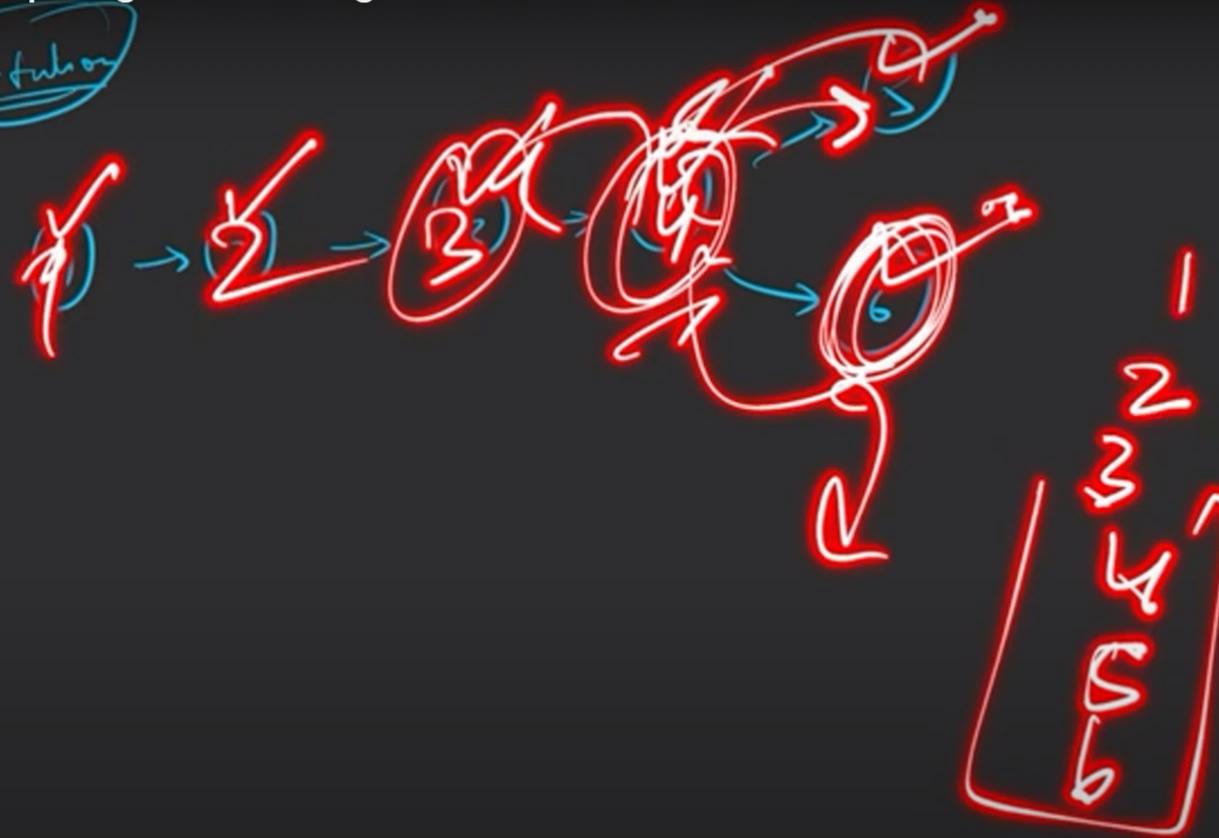
11:53 / 13:29 • Intuition >



⇒ G-21. Topological Sort Algorithm | DFS

2.05

Intuition



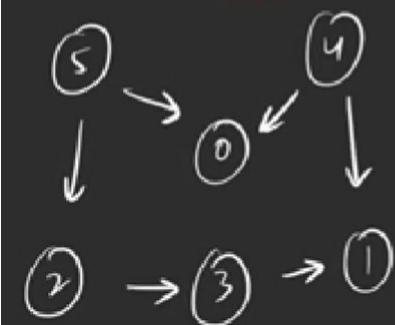
12:02 / 13:29 • Intuition >



G-22. Kahn's Algorithm | Topological Sort Algorithm | BFS

2.05

Topological Sorting



linear ordering of vertices such that
if there is an edge between $u \rightarrow v$, u appears before v in the ordering

$0 \rightarrow \{3\}$
 $1 \rightarrow \{3\}$
 $2 \rightarrow \{1, 3\}$
 $3 \rightarrow \{1, 3\}$
 $4 \rightarrow \{0, 1, 3\}$
 $5 \rightarrow \{0, 1, 2, 3\}$



TUF



0:08 / 13:49



G-22 Kahn's Algorithm | Topological Sort Algorithm | BFS

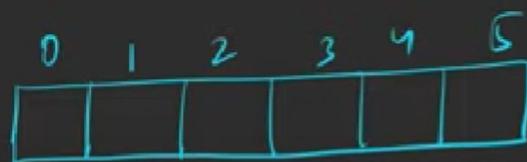
2.05

Q

θ

$4 \rightarrow \{0, 1\}$

$5 \rightarrow \{0, 2\}$



Indegree



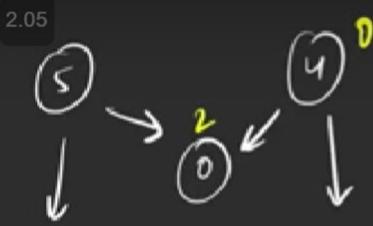
no of incoming edges to a node



4:29 / 13:49



⇒ G-22. Kahn's Algorithm | Topological Sort Algorithm | BFS

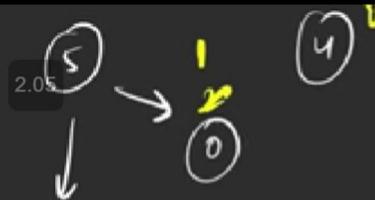


$0 \rightarrow 1 \rightarrow 2$



$0 \rightarrow$
 $1 \rightarrow$
 $2 \rightarrow \{3\}$
 $3 \rightarrow \{1\}$
 $4 \rightarrow \{0, 1\}$
 $5 \rightarrow \{0, 2\}$





$2 \rightarrow 3 \rightarrow 0^{\text{in}}$

4

with indegree 0



0	\rightarrow
1	\rightarrow
2	\rightarrow {3}
3	\rightarrow {1}
4	\rightarrow {0, 1}
5	\rightarrow {0, 2}

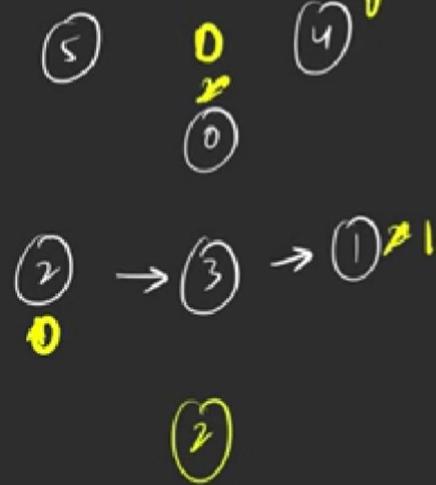
y

0	1	2	3	4	5
x	x	1	1	0	0

Indegree \longrightarrow no of incoming edges to a node



2.05



(*) Remove all nodes
with indegree 0



$0 \rightarrow$
 $1 \rightarrow$
 $2 \rightarrow \{3\}$
 $3 \rightarrow \{1\}$
 $4 \rightarrow \{0, 1\}$
 $5 \rightarrow \{0, 2\}$

4 5 0 2

Table showing node degrees:

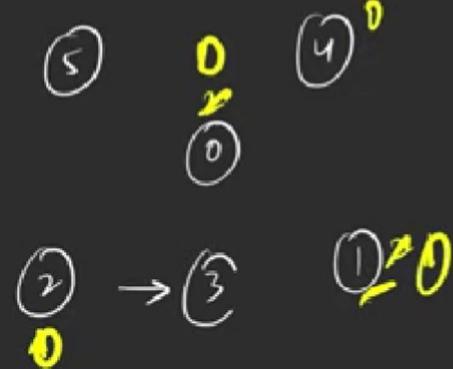
0	1	2	3	4	5
20	11	0	1	0	0



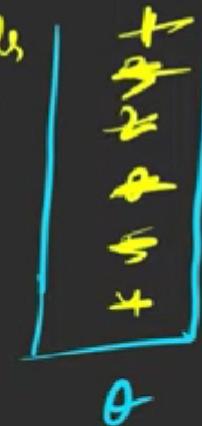
TUG

2.05

Topological Sorting (Kahn's Algorithm / BFS)



(.) Insert all nodes
with indegree 0



0 \rightarrow
1 \rightarrow
2 $\rightarrow \{3\}$
3 $\rightarrow \{1\}$
4 $\rightarrow \{0, 1\}$
5 $\rightarrow \{0, 2\}$

Topo Sort \Rightarrow [4 5 0 2 3 1]

120 of 132

0 1 2 3 4 5

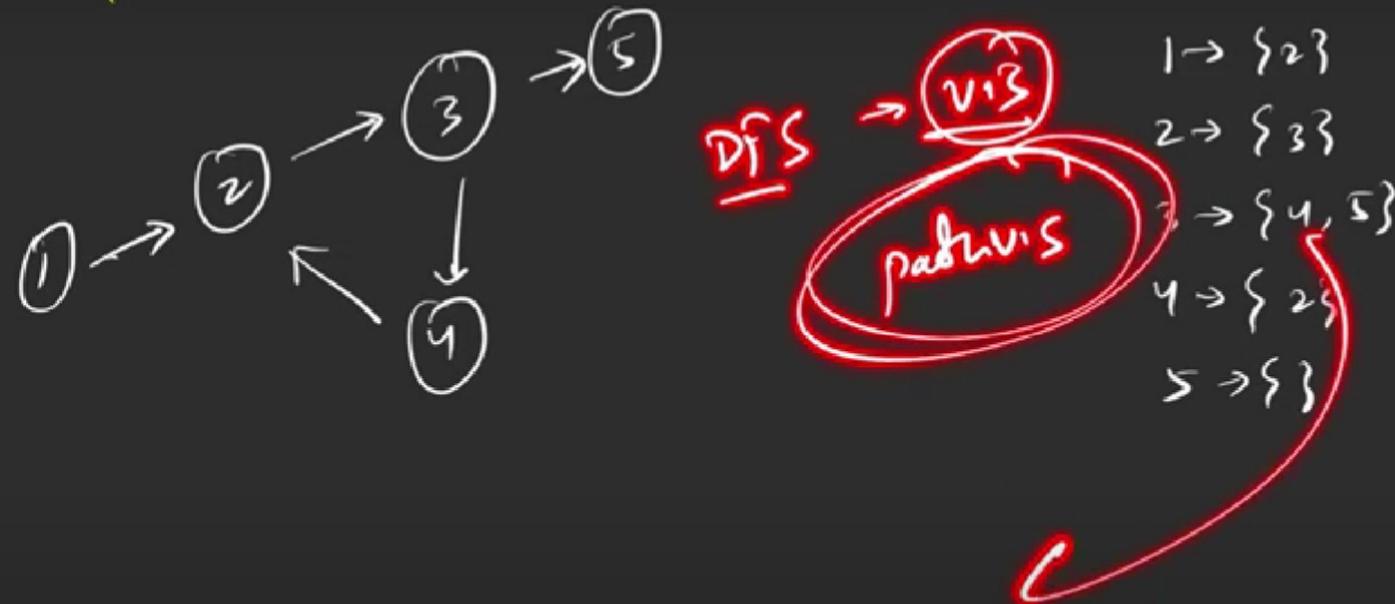
TUF



→ G-23. Detect a Cycle in Directed Graph | Topological Sort | Kahn's Algorithm | BFS

2.05

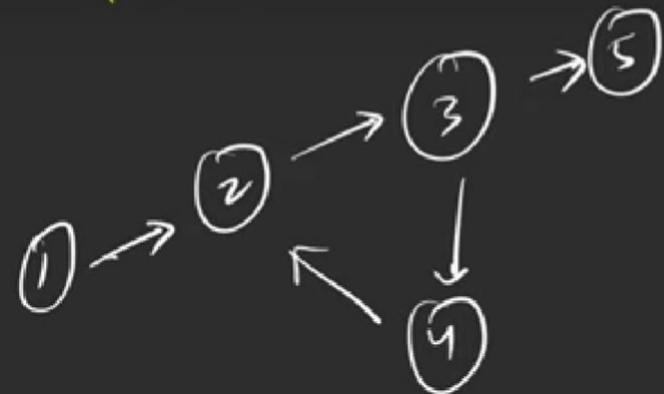
Cycle Detection in Directed Graph → BFS



→ G-23. Detect a Cycle in Directed Graph | Topological Sort | Kahn's Algorithm | BFS

2.05

Cycle Detection in Directed Graph → BFS



~~Kahn's algo~~ 1 → {2, 3}
 2 → {3, 5}
 3 → {4, 5}
 4 → {2, 3}
 5 → {}

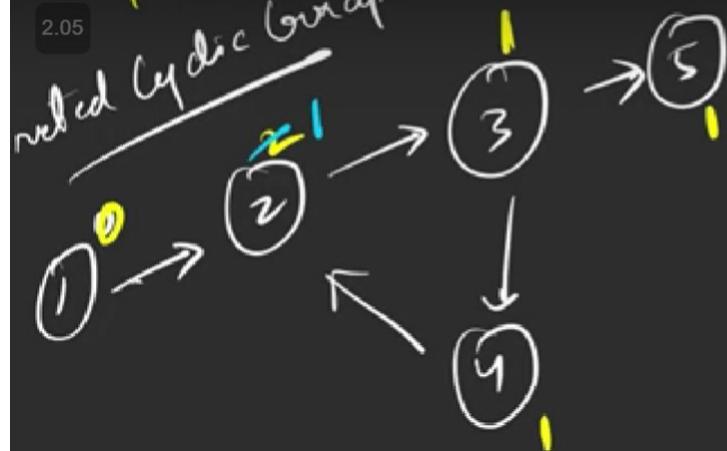


0:47 / 6:16



cycle detection in directed graph | Topological Sort | Kahn's Algorithm | BFS

2.05



Topo Sort

$$\begin{aligned}1 &\rightarrow \{2\} \\2 &\rightarrow \{3\} \\3 &\rightarrow \{4, 5\} \\4 &\rightarrow \{2\} \\5 &\rightarrow \{\}\end{aligned}$$

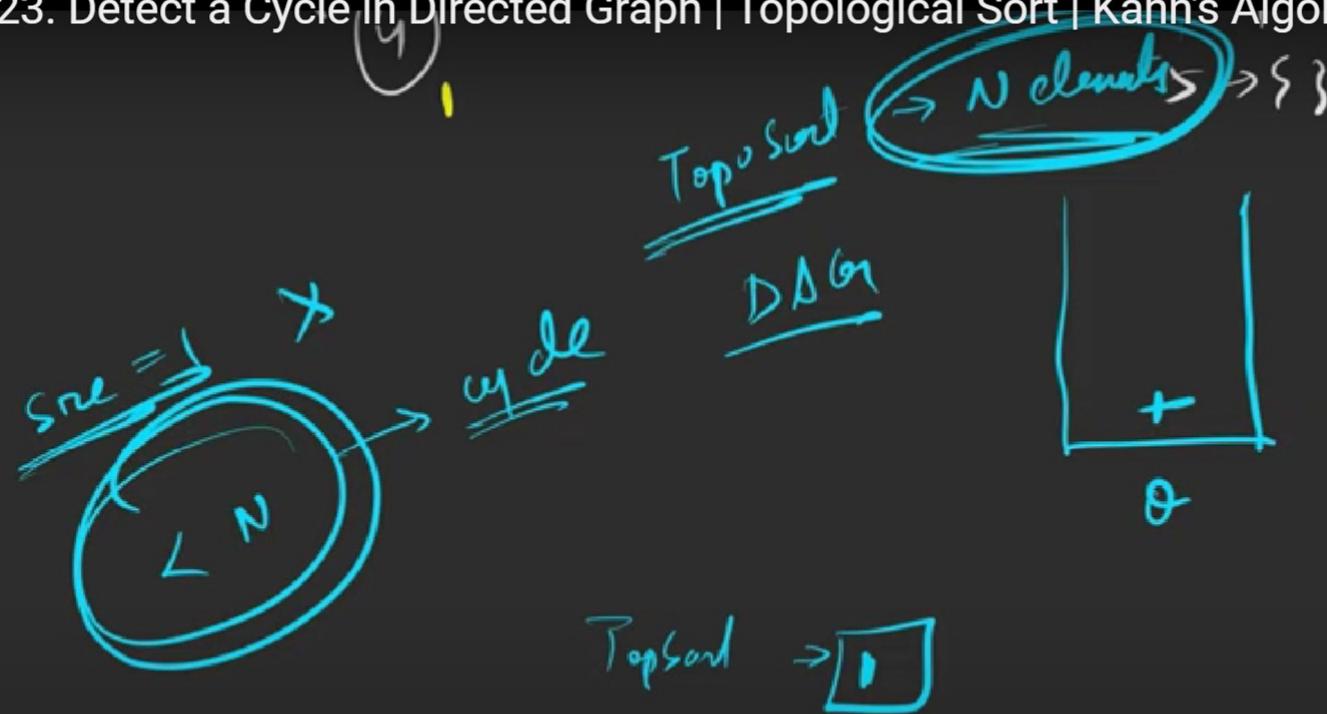


3:17 / 6:16



→ G-23. Detect a Cycle in Directed Graph | Topological Sort | Kahn's Algorithm | BFS

2.05



G-23. Detect a Cycle in Directed Graph | Topological Sort | Kahn's Algorithm | BFS

2.05 Problem Editorial Submissions Comments

Given a Directed Graph with V vertices (Numbered from 0 to $V-1$) and E edges, check whether it contains any cycle or not.

Example 1:

Input:

Output Window

```
C++ (g++ 5.4) +  
6- class Solution {  
7- public:  
8-     // Function to detect cycle in a directed graph.  
9-     bool isCyclic(int V, vector<int> adj[]) {  
10-         int indegree[V] = {0};  
11-         for(int i = 0;i<V;i++) {  
12-             for(auto it : adj[i]) {  
13-                 indegree[it]++;
14-             }
15-         }
16-         queue<int> q;
17-         for(int i = 0;i<V;i++) {
18-             if(indegree[i] == 0) {
19-                 q.push(i);
20-             }
21-         }
22-         int cnt = 0;
23-         // o(v + e)
24-         while(!q.empty()) {
25-             int node = q.front();
26-             q.pop();
27-             cnt++;
28-             // node is in your topo sort
29-             // so please remove it from the indegree
30-             for(auto it : adj[node]) {
31-                 indegree[it]--;
32-                 if(indegree[it] == 0) q.push(it);
33-             }
34-         }
35-         if(cnt == V) return false;
36-         return true;
37-     }
38- }
```

Problem Solved Successfully

You get marks only for the first correct submission if you solve the problem without viewing the full solution.

Test Cases Passed: 410 / 410 Your Total Score: 359

Total Time Taken: 00:00:00.000000 Correct Submission Count: 1

5:39 / 6:16

A video player interface showing a man with a beard speaking. The video is at 5:39 of 6:16. The video player includes standard controls like back, forward, volume, and a progress bar.

G-18. Bipartite Graph | DFS | C++ | Java

Courses Get Hired Events PTD Practice

1/10 Problem Editorial Submissions Comments

Bipartite Graph

Medium Accuracy: 40.1% Submissions: 79088 Points: 4

Given an adjacency list of a graph `adj` of V no. of vertices having 0 based index. Check whether the graph is bipartite or not.

Example 1:

Input:

Output Window

Compilation Results Custom Input

Problem Solved Successfully ✓

You get marks only for the first correct submission if you solve the problem without viewing the full solution.

Test Cases Passed: 825 / 825 Your Total Score: 359

Total Time Taken: 0.47 Correct Submission Count: 1

14:02 / 14:53

359

```
1 // } Driver Code Ends
2 class Solution {
3 private:
4     bool dfs(int node, int col, int color[], vector<int> adj[]) {
5         color[node] = col;
6
7         for(auto it : adj[node]) {
8             if(color[it] == -1) {
9                 if(dfs(it, !col, color, adj) == false) return false;
10            }
11            else if(color[it] == col) {
12                return false;
13            }
14        }
15
16        return true;
17    }
18
19 public:
20     bool isBipartite(int V, vector<int>adj[]){
21         int color[V];
22         for(int i = 0;i<V;i++) color[i] = -1;
23
24         for(int i = 0;i<V;i++) {
25             if(color[i] == -1) {
26                 if(dfs(i, 0, color, adj) == false) return false;
27             }
28         }
29         return true;
30     }
31
32 }
33
34
35 // } Driver Code Ends
```

1.30



Allocate Books

Problem Statement

[Suggest Edit](#)

Given an array 'arr' of integer numbers, 'arr[i]' represents the number of pages in the 'i-th' book.

There are ' m ' number of students, and the task is to allocate all the books to the students.

Allocate books in such a way that:

1. Each student gets at least one book.
2. Each book should be allocated to only one student.
3. Book allocation should be in a contiguous manner.

You have to allocate the book to ' m ' students such that the maximum number of pages assigned to a student is minimum.

If the allocation of books is not possible, return -1.

$$\text{arr}[] = [25 \quad 46 \quad 28 \quad 49 \quad 24]$$

studente - 4

$$25 \mid 46 \mid 28 \mid 49 \ 24 \quad | \boxed{73}$$

$$25 \mid 46 \mid 28 \ 49 \mid 24 \quad | \boxed{77}$$

$$25 \mid 46 \ 28 \mid 49 \mid 24 \quad | \boxed{74}$$

$$25 \ 46 \mid 28 \mid 49 \mid 24 \quad | \boxed{71}$$

TUF

1.30

one student.
3. Book allocation should be in a contiguous manner.

You have to allocate the book to 'm' students such that
the maximum number of pages assigned to a student is
minimum.

If the allocation of books is not possible, return -1.

49
50

51
52

53

1 → 25

2 → 46

3 → 28

4 → 49

5 → 24

BS-18. Allocate Books or Book Allocation | Hard Binary Search

1.30

Problem Statement

Given an array arr of integer numbers, 'arr[i]' represents the number of pages in the i-th book.

There are 'm' number of students, and the task is to allocate all the books to the students.

Allocate books in such a way that:

1. Each student gets at least one book.
2. A book should be allocated to only one student.
3. Book allocation should be in a contiguous manner.

You have to allocate the book to 'm' students such that the maximum number of pages assigned to a student is minimum.

If the allocation of books is not possible, return -1.

arr [] = [25 46 28 49 24]
students = 4 ↗ ↑ ↑ ↘ ↗

49 ↘ 1 → 25
50 ↘ 2 → 46
51 ↘ 3 → 28
52 ↘ 4 → 49
53 ↘

TUF



10:14 / 27:28



1.30

Problem Statement

Given an array, 'arr' of integer numbers, ' $|arr|$ ' represents the number of pages in the '1-bit' book.

There are ' m ' number of students, and the task is to allocate all the books to the students.

Allocate books in such a way that:

1. Each student gets at least one book.
2. Each book should be allocated to only one student.
3. Book allocation should be in a contiguous manner.

You have to allocate the book to ' m ' students such that the maximum number of pages assigned to a student is minimum.

If the allocation of books is not possible, return -1.

arr = [25, 46, 28, 49, 24]

students = 4

Handwritten notes:
arr = [25, 46, 28, 49, 24]
students = 4
1 → 25
2 → 46
3 → 28
4 → 49
5 → 24
6 → 23
7 → 20

TUF

1.30

1. Each student gets at least one book.
2. Each book should be allocated to only one student.
3. Book allocation should be in a contiguous manner.

You have to allocate the book to 'm' students such that the maximum number of pages assigned to a student is minimum.

If the allocation of books is not possible, return -1.

49
50
51
52

53
54
55
56

57
58
59

1 → 25
2 → 46
3 → 28
4 → 19
5 → -

TUF

BS-18. Allocate Books or Book Allocation | Hard Binary Search



arr [] = [25 46 28 49 24]
students = 4 ↑ ↑ ↓ ↓

1 → 25 46

2 → 28 + 49

49
50

51

52

53

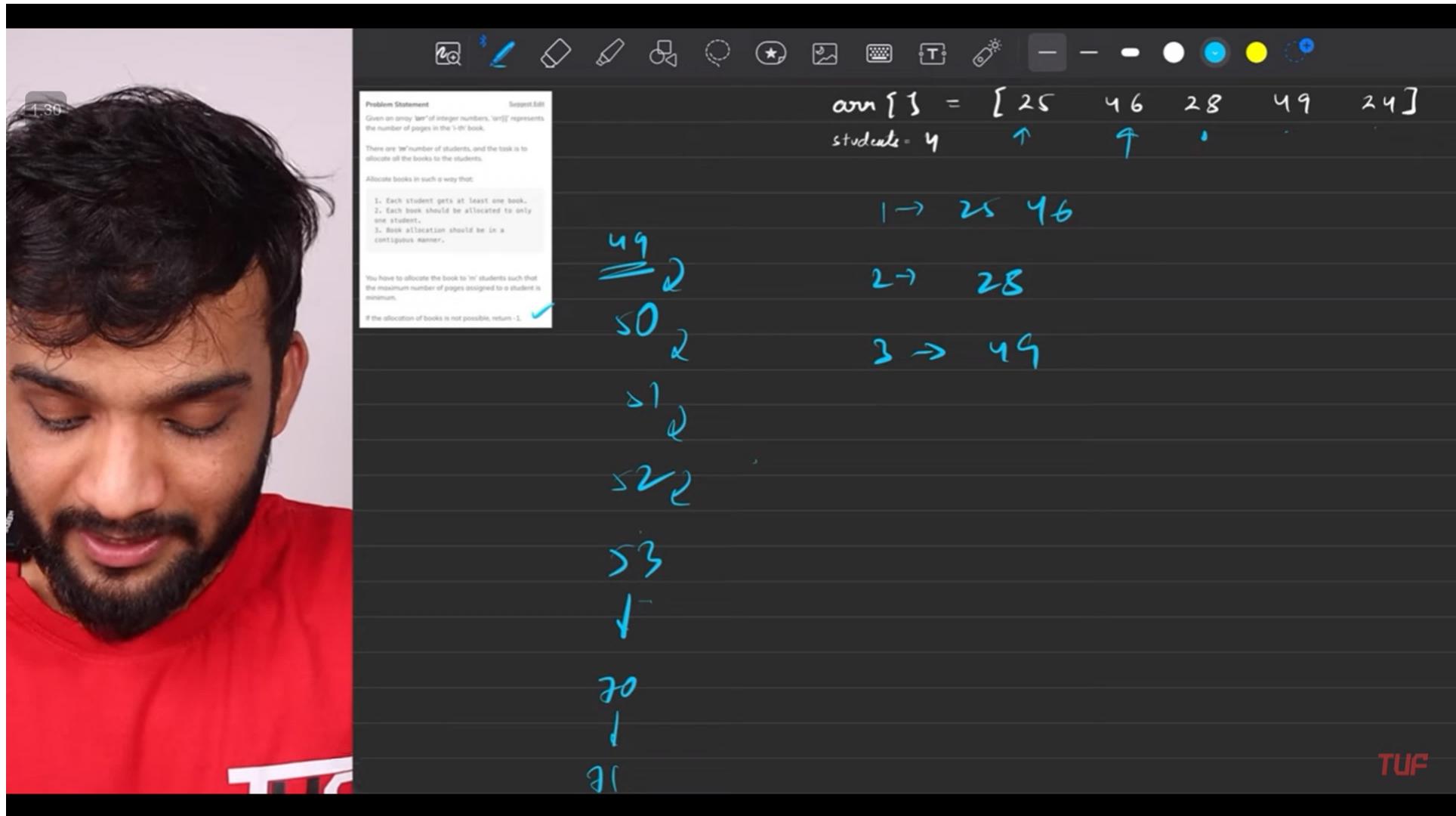
↓

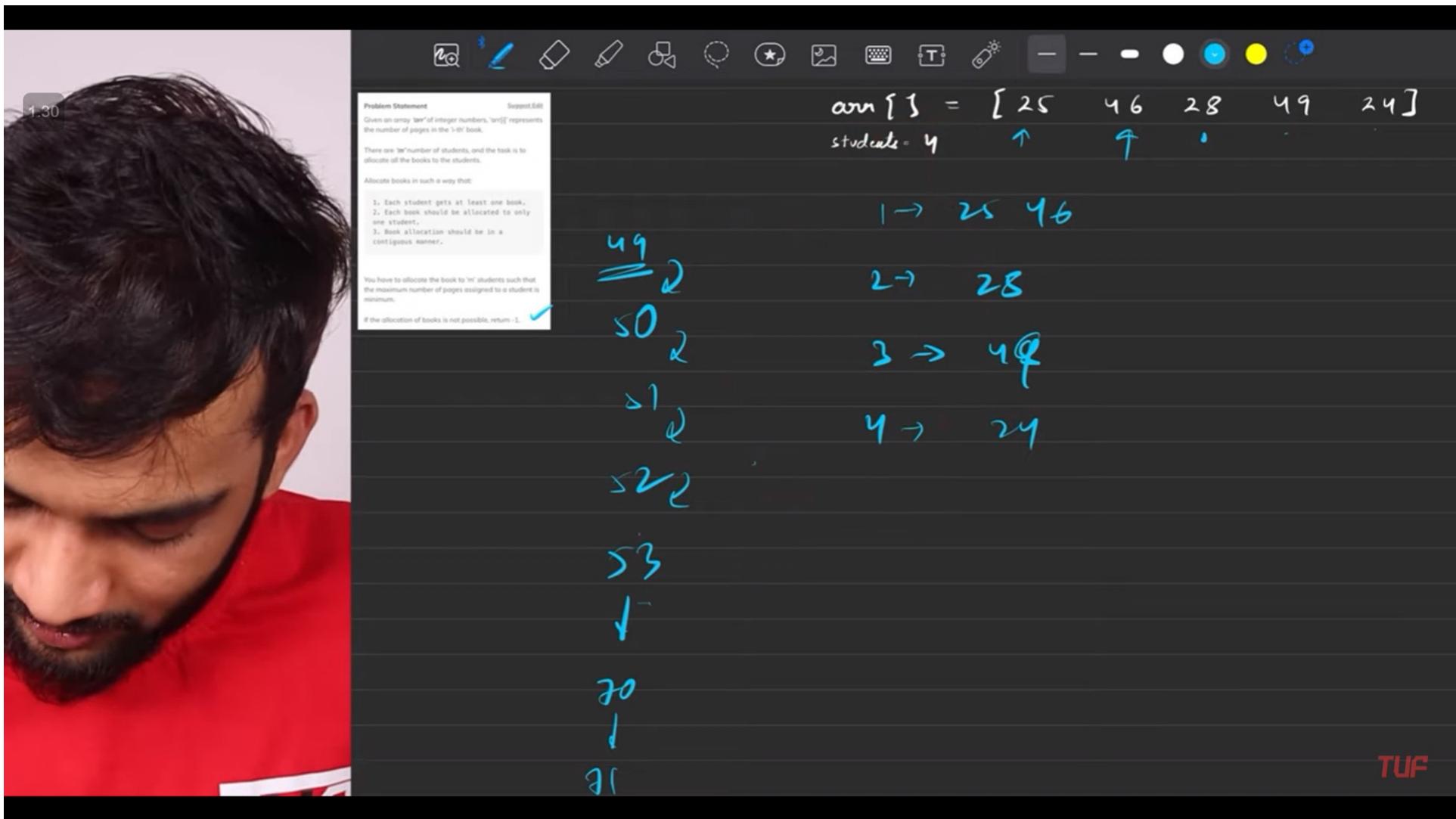
50

↓

TUF







BS-18. Allocate Books or Book Allocation | Hard Binary Search

1.30

$m > n$

Allocate Books

arr [] = [25 46 28 49 24]

student = 4

49

50

51

52

53

↓

TUF



13:16 / 27:28

20

v



BS-18. Allocate Books or Book Allocation | Hard Binary Search

1.30



m > n Allocate Books

Problem Statement
Given an array arr of integer numbers, arr[i] represents the number of pages in the i-th book.
There are m number of students, and the task is to allocate all the books to the students.
Allocate books in such a way that:
1. Each student gets at least one book.
2. Each book should be allocated to only one student.
3. Book allocation should be in a contiguous manner.

You have to allocate the book to m students such that the maximum number of pages assigned to a student is minimum.
If the allocation of books is not possible, return -1.

arr [] = [25 46 28 49 24]
students = 5

high → sum]

49
50
51
52
53
↓

TUF

13:28 / 27:28

1.30



Allocate Books

Given an array 'arr' of integer numbers, 'arr[i]' represents the number of pages in the i-th book.

There are 'm' number of students, and the task is to allocate all the books to the students.

Allocate books in such a way that:

1. Each student gets at least one book.
2. Each book should be allocated to only one student.
3. Book allocation should be in a contiguous manner.

You have to allocate the book to 'm' students such that the maximum number of pages assigned to a student is minimum.

If the allocation of books is not possible, return -1.

$m > n$

$arr = [25 \quad 46 \quad 28 \quad 49 \quad 24]$

$students = 4$

$low = \min(arr)$ $high = \sum(arr)$

TUF

1.30



n > m

Allocate Books

Problem Statement

Given an array ‘arr’ of integer numbers, ‘arr[i]’ represents the number of pages in the ‘i’-th book.

There are ‘m’ number of students, and the task is to allocate all the books to the students.

Allocate books in such a way that:

1. Each student gets at least one book.
2. Each book should be allocated to only one student.
3. Book allocation should be in a contiguous manner.

You have to allocate the book to ‘m’ students such that the maximum number of pages assigned to a student is minimum.

If the allocation of books is not possible, return -1.

$\text{arr} = [25 \quad 46 \quad 28 \quad 49 \quad 24]$
 $\text{students} = 4$

$\text{low} = \min(\text{arr}) \quad \text{high} = \sum(\text{arr})$

$\text{fun}(\text{pages} = \text{low} \rightarrow \text{high})$

{

$\text{cutStu} = \underline{\text{fun}}(\text{arr}, \underline{\text{pages}});$

$\text{if } (\underline{\text{cutStu}} == m)$
 $\quad \quad \quad \text{such pages};$

}

TUF

BS-18. Allocate Books or Book Allocation

Hard Binary Search

1.45

If the allocation of books is not possible, return -1.

$\text{fun}(\underline{\text{pages}} = \text{low} \rightarrow \text{high})$

↳

$\text{cutStu} = \underline{\text{fun}}(\underline{\text{arr}}, \underline{\text{pages}});$

if ($\underline{\text{cutStu}} == m$)
 return pages;

}

$T \sim \underline{o(\sum_{i=1}^n \text{arr}[i]) \times n}$

[25 46 28 49 24]

$\text{fun}(\underline{\text{arr}}, \underline{\text{pages}})$

↳
 $\text{stu} = 1, \text{pagesStudent} = 0,$

1 → 28 71
2 → 28 TUF

▶ ▶ ⏪ 17:58 / 27:28

▶ CC ⚙

BS-18. Allocate Books or Book Allocation | Hard Binary Search

Students = 9

$$\text{low} = \text{max}(\text{arr}) \quad \text{high} = \text{sum}(\text{arr})$$

$$\text{fun}(\text{pages} = \text{low} \rightarrow \text{high})$$

{

$$\text{cutStu} = \text{fun}(\text{arr}, \underline{\text{pages}});$$

$$\text{if } (\underline{\text{cutStu}} == m) \\ \text{such pages;}$$

}



17:34 / 27:28



Word Bre | Striver's S | Short sim | Kth Large | Heap Sort | Online C | (919) ma | Problem | Single Nu | Mega Job | Magic To | + | - | X | Update |

Compile time poly... rohit952513 - LeetC... HTML ASCII Refere... vector - C++ Refere... SDE Off-campus Pl... Striver DP Series : D... Leetcode 75 Questi... Resize image in cm,...

Contest Home Quiz Coding Problems

Time Left: 00:01:50:04

Magic and toy shop

Accuracy: 100.0% Submissions: 3+ Points: 25

Geek goes to a toy shop having **N** toys, the prices of these toys are given as **price** array. He wants to buy all the N toys, but he has only **M** rupees to spend. He has a magical Trident which reduces the price of each toy. This reduced price for each toy is given in **magical_price** array. That is for ith toy ($1 \leq i \leq N$) **price[i]** is the original price and **magical_price[i]** is the price after applying magic. Since applying magic reduces the power of Trident, he wants to apply it as **minimum times** as possible.

Find the minimum number of toys on which he should apply this magic so as to buy all the N toys for **atmost** **M** rupees. If it is not possible for Geek to buy the toys even after applying the magic on all the toys the return **-1**.

Example 1:

```
C++ (g++ 5.4) *
15 int t2=accumulate(price.begin(),price.end(),0);
16 if(t2<=m){
17     return 0;
18 }
19 std::vector<int> effectivePrice(price); // Initialize effectivePrice with original
20 std::sort(effectivePrice.begin(), effectivePrice.end()); // Sort the effectivePrice
21
22 int totalCost = 0;
23 int magicCount = 0;
24
25 for (int toyPrice : effectivePrice) {
26     if (toyPrice + totalCost > m) {
27         break;
28     }
29
30     totalCost += (toyPrice - magical_price[std::find(price.begin(), price.end(), toyPrice)]);
31     magicCount++;
32 }
33
34 if (totalCost <= m) {
35     return magicCount;
36 } else {
37     return -1;
38 }
39 }
40 };
41 }
```

Custom Input Compile & Run Submit

Cloudy 32°C Search b W 8:43 PM ENG IN 7/5/2023 2

Word Bre | Striver's S | Short sim | Kth Large | Heap Sort | Online C | (919) ma | Problem | Single Nu | Mega Job | Magic To | + | - | X | Update |

Compile time poly... rohit952513 - LeetC... HTML ASCII Refere... vector - C++ Refere... SDE Off-campus Pl... Striver DP Series : D... Leetcode 75 Questi... Resize image in cm,...

Contest Home Quiz Coding Problems

Time Left: 00:01:50:00

Problem Submissions

possible for Geek to buy the toys even after applying the magic on all the toys the return -1.

Example 1:

Input:
N = 5, M = 13
price = {3,4,6,2,4}
magical_price = {1,2,5,1,3}

Output:
4

Explanation:
Geek can apply the magic on first four toys so as to buy all the toys for 13. He will require minimum 4 magical operations.

C++ (g++ 5.4) *

```
15 int t2=accumulate(price.begin(),price.end(),0);
16 if(t2<=m){
17     return 0;
18 }
19 std::vector<int> effectivePrice(price); // Initialize effectivePrice with original
20 std::sort(effectivePrice.begin(), effectivePrice.end()); // Sort the effectivePrice
21
22 int totalCost = 0;
23 int magicCount = 0;
24
25 for (int toyPrice : effectivePrice) {
26     if (toyPrice + totalCost > m) {
27         break;
28     }
29
30     totalCost += (toyPrice - magical_price[std::find(price.begin(), price.end(), toyPrice)]);
31     magicCount++;
32 }
33
34 if (totalCost <= m) {
35     return magicCount;
36 } else {
37     return -1;
38 }
39 }
40 }
41 
```

Custom Input Compile & Run Submit

Input: 32°C Cloudy

Search b W S 8:43 PM 7/5/2023 ENG IN 2

Word Bre | Striver's S | Short sim | Kth Large | Heap Sort | Online C | (919) ma | Problem | Single Nu | Mega Job | Magic To | + | - | X

Compile time poly... rohit952513 - LeetC... HTML ASCII Refere... vector - C++ Refere... SDE Off-campus Pl... Striver DP Series : D... Leetcode 75 Questi... Resize image in cm,...

Contest Home Quiz Coding Problems

Time Left: 00:01:49:57

Problem Submissions

Explanation:
Geek can apply the magic on first four toys so as to buy all the toys for 13. He will require minimum 4 magical operations.

Example 2:

Input:
 $N = 3, M = 6$
price = {4,3,4}
magical_price = {2,2,3}

Output:
-1

Explanation:
Even after applying the maigc on all the toys, he cannot buy all the toys at rupees 6.

C++ (g++ 5.4)

```
15 int t2=accumulate(price.begin(),price.end(),0);
16 if(t2<=m){
17     return 0;
18 }
19 std::vector<int> effectivePrice(price); // Initialize effectivePrice with original
20 std::sort(effectivePrice.begin(), effectivePrice.end()); // Sort the effectivePrice
21
22 int totalCost = 0;
23 int magicCount = 0;
24
25 for (int toyPrice : effectivePrice) {
26     if (toyPrice + totalCost > m) {
27         break;
28     }
29
30     totalCost += (toyPrice - magical_price[std::find(price.begin(), price.end(), toyPrice)]);
31     magicCount++;
32 }
33
34 if (totalCost <= m) {
35     return magicCount;
36 } else {
37     return -1;
38 }
39 }
40 };
41 
```

Custom Input Compile & Run Submit

32°C Cloudy Search b W 8:43 PM 7/5/2023 ENG IN 2

Word Bre | Striver's S | Short sim | Kth Large | Heap Sort | Online C | (919) ma | Problem | Single Nu | Mega Job | Meg X | Count Be | + | - | X

Compile time poly... rohit952513 - LeetC... HTML ASCII Refere... vector - C++ Refere... SDE Off-campus Pl... Striver DP Series : D... Leetcode 75 Questi... Resize image in cm,... »

Contest Home Quiz Coding Problems

Time Left : 00:01:15:39

Problem Submissions

Count beautiful strings

Accuracy: 6.18% Submissions: 1K+ Points: 35

Geek love strings having **atmost one** character whose frequency is **odd** and rest all (if exists) characters have **even** frequencies, he calls such strings as **beautiful strings**. Geek has a **box** containing **n** strings. You have to help him in counting number of pairs **(i,j)** (**0<=i<j<n-1**) such that a string formed by concatenating string at index **i** with string at index **j** is a beautiful string.

Note: Strings consist of only lowercase English letters.

Example 1:

Input:
n = 3
box = {bbcb, abccc, abc}

Output:

```
//{ } Driver Code Ends
//User function Template for C++
class Solution{
public:
    long long noOfPairs(vector<string> &box)
    {
        // Write your code here.
        long long count = 0;
        int n = box.size();
        for (int i = 0; i < n - 1; i++) {
            unordered_map<char, int> freq;
            // Calculate the frequency of characters in string at index i
            for (char c : box[i])
                freq[c]++;
            for (int j = i + 1; j < n; j++) {
                // Calculate the frequency of characters in string at index j
                for (char c : box[j])
                    freq[c]++;
                int oddCount = 0, evenCount = 0;
                // Count the number of characters with odd and even frequencies
                for (auto it = freq.begin(); it != freq.end(); it++) {
                    if (it->second % 2 == 0)
                        evenCount++;
                    else
                        oddCount++;
                }
                if (oddCount == 1)
                    count++;
            }
        }
        return count;
    }
};

Custom Input Compile & Run Submit
```

31°C Partly cloudy

Search

b W S 2 9:22 PM 7/5/2023

Word Bre | Striver's S | Short sim | Kth Large | Heap Sort | Online C | (919) ma | Problem | Single Nu | Mega Job | Meg X | Count Be | + | - | X

Compile time poly... rohit952513 - LeetC... HTML ASCII Refere... vector - C++ Refere... SDE Off-campus Pl... Striver DP Series : D... Leetcode 75 Questi... Resize image in cm,... Update

Contest Home Quiz Coding Problems

Time Left : 00:01:15:35

Problem Submissions

Input:

```
n = 3
box = {bbcb, abccc, abc}
```

Output:

```
3
```

Explanation:

Pairs which form beautiful string are (1,2), (1,3) and (2,3).

Concatenating 'bbcb' and 'abccc' we get 'bbcbabccc' in which characters 'b' and 'c' has frequency of 4, and only 'a' has odd frequency. So the pair (1,2) forms a beautiful string.

Concatenating 'bbcb' and 'abc' we get 'bbcbabc' in which characters 'b' and 'c' has frequency of 4 and 2 respectively, and only 'a' has odd frequency. So the pair (1,3) forms a beautiful string.

C++ (g++ 5.4) *

```
1 // [{ }] Driver Code Ends
2 // User function Template for C++
3
4 class Solution{
5 public:
6     long long noOfPairs(vector<string> &box)
7     {
8         // Write your code here.
9         long long count = 0;
10        int n = box.size();
11
12        for (int i = 0; i < n - 1; i++) {
13            unordered_map<char, int> freq;
14
15            // Calculate the frequency of characters in string at index i
16            for (char c : box[i])
17                freq[c]++;
18
19            for (int j = i + 1; j < n; j++) {
20                // Calculate the frequency of characters in string at index j
21                for (char c : box[j])
22                    freq[c]++;
23
24                int oddCount = 0, evenCount = 0;
25
26                // Count the number of characters with odd and even frequencies
27                for (auto it = freq.begin(); it != freq.end(); it++) {
28
29                    if (it->second % 2 == 0)
30                        evenCount++;
31                    else
32                        oddCount++;
33
34                }
35
36                if (oddCount > 0 && evenCount > 0)
37                    count++;
38            }
39        }
40    }
41
42 }
```

Custom Input Compile & Run Submit

31°C Partly cloudy

Search

b W S 2 ENG IN 9:22 PM 7/5/2023

Word Bre | Striver's S | Short sim | Kth Large | Heap Sort | Online C | (919) ma | Problem | Single Nu | Mega Job | Count Be | + | - | X | Update |

Compile time poly... rohit952513 - LeetC... HTML ASCII Refere... vector - C++ Refere... SDE Off-campus Pl... Striver DP Series : D... Leetcode 75 Questi... Resize image in cm,...

Contest Home Quiz Coding Problems

Time Left : 00:01:15:30

Problem Submissions

Concatenating 'abccc' and 'abc' we get 'abcccabc' in which all the characters have even frequency. So the pair (2,3) also forms a beautiful string.

Example 2:

Input:
n = 7
box = {aaaa, abba, ccc, caa, cbba, acaac, bcb}

Output:
16

Explanation:
There are total 16 pairs which form beautiful strings.
Pairs which **do not** form beautiful string are (1,5), (2,5), (3,6), (4,6) and (6,7).

Your task:
You don't need to read input or print anything. Your task is to

C++ (g++ 5.4)

```
1 //{{ }} Driver Code Ends
2 //User function Template for C++
3
4 class Solution{
5 public:
6     long long noOfPairs(vector<string> &box)
7     {
8         // Write your code here.
9         long long count = 0;
10        int n = box.size();
11
12        for (int i = 0; i < n - 1; i++) {
13            unordered_map<char, int> freq;
14
15            // Calculate the frequency of characters in string at index i
16            for (char c : box[i])
17                freq[c]++;
18
19            for (int j = i + 1; j < n; j++) {
20                // Calculate the frequency of characters in string at index j
21                for (char c : box[j])
22                    freq[c]++;
23
24                int oddCount = 0, evenCount = 0;
25
26                // Count the number of characters with odd and even frequencies
27                for (auto it = freq.begin(); it != freq.end(); it++)
28                {
29                    if (it->second % 2 == 0)
30                        evenCount++;
31                    else
32                        oddCount++;
33                }
34
35                if (oddCount >= 1 && evenCount >= 1)
36                    count++;
37            }
38        }
39    }
40
41
42 }
```

Custom Input Compile & Run Submit

31°C Partly cloudy

Search

b W S 2 9:22 PM 7/5/2023 ENG IN

Problems Courses Get Hired Events </POTD

Practice

40%

Problem Editorial Submissions Comments

Prerequisite Tasks

Medium Accuracy: 50.33% Submissions: 29733 Points: 4

There are a total of N tasks, labeled from 0 to N-1. Some tasks may have prerequisites, for example to do task 0 you have to first complete task 1, which is expressed as a pair: [0, 1]

Given the total number of tasks N and a list of prerequisite pairs P, find if it is possible to finish all tasks.

Example 1:

Input:
N = 4, P = 3
prerequisites = {{1,0},{2,1},{3,2}}
Output: .
Yes
Explanation:
To do task 1 you should have completed task 0, and to do task 2 you should have finished task 1, and to do task 3 you should have finished task 2. So it is possible.

C++ (g++ 5.4) Average Time: 26m

```
1 // Driver Code Ends
2 class Solution {
3 public:
4     bool isPossible(int N, vector<pair<int, int>>& prerequisites) {
5         // Code here
6     }
7 };
8 // Driver Code Ends
```

[0, 1] → | → 0

0 → | → 1 → 2 → 3

True

Input:

Compile & Run

See your GFG solution

Steiner's Sol is ulta

→ G-24. Course Schedule I and II | Pre-requisite Tasks | Topological Sort

1.30 Course Schedule - I & II

1, 0
2, 1
3, 2

1

TUF

► G-24. Course Schedule I and II | Pre-requisite Tasks | Topological Sort

1.75

Course Schedule - I & II



1:46 / 11:31



Course Schedule - I & II

1, 0
2, 1
3, 2

✓
—
—

3 2 1 0

yes

1	2
4	3
2	4
4	1



IIT bombay Resu | Word Break II | Striver's SDE She | Heap Sort - Data | Single Number II | youtube.com/watch?v=WAOfKpxYHR8&list=PLgUwDviBlf0rGEWe64KWa... | Maximum Num... | Queue using Sta... | +

Compile time poly... rohit952513 - LeetC... HTML ASCII Refere... vector - C++ Refere... SDE Off-campus Pl... Striver DP Series : D... Leetcode 75 Questi... Resize image in cm,...

YouTube IN graph striver

Course Schedule - I & II

1, 0 ✓
2, 1 ✓
3, 2 ✓

3 2 1 0 Yes

1 2
4 3
2 4
1

1 → 2 → 4 → 3

Striver's Graph Series | Plat...

take U forward - 16 / 30

G-1. Introduction to Graph | Types... 13:43 take U forward

G-2. Graph Representation ... 16:04 take U forward

G-4. What are Connected... 7:07 take U forward

G-5. Breadth-First Search... 19:39 take U forward

G-24. Course Schedule I and II | Pre-requisite Tasks | Topological Sort

take U forward

28°C Near record

Search

11:26 AM 7/8/2023

Press Esc to exit full screen

Yes

3 2 1 0

1, 0 ✓
2, 1 ✓
3, 2 ✓

1 2 ✓
4 3 ✓
2 4 ✓
4 ! X

NO

Diagram showing a sequence of nodes 1, 2, 3, 4 connected by arrows. Node 1 points to node 2, node 2 points to node 3, node 3 points to node 4, and node 4 points back to node 1, forming a cycle.

The video call interface features a dark background with white text and graphics. On the left, there's a hand-drawn diagram of two tables and some numbers. The top table has three rows: [1, 0], [2, 1], and [3, 2], each marked with a checkmark. Below it is another table with four rows: [1, 2], [4, 3], [2, 4], and [4, !], with the last row marked with a red X. To the right of these tables are the numbers 3, 2, 1, and 0. A large orange arrow points from the text 'NO' towards the sequence of numbers. To the right of the numbers is a hand-drawn diagram of four nodes (circles) labeled 1, 2, 3, and 4, connected in a cycle: 1 → 2 → 3 → 4 → 1. On the right side of the screen, a young man with a beard and short hair, wearing a red t-shirt, is gesturing with his hands while speaking. He is holding a white pen in his right hand. The video call interface includes standard controls like 'Esc' to exit full screen and a 'Yes' button.

G-24. Course Schedule I and II | Pre-requisite Tasks | Topological Sort



1.75

Topo Sort → Yes

$u \rightarrow v$

(u, v)

u v

14 of 21



5:04 / 11:31



G-24. Course Schedule I and II | Pre-requisite Tasks | Topological Sort



1.75

Topo Sort → Yes

$u \rightarrow v$

(u, v)

u v



14 of 21



5:04 / 11:31



Topo Sort → Yes
(DAG)
(Directed Graph)
(Acyclic Graph)
 $u \rightarrow v$
 (u, v)

Possible → Yes
No



G-24. Course Schedule I and II | Pre-requisite Tasks | Topological Sort

Problems Courses Get Hired Events </> POTD Practice 40!

1.75 </> Problem Editorial Submissions Comments

Prerequisite Tasks

Medium Accuracy

There are a total of 1 prerequisites, for example, which is expressed as Given the total number, it is possible to finish.

Example 1:

Input:
N = 4, P = 3
prerequisites = {
 {1, 0},
 {1, 2},
 {2, 0},
 {2, 3}
}

Output:
Yes

Explanation:
To do task 1 you need to do task 0, and to do task 2, you need to have finished tasks 0 and 1. You should have finished tasks 0, 1, 2 to do task 3.

Example 2:

```
C++ (g++ 5.4) Average Time: 26m
public:
    bool isPossible(int V, vector<pair<int, int>>& prerequisites) {
        vector<int> adj[V];
        for(auto it : prerequisites) {
            adj[it.first].push_back(it.second);
        }

        int indegree[V] = {0};
        for(int i = 0; i < V; i++) {
            for(auto it : adj[i]) {
                indegree[it]++;
            }
        }

        queue<int> q;
        for(int i = 0; i < V; i++) {
            if(indegree[i] == 0) {
                q.push(i);
            }
        }

        vector<int> topo;
        while(!q.empty()) {
            int node = q.front();
            q.pop();
            topo.push_back(node);
            // node is in your topo sort
            // so please remove it from the indegree

            for(auto it : adj[node]) {
                indegree[it]--;
                if(indegree[it] == 0) q.push(it);
            }
        }

        if(topo.size() == V) return true;
        return false;
    }
}
```

Input: 9:07 / 11:31

Compile & Run CC Settings Full Screen

G-24. Course Schedule I and II | Pre-requisite Tasks | Topological Sort

1.75 Problem

Prerequisite Tasks

Medium Accuracy

There are a total of 1 prerequisites, for example, which is expressed as Given the total number, it is possible to finish.

Example 1:

Input:
N = 4, P = 3
prerequisites = {{1, 0}, {1, 2}, {2, 0}}
Output:
Yes
Explanation:
To do task 1 you need to do task 0, and to do task 2, you have finished tasks 0 and 1. So, you should have finished tasks 0, 1, and 2 to do task 3.

Example 2:

```
class Solution {
public boolean isPossible(int V, List<List<Integer>> prerequisites) {
    // Form a graph
    ArrayList<ArrayList<Integer>> adj = new ArrayList<ArrayList<Integer>>();
    for(int i = 0; i < V; i++) {
        adj.add(new ArrayList<Integer>());
    }
    int m = prerequisites.length;
    for(int i = 0; i < m; i++) {
        adj.get(prerequisites.get(i).get(0)).add(prerequisites.get(i).get(1));
    }

    int indegree[] = new int[V];
    for(int i = 0; i < V; i++) {
        for(int it : adj.get(i)) {
            indegree[it]++;
        }
    }

    Queue<Integer> q = new LinkedList<Integer>();
    for(int i = 0; i < V; i++) {
        if(indegree[i] == 0) {
            q.add(i);
        }
    }

    List<Integer> topo = new ArrayList<Integer>();
    // O(V + E)
    while(!q.isEmpty()) {
        int node = q.peek();
        q.remove();
        topo.add(node);
        // node is in your topo sort
        // so please remove it from the indegree
        for(int it : adj.get(node)) {
            indegree[it]--;
            if(indegree[it] == 0) q.add(it);
        }
    }

    if(topo.size() == V) return true;
    return false;
}
}
```

C++ (g++ 5.4) Average Time: 26m

```
for(auto it : prerequisites) {
    adj[it.first].push_back(it.second);
}

int indegree[V] = {0};
for(int i = 0; i < V; i++) {
    for(auto it : adj[i]) {
        indegree[it]++;
    }
}

queue<int> q;
for(int i = 0; i < V; i++) {
    if(indegree[i] == 0) {
        q.push(i);
    }
}

vector<int> topo;
while(!q.empty()) {
    int node = q.front();
    q.pop();
    topo.push_back(node);
    // node is in your topo sort
    // so please remove it from the indegree

    for(auto it : adj[node]) {
        indegree[it]--;
        if(indegree[it] == 0) q.push(it);
    }
}

if(topo.size() == V) return true;
return false;
};

// Driver Code Ends
```

Input: 9:11 / 11:31

Output: CC Gears Full Screen

G-26. Alien Dictionary - Topological Sort

Problems Courses Get Hired Events <>POTD Practice 409

1.00 Problem Editorial Submissions Comments C++ (g++ 5.4)

Given a sorted dictionary of an alien language having N words and k starting alphabets of standard dictionary. Find the order of characters in the alien language.

Note: Many orders may be possible for a particular test case, thus you may return any valid order and output will be 1 if the order of string returned by the function is correct else 0 denoting incorrect string returned.

Example 1:

Input:
N = 5, K = 4
dict = {"baa", "abcd", "abca", "cab", "cad"}
Output:
1

Explanation:
Here order of characters is
'b', 'd', 'a', 'c' Note that words are sorted
and in the given language "baa" comes before
"abcd", therefore 'b' is before 'a' in output.
Similarly we can find other orders.

Example 2:

Input:
N = 3, K = 3



```
1 // } Driver Code Ends
8 // User Function Template for C++
9
10 class Solution{
11     public:
12     string findOrder(string dict[], int N, int K) {
13         //code here
14     }
15 };
16 // } Driver Code Ends
```

Settings

⇒ G-26. Alien Dictionary - Topological Sort



Alien Dictionary

2.20

Alien

$n=4$

baa

abcd

abca

cab

cad

a b c d e ... myz

abca

ab cd

ba a

ca b

ca d

find out the alien order?

abcd



3:06 / 20:53



—

$n=4$

baa
abcd
abca
cab
cad

abca
abcd
baa
cab
cad

find out the alien order?

abcd

(b d a c)

→ alien order.



⇒ G-26. Alien Dictionary - Topological Sort

2.20

baa
abcd
abca
cab
cad

abca
abcd
baa
cab
cad

abcd

Topo Sort

find out the alien order?

(b d a c)

→ alien order.



➡ G-26. Alien Dictionary - Topological Sort

2.20

Alien Dictionary
Alien

$k = 4$

baa
abcd
abca
cab
cad

a b c d
0 1 2 3

baa
abcd

b → a



⇒ G-26. Alien Dictionary - Topological Sort

2.20

Alien Dictionary

Alien

$k = 4$

baa
abcd
abca
cab
cad

a b c d
0 1 2 3

a b c a
c a b



15 of 21

8:05 / 20:53



⇒ G-26. Alien Dictionary - Topological Sort

2.20

Alien

$k=4$

baa

abcd

abca

cab

cad

a b c d
0 1 2 3

cab
cad



15 of 21



8:42 / 20:53



⇒ G-26. Alien Dictionary - Topological Sort

2.20



D Gr



9:42 / 20:53



➡ G-26. Alien Dictionary - Topological Sort

2.20

Alien Dictionary

Alien

K = 5

baa

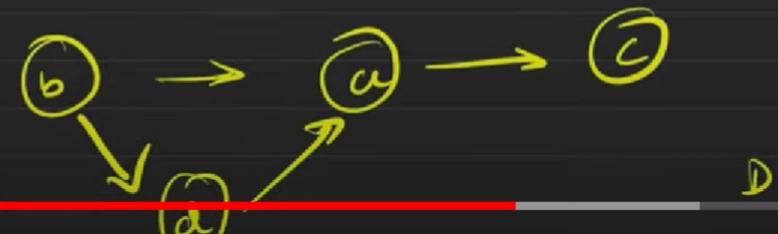
abcd

abca

cab

cad

a b c d e
0 1 2 3 4



Alicen

$i=0$ baa
 $i=1$ abcd
 $i=2$ abca
 $i=3$ cab
cad

$K = 5$

0 1 2 3 4

Topo Sort

$S_1 = \text{arr}[i] \rightarrow$
 $S_2 = \text{arr}[i+1] \rightarrow$



Alicen

i=0	baa
1	abcd
2	abca
3	cab
	cad

K = 5

0 1 2 3 4

Topo Sort

$$S_1 = \text{arr}[q_i] \rightarrow \boxed{a b c a}$$
$$S_2 = \text{arr}[q_{i+1}] \rightarrow \boxed{c a b}$$


DG₂



G-26. Alien Dictionary - Topological Sort

Practice

```
P 2.20
    topo.add(node);
    // node is in your topo sort
    // so please remove it from the indegree
}
return topo;
}
public String findOrder(String [] dict, int N, int K)
{
    List<List<Integer>> adj = new ArrayList<>();
    for(int l = 0;l<K;l++) {
        adj.add(new ArrayList<>());
    }

    for(int l = 0;l<N-1;l++) {
        String s1 = dict[l];
        String s2 = dict[l+1];
        int len = Math.min(s1.length(), s2.length());
        for(int ptr = 0; ptr < len; ptr++) {
            if(s1.charAt(ptr) != s2.charAt(ptr)) {
                adj.get(s1.charAt(ptr) - 'a').add(s2.charAt(ptr) - 'a');
                break;
            }
        }
    }

    List<Integer> topo = topoSort(K, adj);
    String ans = "";
    for(int lt : topo) {
        ans = ans + (char)(lt + (int)('a'));
    }
    return ans;
}
```

Similarly we can find other orders.

Example 2:

Input:

N = 3, K = 2

```
C++ (g++ 5.4) ▶
1 * // } Driver Code Ends
8 // User Function Template for C++
9
10 * class Solution{
11     public:
12     string findOrder(string dict[], int N, int K) {
13         vector<int>adj[K];
14         for(int i = 0;i<N-1;i++) {
15             string s1 = dict[i];
16             string s2 = dict[i+1];
17             int len = min(s1.size(), s2.size());
18             for(int ptr = 0; ptr < len; ptr++) {
19                 if(s1[ptr] != s2[ptr]) {
20                     adj[s1[ptr] - 'a'].push_back(s2[ptr] - 'a');
21                     break;
22                 }
23             }
24         }
25     }
26     |
27 }
28 };
29 * // } Driver Code Ends
```



Time: 16:39 / 20:53 Sun Compile & Run



G-26. Alien Dictionary - Topological Sort
 Problems Courses Get Hired Events <> POTD

 409

2.20
c/c++ Prob

```

class Solution {
    private List<Integer> topoSort(int V, List<List<Integer>> adj) {
        int[] indegree = new int[V];
        for(int i = 0; i < V; i++) {
            for(int it : adj.get(i)) {
                indegree[it]++;
            }
        }
        Queue<Integer> q = new LinkedList<Integer>();
        for(int i = 0; i < V; i++) {
            if(indegree[i] == 0) {
                q.add(i);
            }
        }
        List<Integer> topo = new ArrayList<Integer>();
        while(!q.isEmpty()) {
            int node = q.peek();
            q.remove();
            topo.add(node);
            // node is in your topo sort
            // so please remove it from the indegree
            for(int it : adj.get(node)) {
                indegree[it]--;
                if(indegree[it] == 0) q.add(it);
            }
        }
        return topo;
    }

    public String findOrder(String[] dict, int N, int K) {
        List<List<Integer>> adj = new ArrayList<List<Integer>>();
        for(int i = 0; i < K; i++) {
            adj.add(new ArrayList<Integer>());
        }
        for(int i = 0; i < N-1; i++) {
            String s1 = dict[i];
            String s2 = dict[i+1];
            int len = Math.min(s1.length(), s2.length());
            for(int ptr = 0; ptr < len; ptr++) {
                if(s1.charAt(ptr) != s2.charAt(ptr)) {
                    adj.get(s1.charAt(ptr) - 'a').add(s2.charAt(ptr) - 'a');
                    break;
                }
            }
        }
        List<Integer> topo = topoSort(K, adj);
        String ans = "";
        for(int it : topo) {
            ans = ans + (char)(it + (int)('a'));
        }
        return ans;
    }
}

```

C++ (g++ 5.4) ▶

// Driver Code Ends
// User Function Template for C++

```

class Solution{
private:
    vector<int> topoSort(int V, vector<int> adj[]) {
        int indegree[V] = {0};
        for(int i = 0; i < V; i++) {
            for(auto it: adj[i]) {
                indegree[it]++;
            }
        }
        queue<int> q;
        for(int i = 0; i < V; i++) {
            if(indegree[i] == 0) {
                q.push(i);
            }
        }
        vector<int> topo;
        while(!q.empty()) {
            int node = q.front();
            q.pop();
            topo.push_back(node);
            // node is in your topo sort
            // so please remove it from the indegree
            for(auto it : adj[node]) {
                indegree[it]--;
                if(indegree[it] == 0) q.push(it);
            }
        }
        return topo;
    }

public:
    string findOrder(string dict[], int N, int K) {
        vector<int> adj[K];
        for(int i = 0; i < N-1; i++) {

```

◀ ▶ ⏪ ⏩
16:54 / 20:53
CC
Settings
#

Problems Courses Get Hired Events </> POTD

Prob

```

class Solution {
    private List<Integer> topoSort(int V, List<List<Integer>> adj) {
        int[] indegree = new int[V];
        for(int i = 0; i < V; i++) {
            for(int it : adj.get(i)) {
                indegree[it]++;
            }
        }
        Queue<Integer> q = new LinkedList<>();
        for(int i = 0; i < V; i++) {
            if(indegree[i] == 0) {
                q.add(i);
            }
        }
        List<Integer> topo = new ArrayList<>();
        while(!q.isEmpty()) {
            int node = q.peek();
            q.remove();
            topo.add(node);
            // node is in your topo sort
            // so please remove it from the indegree
            for(int it : adj.get(node)) {
                indegree[it]--;
                if(indegree[it] == 0) q.add(it);
            }
        }
        return topo;
    }
    public String findOrder(String dict[], int N, int K) {
        List<List<Integer>> adj = new ArrayList<>();
        for(int i = 0; i < K; i++) {
            adj.add(new ArrayList<>());
        }
        for(int i = 0; i < N - 1; i++) {
            String s1 = dict[i];
            String s2 = dict[i + 1];
            int len = Math.min(s1.length(), s2.length());
            for(int ptr = 0; ptr < len; ptr++) {
                if(s1.charAt(ptr) != s2.charAt(ptr)) {
                    adj.get(s1.charAt(ptr) - 'a').add(s2.charAt(ptr) - 'a');
                    break;
                }
            }
        }
        List<Integer> topo = topoSort(K, adj);
        String ans = "";
        for(int it : topo) {
            ans += (char)(it + (int)('a'));
        }
        return ans;
    }
}

```

Note: Many or may return any returned by th returned.

Example 1:

Input:
N = 5, K =
dict = {"baa"}
Output:
1

Explanation:
Here order
'b', 'd', 'a', '
and in the {
"abcd", ther
Similarly we

Example 2:

Input:
N = 3, K =
dict = {"caa", "aaa", "aab"}
the alien langu

Practice

409

```

C++ (g++ 5.4) ~
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63

```

public:

```

string findOrder(string dict[], int N, int K) {
    vector<int> adj[K];
    for(int i = 0; i < N - 1; i++) {
        string s1 = dict[i];
        string s2 = dict[i + 1];
        int len = min(s1.size(), s2.size());
        for(int ptr = 0; ptr < len; ptr++) {
            if(s1[ptr] != s2[ptr]) {
                adj[s1[ptr] - 'a'].push_back(s2[ptr] - 'a');
                break;
            }
        }
    }
}

```

Driver Code Ends

Compile &



Problems Courses Get Hired Events </>POTD

</> Prob

```

class Solution {
    private List<Integer> topoSort(int V, List<List<Integer>> adj) {
        int[] indegree = new int[V];
        for(int i = 0; i < V; i++) {
            for(int it : adj.get(i)) {
                indegree[it]++;
            }
        }

        Queue<Integer> q = new LinkedList<>();
        for(int i = 0; i < V; i++) {
            if(indegree[i] == 0)
                q.add(i);
        }

        List<Integer> topo = new ArrayList<>();
        while(!q.isEmpty()) {
            int node = q.peek();
            q.remove();
            topo.add(node);
            // node is in your topo sort
            // so please remove it from the indegree
            for(int it : adj.get(node)) {
                indegree[it]--;
                if(indegree[it] == 0)
                    q.add(it);
            }
        }

        return topo;
    }

    public String findOrder(String[] dict, int N, int K) {
        List<List<Integer>> adj = new ArrayList<>();
        for(int i = 0; i < K; i++)
            adj.add(new ArrayList<>());

        for(int i = 0; i < K - 1; i++) {
            String s1 = dict[i];
            String s2 = dict[i + 1];
            int len = Math.min(s1.length(), s2.length());
            for(int ptr = 0; ptr < len; ptr++) {
                if(s1.charAt(ptr) != s2.charAt(ptr)) {
                    adj.get(i).add(s1.charAt(ptr) - 'a');
                    adj.get(i + 1).add(s2.charAt(ptr) - 'a');
                    break;
                }
            }
        }

        List<Integer> topo = topoSort(K, adj);
        String ans = "";
        for(int it : topo) {
            ans = ans + (char)(it + (int)('a'));
        }

        return ans;
    }
}

```

Expected Time: maximum length
Expected Space:
Constraints:
 $1 \leq N, M \leq 300$
 $1 \leq K \leq 26$
 $1 \leq \text{Length of } v$

[View Bookmarks](#)

Company T

Output Will

Compilation Results

prog.cpp
Solution
prog.cpp
std::vector
std::string

Expected Output:

AE Practice

409

```

C++ (g++ 5.4) +
28     vector<int> topo;
29     while(!q.empty()) {
30         int node = q.front();
31         q.pop();
32         topo.push_back(node);
33         // node is in your topo sort
34         // so please remove it from the indegree
35
36         for(auto it : adj[node]) {
37             indegree[it]--;
38             if(indegree[it] == 0) q.push(it);
39         }
40
41     }
42
43     return topo;
44 }
public:
string findOrder(string dict[], int N, int K) {
    vector<int>adj[K];
    for(int i = 0; i < N - 1; i++) {
        string s1 = dict[i];
        string s2 = dict[i + 1];
        int len = min(s1.size(), s2.size());
        for(int ptr = 0; ptr < len; ptr++) {
            if(s1[ptr] != s2[ptr]) {
                adj[i].push_back(s1[ptr] - 'a');
                adj[i + 1].push_back(s2[ptr] - 'a');
                break;
            }
        }
    }

    vector<int> topo = topoSort(K, adj);
    string ans = "";
    for(auto it : topo) {
        ans = ans + char(it + 'a');
    }
    return ans;
}

```

Compile &



Problems Courses Get Hired Events </> POTD

</> Prob

```

class Solution {
    private List<Integer> topoSort(int V, List<List<Integer>> adj) {
        int[] indegree = new int[V];
        for(int i = 0; i < V; i++) {
            for(int it : adj.get(i)) {
                indegree[it]++;
            }
        }

        Queue<Integer> q = new LinkedList<>();
        for(int i = 0; i < V; i++) {
            if(indegree[i] == 0) {
                q.add(i);
            }
        }

        List<Integer> topo = new ArrayList<>();
        while(!q.isEmpty()) {
            int node = q.peek();
            q.remove();
            topo.add(node);
            // node is in your topo sort
            // so please remove it from the indegree
            for(int it : adj.get(node)) {
                indegree[it]--;
                if(indegree[it] == 0) q.add(it);
            }
        }

        return topo;
    }

    public String findOrder(String[] dict, int N, int K) {
        List<List<Integer>> adj = new ArrayList<>();
        for(int i = 0; i < K; i++) {
            adj.add(new ArrayList<>());
        }

        for(int i = 0; i < N - 1; i++) {
            String s1 = dict[i];
            String s2 = dict[i + 1];
            int len = Math.min(s1.length(), s2.length());
            for(int ptr = 0; ptr < len; ptr++) {
                if(s1.charAt(ptr) != s2.charAt(ptr)) {
                    adj.get(s1.charAt(ptr) - 'a').add(s2.charAt(ptr) - 'a');
                    break;
                }
            }
        }

        List<Integer> topo = topoSort(K, adj);
        String ans = "";
        for(int it : topo) {
            ans = ans + (char)(it + (int)('a'));
        }

        return ans;
    }
}

```

Expected Output

Output Will Be

Compilation Results

prog.cpp
Solution
prog.cpp
std::vector
std::string

Expected Output

Practice

409

```

C++ (g++ 5.4) ~

28     vector<int> topo;
29     while(!q.empty()) {
30         int node = q.front();
31         q.pop();
32         topo.push_back(node);
33         // node is in your topo sort
34         // so please remove it from the indegree
35
36         for(auto it : adj[node]) {
37             indegree[it]--;
38             if(indegree[it] == 0) q.push(it);
39         }
40
41     }
42
43     return topo;
44 }
public:
45 string findOrder(string dict[], int N, int K) {
46     vector<int>adj[K];
47     for(int i = 0; i < N - 1; i++) {
48         string s1 = dict[i];
49         string s2 = dict[i + 1];
50         int len = min(s1.size(), s2.size());
51         for(int ptr = 0; ptr < len; ptr++) {
52             if(s1[ptr] != s2[ptr]) {
53                 adj[s1[ptr] - 'a'].push_back(s2[ptr] - 'a');
54                 break;
55             }
56         }
57     }
58
59     vector<int> topo = topoSort(K, adj);
60     string ans = "";
61     for(auto it : topo) {
62         ans = ans + char(it + 'a');
63     }
64     return ans;
65 }
66 } // Driver Code Ends

```

Compile &



⇒ G-26. Alien Dictionary - Topological Sort

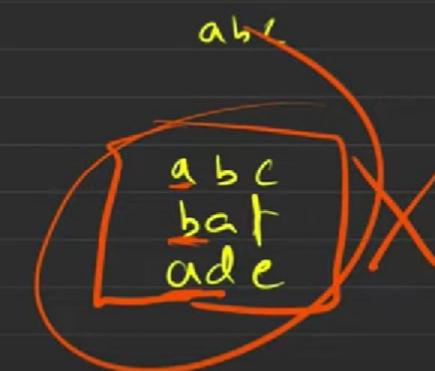
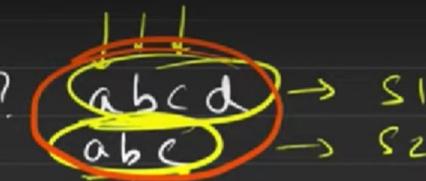
2.20

When the order is not possible?

No!

~~abcde~~ → Top Sort X abc abed

a < b < a



G-27. Shortest Path in Directed Acyclic Graph - Topological Sort

1.0 Problem Submissions Doubt Support Practice

Shortest Path in Directed Acyclic Graph

Medium Accuracy: 50.0% Submissions: 10 Points: 4

Given a Directed acyclic Graph. Find the shortest path from src(0) to all the vertex.

Example:

```
Input:  
n = 6, m= 7  
adj=[[0,1,2],[0,4,1],[4,5,4],[4,2,2],[1,2,3],[2,3,6],[5,3]  
src=0  
Output:  
0 2 3 6 1 5
```

Your Task:

You don't need to print or input anything. Complete the function shortestPath() which takes an directed acyclic graph, an integer n and an integer src as the input parameters and returns an integer, denoting the vector of distance from src to all nodes.

Constraint:

```
1<=n,m<=100  
1<=adj[i][j]<=100
```

Expected Time Complexity: O(N+E), where N is the number of nodes and E is edges

Expected Space Complexity: O(N)

View Bookmarked Problems

Discussions (1 Threads) Most Recent

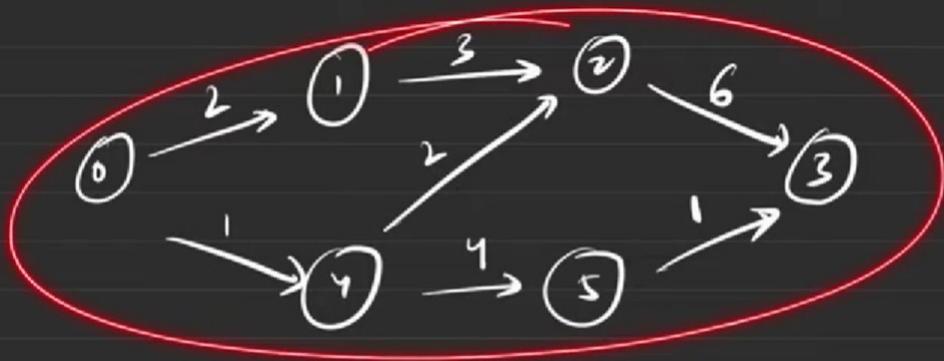
Most Recent

Average Time: 20m

Compile & Run

CC

```
1. // } Driver Code Ends  
8 // User Function Template for C++  
9 class Solution {  
10 public:  
11     vector<int> shortestPath( int N, int M, vector<int> edges){  
12         // code here  
13     }  
14 };  
15  
16  
17 // } Driver Code Ends
```

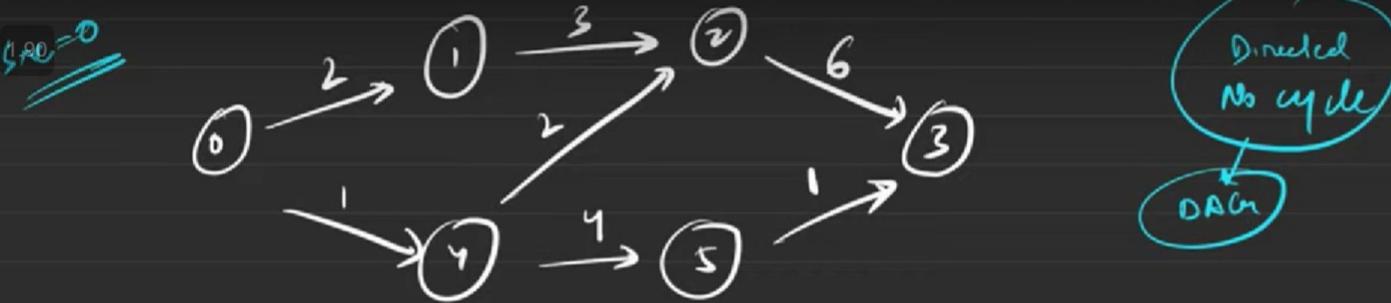


DAG

dist \rightarrow 0 2 3 6 1 5



→ G-27. Shortest Path in Directed Acyclic Graph - Topological Sort



dist → 0 2 3 6 1 5

shortest distance to all.

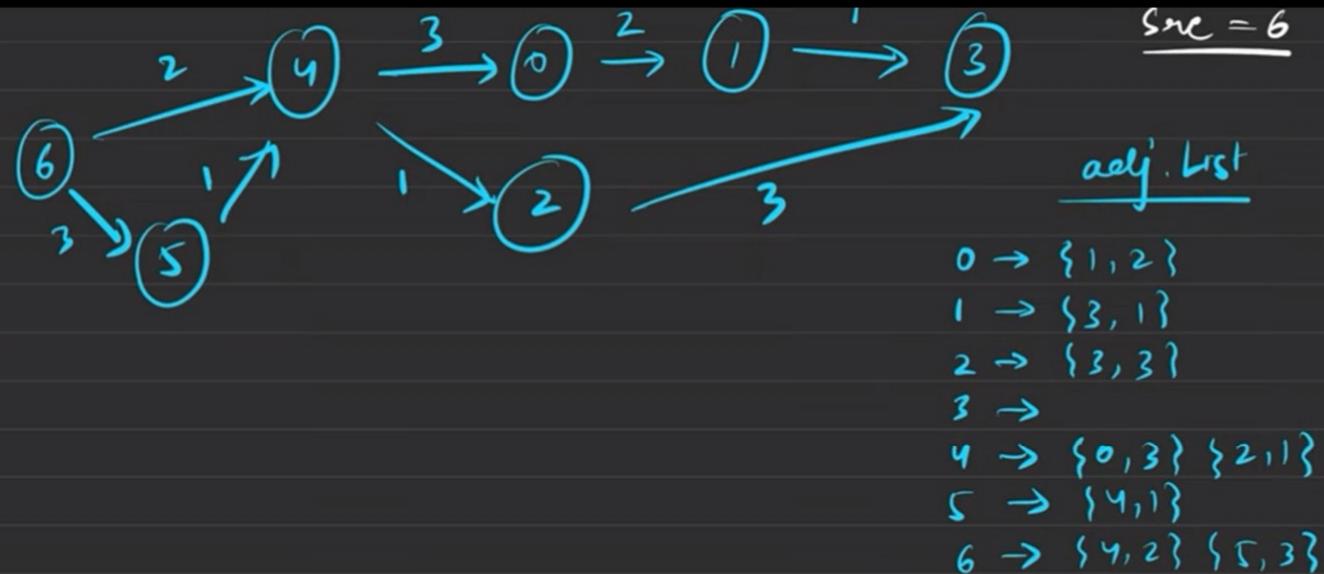
n-1



2:04 / 26:35



1.90



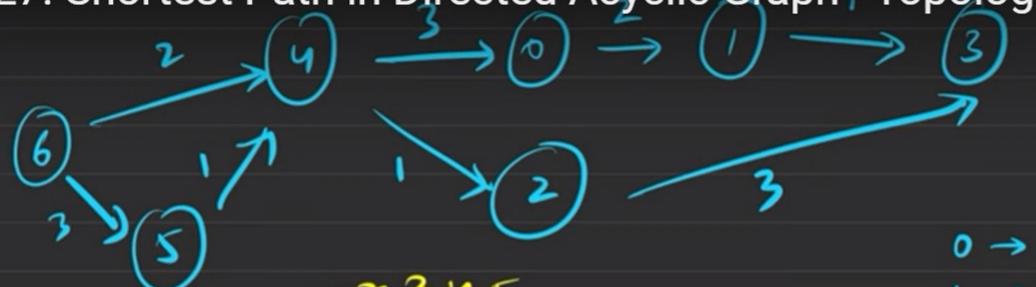
Step 1 → Do a Topo
S_u



→ G-27. Shortest Path in Directed Acyclic Graph - Topological Sort

DAG

1.90



$\text{fun}(i=0 \rightarrow 6)$

$\tilde{y}([v_i, s_i])$
 $dys(i)$

s_u
2
0
1
3
stack

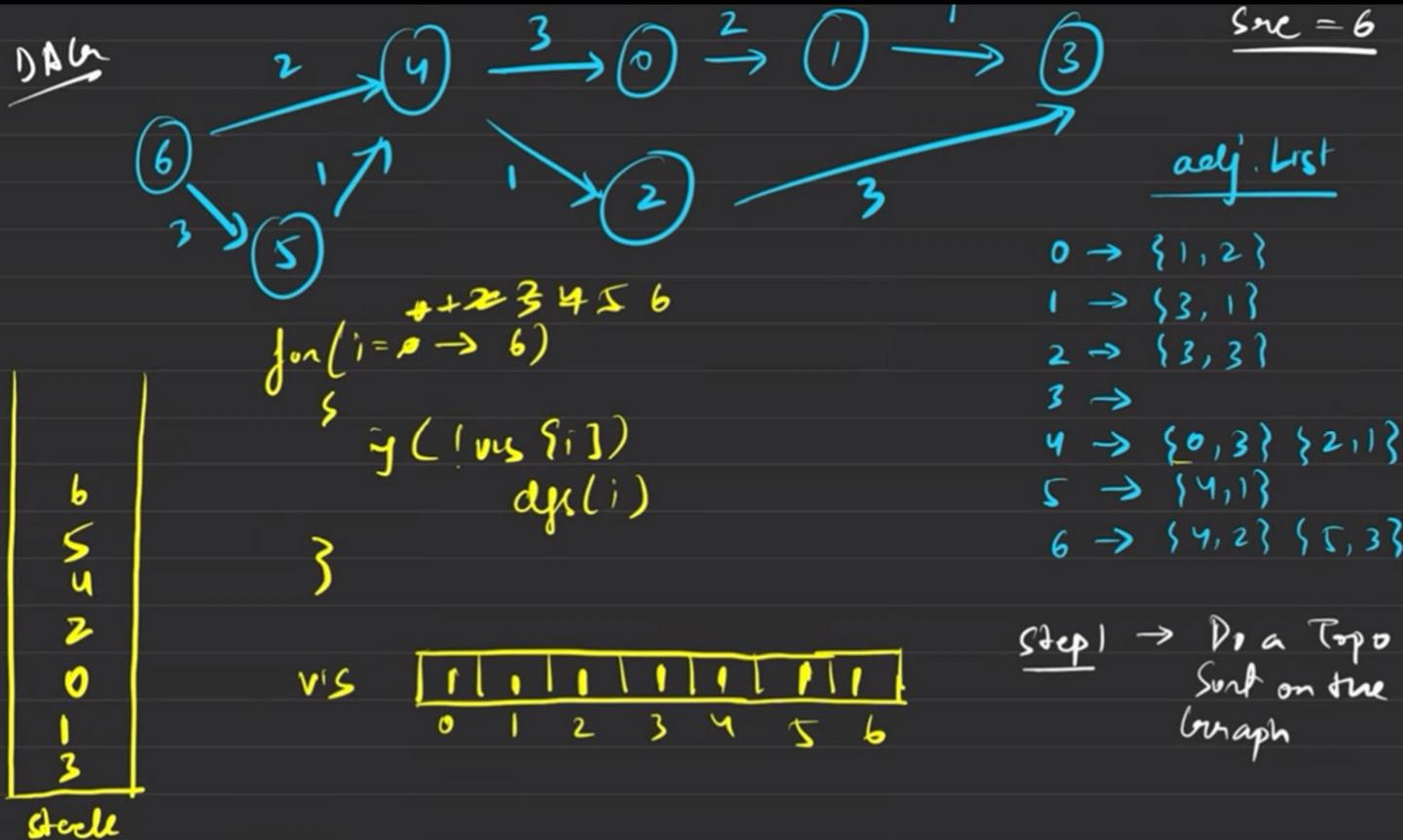
vis
0 1 2 3 4 5 6

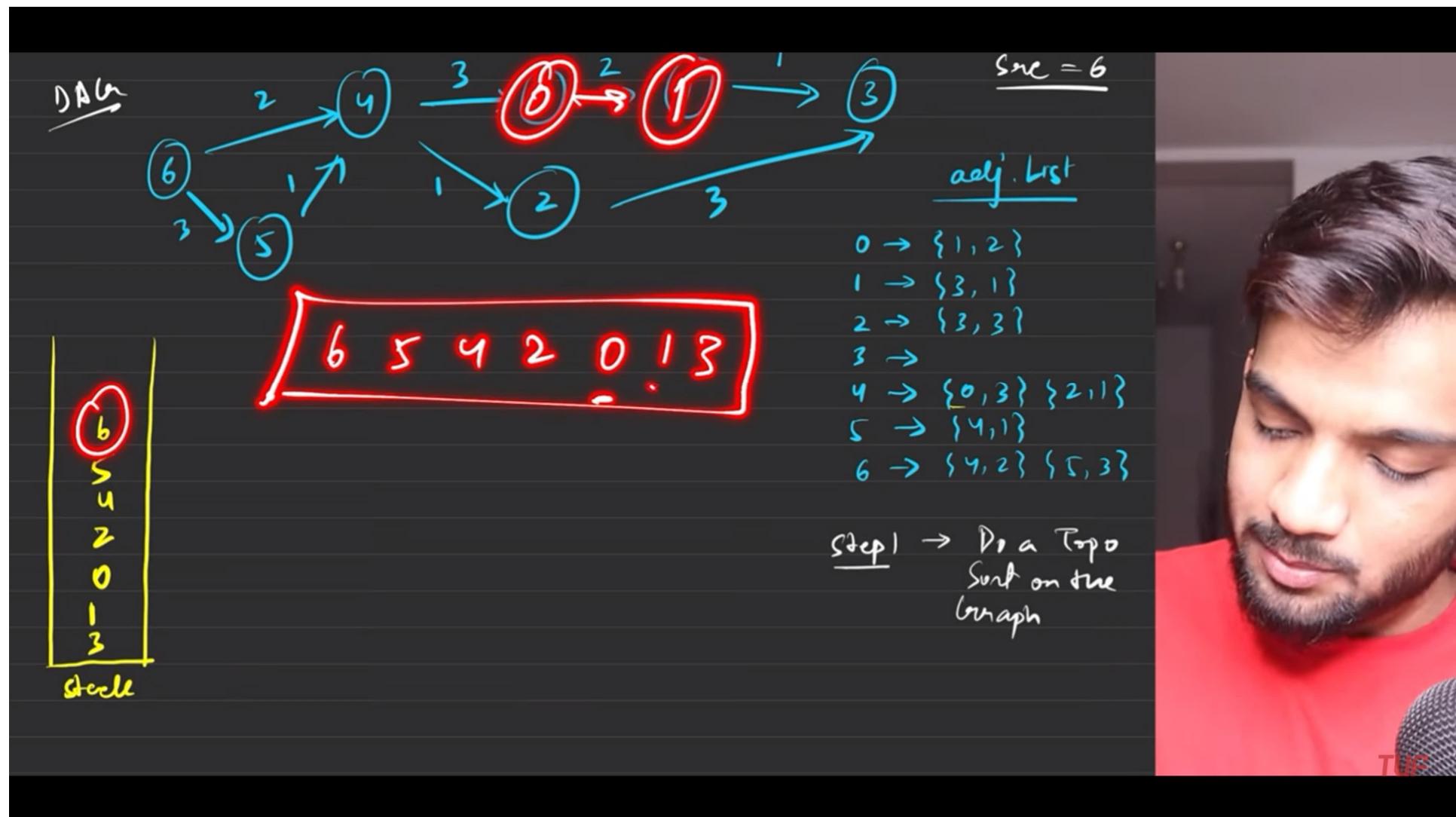
adj. List

$0 \rightarrow \{1, 2\}$
 $1 \rightarrow \{3, 1\}$
 $2 \rightarrow \{3, 3\}$
 $3 \rightarrow$
 $4 \rightarrow \{0, 3\} \{2, 1\}$
 $5 \rightarrow \{4, 1\}$
 $6 \rightarrow \{4, 2\} \{5, 3\}$

Step 1 → Do a Topo
Sort on the
graph







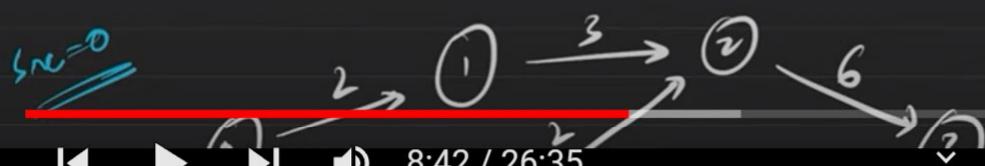
⇒ G-27. Shortest Path in Directed Acyclic Graph - Topological Sort

1.90
6
5
4
3
2
0
1
3
Stack

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

Step 1 → Do a Topo
Sort on the
graph

Step → Take one
node out of
stack &
remove edges



Directed
No cycle



DAG



size = 6

adj. List

- 0 → {1, 2}
- 1 → {3, 1}
- 2 → {3, 3}
- 3 →
- 4 → {0, 3} {2, 1}
- 5 → {4, 1}
- 6 → {4, 2} {5, 3}

b
S
u
z
0
1
3
Stack

$$\text{dist}[7] = [\alpha | \alpha | \alpha | \alpha | \alpha | \alpha | \alpha]$$

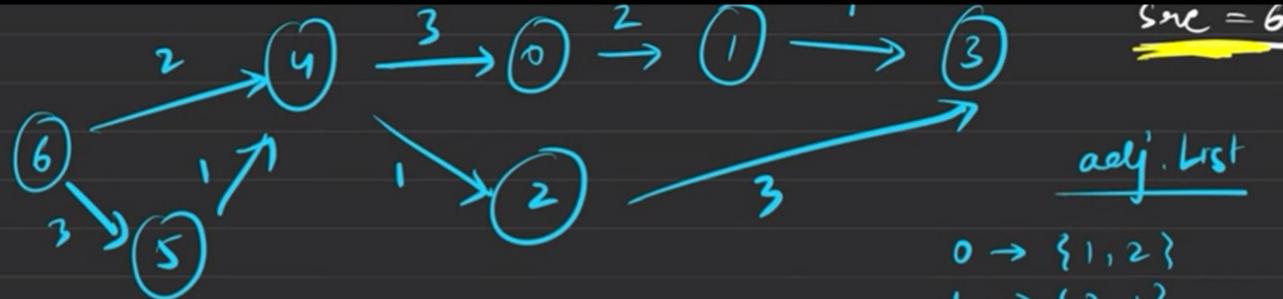
0	1	2	3	4	5	6
---	---	---	---	---	---	---

Step 1 → Do a Topo Sort on the Graph

Step → Take the nodes out of stack &



DAL



suc = 6

adj. List

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

node = 6, dist = 0

+3

+2

node = 5
dist = 3

node = 4
dist = 2

Stack

6
5
4
2
0
1
3

dist[7] = [d | d | d | d | d | d | d]
0 1 2 3 4 5 6

Step 1 → Do a Topo Sort on the Graph

Step → Take the nodes out of stack &



G-27. Shortest Path in Directed Acyclic Graph - Topological Sort

DAG
1.90



node = 5, dist = 3

✓ + 1

node = 4
dist = 4

Stack

5
2
0
1
3

dist[7] = [α | α | α | α | α | α | α]
0 1 2 3 4 5 6

Step 1 → Do a Topo
Sort on the
Graph

Step → Take the
nodes out of
stack &

adj. List

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



12:10 / 26:35





node = 0, dist = 5

+ 2

node = 1
dist = 7



dist[7] = [5 7 7 6 2 3 0]
0 1 2 3 4 5 6

Step 1 → Do a Topo Sort on the Graph

Step → Take one node out of stack &



G-27. Shortest Path in Directed Acyclic Graph - Topological Sort

DAG
1.90

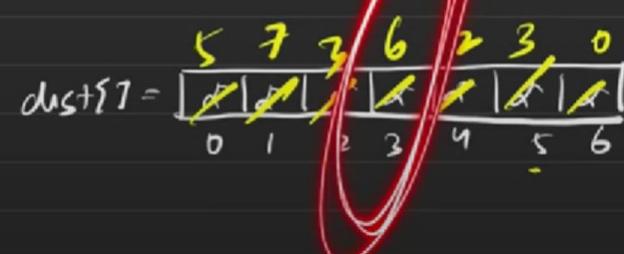


node = 1 , dist = 7

+1

node = 3

dist = 9



Step 1 → Do a Topo Sort on the graph

Step → Take one node out of stack &



14:55 / 26:35



G-27. Shortest Path in Directed Acyclic Graph - Topological Sort



1:90 problems

Shortest Medium

Given a Di vertex.

Example:

Input:
n = 6,
adj=[[0,1,2,3,4,5],
[1,2,3,4,5,6],
[2,3,4,5,6,7],
[3,4,5,6,7,8],
[4,5,6,7,8,9],
[5,6,7,8,9,10],
[6,7,8,9,10,11],
[7,8,9,10,11,12],
[8,9,10,11,12,13],
[9,10,11,12,13,14],
[10,11,12,13,14,15],
[11,12,13,14,15,16],
[12,13,14,15,16,17],
[13,14,15,16,17,18],
[14,15,16,17,18,19],
[15,16,17,18,19,20],
[16,17,18,19,20,21],
[17,18,19,20,21,22],
[18,19,20,21,22,23],
[19,20,21,22,23,24],
[20,21,22,23,24,25],
[21,22,23,24,25,26],
[22,23,24,25,26,27],
[23,24,25,26,27,28],
[24,25,26,27,28,29],
[25,26,27,28,29,30],
[26,27,28,29,30,31],
[27,28,29,30,31,32],
[28,29,30,31,32,33],
[29,30,31,32,33,34],
[30,31,32,33,34,35],
[31,32,33,34,35,36],
[32,33,34,35,36,37],
[33,34,35,36,37,38],
[34,35,36,37,38,39],
[35,36,37,38,39,40],
[36,37,38,39,40,41],
[37,38,39,40,41,42],
[38,39,40,41,42,43],
[39,40,41,42,43,44],
[40,41,42,43,44,45],
[41,42,43,44,45,46],
[42,43,44,45,46,47],
[43,44,45,46,47,48],
[44,45,46,47,48,49],
[45,46,47,48,49,50],
[46,47,48,49,50,51],
[47,48,49,50,51,52],
[48,49,50,51,52,53],
[49,50,51,52,53,54],
[50,51,52,53,54,55],
[51,52,53,54,55,56],
[52,53,54,55,56,57],
[53,54,55,56,57,58],
[54,55,56,57,58,59],
[55,56,57,58,59,60],
[56,57,58,59,60,61],
[57,58,59,60,61,62],
[58,59,60,61,62,63],
[59,60,61,62,63,64],
[60,61,62,63,64,65],
[61,62,63,64,65,66],
[62,63,64,65,66,67],
[63,64,65,66,67,68],
[64,65,66,67,68,69],
[65,66,67,68,69,70],
[66,67,68,69,70,71],
[67,68,69,70,71,72],
[68,69,70,71,72,73],
[69,70,71,72,73,74],
[70,71,72,73,74,75],
[71,72,73,74,75,76],
[72,73,74,75,76,77],
[73,74,75,76,77,78],
[74,75,76,77,78,79],
[75,76,77,78,79,80],
[76,77,78,79,80,81],
[77,78,79,80,81,82],
[78,79,80,81,82,83],
[79,80,81,82,83,84],
[80,81,82,83,84,85],
[81,82,83,84,85,86],
[82,83,84,85,86,87],
[83,84,85,86,87,88],
[84,85,86,87,88,89],
[85,86,87,88,89,90],
[86,87,88,89,90,91],
[87,88,89,90,91,92],
[88,89,90,91,92,93],
[89,90,91,92,93,94],
[90,91,92,93,94,95],
[91,92,93,94,95,96],
[92,93,94,95,96,97],
[93,94,95,96,97,98],
[94,95,96,97,98,99],
[95,96,97,98,99,100],
[96,97,98,99,100,101],
[97,98,99,100,101,102],
[98,99,100,101,102,103],
[99,100,101,102,103,104],
[100,101,102,103,104,105],
[101,102,103,104,105,106],
[102,103,104,105,106,107],
[103,104,105,106,107,108],
[104,105,106,107,108,109],
[105,106,107,108,109,110],
[106,107,108,109,110,111],
[107,108,109,110,111,112],
[108,109,110,111,112,113],
[109,110,111,112,113,114],
[110,111,112,113,114,115],
[111,112,113,114,115,116],
[112,113,114,115,116,117],
[113,114,115,116,117,118],
[114,115,116,117,118,119],
[115,116,117,118,119,120],
[116,117,118,119,120,121],
[117,118,119,120,121,122],
[118,119,120,121,122,123],
[119,120,121,122,123,124],
[120,121,122,123,124,125],
[121,122,123,124,125,126],
[122,123,124,125,126,127],
[123,124,125,126,127,128],
[124,125,126,127,128,129],
[125,126,127,128,129,130],
[126,127,128,129,130,131],
[127,128,129,130,131,132],
[128,129,130,131,132,133],
[129,130,131,132,133,134],
[130,131,132,133,134,135],
[131,132,133,134,135,136],
[132,133,134,135,136,137],
[133,134,135,136,137,138],
[134,135,136,137,138,139],
[135,136,137,138,139,140],
[136,137,138,139,140,141],
[137,138,139,140,141,142],
[138,139,140,141,142,143],
[139,140,141,142,143,144],
[140,141,142,143,144,145],
[141,142,143,144,145,146],
[142,143,144,145,146,147],
[143,144,145,146,147,148],
[144,145,146,147,148,149],
[145,146,147,148,149,150],
[146,147,148,149,150,151],
[147,148,149,150,151,152],
[148,149,150,151,152,153],
[149,150,151,152,153,154],
[150,151,152,153,154,155],
[151,152,153,154,155,156],
[152,153,154,155,156,157],
[153,154,155,156,157,158],
[154,155,156,157,158,159],
[155,156,157,158,159,160],
[156,157,158,159,160,161],
[157,158,159,160,161,162],
[158,159,160,161,162,163],
[159,160,161,162,163,164],
[160,161,162,163,164,165],
[161,162,163,164,165,166],
[162,163,164,165,166,167],
[163,164,165,166,167,168],
[164,165,166,167,168,169],
[165,166,167,168,169,170],
[166,167,168,169,170,171],
[167,168,169,170,171,172],
[168,169,170,171,172,173],
[169,170,171,172,173,174],
[170,171,172,173,174,175],
[171,172,173,174,175,176],
[172,173,174,175,176,177],
[173,174,175,176,177,178],
[174,175,176,177,178,179],
[175,176,177,178,179,180],
[176,177,178,179,180,181],
[177,178,179,180,181,182],
[178,179,180,181,182,183],
[179,180,181,182,183,184],
[180,181,182,183,184,185],
[181,182,183,184,185,186],
[182,183,184,185,186,187],
[183,184,185,186,187,188],
[184,185,186,187,188,189],
[185,186,187,188,189,190],
[186,187,188,189,190,191],
[187,188,189,190,191,192],
[188,189,190,191,192,193],
[189,190,191,192,193,194],
[190,191,192,193,194,195],
[191,192,193,194,195,196],
[192,193,194,195,196,197],
[193,194,195,196,197,198],
[194,195,196,197,198,199],
[195,196,197,198,199,200],
[196,197,198,199,200,201],
[197,198,199,200,201,202],
[198,199,200,201,202,203],
[199,200,201,202,203,204],
[200,201,202,203,204,205],
[201,202,203,204,205,206],
[202,203,204,205,206,207],
[203,204,205,206,207,208],
[204,205,206,207,208,209],
[205,206,207,208,209,210],
[206,207,208,209,210,211],
[207,208,209,210,211,212],
[208,209,210,211,212,213],
[209,210,211,212,213,214],
[210,211,212,213,214,215],
[211,212,213,214,215,216],
[212,213,214,215,216,217],
[213,214,215,216,217,218],
[214,215,216,217,218,219],
[215,216,217,218,219,220],
[216,217,218,219,220,221],
[217,218,219,220,221,222],
[218,219,220,221,222,223],
[219,220,221,222,223,224],
[220,221,222,223,224,225],
[221,222,223,224,225,226],
[222,223,224,225,226,227],
[223,224,225,226,227,228],
[224,225,226,227,228,229],
[225,226,227,228,229,230],
[226,227,228,229,230,231],
[227,228,229,230,231,232],
[228,229,230,231,232,233],
[229,230,231,232,233,234],
[230,231,232,233,234,235],
[231,232,233,234,235,236],
[232,233,234,235,236,237],
[233,234,235,236,237,238],
[234,235,236,237,238,239],
[235,236,237,238,239,240],
[236,237,238,239,240,241],
[237,238,239,240,241,242],
[238,239,240,241,242,243],
[239,240,241,242,243,244],
[240,241,242,243,244,245],
[241,242,243,244,245,246],
[242,243,244,245,246,247],
[243,244,245,246,247,248],
[244,245,246,247,248,249],
[245,246,247,248,249,250],
[246,247,248,249,250,251],
[247,248,249,250,251,252],
[248,249,250,251,252,253],
[249,250,251,252,253,254],
[250,251,252,253,254,255],
[251,252,253,254,255,256],
[252,253,254,255,256,257],
[253,254,255,256,257,258],
[254,255,256,257,258,259],
[255,256,257,258,259,260],
[256,257,258,259,260,261],
[257,258,259,260,261,262],
[258,259,260,261,262,263],
[259,260,261,262,263,264],
[260,261,262,263,264,265],
[261,262,263,264,265,266],
[262,263,264,265,266,267],
[263,264,265,266,267,268],
[264,265,266,267,268,269],
[265,266,267,268,269,270],
[266,267,268,269,270,271],
[267,268,269,270,271,272],
[268,269,270,271,272,273],
[269,270,271,272,273,274],
[270,271,272,273,274,275],
[271,272,273,274,275,276],
[272,273,274,275,276,277],
[273,274,275,276,277,278],
[274,275,276,277,278,279],
[275,276,277,278,279,280],
[276,277,278,279,280,281],
[277,278,279,280,281,282],
[278,279,280,281,282,283],
[279,280,281,282,283,284],
[280,281,282,283,284,285],
[281,282,283,284,285,286],
[282,283,284,285,286,287],
[283,284,285,286,287,288],
[284,285,286,287,288,289],
[285,286,287,288,289,290],
[286,287,288,289,290,291],
[287,288,289,290,291,292],
[288,289,290,291,292,293],
[289,290,291,292,293,294],
[290,291,292,293,294,295],
[291,292,293,294,295,296],
[292,293,294,295,296,297],
[293,294,295,296,297,298],
[294,295,296,297,298,299],
[295,296,297,298,299,300],
[296,297,298,299,300,301],
[297,298,299,300,301,302],
[298,299,300,301,302,303],
[299,300,301,302,303,304],
[300,301,302,303,304,305],
[301,302,303,304,305,306],
[302,303,304,305,306,307],
[303,304,305,306,307,308],
[304,305,306,307,308,309],
[305,306,307,308,309,310],
[306,307,308,309,310,311],
[307,308,309,310,311,312],
[308,309,310,311,312,313],
[309,310,311,312,313,314],
[310,311,312,313,314,315],
[311,312,313,314,315,316],
[312,313,314,315,316,317],
[313,314,315,316,317,318],
[314,315,316,317,318,319],
[315,316,317,318,319,320],
[316,317,318,319,320,321],
[317,318,319,320,321,322],
[318,319,320,321,322,323],
[319,320,321,322,323,324],
[320,321,322,323,324,325],
[321,322,323,324,325,326],
[322,323,324,325,326,327],
[323,324,325,326,327,328],
[324,325,326,327,328,329],
[325,326,327,328,329,330],
[326,327,328,329,330,331],
[327,328,329,330,331,332],
[328,329,330,331,332,333],
[329,330,331,332,333,334],
[330,331,332,333,334,335],
[331,332,333,334,335,336],
[332,333,334,335,336,337],
[333,334,335,336,337,338],
[334,335,336,337,338,339],
[335,336,337,338,339,340],
[336,337,338,339,340,341],
[337,338,339,340,341,342],
[338,339,340,341,342,343],
[339,340,341,342,343,344],
[340,341,342,343,344,345],
[341,342,343,344,345,346],
[342,343,344,345,346,347],
[343,344,345,346,347,348],
[344,345,346,347,348,349],
[345,346,347,348,349,350],
[346,347,348,349,350,351],
[347,348,349,350,351,352],
[348,349,350,351,352,353],
[349,350,351,352,353,354],
[350,351,352,353,354,355],
[351,352,353,354,355,356],
[352,353,354,355,356,357],
[353,354,355,356,357,358],
[354,355,356,357,358,359],
[355,356,357,358,359,360],
[356,357,358,359,360,361],
[357,358,359,360,361,362],
[358,359,360,361,362,363],
[359,360,361,362,363,364],
[360,361,362,363,364,365],
[361,362,363,364,365,366],
[362,363,364,365,366,367],
[363,364,365,366,367,368],
[364,365,366,367,368,369],
[365,366,367,368,369,370],
[366,367,368,369,370,371],
[367,368,369,370,371,372],
[368,369,370,371,372,373],
[369,370,371,372,373,374],
[370,371,372,373,374,375],
[371,372,373,374,375,376],
[372,373,374,375,376,377],
[373,374,375,376,377,378],
[374,375,376,377,378,379],
[375,376,377,378,379,380],
[376,377,378,379,380,381],
[377,378,379,380,381,382],
[378,379,380,381,382,383],
[379,380,381,382,383,384],
[380,381,382,383,384,385],
[381,382,383,384,385,386],
[382,383,384,385,386,387],
[383,384,385,386,387,388],
[384,385,386,387,388,389],
[385,386,387,388,389,390],
[386,387,388,389,390,391],
[387,388,389,390,391,392],
[388,389,390,391,392,393],
[389,390,391,392,393,394],
[390,391,392,393,394,395],
[391,392,393,394,395,396],
[392,393,394,395,396,397],
[393,394,395,396,397,398],
[394,395,396,397,398,399],
[395,396,397,398,399,400],
[396,397,398,399,400,401],
[397,398,399,400,401,402],
[398,399,400,401,402,403],
[399,400,401,402,403,404],
[400,401,402,403,404,405],
[401,402,403,404,405,406],
[402,403,404,405,406,407],
[403,404,405,406,407,408],
[404,405,406,407,408,409],
[405,406,407,408,409,410],
[406,407,408,409,410,411],
[407,408,409,410,411,412],
[408,409,410,411,412,413],
[409,410,411,412,413,414],
[410,411,412,413,414,415],
[411,412,413,414,415,416],
[412,413,414,415,416,417],
[413,414,415,416,417,418],
[414,415,416,417,418,419],

G-27. Shortest Path in Directed Acyclic Graph - Topological Sort



1:90 problems

Shortest Path in DAG

Medium

Given a DAG, find the shortest path from source vertex to target vertex.

Example:

Input:
n = 6,
adj = [[0, 1, 2], [1, 2, 3], [2, 3, 4], [3, 4, 5], [4, 5, 0], [5, 0, 1]]
src=0
Output:
0 2 3 1 4 5

Your Task

You don't need to print anything. You just need to implement the shortestPath() which takes the input parameters from src to target and returns the shortest path.

Constraints

1<=n,m<=1000
1<=adj[i][j]<=1000

Expected Output

edges

Expected Input

View Bookmarks

Discuss

```
C++ (g++ 5.4) Practice Test against custom input
1 // Driver Code Ends
8 // User Function Template for C++
9 class Solution {
10 private:
11     void topoSort(int node, vector<pair<int,int>> adj[], int vis[], stack<int> &st) {
12         vis[node] = 1;
13         for(int i = 0;i<adj[node].size();i++) {
14             int v = adj[node].get(i).first;
15             if(vis[v] == 0) {
16                 topoSort(v, adj, vis, st);
17             }
18             st.push(node);
19         }
20     }
21     public:
22         vector<int> shortestPath( int N,int M, vector<int> edges[]){
23             vector<pair<int,int>> adj[N];
24             for(int i = 0;i<M;i++) {
25                 int u = edges[i][0];
26                 int v = edges[i][1];
27                 int wt = edges[i][2];
28                 adj[u].push_back({v, wt});
29             }
30             // Find the topo sort
31             int vis[N] = {0};
32             stack<int> st;
33             for(int i = 0;i<N;i++) {
34                 if(!vis[i]) {
35                     topoSort(i, adj, vis, st);
36                 }
37             }
38             // Step 2 do the distance thing
39             vector<int> dist(N);
40             while(!st.isEmpty()) {
41                 int node = st.peek();
42                 st.pop();
43                 for(int i = 0;i<adj[node].size();i++) {
44                     int v = adj[node].get(i).first;
45                     int wt = adj[node].get(i).second;
46                     if(dist[node] + wt < dist[v]) {
47                         dist[v] = wt + dist[node];
48                     }
49                 }
50             }
51             return dist;
52         }
53     };
54 }
```

20:43 / 26:35

G-27. Shortest Path in Directed Acyclic Graph - Topological Sort

1:19 Problem

Shortest Medium

Given a Di vertex.

Example:

Input:
n = 6,
adj=[[0,1,2,3,4,5],
[1,2,3,4,5,6],
[2,3,4,5,6,7],
[3,4,5,6,7,8],
[4,5,6,7,8,9],
[5,6,7,8,9,10],
[6,7,8,9,10,11],
[7,8,9,10,11,12],
[8,9,10,11,12,13],
[9,10,11,12,13,14],
[10,11,12,13,14,15],
[11,12,13,14,15,16],
[12,13,14,15,16,17],
[13,14,15,16,17,18],
[14,15,16,17,18,19],
[15,16,17,18,19,20],
[16,17,18,19,20,21],
[17,18,19,20,21,22],
[18,19,20,21,22,23],
[19,20,21,22,23,24],
[20,21,22,23,24,25],
[21,22,23,24,25,26],
[22,23,24,25,26,27],
[23,24,25,26,27,28],
[24,25,26,27,28,29],
[25,26,27,28,29,30],
[26,27,28,29,30,31],
[27,28,29,30,31,32],
[28,29,30,31,32,33],
[29,30,31,32,33,34],
[30,31,32,33,34,35],
[31,32,33,34,35,36],
[32,33,34,35,36,37],
[33,34,35,36,37,38],
[34,35,36,37,38,39],
[35,36,37,38,39,40],
[36,37,38,39,40,41],
[37,38,39,40,41,42],
[38,39,40,41,42,43],
[39,40,41,42,43,44],
[40,41,42,43,44,45],
[41,42,43,44,45,46],
[42,43,44,45,46,47],
[43,44,45,46,47,48],
[44,45,46,47,48,49],
[45,46,47,48,49,50],
[46,47,48,49,50,51],
[47,48,49,50,51,52],
[48,49,50,51,52,53],
[49,50,51,52,53,54],
[50,51,52,53,54,55],
[51,52,53,54,55,56],
[52,53,54,55,56,57],
[53,54,55,56,57,58],
[54,55,56,57,58,59],
[55,56,57,58,59,60],
[56,57,58,59,60,61],
[57,58,59,60,61,62],
[58,59,60,61,62,63],
[59,60,61,62,63,64],
[60,61,62,63,64,65],
[61,62,63,64,65,66],
[62,63,64,65,66,67],
[63,64,65,66,67,68],
[64,65,66,67,68,69],
[65,66,67,68,69,70],
[66,67,68,69,70,71],
[67,68,69,70,71,72],
[68,69,70,71,72,73],
[69,70,71,72,73,74],
[70,71,72,73,74,75],
[71,72,73,74,75,76],
[72,73,74,75,76,77],
[73,74,75,76,77,78],
[74,75,76,77,78,79],
[75,76,77,78,79,80],
[76,77,78,79,80,81],
[77,78,79,80,81,82],
[78,79,80,81,82,83],
[79,80,81,82,83,84],
[80,81,82,83,84,85],
[81,82,83,84,85,86],
[82,83,84,85,86,87],
[83,84,85,86,87,88],
[84,85,86,87,88,89],
[85,86,87,88,89,90],
[86,87,88,89,90,91],
[87,88,89,90,91,92],
[88,89,90,91,92,93],
[89,90,91,92,93,94],
[90,91,92,93,94,95],
[91,92,93,94,95,96],
[92,93,94,95,96,97],
[93,94,95,96,97,98],
[94,95,96,97,98,99],
[95,96,97,98,99,100],
[96,97,98,99,100,101],
[97,98,99,100,101,102],
[98,99,100,101,102,103],
[99,100,101,102,103,104],
[100,101,102,103,104,105],
[101,102,103,104,105,106],
[102,103,104,105,106,107],
[103,104,105,106,107,108],
[104,105,106,107,108,109],
[105,106,107,108,109,110],
[106,107,108,109,110,111],
[107,108,109,110,111,112],
[108,109,110,111,112,113],
[109,110,111,112,113,114],
[110,111,112,113,114,115],
[111,112,113,114,115,116],
[112,113,114,115,116,117],
[113,114,115,116,117,118],
[114,115,116,117,118,119],
[115,116,117,118,119,120],
[116,117,118,119,120,121],
[117,118,119,120,121,122],
[118,119,120,121,122,123],
[119,120,121,122,123,124],
[120,121,122,123,124,125],
[121,122,123,124,125,126],
[122,123,124,125,126,127],
[123,124,125,126,127,128],
[124,125,126,127,128,129],
[125,126,127,128,129,130],
[126,127,128,129,130,131],
[127,128,129,130,131,132],
[128,129,130,131,132,133],
[129,130,131,132,133,134],
[130,131,132,133,134,135],
[131,132,133,134,135,136],
[132,133,134,135,136,137],
[133,134,135,136,137,138],
[134,135,136,137,138,139],
[135,136,137,138,139,140],
[136,137,138,139,140,141],
[137,138,139,140,141,142],
[138,139,140,141,142,143],
[139,140,141,142,143,144],
[140,141,142,143,144,145],
[141,142,143,144,145,146],
[142,143,144,145,146,147],
[143,144,145,146,147,148],
[144,145,146,147,148,149],
[145,146,147,148,149,150],
[146,147,148,149,150,151],
[147,148,149,150,151,152],
[148,149,150,151,152,153],
[149,150,151,152,153,154],
[150,151,152,153,154,155],
[151,152,153,154,155,156],
[152,153,154,155,156,157],
[153,154,155,156,157,158],
[154,155,156,157,158,159],
[155,156,157,158,159,160],
[156,157,158,159,160,161],
[157,158,159,160,161,162],
[158,159,160,161,162,163],
[159,160,161,162,163,164],
[160,161,162,163,164,165],
[161,162,163,164,165,166],
[162,163,164,165,166,167],
[163,164,165,166,167,168],
[164,165,166,167,168,169],
[165,166,167,168,169,170],
[166,167,168,169,170,171],
[167,168,169,170,171,172],
[168,169,170,171,172,173],
[169,170,171,172,173,174],
[170,171,172,173,174,175],
[171,172,173,174,175,176],
[172,173,174,175,176,177],
[173,174,175,176,177,178],
[174,175,176,177,178,179],
[175,176,177,178,179,180],
[176,177,178,179,180,181],
[177,178,179,180,181,182],
[178,179,180,181,182,183],
[179,180,181,182,183,184],
[180,181,182,183,184,185],
[181,182,183,184,185,186],
[182,183,184,185,186,187],
[183,184,185,186,187,188],
[184,185,186,187,188,189],
[185,186,187,188,189,190],
[186,187,188,189,190,191],
[187,188,189,190,191,192],
[188,189,190,191,192,193],
[189,190,191,192,193,194],
[190,191,192,193,194,195],
[191,192,193,194,195,196],
[192,193,194,195,196,197],
[193,194,195,196,197,198],
[194,195,196,197,198,199],
[195,196,197,198,199,200],
[196,197,198,199,200,201],
[197,198,199,200,201,202],
[198,199,200,201,202,203],
[199,200,201,202,203,204],
[200,201,202,203,204,205],
[201,202,203,204,205,206],
[202,203,204,205,206,207],
[203,204,205,206,207,208],
[204,205,206,207,208,209],
[205,206,207,208,209,210],
[206,207,208,209,209,211],
[207,208,209,209,211,212],
[208,209,209,211,212,213],
[209,209,211,212,213,214],
[210,209,211,212,213,215],
[211,209,212,213,214,216],
[212,209,213,214,215,217],
[213,209,214,215,216,218],
[214,209,215,216,217,219],
[215,209,216,217,218,220],
[216,209,217,218,219,221],
[217,209,218,219,220,222],
[218,209,219,220,221,223],
[219,209,220,221,222,224],
[220,209,221,222,223,225],
[221,209,222,223,224,226],
[222,209,223,224,225,227],
[223,209,224,225,226,228],
[224,209,225,226,227,229],
[225,209,226,227,228,230],
[226,209,227,228,229,231],
[227,209,228,229,230,232],
[228,209,229,230,231,233],
[229,209,230,231,232,234],
[230,209,231,232,233,235],
[231,209,232,233,234,236],
[232,209,233,234,235,237],
[233,209,234,235,236,238],
[234,209,235,236,237,239],
[235,209,236,237,238,240],
[236,209,237,238,239,241],
[237,209,238,239,240,242],
[238,209,239,240,241,243],
[239,209,240,241,242,244],
[240,209,241,242,243,245],
[241,209,242,243,244,246],
[242,209,243,244,245,247],
[243,209,244,245,246,248],
[244,209,245,246,247,249],
[245,209,246,247,248,250],
[246,209,247,248,249,251],
[247,209,248,249,250,252],
[248,209,249,250,251,253],
[249,209,250,251,252,254],
[250,209,251,252,253,255],
[251,209,252,253,254,256],
[252,209,253,254,255,257],
[253,209,254,255,256,258],
[254,209,255,256,257,259],
[255,209,256,257,258,260],
[256,209,257,258,259,261],
[257,209,258,259,260,262],
[258,209,259,260,261,263],
[259,209,260,261,262,264],
[260,209,261,262,263,265],
[261,209,262,263,264,266],
[262,209,263,264,265,267],
[263,209,264,265,266,268],
[264,209,265,266,267,269],
[265,209,266,267,268,270],
[266,209,267,268,269,271],
[267,209,268,269,270,272],
[268,209,269,270,271,273],
[269,209,270,271,272,274],
[270,209,271,272,273,275],
[271,209,272,273,274,276],
[272,209,273,274,275,277],
[273,209,274,275,276,278],
[274,209,275,276,277,279],
[275,209,276,277,278,280],
[276,209,277,278,279,281],
[277,209,278,279,280,282],
[278,209,279,280,281,283],
[279,209,280,281,282,284],
[280,209,281,282,283,285],
[281,209,282,283,284,286],
[282,209,283,284,285,287],
[283,209,284,285,286,288],
[284,209,285,286,287,289],
[285,209,286,287,288,290],
[286,209,287,288,289,291],
[287,209,288,289,290,292],
[288,209,289,290,291,293],
[289,209,290,291,292,294],
[290,209,291,292,293,295],
[291,209,292,293,294,296],
[292,209,293,294,295,297],
[293,209,294,295,296,298],
[294,209,295,296,297,299],
[295,209,296,297,298,300],
[296,209,297,298,299,301],
[297,209,298,299,300,302],
[298,209,299,300,301,303],
[299,209,300,301,302,304],
[300,209,301,302,303,305],
[301,209,302,303,304,306],
[302,209,303,304,305,307],
[303,209,304,305,306,308],
[304,209,305,306,307,309],
[305,209,306,307,308,310],
[306,209,307,308,309,311],
[307,209,308,309,310,312],
[308,209,309,310,311,313],
[309,209,310,311,312,314],
[310,209,311,312,313,315],
[311,209,312,313,314,316],
[312,209,313,314,315,317],
[313,209,314,315,316,318],
[314,209,315,316,317,319],
[315,209,316,317,318,320],
[316,209,317,318,319,321],
[317,209,318,319,320,322],
[318,209,319,320,321,323],
[319,209,320,321,322,324],
[320,209,321,322,323,325],
[321,209,322,323,324,326],
[322,209,323,324,325,327],
[323,209,324,325,326,328],
[324,209,325,326,327,329],
[325,209,326,327,328,330],
[326,209,327,328,329,331],
[327,209,328,329,330,332],
[328,209,329,330,331,333],
[329,209,330,331,332,334],
[330,209,331,332,333,335],
[331,209,332,333,334,336],
[332,209,333,334,335,337],
[333,209,334,335,336,338],
[334,209,335,336,337,339],
[335,209,336,337,338,340],
[336,209,337,338,339,341],
[337,209,338,339,340,342],
[338,209,339,340,341,343],
[339,209,340,341,342,344],
[340,209,341,342,343,345],
[341,209,342,343,344,346],
[342,209,343,344,345,347],
[343,209,344,345,346,348],
[344,209,345,346,347,349],
[345,209,346,347,348,350],
[346,209,347,348,349,351],
[347,209,348,349,350,352],
[348,209,349,350,351,353],
[349,209,350,351,352,354],
[350,209,351,352,353,355],
[351,209,352,353,354,356],
[352,209,353,354,355,357],
[353,209,354,355,356,358],
[354,209,355,356,357,359],
[355,209,356,357,358,360],
[356,209,357,358,359,361],
[357,209,358,359,360,362],
[358,209,359,360,361,363],
[359,209,360,361,362,364],
[360,209,361,362,363,365],
[361,209,362,363,364,366],
[362,209,363,364,365,367],
[363,209,364,365,366,368],
[364,209,365,366,367,369],
[365,209,366,367,368,370],
[366,209,367,368,369,371],
[367,209,368,369,370,372],
[368,209,369,370,371,373],
[369,209,370,371,372,374],
[370,209,371,372,373,375],
[371,209,372,373,374,376],
[372,209,373,374,375,377],
[373,209,374,375,376,378],
[374,209,375,376,377,379],
[375,209,376,377,378,380],
[376,209,377,378,379,381],
[377,209,378,379,380,382],
[378,209,379,380,381,383],
[379,209,380,381,382,384],
[380,209,381,382,383,385],
[381,209,382,383,384,386],
[382,209,383,384,385,387],
[383,209,384,385,386,388],
[384,209,385,386,387,389],
[385,209,386,387,388,390],
[386,209,387,388,389,391],
[387,209,388,389,390,392],
[388,209,389,390,391,393],
[389,209,390,391,392,394],
[390,209,391,392,393,395],
[391,209,392,393,394,396],
[392,209,393,394,395,397],
[393,209,394,395,396,398],
[394,209,395,396,397,399],
[395,209,396,397,398,400],
[396,209,397,398,399,401],
[397,209,398,399,400,402],
[398,209,399,400,401,403],
[399,209,400,401,402,404],
[400,209,401,402,403,405],
[401,209,402,403,404,406],
[402,209,403,404,405,407],
[403,209,404,405,406,408],
[404,209,405,406,407,409],
[405,209,406,407,408,410],
[406,209,407,408,409,411],
[407,209,408,409,410,412],
[408,209,409,410,411,413],
[409,209,410,411,412,414],
[410,209,411,412,413,415],
[411,209,412,413,414,416],
[412,209,413,414,415,417],
[413,209,414,415,416,418],
[414,209,415,416,417,419],
[415,209,416,417,418,420],
[416,209,

G-27. Shortest Path in Directed Acyclic Graph - Topological Sort

Practice

1:90 Problem Submissions Doubt Support

Shortest Path in Directed Acyclic Graph

Medium Accuracy: 50.0% Submissions: 10 Points: 4

Given a Directed acyclic Graph. Find the shortest path from src(0) to all the vertex.

Example:

```
Input:  
n = 6, m= 7  
adj=[[0,1,2],[0,4,1],[4,5,4],[4,2,2],[1,2,3],[2,3,6],[5,3]  
src=0  
Output:  
0 2 3 6 1 5
```

Your Task:

You don't need to print or input anything. Complete the function shortestPath() which takes an directed acyclic graph, an integer n and an integer src as the input parameters and returns an integer, denoting the vector of distance from src to all nodes.

Constraint:

```
1<=n,m<=100  
1<=adj[i][j]<=100
```

Expected Time Complexity: O(N+E), where N is the number of nodes and E is edges

Expected Space Complexity: O(N)

View Bookmarked Problems

Discussions (1 Threads) Most Recent

```
C++ (g++ 5.4) Test against custom input
```

```
55         dist[v] = dist[node] + wt;  
56     }  
57 }  
58 return dist;  
59 }  
60 };  
61 };  
62 };  
63 };  
64 };  
65 // { Driver Code Starts.  
66 int main() {  
67     int t;  
68     cin >> t;  
69     while (t--) {  
70         int n, m;  
71         cin >> n >> m;  
72         vector<int> edges[m];  
73         for (int i = 0; i < m; i++) {  
74             int u, v, wt;  
75             cin >> u >> v >> wt;  
76             edges[i].push_back(u);  
77             edges[i].push_back(v);  
78             edges[i].push_back(wt);  
79         }  
80         // string s; cin>>s;  
81         Solution obj;  
82         vector<int> res = obj.shortestPath(n, m, edges);  
83         for (auto x : res){  
84             if(x==1e9)cout<<"INF "  
85             else cout<<x<< " ";  
86         }  
87         cout<<"\n";  
88     }  
89 }  
90 } // } Driver Code Ends
```

Compile & Run



G-27. Shortest Path in Directed Acyclic Graph - Topological Sort

Practice

1:90 Problem Submissions Doubt Support

Shortest Path in Directed Acyclic Graph Medium Accuracy: 50.0% Submissions: 10 Points: 4

Given a Directed acyclic Graph. Find the shortest path from $\text{src}(0)$ to all the vertex.

Example:

```
Input:  
n = 6, m = 7  
adj=[[0,1,2],[0,4,1],[4,5,4],[4,2,2],[1,2,3],[2,3,6],[5,3]  
src=0  
Output:  
0 2 3 6 1 5
```

Your Task:

You don't need to print or input anything. Complete the function `shortestPath()` which takes an directed acyclic graph, an integer n and an integer src as the input parameters and returns an integer, denoting the vector of distance from src to all nodes.

Constraint:

$1 \leq n, m \leq 100$
 $1 \leq \text{adj}[i][j] \leq 100$

Expected Time Complexity: $O(N+E)$, where N is the number of nodes and E is edges

Expected Space Complexity: $O(N)$

View Bookmarked Problems

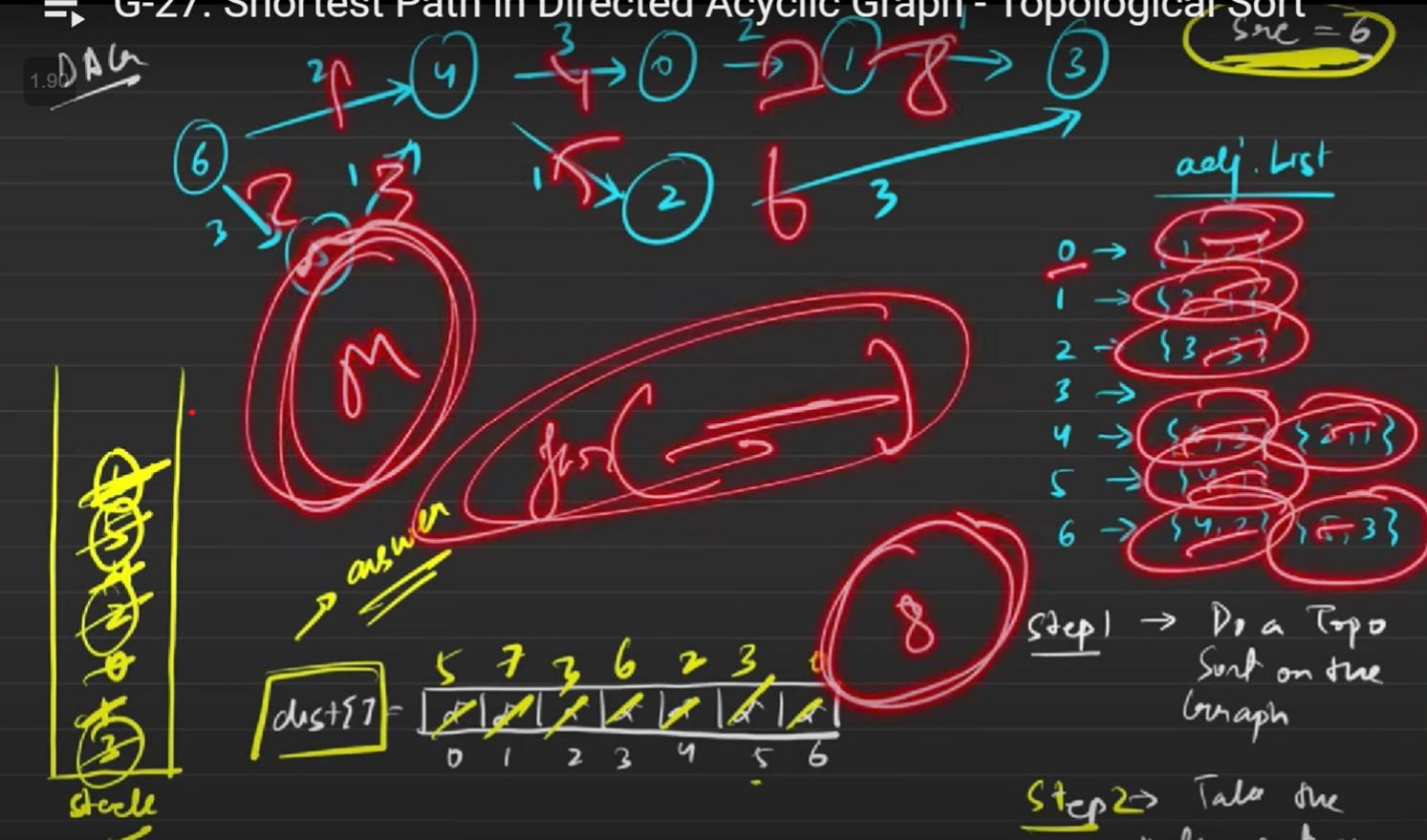
Discussions (1 Threads) Most Recent ▾

```
C++ (g++ 5.4) Test against custom input
```

```
43     vector<int> dist(N);
44     for(int i = 0;i<N;i++) dist[i] = 1e98;
45     dist[0] = 0;
46     while(!st.empty()) {
47         int node = st.top();
48         st.pop();
49
50         for(auto it : adj[node]) {
51             int v = it.first;
52             int wt = it.second;
53
54             if(dist[node] + wt < dist[v]) {
55                 dist[v] = dist[node] + wt;
56             }
57         }
58         return dist;
59     }
60 }
61 }
62 }
63
64 // { Driver Code Starts.
65 int main() {
66     int t;
67     cin >> t;
68     while (t--) {
69         int n, m;
70         cin >> n >> m;
71         vector<int> edges[m];
72         for(int i = 0;i<m;i++) {
73             int u, v, wt;
74             cin >> u >> v >> wt;
75
76             edges[i].push_back(u);
77             edges[i].push_back(v);
78             edges[i].push_back(wt);
79         }
80     }
81 }
```



→ G-27. Shortest Path in Directed Acyclic Graph - Topological Sort



G-27. Shortest Path in Directed Acyclic Graph - Topological Sort

Practice

1:90 Problem Submissions Doubt Support

C++ (g++ 5.4) Test against custom input

```
26.     for(int i = 0;i<M;i++) {
27.         int u = edges[i][0];
28.         int v = edges[i][1];
29.         int wt = edges[i][2];
30.         adj[u].push_back({v, wt});
31.     }
32.
33.     // find the topo sort
34.     int vis[N] = {0};
35.     stack<int> st;
36.     for(int i = 0;i<N;i++) {
37.         if(!vis[i]) {
38.             topoSort(i, adj, vis, st);
39.         }
40.     }
41.
42.     // step 2 do the distance thing
43.     vector<int> dist(N);
44.     for(int i = 0;i<N;i++) dist[i] = 1e9;
45.     dist[0] = 0;
46.     // O(N + M)
47.     while(!st.empty()) {
48.         int node = st.top();
49.         st.pop();
50.
51.         for(auto it : adj[node]) {
52.             int v = it.first;
53.             int wt = it.second;
54.
55.             if(dist[node] + wt < dist[v]) {
56.                 dist[v] = dist[node] + wt;
57.             }
58.         }
59.     }
60.     return dist;
61. }
62. }
```

Shortest Path in Directed Acyclic Graph

Medium Accuracy: 50.0% Submissions:10 Points: 4

Given a Directed acyclic Graph. Find the shortest path from src(0) to all the vertex.

Example:

```
Input:  
n = 6, m= 7  
adj=[[0,1,2],[0,4,1],[4,5,4],[4,2,2],[1,2,3],[2,3,6],[5,3]  
src=0  
Output:  
0 2 3 6 1 5
```

Your Task:

You don't need to print or input anything. Complete the function shortestPath() which takes an directed acyclic graph, an integer n and an integer src as the input parameters and returns an integer, denoting the vector of distance from src to all nodes.

Constraint:

```
1<=n,m<=100  
1<=adj[i][j]<=100
```

Expected Time Complexity: $O(N+E)$, where N is the number of nodes and E is edges

Expected Space Complexity: $O(N)$

View Bookmarked Problems

Discussions (1 Threads) Most Recent

Most Recent

Compile & Run

22:40 / 26:35

CC G #



G-27. Shortest Path in Directed Acyclic Graph - Topological Sort

Practice



1:19 Problem

Submissions

Doubt Support

Shortest Path in Directed Acyclic Graph

Medium Accuracy: 50.0% Submissions: 10 Points: 4

Given a Directed acyclic Graph. Find the shortest path from src(0) to all the vertex.

Example:

```
Input:  
n = 6, m= 7  
adj=[[0,1,2],[0,4,1],[4,5,4],[4,2,2],[1,2,3],[2,3,6],[5,3]  
src=0  
Output:  
0 2 3 6 1 5
```

Your Task:

You don't need to print or input anything. Complete the function shortestPath() which takes an directed acyclic graph, an integer n and an integer src as the input parameters and returns an integer, denoting the vector of distance from src to all nodes.

Constraint:

```
1<=n,m<=100  
1<=adj[i][j]<=100
```

Expected Time Complexity: O(N+E), where N is the number of nodes and E is edges

Expected Space Complexity: O(N)

[View Bookmarked Problems](#)

Discussions (1 Threads)

Most Recent

Compile & Run



23:02 / 26:35

```
26.     for(int i = 0;i<M;i++) {  
27.         int u = edges[i][0];  
28.         int v = edges[i][1];  
29.         int wt = edges[i][2];  
30.         adj[u].push_back({v, wt});  
31.     }  
32.  
33.     // find the topo sort  
34.     // O(N + M)  
35.     int vis[N] = {0};  
36.     stack<int> st;  
37.     for(int i = 0;i<N;i++) {  
38.         if(!vis[i]) {  
39.             topoSort(i, adj, vis, st);  
40.         }  
41.     }  
42.  
43.     // step 2 do the distance thing  
44.     vector<int> dist(N);  
45.     for(int i = 0;i<N;i++) dist[i] = 1e9;  
46.     dist[0] = 0;  
47.     // O(N + M)  
48.     while(!st.empty()) {  
49.         int node = st.top();  
50.         st.pop();  
51.  
52.         for(auto it : adj[node]) {  
53.             int v = it.first;  
54.             int wt = it.second;  
55.  
56.             if(dist[node] + wt < dist[v]) {  
57.                 dist[v] = dist[node] + wt;  
58.             }  
59.         }  
60.     }  
61.     return dist;  
62.
```



IIT bombay | Word Break | Striver's SDE | Heap Sort - | Single Num | Maximum N | (931) G-27. | Shortest Pat | Shortest Pat | ChatGPT | + | X

Compile time poly... rohit952513 - LeetC... HTML ASCII Refere... vector - C++ Refere... SDE Off-campus Pl... Striver DP Series : D... Leetcode 75 Questi... Resize image in cm...

Problems Courses Geek-O-Lympics Events POTD

GeeksforGeeks

Average Time: 20m Your Time: 25m 14s

128 up

Problem Submissions Comments

Shortest path in Directed Acyclic Graph

Medium Accuracy: 48.48% Submissions: 36K+ Points: 4

Sharpen up your programming skills, participate in coding contests & explore high-paying jobs

Given a Directed Acyclic Graph of N vertices from 0 to N-1 and a 2D Integer array(or vector) edges[][] of length M, where there is a directed edge from edge[i][0] to edge[i][1] with a distance of edge[i][2] for all i.

Find the shortest path from src(0) vertex to all the vertices and if it is impossible to reach any vertex, then return -1 for that vertex.

Example1:

Input:
N = 4, M = 2
edges = [[0, 1, 2], [0, 2, 1]]

```
int wt=edges[i][2];
adj[u].push_back({v,wt});
}
// we will do topo sort now
int vis[n]={0};
stack<int>st;
for(int i=0;i<n;i++){
    if(!vis[i]){
        topsort(i,vis,adj,st);
    }
}

//now will calculate dis ,the final
vector<int>dis(n);
for(int i=0;i<n;i++){
    dis[i]=1e9;
}
dis[0]=0;
while(!st.empty()){
    int node=st.top();
    st.pop();
    for(auto &it:adj[node]){
        int v=it.first;
        int wt=it.second;
        if(dis[v]>dis[node]+wt){
            dis[v]=dis[node]+wt;
        }
    }
}
```

Custom Input Compile & Run Submit

Cloudy 27°C ENG IN 7:48 PM 7/8/2023

IIT bombay | Word Break | Striver's SDE | Heap Sort - | Single Num | Maximum N | (931) G-27. | Shortest Pat | Shortest Pat | ChatGPT | + | - | X

Compile time poly... rohit952513 - LeetC... HTML ASCII Refere... vector - C++ Refere... SDE Off-campus Pl... Striver DP Series : D... Leetcode 75 Questi... Resize image in cm...

Problems Courses Geek-O-Lympics Events POTD

edge[i][2] for all i.

Find the shortest path from **src(0)** vertex to all the vertices and if it is impossible to reach any vertex, then return -1 for that vertex.

Example1:

Input:
N = 4, M = 2
edge = [[0,1,2],[0,2,1]]
Output:
0 2 1 -1

Explanation:
Shortest path from 0 to 1 is 0->1 with edge weight 2.
Shortest path from 0 to 2 is 0->2 with edge weight 1.
There is no way we can reach 3, so it's -1 for 3.

Example2:

```
C++ (g++ 5.4) Average Time: 20m Your Time: 25m 14s
```

```
40 }  
41 }  
42 //now will calculate dis ,the final  
43 vector<int>dis(n);  
44 for(int i=0;i<n;i++){  
45     dis[i]=1e9;  
46 }  
47 dis[0]=0;  
48 while(!st.empty()){  
49     int node=st.top();  
50     st.pop();  
51     for(auto &it:adj[node]){  
52         int v=it.first;  
53         int wt=it.second;  
54         if(dis[v]>dis[node]+wt){  
55             dis[v]=dis[node]+wt;  
56         }  
57     }  
58 }  
59 for (int i = 0; i < n; i++) {  
60     if (dis[i] == 1e9) dis[i] = -1;  
61 }  
62 return dis;
```

Custom Input Compile & Run Submit

Cloudy 27°C ENG IN 7:48 PM 7/8/2023



≡

Practice

Problem Submission Doubt Support

Shortest path in Undirected Graph having unit distance

Medium Accuracy: 33.33% Submissions: 9 Points: 4

You are given an Undirected Graph having unit weight, find the shortest distance from src to all the points, if path is not possible then put -1.

Example:

```
Input:  
n = 9, m= 10  
edges=[[0,1],[0,3],[3,4],[4 ,5] [5, 6],[1,2],  
src=0  
Output:  
0 1 2 1 2 3 3 4 4
```

Your Task:

You don't need to print or input anything. Complete the function **shortestPath()** which takes a undirected graph, an integer n and an integer src as the input parameters and returns an integer, denoting the vector of distance from src to all nodes.

Constraint:

$1 \leq n, m \leq 100$
 $1 \leq adj[i][j] \leq 100$

Expected Time Complexity: $O(N + E)$, where N is the number of nodes and E is the number of edges.

C++ (g++ 5.4) Test against custom input

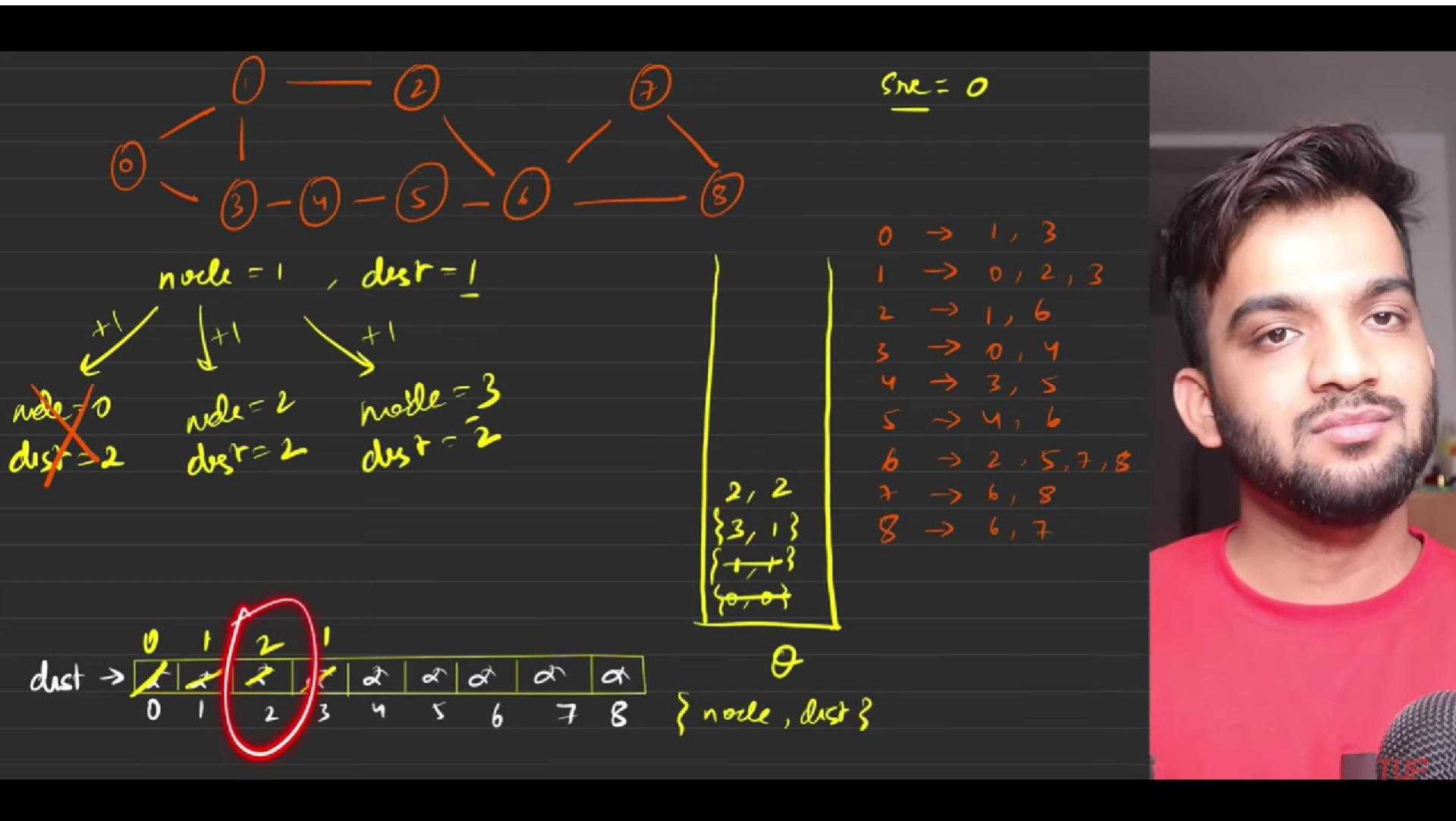
```
1 // } Driver Code Ends  
2 // User Function Template for C++  
3 class Solution {  
4 public:  
5     vector<int> shortestPath(vector<int> edges, int N, int M,  
6                                /  
7                                /  
8                                /  
9                                /  
10 };  
11 // } Driver Code Ends
```

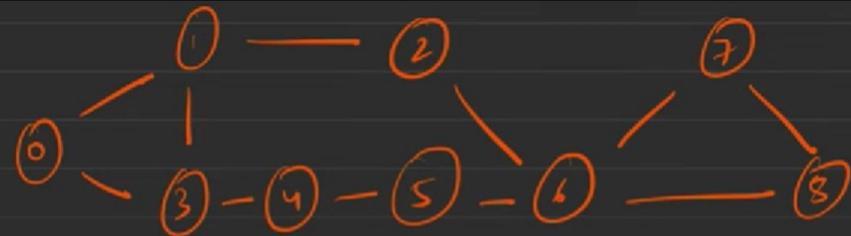
Average Time: 20ms

Compile & Run

TUF







src = 0

$\text{node} = 0$ $\text{dist} = 0$
 $\text{node} = 1$ $\text{dist} = 1$
 $\text{node} = 2$ $\text{dist} = 2$
 $\text{node} = 3$ $\text{dist} = 1$
 $\text{node} = 4$ $\text{dist} = 2$



$0 \rightarrow 1, 3$
 $1 \rightarrow 0, 2, 3$
 $2 \rightarrow 1, 6$
 $3 \rightarrow 0, 4$
 $4 \rightarrow 3, 5$
 $5 \rightarrow 4, 6$
 $6 \rightarrow 2, 5, 7, 8$
 $7 \rightarrow 6, 8$
 $8 \rightarrow 6, 7$

$\text{dist} \rightarrow$

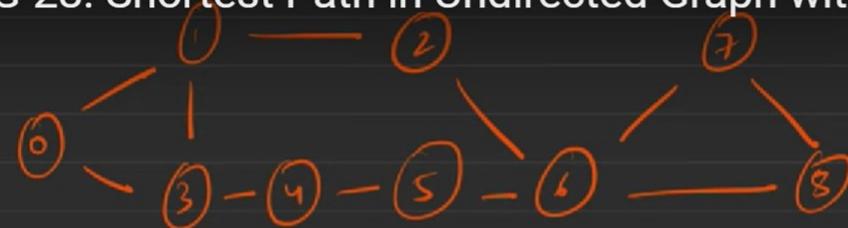
0	1	2	1	0	0	0	0	0
0	1	2	3	4	5	6	7	8

 $\{\text{node}, \text{dist}\}$



→ G-28. Shortest Path in Undirected Graph with Unit Weights

1.75

src = 0

$mole = 3$ $dist = 1$

$\cancel{mole = 0}$ $\cancel{dist = 2}$

$+1$

$node = 4$

$dist = 2$



0	→	1, 3
1	→	0, 2, 3
2	→	1, 6
3	→	0, 4
4	→	3, 5
5	→	4, 6
6	→	2, 5, 7, 8
7	→	6, 8
8	→	6, 7

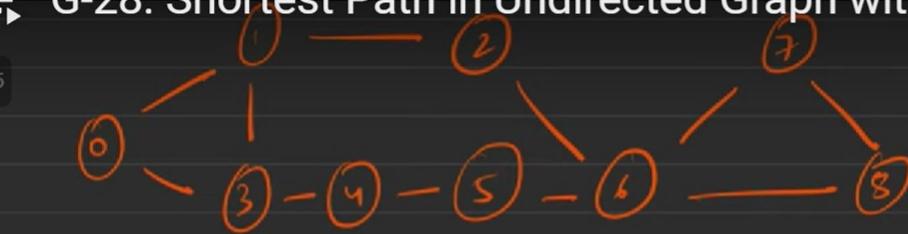


6:24 / 16:31

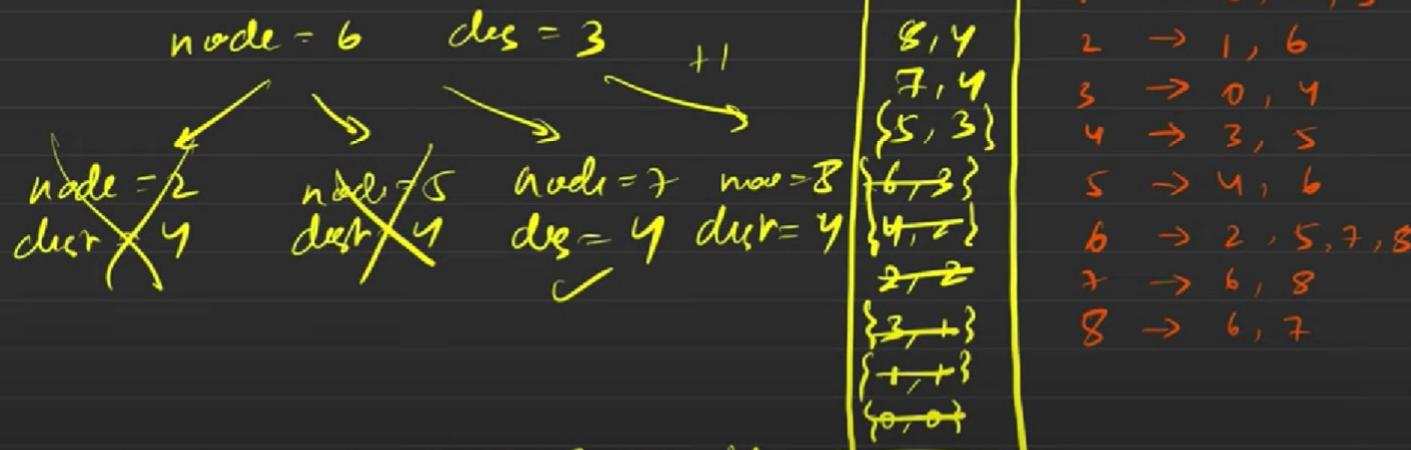


→ G-28. Shortest Path in Undirected Graph with Unit Weights

1.75



s_{src} = 0

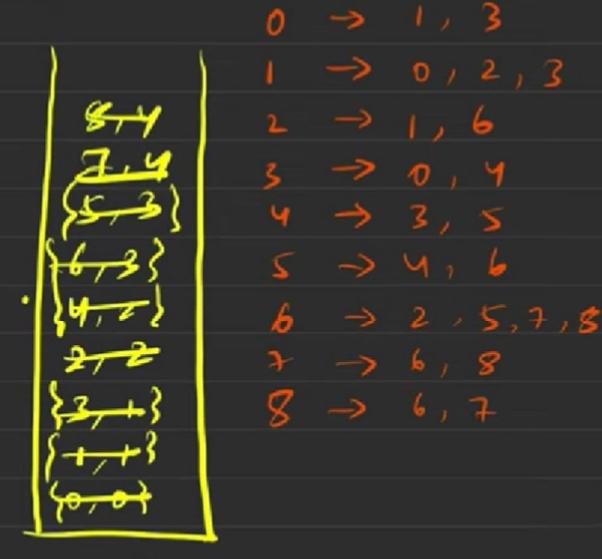


8:57 / 16:31





$\alpha \rightarrow \text{int}$



≡

Practice

C++ (g++ 5.4) Test against custom input

Last Backup C

Shortest path in Undirected Graph having unit distance

Medium Accuracy: 50.0% Submissions: 18 Points: 4

You are given an Undirected Graph having unit weight. Find the shortest path from src(0) to all the vertex and if it is unreachable to reach any vertex, then return -1 for that vertex.

Example:

```
Input:  
n = 9, m = 10  
edges=[[0,1],[0,3],[3,4],[4 ,5]  
. [5, 6].[1,2].[2,6].[6,7].[7,8].[6,8]]
```

Output Window

Test Cases Processed: 24 / 51

```
#include <iostream>  
using namespace std;  
  
// } Driver Code Ends  
// User function Template for C++  
class Solution {  
public:  
    vector<int> shortestPath(vector<vector<int>> &edges, int N, int M, int src){  
        vector<int> adj[N];  
        for(auto it : edges) {  
            adj[it[0]].push_back(it[1]);  
            adj[it[1]].push_back(it[0]);  
        }  
  
        int dist[N];  
        for(int i = 0;i<N;i++) dist[i] = 1e9;  
        dist[src] = 0;  
        queue<int> q;  
        q.push(src);  
        while(!q.empty()) {  
            int node = q.front();  
            q.pop();  
            for(auto it : adj[node]) {  
                if(dist[node] + 1 < dist[it]) {  
                    dist[it] = 1 + dist[node];  
                    q.push(it);  
                }  
            }  
        }  
        vector<int> ans(N, -1);  
        for(int i = 0;i<N;i++) {  
            if(dist[i] != 1e9) {  
                ans[i] = dist[i];  
            }  
        }  
        return ans;  
    }  
};
```

Average Time: 20m

Compile & Run



≡

Practice

C++ (g++ 5.4) - Test against custom input

Last Backup C

Shortest path in Undirected Graph having unit distance

Medium Accuracy: 50.0% Submissions: 18 Points: 4

You are given an Undirected Graph having unit weight. Find the shortest path from `src(0)` to all the vertex and if it is unreachable to reach any vertex, then return -1 for that vertex.

Example:

```
Input:  
n = 9, m = 10  
edges=[[0,1],[0,3],[3,4],[4 ,5],  
,[5, 6],[1,2],[2,6],[6,7],[7,8],[6,8]]  
src=0  
Output:  
0 1 2 1 2 3 3 4 4
```

Your Task:

You don't need to print or input anything. Complete the function `shortestPath()` which takes a 2d vector or array `edges` representing the edges of undirected graph with unit weight, an integer `N` as number nodes, an integer `M` as number of edges and an integer `src` as the input parameters and returns an integer array or vector, denoting the **vector of distance from src to all nodes**.

Constraint:

$1 \leq n, m \leq 100$
 $1 \leq adj[i][j] \leq 100$

Expected Time Complexity: $O(N + E)$, where N is the number of nodes and E is edges

Expected Space Complexity: $O(N)$

[View Bookmarked Problems](#)

[Discussions \(5 Threads\)](#)

Most Recent

Average Time: 20m

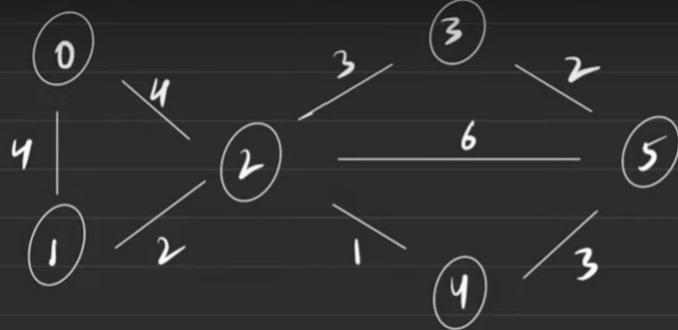
Compile & Run



```
4 #include <bits/stdc++.h>
5 using namespace std;
6
7 // } Driver Code Ends
8 // User Function Template for C++
9 class Solution {
10 public:
11     vector<int> shortestPath(vector<vector<int>>& edges, int N, int M, int src){
12         vector<int> adj[N];
13         for(auto it : edges) {
14             adj[it[0]].push_back(it[1]);
15             adj[it[1]].push_back(it[0]);
16         }
17
18         int dist[N];
19         for(int i = 0;i<N;i++) dist[i] = 1e9;
20         dist[src] = 0;
21         queue<int> q;
22         q.push(src);
23         while(!q.empty()) {
24             int node = q.front();
25             q.pop();
26             for(auto it : adj[node]) {
27                 if(dist[node] + 1 < dist[it]) {
28                     dist[it] = 1 + dist[node];
29                     q.push(it);
30                 }
31             }
32         }
33         vector<int> ans(N, -1);
34         for(int i = 0;i<N;i++) {
35             if(dist[i] != 1e9) {
36                 ans[i] = dist[i];
37             }
38         }
39         return ans;
40     }
41 }
```

~~SN~~ G-32. Dijkstra's Algorithm - Using Priority Queue - C++ and Java - Part 1

Adj. List



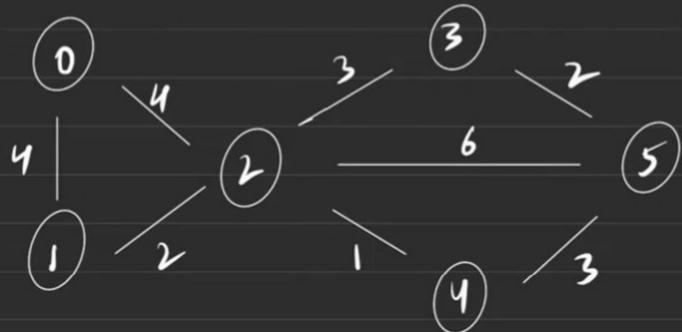
0 $d=0$

$0 \rightarrow \{1, 4\} \{2, 4\}$
 $1 \rightarrow \{0, 4\} \{2, 2\}$
 $2 \rightarrow \{0, 4\} \{1, 2\} \{3, 3\} \{4, 1\} \{5, 6\}$
 $3 \rightarrow \{2, 3\} \{5, 2\}$
 $4 \rightarrow \{2, 1\} \{5, 3\}$
 $5 \rightarrow \{2, 6\} \{3, 2\} \{4, 3\}$



Dijkstra's Algorithm

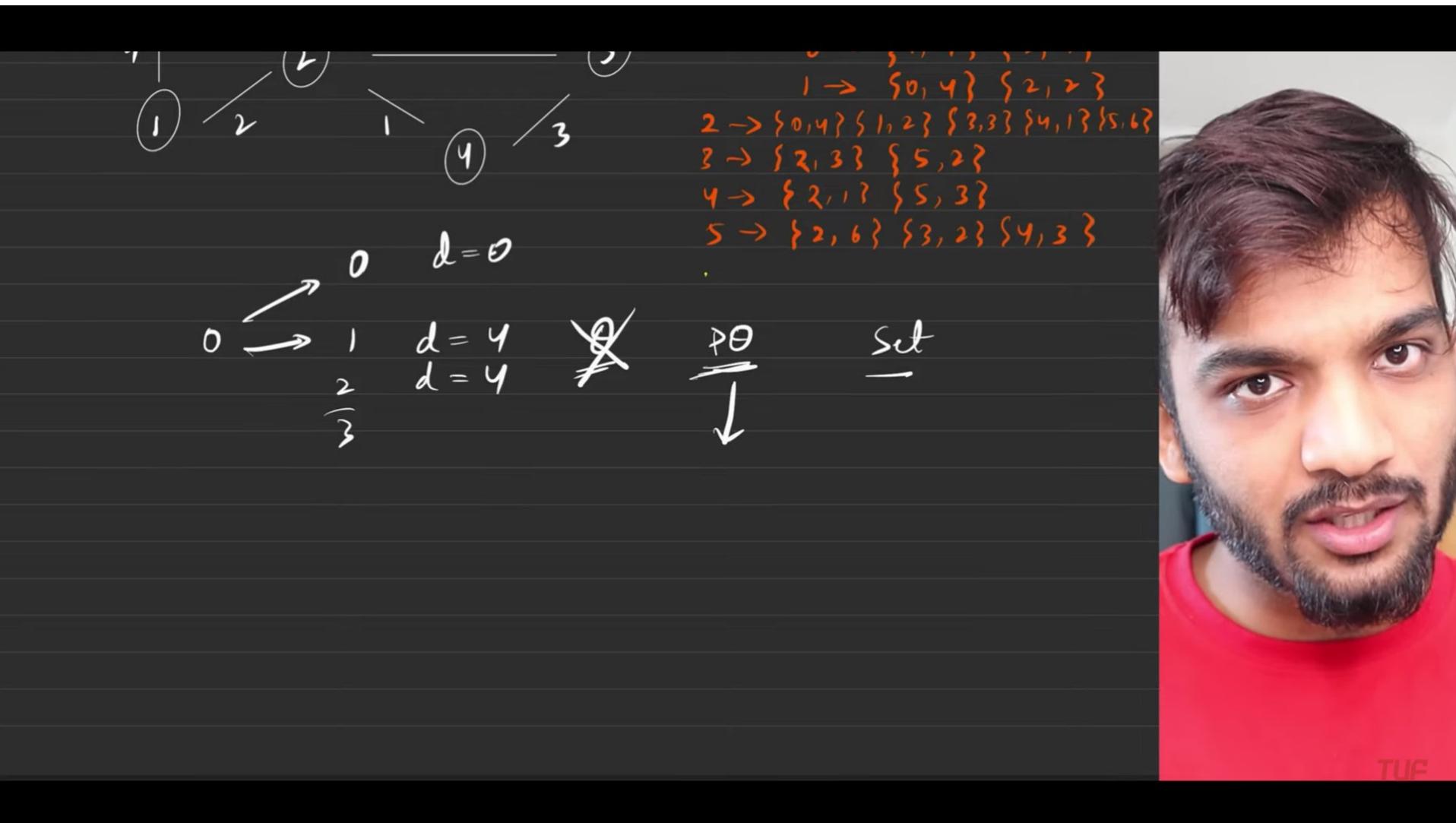
Press Esc to exit full screen



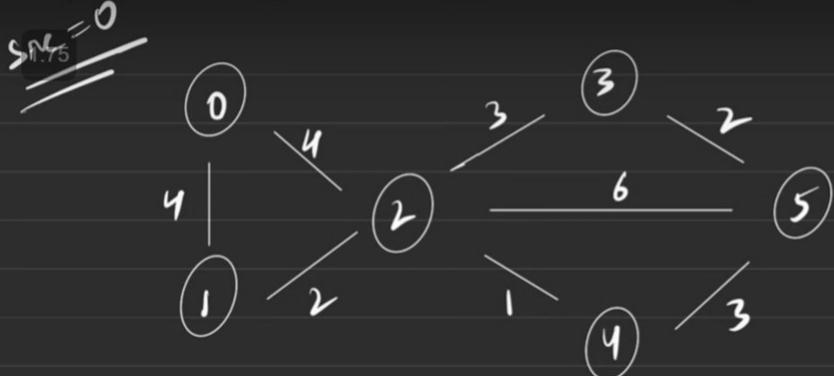
Adj. List

$0 \rightarrow \{1, 4\} \{2, 4\}$
 $1 \rightarrow \{0, 4\} \{2, 2\}$
 $2 \rightarrow \{0, 4\} \{1, 2\} \{3, 3\} \{4, 1\} \{5, 6\}$
 $3 \rightarrow \{2, 3\} \{5, 2\}$
 $4 \rightarrow \{2, 1\} \{5, 3\}$
 $5 \rightarrow \{2, 6\} \{3, 2\} \{4, 3\}$





→ G-32. Dijkstra's Algorithm - Using Priority Queue - C++ and Java - Part 1

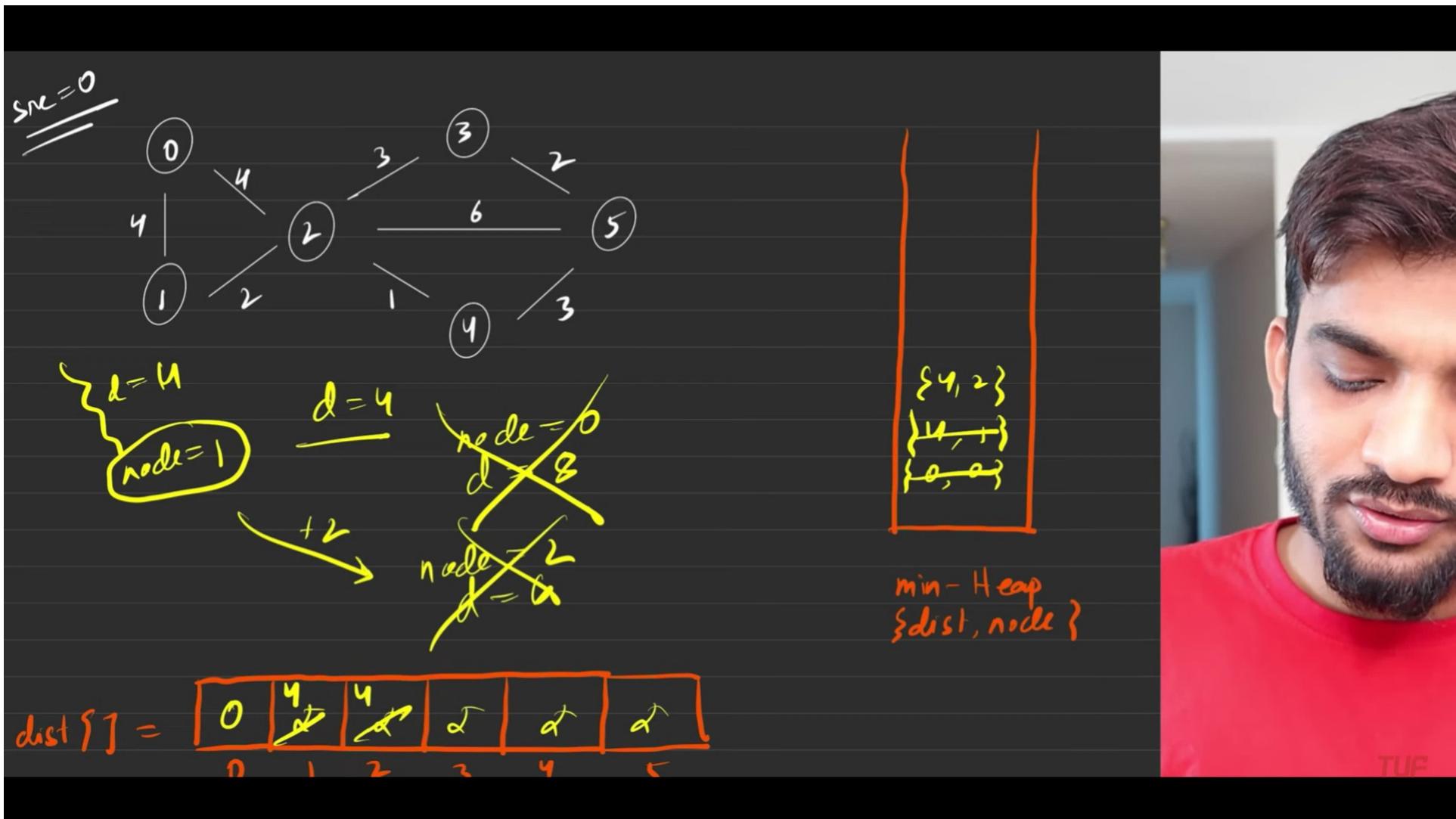


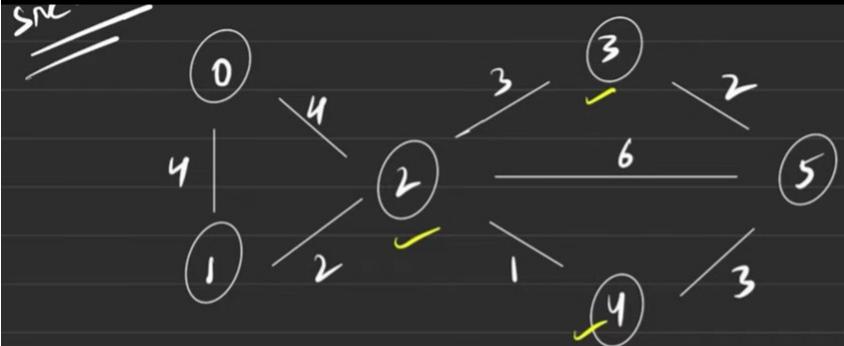
$d = 0$
 $\text{node} = 0$ $+ 4$ $\text{node} = 1$
 $+ 4$ $d = 4$
 $+ 4$ $\text{node} = 2$
 $d = 4$



min-Heap
{dist, node}







min-Heap
{dist, node?}

dist[] = [

0	4	4	7	5	10	6
0	1	2	3	4	5	6



G-32. Dijkstra's Algorithm - Using Priority Queue - C++ and Java - Part 1

Problems Courses Get Hired Contests </> POTD Practice 435

1.75 Problem Editorial Submissions Comments C++ (g++ 5.4) Average Time: 25m

Given a weighted, undirected and connected graph of V vertices and E edges, Find the shortest distance of all the vertex's from the source vertex S.

Note: The Graph doesn't contain any negative weight cycle.

Example 1:

Input:
V = 2, E = 1
u = 0, v = 1, w = 9
adj [] = {{{1, 9}}, {{0, 9}}}
S = 0

Output:
0 9

Explanation:

The source vertex is 0. Hence, the shortest distance of vertex 1 from vertex 0 is 9.

```
1 //} Driver Code Ends
2 class Solution
3 {
4     public:
5         //Function to find the shortest distance of all the vertices
6         //from the source vertex S.
7         vector<int> dijkstra(int V, vector<vector<int>> adj[], int S)
8     {
9         vector<int> dist(V, INT_MAX);
10        dist[S] = 0;
11        priority_queue<pair<int, int>, vector<pair<int, int>>, greater<pair<int, int>> pq;
12        pq.push({0, S});
13        while (!pq.empty())
14        {
15            auto curr = pq.top();
16            pq.pop();
17            int currNode = curr.second;
18            int currDist = curr.first;
```

Custom Input Compile & Run CC HD

14:34 / 22:41

G-32. Dijkstra's Algorithm - Using Priority Queue - C++ and Java - Part 1

Problems Courses Get Hired Contests </>POTD Practice 435

1.75 </> Problem Editorial Submissions Comments C++ (g++ 5.4) Average Time: 25m

Given a weighted, undirected and connected graph of V vertices and E edges, Find the shortest distance of all the vertex's from the source vertex S.

Note: The Graph doesn't contain any negative weight cycle.

Example 1:

Input:

```
V = 2, E = 1
u = 0, v = 1, w = 9
adj [] = {{ {1, 9}}, {{0, 9}}}
S = 0
```

Output Window

Compilation Results Custom Input

Problem Solved Successfully ✓

You get marks only for the first correct submission if you solve the problem without viewing the full solution.

Test Cases Passed: 131 / 131 Your Total Score: 435

Total Time Taken: 0.00 Correct Submission Count: 17:59 / 22:41 Custom Input Compile & Run

1 // } Driver Code Ends
2 class Solution
3 {
4 public:
5 //Function to find the shortest distance of all the vertices
6 //from the source vertex S.
7 vector<int> dijkstra(int V, vector<vector<int>> adj[], int S)
8 {
9 priority_queue<pair<int,int>,vector<pair<int,int>>,greater<pair<int,int>>> pq;
10 vector<int> dist(V);
11 for(int i = 0;i<V;i++) dist[i] = 1e9;
12
13 dist[S] = 0;
14 pq.push({0,S});
15
16 while(!pq.empty())
17 {
18 int dis = pq.top().first;
19 int node = pq.top().second;
20 pq.pop();
21
22 for(auto it : adj[node])
23 {
24 int edgeWeight = it[1];
25 int adjNode = it[0];
26
27 if(dis + edgeWeight < dist[adjNode])
28 {
29 dist[adjNode] = dis + edgeWeight;
30 pq.push({dist[adjNode], adjNode});
31 }
32 }
33 }
34 return dist;
35 }
36 }
37
38 // } Driver Code Ends

can be solved
using queue
only →
done on
fbg



Dijkstra uses the following

→ negative cycle



G-32. Dijkstra's Algorithm - Using Priority Queue - C++ and Java - Part 1

1.75

Dijkstra does not work in
→ negative weight
→ negative cycle



18:19 / 22:41



$s^{re} = 0$



-4	
0	-2
0	1

$(-4, 0)$
$(-2, 1)$
$\{0, 0\}$

PQ



TUG

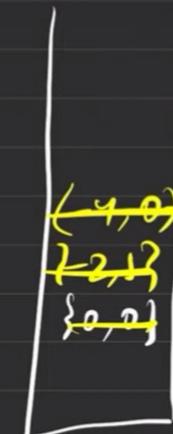
→ G-32. Dijkstra's Algorithm - Using Priority Queue - C++ and Java - Part 1

src = 0

{ infinite loop }



node = 0
d = -6



PQ



20:33 / 22:41



$TCL \rightarrow E$ ~~$\log V$~~ \rightarrow no. of nodes
Total no. of edges

(Q) Why a ~~\log~~ is used
 θ



TUF

⇒ G-34. Dijkstra's Algorithm - Why PQ and not Q, Intuition, Time Complexity Derivat... ⏱ ➔

1.90

```
while (!PQ.empty()) → V
    {
        dis, node = PQ.top() → log(heaps size)
        ne ← for (iterate on adjacent nodes) Total no. of edges
            {
                if (conditional check)
                    update dist
                    push in PQ
            }
    }
```



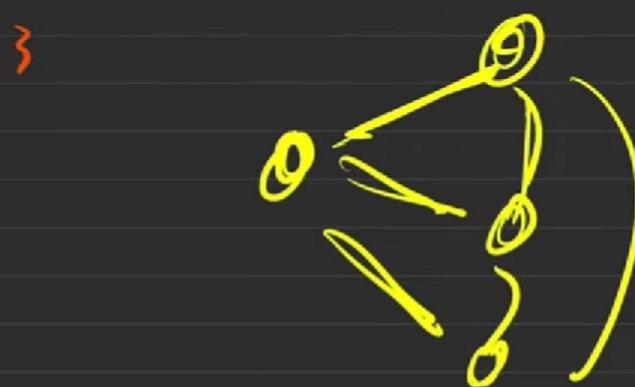
8:00 / 14:29



`ne ← for (iterate on adjacent nodes) no. of edges`

if (conditional check)
 update dict
 push in PD

4 nodes



→ G-34. Dijkstra's Algorithm - Why PQ and not Q, Intuition, Time Complexity Derivat...



1.90

$\text{dis}, \text{node} = \text{top}(L)$

$\rightarrow \log(\text{heapsize})$

\downarrow

no. of nodes

Total
no. of edges

$\text{ne} \leftarrow \text{fun}(\text{iterate on adjacent nodes})$

\downarrow
 $\text{if } (\text{conditional check})$
 update dist
 push in PQ

\downarrow
 4 nodes



9:17 / 14:29



⇒ G-34. Dijkstra's Algorithm - Why PQ and not Q, Intuition, Time Complexity Derivat... ⏱ ➔

1.90

$\Theta(V \times (\text{pop vertex from min heap} + \text{no. of edges on each vertex} \times \text{push into PQ}))$

$\Theta(V \times (\log(\text{heap size}) + ne \times \log(\text{heap size})))$

$\Theta(V \times (\log(\text{heap size}) \times (ne + 1)))$

one vertex can have
 $(V-1)$ edges.

$\Theta(V \times V \times \log(\text{heap size}))$

$\Theta(V^2 \times \log(\text{heap size}))$

every row compresses
it's like all the



10:52 / 14:29



⇒ G-34. Dijkstra's Algorithm - Why PQ and not Q, Intuition, Time Complexity Derivat... ⏱ ➔

1.90

$\Theta(V \times (\text{pop vertex from min heap} + \text{no. of edges on each vertex} \times \text{push into PQ}))$

$\Theta(V \times (\log(\text{heap size}) + ne \times \log(\text{heap size})))$

$\Theta(V \times (\log(\text{heap size}) \times (ne + 1)))$

one vertex can have
 $(V-1)$ edges.

$\Theta(V \times V \times \log(\text{heap size}))$

$\Theta(V^2 \times \log(\text{heap size}))$

every row compresses
it's like all the

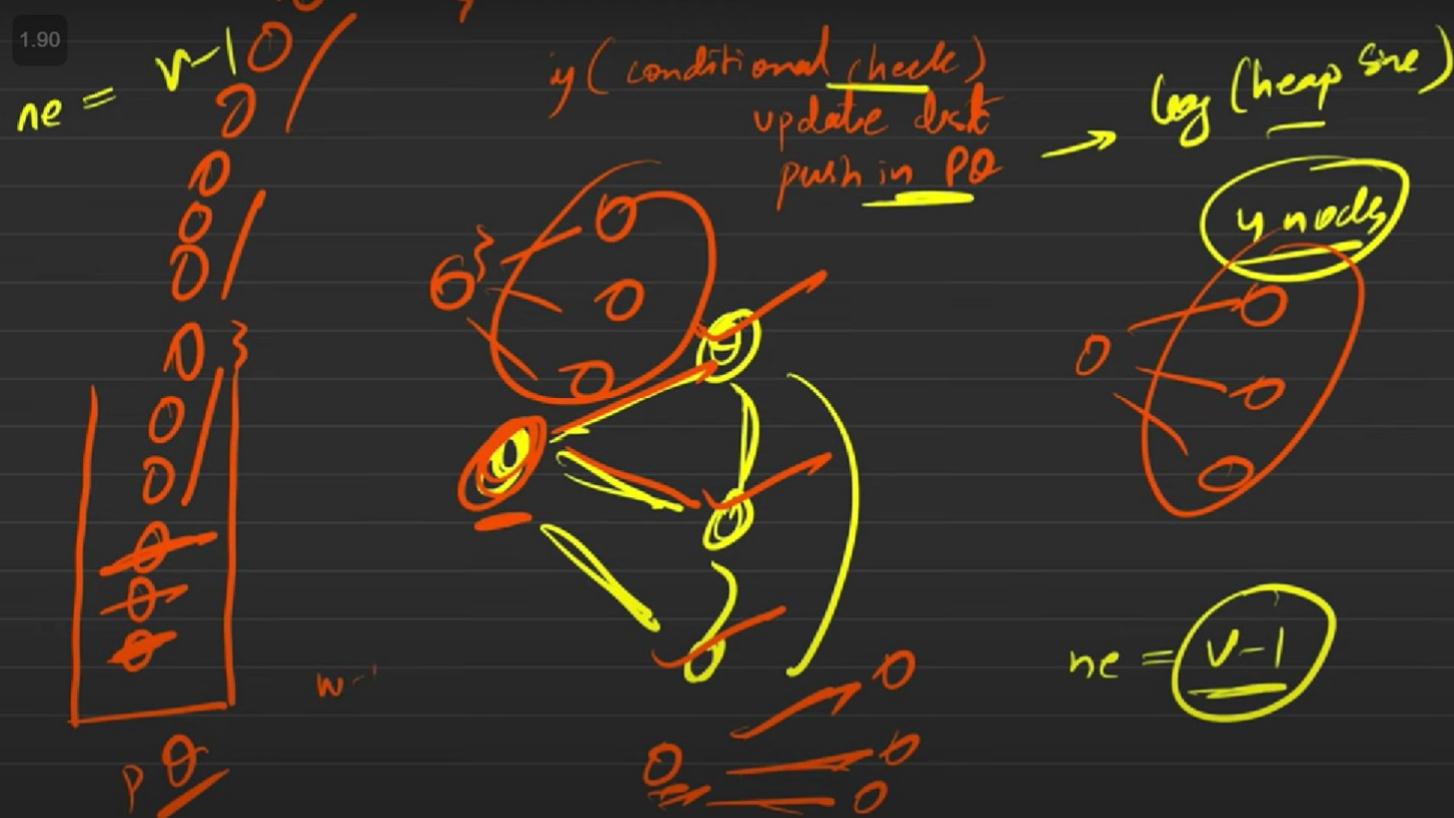


10:52 / 14:29



→ G-34. Dijkstra's Algorithm - Why PQ and not Q, Intuition, Time Complexity Derivat... ⏱ ➔

1.90



→ G-34. Dijkstra's Algorithm - Why PQ and not Q, Intuition, Time Complexity Derivat... ⏱ ➔

1.90

~~o ($V \times (\log(\text{heap size}) + ne \times \log(\text{heap size}))$)~~

~~o ($V \times (\log(\text{heap size}) \times (ne + 1))$)~~

$\circ (V \times (\log(\text{heap size}) + ne \times \log(\text{heap size}))$

$\circ (V \times (\log(\text{heap size}) \times (ne + 1)))$

$\circ (\cancel{V} \times \cancel{V} \times \log(\text{heap size}))$

$\circ (V^2 \times \log(\text{heap size}))$

$\circ (V^2 \times \log(V^2))$

$\circ (V^2 \times 2 \log V)$

$\circ (E \times 2 \times \log V) \approx O(E \log V)$

one vertex can have
 $(V-1)$ edges.

every one compresses
gets all the
nodes in

$E = V^2$

Total edges

27 of 34



12:57 / 14:29

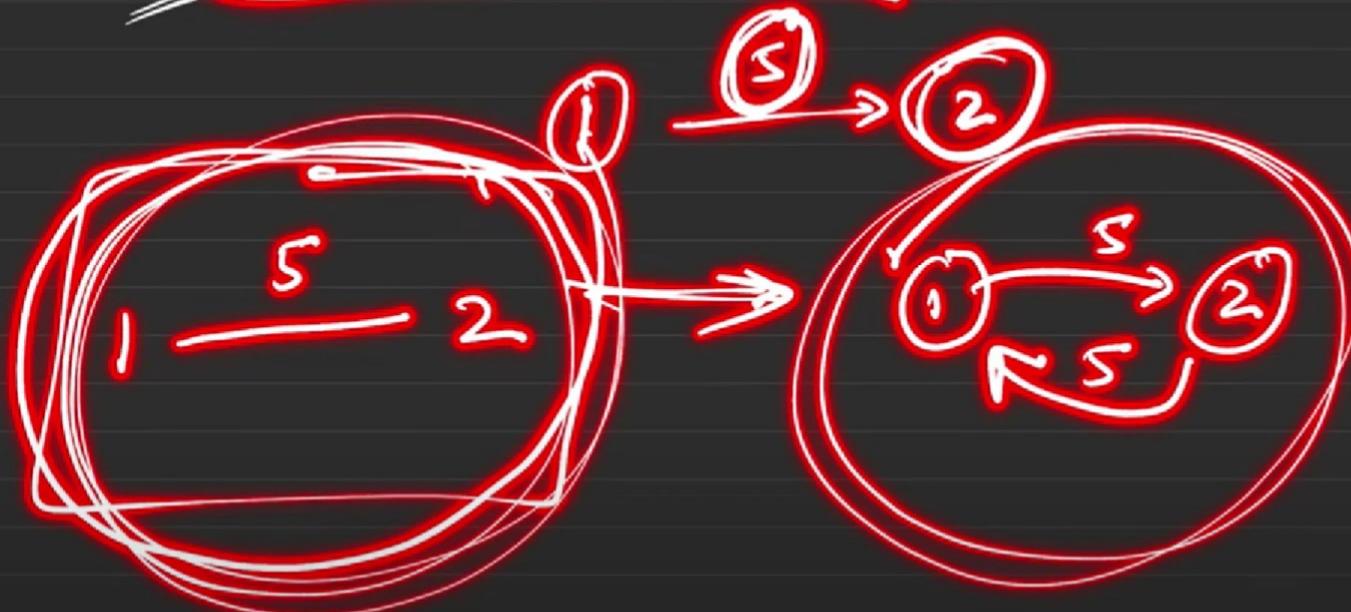


→ G-41. Bellman Ford Algorithm

2.05

Bellman Ford Algorithm

→ helps you to detect negative cycles.



2:24 / 27:42 • Introduction >

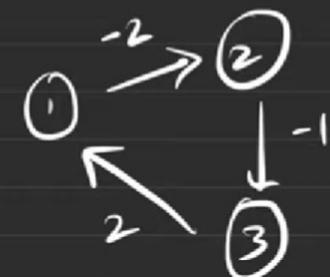


⇒ G-41. Bellman Ford Algorithm

2.05

Dm

Bellman Ford Algorithm → helps you to detect negative cycles.



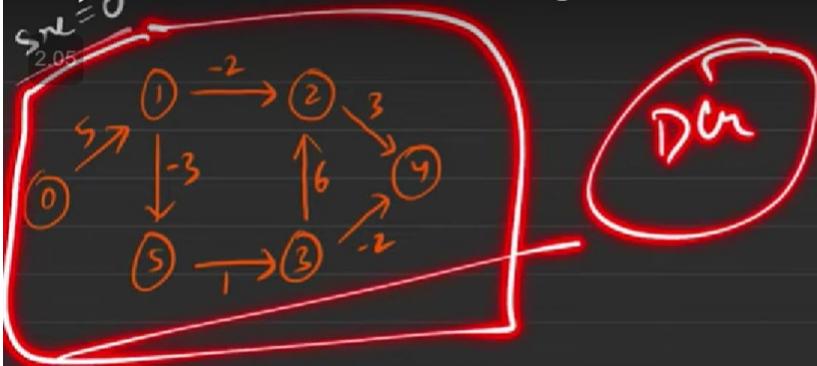
$$\begin{aligned} \text{Path weight} &= -2 - 1 + 2 \\ &= -1 \end{aligned}$$



3:46 / 27:42 • Introduction >



G-41. Bellman Ford Algorithm



(u, v, wt)

$(3, 2, 6)$

$(5, 3, 1)$

$(0, 1, -5)$

$(1, 5, -3)$

$(1, 2, -2)$

$(3, 4, -2)$

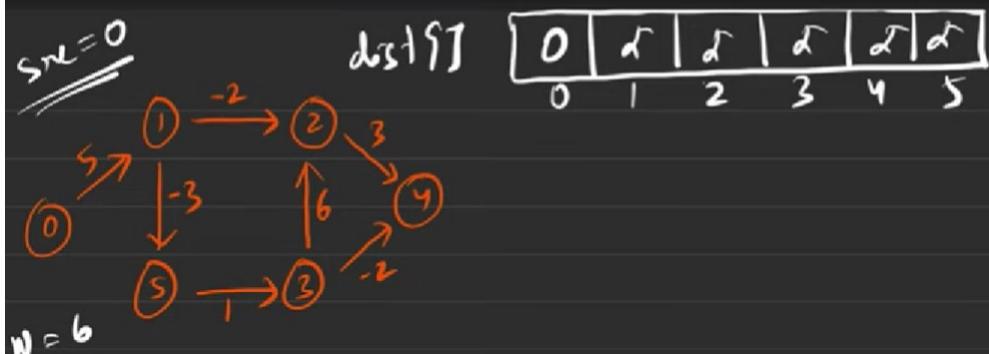
$(2, 4, 3)$



→ G-41. Bellman Ford Algorithm

Implementation

2.05



* Relax all the edges
N-1 times sequentially

* Relax

$$\text{if } (\text{dist}[u] + \text{wt} < \text{dist}[v]) \\ \text{dist}[v] = \text{dist}[u] + \text{wt}$$

(u, v, wt)

$(3, 2, 6)$

$(5, 3, 1)$

$(0, 1, 5)$

$(1, 5, -3)$

$(1, 2, -2)$

$(3, 4, -2)$

$(2, 4, 3)$

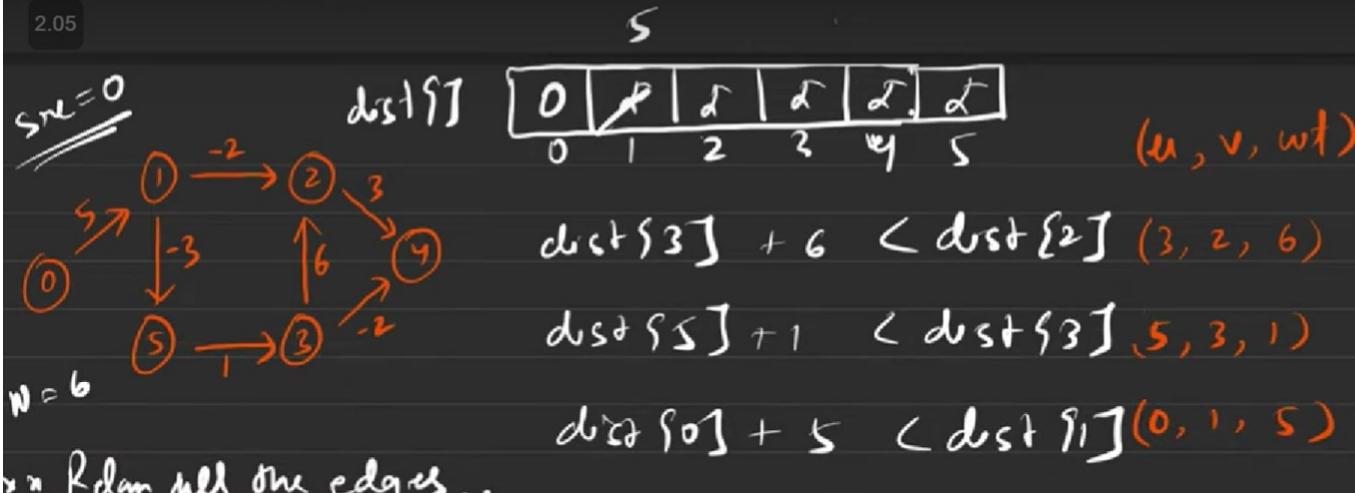


7:50 / 27:42 • Implementation >



⇒ G-41. Bellman Ford Algorithm

2.05



• Relax all the edges
N-1 times sequentially

• Relax

$$\text{if } (\text{dist}[u] + \text{wt} < \text{dist}[v]) \\ \text{dist}[v] = \text{dist}[u] + \text{wt}$$

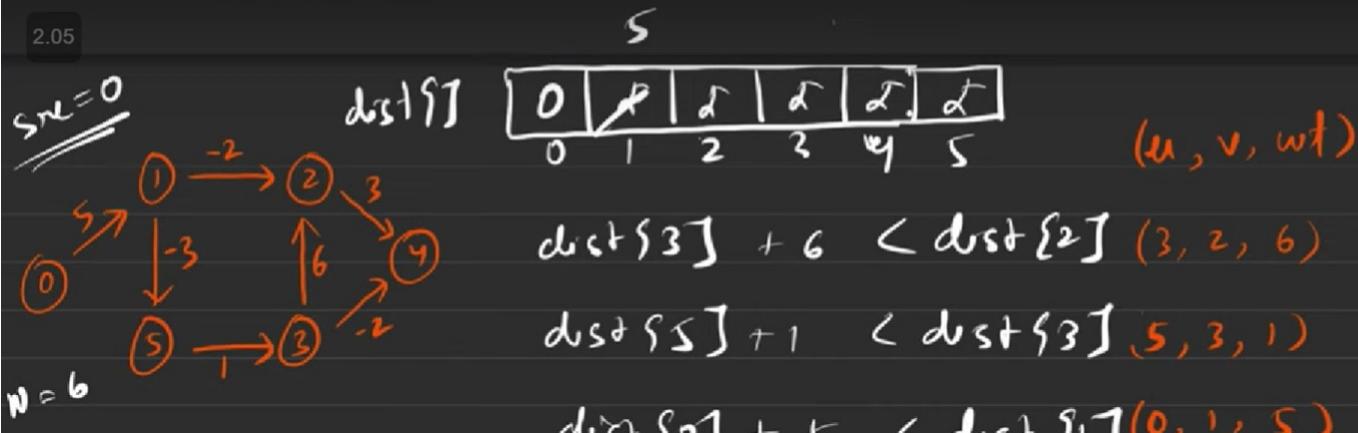


9:56 / 27:42 • Relaxation >



→ G-41. Bellman Ford Algorithm

2.05



• Relax all the edges

$N-1$ times sequentially

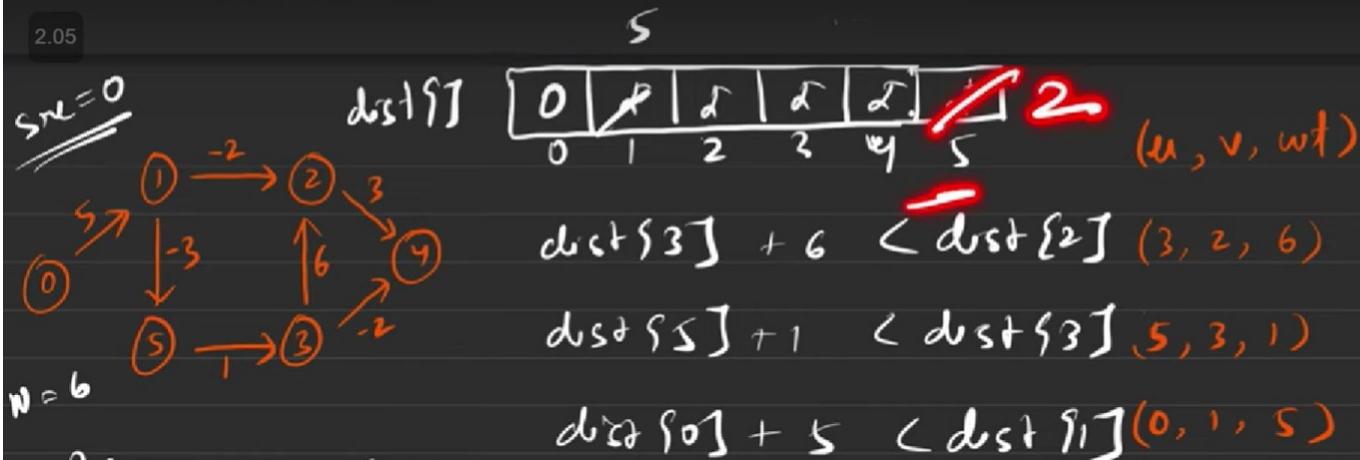
• Relax

$$\text{if } (\text{dist}[u] + \text{wt} < \text{dist}[v]) \\ \text{dist}[v] = \text{dist}[u] + \text{wt}$$



→ G-41. Bellman Ford Algorithm

2.05

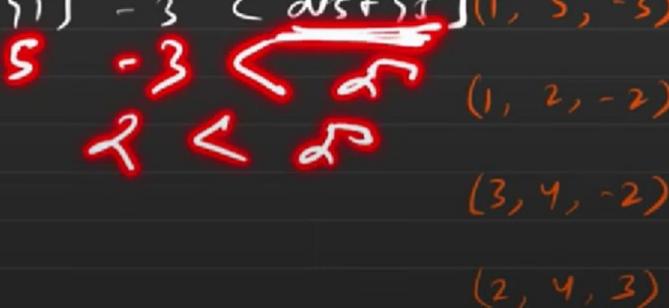


• Relax all the edges

$N-1$ times sequentially

• Relax

$$\text{if } (dist\{u\} + wt < dist\{v\}) \\ dist\{v\} = dist\{u\} + wt$$



10:34 / 27:42 • Relaxation >



+ 8 edges can be



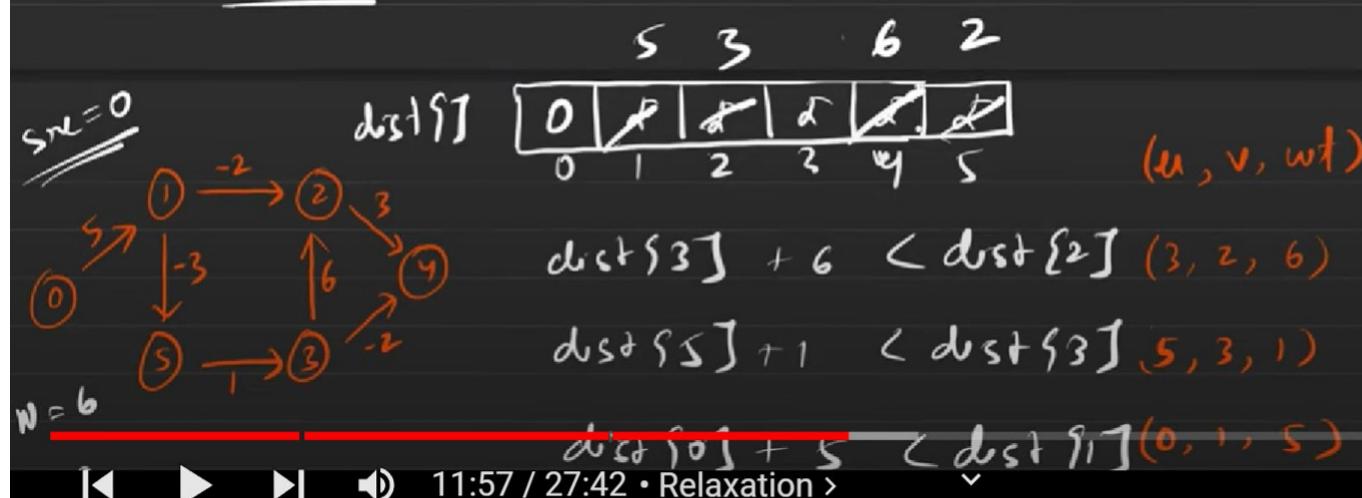
⇒ G-41. Bellman Ford Algorithm

2.05



1st
✓

5 iteration



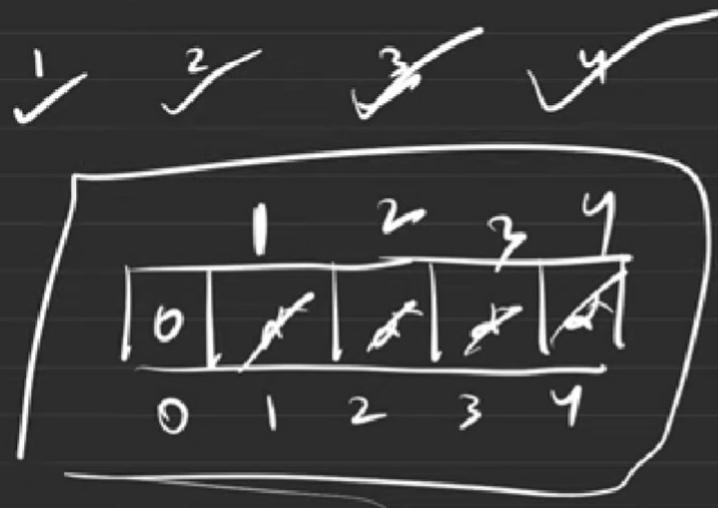
Intuition \rightarrow why N^{-1} ??

xx edges can
be in any
order...



sm

(u, v, wt)



$$\text{dist}[3] + 1 < \text{dist}[4](3, 4, 1)$$

$$\text{dist}[2] + 1 < \text{dist}[3](2, 3, 1)$$

$$\text{dist}[1] + 1 < \text{dist}[2](1, 2, 1)$$

$$\text{dist}[0] + 1 < \text{dist}[1](0, 1, 1)$$



TUF

► G-41. Bellman Ford Algorithm

2.05

Intuition

→ why $N-1$??

xx edges can
be in any
order...



$$\begin{aligned} & dist[3] + 1 < dist[4](3, 4, 1) \\ & dist[2] + 1 < dist[3](2, 3, 1) \\ & dist[1] + 1 < dist[2](1, 2, 1) \\ & \langle dist[1](0, 1, 1) \rangle \end{aligned}$$

Since in a graph of N nodes, in worst case, you will take $N-1$ edges to reach from the first to the last, thereby we iterate for $N-1$ iterations.

Try drawing a graph which takes more than $N-1$ edges for any path, it is not possible.



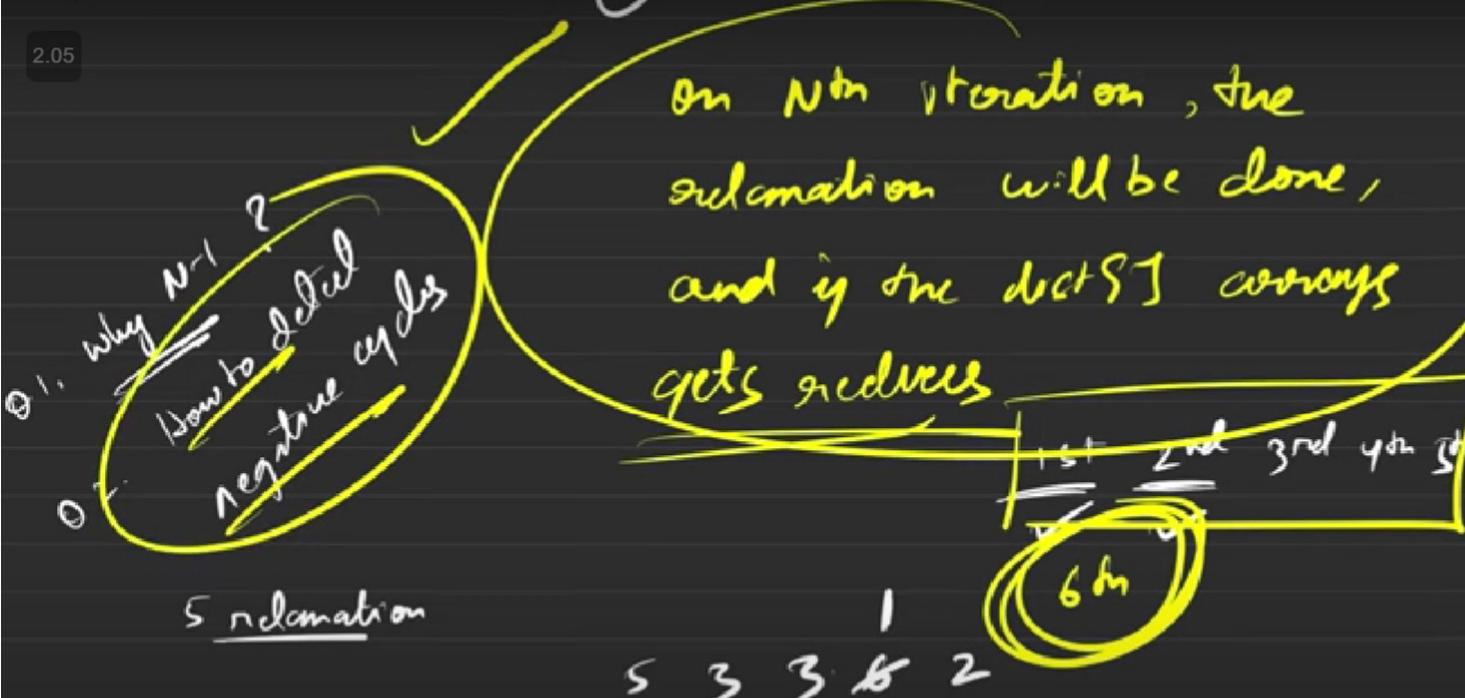
21:02 / 27:42 • Relaxation >



⇒ G-41. Bellman Ford Algorithm

2.05

on N^{th} iteration, the
relaxation will be done,
and if the dist $[S]$ coverage
gets reduces



$$s^{rel} = 0$$

dist $[0] = [0 \ 1 \ 2 \ 3 \ 4 \ 5]$

$$0 \xrightarrow{-2} 1 \xrightarrow{3} 2 \quad \text{dist}[1] = [0 \ 1 \ 2 \ 3 \ 4 \ 5]$$
$$1 \xrightarrow{3} 2 \quad \text{dist}[2] = [0 \ 1 \ 2 \ 3 \ 4 \ 5]$$
$$2 \xrightarrow{2} 3 \quad \text{dist}[3] = [0 \ 1 \ 2 \ 3 \ 4 \ 5]$$
$$3 \xrightarrow{1} 4 \quad \text{dist}[4] = [0 \ 1 \ 2 \ 3 \ 4 \ 5]$$
$$4 \xrightarrow{2} 5 \quad \text{dist}[5] = [0 \ 1 \ 2 \ 3 \ 4 \ 5]$$

(u, v, wt)

$$\text{dist}[2] + 3 < \text{dist}[2] (2, 2, 6)$$



Problems Courses Get Hired Contests POTD

Practice

Problem Editorial Submissions Comments

Distance from the Source (Bellman-Ford Algo Algorithm)

Medium Accuracy: 50.02% Submissions: 22250 Points: 4

Given a weighted, directed and connected graph of V vertices and E edges, Find the shortest distance of all the vertex's from the source vertex S.

Note: If the Graph contains a negative cycle then return an array consisting of only -1.

Example 1:

Input:

```
E = [[0,1,9]]
S = 0
```

Output:

```
0 9
```

```
C++ (g++ 5.4) Average Time: 25m
1 // Driver Code Ends
6 // User function Template for C++
7
8 class Solution {
9 public:
10    /* Function to implement Bellman Ford
11    * edges: vector of vectors which represents the graph
12    * S: source vertex to start traversing graph with
13    * V: number of vertices
14    */
15    vector<int> bellman_ford(int V, vector<vector<int>>& edges, int S) {
16        // Code here
17    }
18 };
19 // } Driver Code Ends
```

Custom Input

G-41. Bellman Ford Algorithm

Problems Courses Get Hired Contests ePOTD

Practice

2.05

Problem Editorial Submissions Comments

2 to 0. This has a distance of 1.
For nodes 2 to 1, we can follow the path-
2->0->1, which has a distance of $1+5 = 6$.

Your Task:

You don't need to read input or print anything. Your task is to complete the function `bellman_ford()` which takes a number of vertices `V` and an `E`-sized list of lists of three integers where the three integers are `u, v, and w`; denoting there's an edge from `u` to `v`, which has a weight of `w` and source node `S` as input parameters and returns a list of integers where the `i`th integer denotes the distance of an `i`th node from the source node.

If some node isn't possible to visit, then its distance should be `100000000`(`1e8`). Also, If the Graph contains a negative cycle then return an array consisting of only `-1`.

Expected Time Complexity: `O(V*E)`.

Expected Auxiliary Space: `O(V)`.

Constraints:

$1 \leq V \leq 500$

```
1 // } Driver Code Ends
6 // User function Template for C++
7
8 class Solution {
9 public:
10    /* Function to implement Bellman Ford
11     * edges: vector of vectors which represents the graph
12     * S: source vertex to start traversing graph with
13     * V: number of vertices
14    */
15    vector<int> bellman_ford(int V, vector<vector<int>>& edges, int S) {
16        // Code here
17    }
18 };
19 // } Driver Code Ends
```



23:26 / 27:42 • Code >



Custom Input Com



G-41. Bellman Ford Algorithm

Problems Courses Get Hired Contests < POTD

Practice

2.05 Problem

Editorial

Submissions

Comments

C++ (g++ 5.4) Average Time: 25m

Z=0. This has a distance of 1.
For nodes 2 to 1, we can follow the path-
2-0-1, which has a distance of $1+5 = 6$,

Your Task:

You don't need to read input or print anything. Your task is to complete the function **bellman_ford()** which takes a number of vertices **V** and an **E**-sized list of lists of three integers where the three integers are **u,v**, and **w**; denoting there's an edge from **u** to **v**, which has a weight of **w** and source node **S** as input parameters and returns a list of integers where the *i*th integer denotes the distance of an *i*th node from the source node.

If some node isn't possible to visit, then its distance should be 100000000(1e8). Also, If the Graph contains a negative cycle then return an array consisting of only -1.

Expected Time Complexity: $O(V \cdot E)$.

Expected Auxiliary Space: $O(V)$.

Constraints:

$1 \leq V \leq 500$

```
1 // } Driver Code Ends
6 // User function Template for C++
7
8 class Solution {
9 public:
10    /* `Function to implement Bellman Ford
11     * edges: vector of vectors which represents the graph
12     * S: source vertex to start traversing graph with
13     * V: number of vertices
14     */
15    vector<int> bellman_ford(int V, vector<vector<int>>& edges, int S) {
16        // Code here
17    }
18 };
19 // } Driver Code Ends
```



Settings



23:26 / 27:42 • Code >



Custom Input Con

Problems Courses Get Hired Contests ↗POTD

Practice

Problem Editorial Submissions Comments

Distance from the Source (Bellman-Ford Algorithm)

Medium Accuracy: 50.02% Submissions: 22250 Points: 4

Given a weighted, directed and connected graph of V vertices and E edges, Find the shortest distance of all the vertex's from the source vertex S.

Note: If the Graph contains a negative cycle then return an array consisting of only -1.

Example 1:

Input:

E = [[0,1,9]]

S = 0

Output:

0 9

Explanation:

Shortest distance of all nodes from

C++ (g++ 5.4) Average Time: 25m

```
6 // User function Template for C++
7
8 class Solution {
9 public:
10    /* Function to implement Bellman Ford
11     * edges: vector of vectors which represents the graph
12     * S: source vertex to start traversing graph with
13     * V: number of vertices
14 */
15    vector<int> bellman_ford(int V, vector<vector<int>>& edges, int S) {
16        vector<int> dist(V, 1e8);
17        dist[S] = 0;
18        for(int i = 0;i<V-1;i++) {
19            for(auto it : edges) {
20                int u = it[0];
21                int v = it[1];
22                int wt = it[2];
23                if(dist[u] != 1e8 && dist[u] + wt < dist[v]) {
24                    dist[v] = dist[u] + wt;
25                }
26            }
27        }
28        // Nth relaxation to check negative cycle
29        for(auto it : edges) {
30            int u = it[0];
31            int v = it[1];
32            int wt = it[2];
33            if(dist[u] != 1e8 && dist[u] + wt < dist[v]) {
34                return {-1};
35            }
36        }
37
38
39        return dist;
40    }
41 };
42
43 // } Driver Code Ends
```

Custom Input

TU

G-41. Bellman Ford Algorithm

Problems Courses Get Hired Contests </POTD

2.05 Problem Editorial Submissions Comments

Distance from the Source (Bellman-Ford Algorithm)

Medium Accuracy: 50.02% Submissions: 22250 Points: 4

Given a weighted, directed and connected graph of V vertices and E edges, Find the shortest distance of all the vertex's from the source vertex S.

Note: If the Graph contains a negative cycle then return an array consisting of only -1.

Example 1:

Input:

E = [[0,1,9]]
S = 0
Output:
0 9

Explanation:
Shortest distance of all nodes from

C++ (g++ 5.4) Average Time: 25m

```
6 // User Function Template for C++
7
8 class Solution {
9 public:
10    /* Function to implement Bellman Ford
11     * edges: vector of vectors which represents the graph
12     * S: source vertex to start traversing graph with
13     * V: number of vertices
14 */
15    vector<int> bellman_ford(int V, vector<vector<int>>& edges, int S) {
16        vector<int> dist(V, 1e8);
17        dist[S] = 0;
18        for(int i = 0;i<V-1;i++) {
19            for(auto it : edges) {
20                int u = it[0];
21                int v = it[1];
22                int wt = it[2];
23                if(dist[u] != 1e8 && dist[u] + wt < dist[v]) {
24                    dist[v] = dist[u] + wt;
25                }
26            }
27        }
28        // Nth relaxation to check negative cycle
29        for(auto it : edges) {
30            int u = it[0];
31            int v = it[1];
32            int wt = it[2];
33            if(dist[u] != 1e8 && dist[u] + wt < dist[v]) {
34                return {-1};
35            }
36        }
37
38
39        return dist;
40    }
41 }
42
43 // Driven Code Ends
```

Custom Input Con

◀ ▶ ⏪ ⏩ 25:55 / 27:42 • Code > 🔍 CC HD #

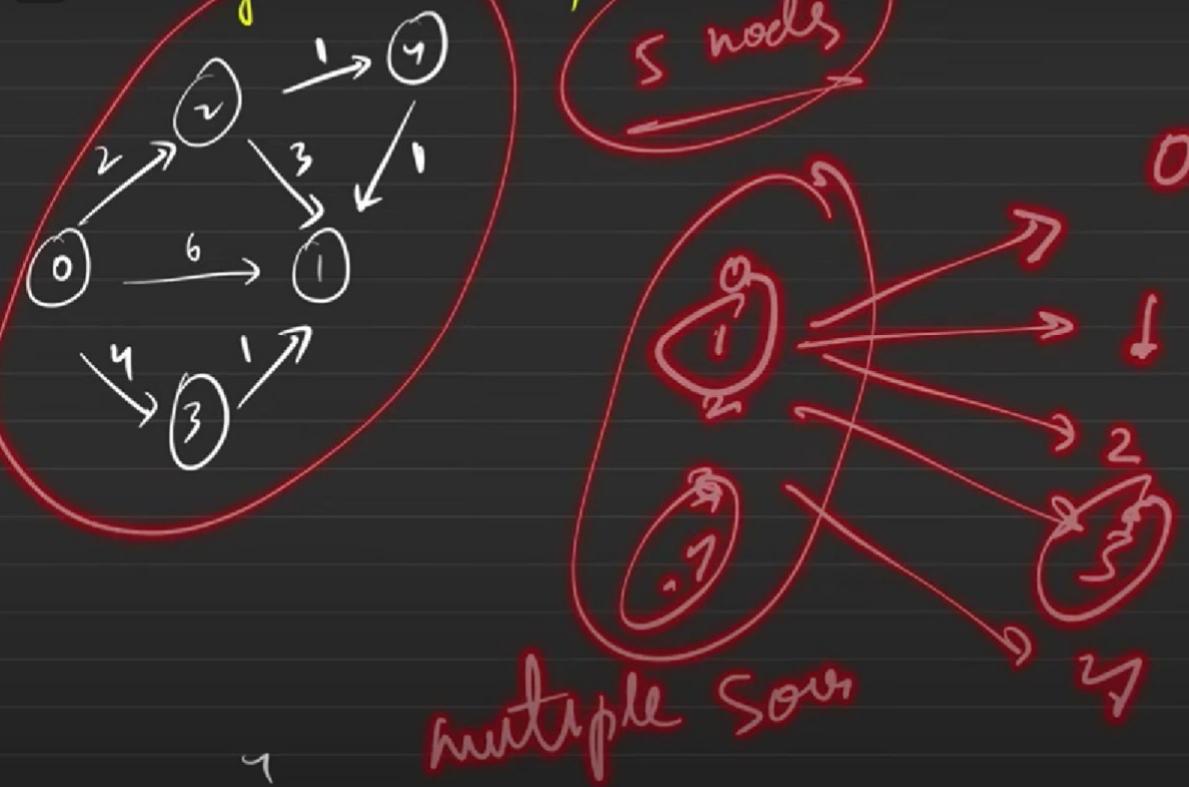
G-42. Floyd Warshall Algorithm

2.05

Floyd Warshall Algorithm

Algorithm

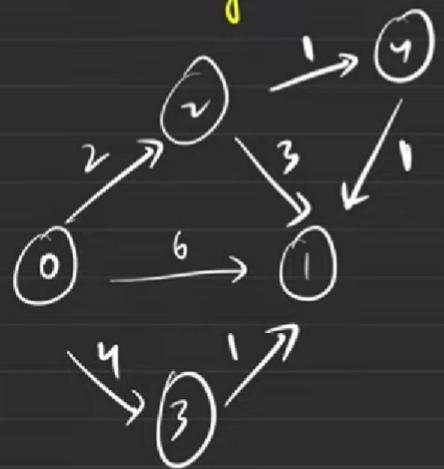
5 nodes

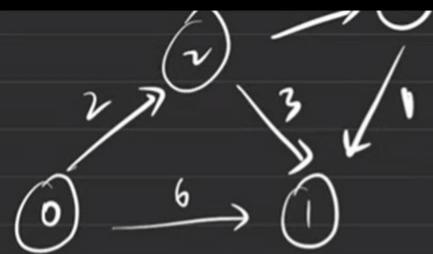


The Floyd Warshall Algorithm is for solving all pairs of shortest-path problems. The problem is to find the shortest distances between every pair of vertices in a given edge-weighted directed Graph.

It is an algorithm for finding the shortest path between all the pairs of vertices in a weighted graph. This algorithm follows the dynamic programming approach to find the shortest path.

Floyd Warshall Algorithm \rightarrow (multisource shortest Path)
 \rightarrow detect negative cycle





Note \rightarrow goes via every vertex/nodes.

0 1

$$d[0][1] \rightarrow (0 \rightarrow 1)$$

$$(0 \rightarrow 2) + (2 \rightarrow 1) \\ 2 \qquad \qquad \qquad 3 = 5$$

$$(0 \rightarrow 3) + (3 \rightarrow 1) \\ 4 \qquad \qquad \qquad 1 = 5$$

7

$$(0 \rightarrow 1) + (1 \rightarrow 1) \\ 3 \qquad \qquad \qquad 1 = 4$$



Battery status: 25% available (plugged in)

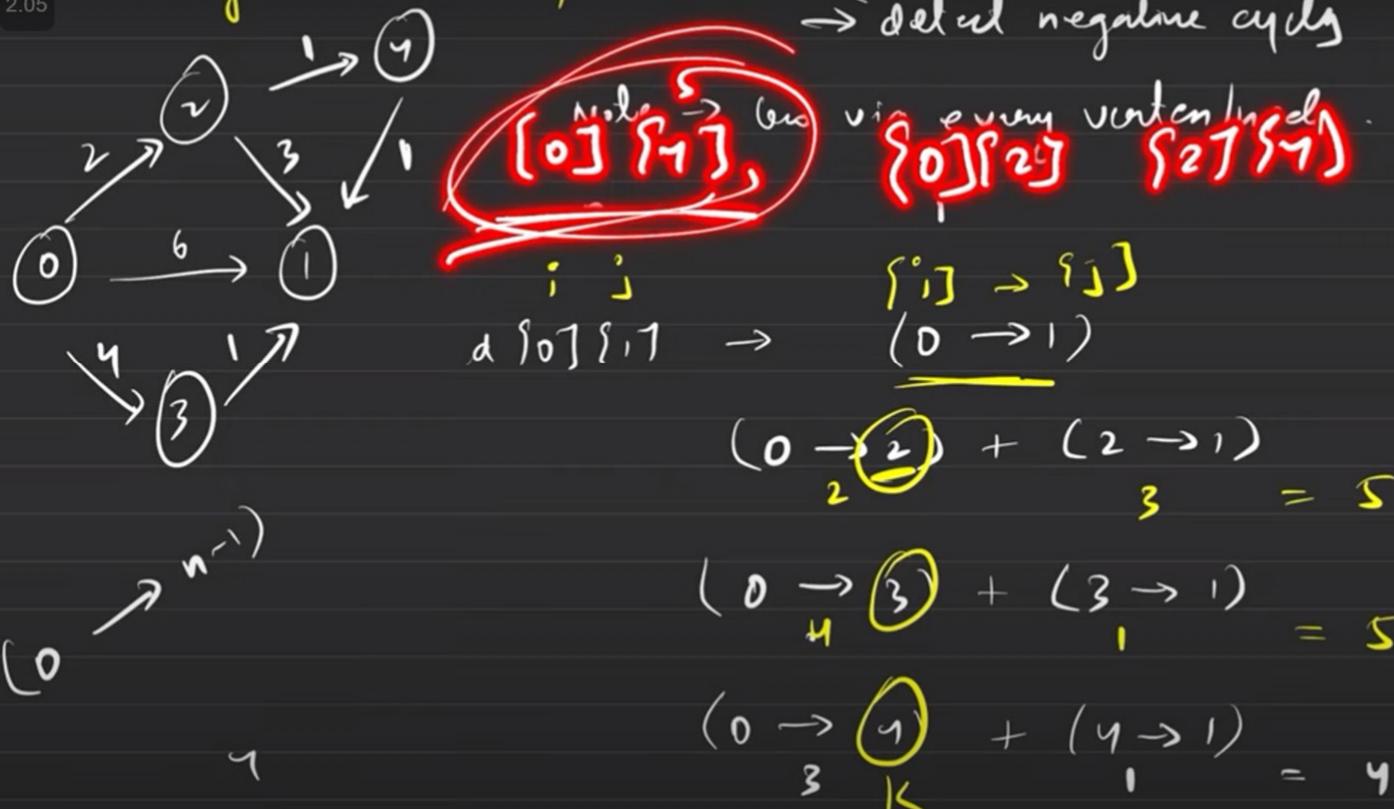
TUF

⇒ G-42. Floyd Warshall Algorithm

Floyd Warshall Algorithm

→ (multisource shortest Path)
→ detect negative cycle

2.05



$d_{01} = 2$ (from d_{01})

6:23 / 30:12

CC #

⇒ G-42. Floyd Warshall Algorithm

Floyd Warshall Algorithm

→ (multisource Shortest Path)
→ detect negative cycle



Note → Go via every vertex/node.

$$d[s_0][s_1] \rightarrow \begin{cases} s_0 \\ s_1 \end{cases} \quad [s_{ij} \rightarrow s_j]$$

$$(0 \rightarrow 2) + (2 \rightarrow 1) = 5$$

Pre computed

$$\begin{aligned} (0 \rightarrow 3) &+ (3 \rightarrow 1) = 5 \\ (0 \rightarrow 1) &+ (4 \rightarrow 1) = 4 \end{aligned}$$

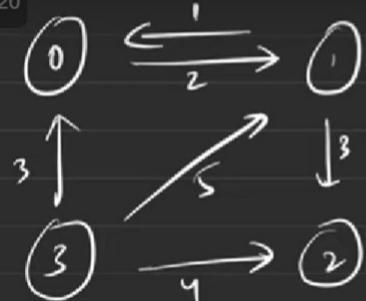
$$d[s_0][s_1] + d[s_1][s_2]$$

6:57 / 30:12



→ G-42. Floyd Warshall Algorithm

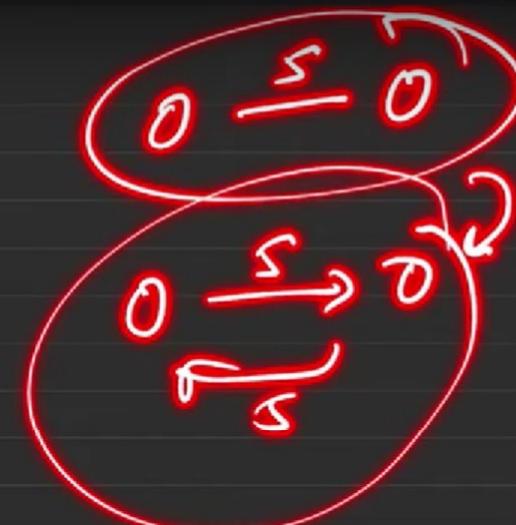
2.20



A 4x4 weight matrix for the graph:

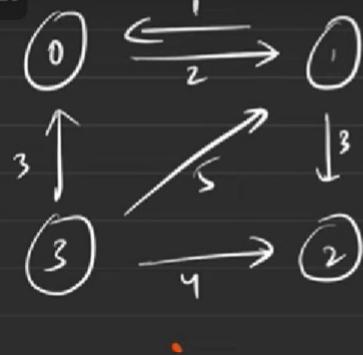
$$\begin{matrix} & 0 & 1 & 2 & 3 \\ 0 & 0 & 2 & 1 & \infty \\ 1 & \infty & 0 & 3 & \infty \\ 2 & \infty & \infty & 0 & 2 \\ 3 & 3 & 5 & 4 & 0 \end{matrix}$$

Below the matrix is the label "wst".



⇒ G-42. Floyd Warshall Algorithm

2.20



$$\text{Initial Adjacency Matrix:}$$
$$A = \begin{bmatrix} 0 & 1 & 2 & 3 \\ 0 & 0 & 1 & \infty \\ 1 & 0 & 0 & 1 \\ 2 & \infty & 0 & 0 \\ 3 & 5 & 4 & 0 \end{bmatrix}$$

wst

$$\text{Resultant Matrix after Step 1:}$$
$$A' = \begin{bmatrix} 0 & 1 & 2 & 3 \\ 0 & 0 & 2 & \infty \\ 1 & 0 & 0 & 3 \\ 2 & \infty & 0 & 0 \\ 3 & 3 & 4 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 & 1 & 2 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 1 & 1 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 1 & 2 \end{bmatrix}$$

1 + α

α

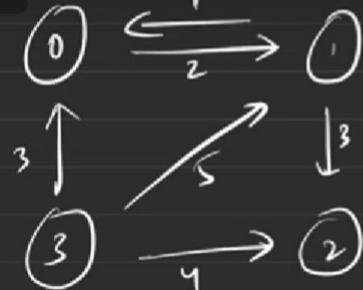
$\begin{bmatrix} 1 & 1 & 1 & 3 \end{bmatrix}$



⇒ G-42. Floyd Warshall Algorithm



2.20

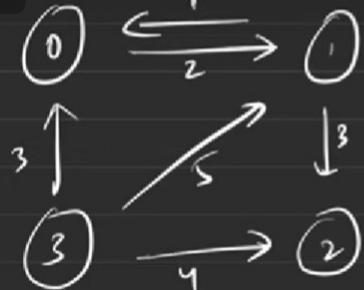
wst

$$\{0\} \cap \{2\} \rightarrow \cancel{\{0\} \cap \{0\}} + \{0\} \cap \{2\}$$



⇒ G-42. Floyd Warshall Algorithm

2.20



An initial weight matrix w_{ij} for a 4x4 graph:

$$w = \begin{bmatrix} 0 & 1 & 2 & 3 \\ 0 & 0 & 2 & 1 & \infty \\ 1 & \infty & 0 & 3 & \infty \\ 2 & \infty & \infty & 0 & 2 \\ 3 & 3 & 5 & 4 & 0 \end{bmatrix}$$

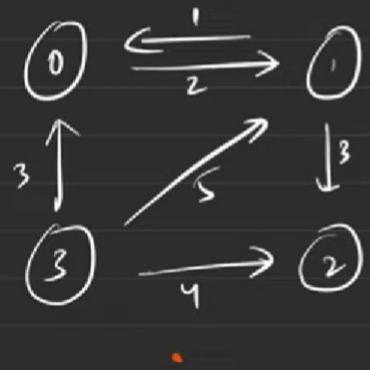
The transpose of the weight matrix is also shown:

$$w^T = \begin{bmatrix} 0 & 1 & 2 & 3 \\ 0 & 0 & 2 & 1 & \infty \\ 1 & \infty & 0 & 3 & \infty \\ 2 & \infty & \infty & 0 & 2 \\ 3 & 3 & 5 & 4 & 0 \end{bmatrix}$$

Below the matrices, the label "wst" is written.

A diagram illustrating the Floyd-Warshall update step. It shows the set of nodes $\{1\}$ and $\{0\}$ being combined to form the set $\{1, 3\}$. The set $\{1, 3\}$ is enclosed in a red oval, and the set $\{0\}$ is enclosed in another red oval. An arrow points from the union of these sets to the original sets, indicating the update process.





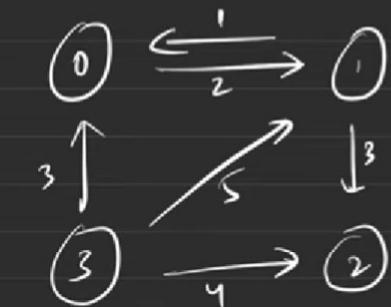
$$\begin{matrix}
 & 0 & 1 & 2 & 3 \\
 0 & \left[\begin{matrix} 0 & 2 & 1 & \alpha \\ 1 & 0 & 3 & \alpha \\ 2 & \alpha & 0 & \alpha \\ 3 & \alpha & 4 & 0 \end{matrix} \right] & \xrightarrow{\text{row } 0} & \left[\begin{matrix} 0 & 1 & 2 & 3 \\ 0 & 2 & \alpha & \alpha \\ 1 & 0 & 0 & 0 \\ 2 & \alpha & 0 & 0 \\ 3 & \alpha & 0 & 0 \end{matrix} \right] \\
 & \underline{\text{wst}} & &
 \end{matrix}$$

$$\begin{bmatrix} 1 & 3 & 9 & 2 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 3 & 9 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 9 & 2 \end{bmatrix} \\
 1 \quad + \quad \alpha$$



⇒ G-42. Floyd Warshall Algorithm

2.20



$$\text{Initial Adjacency Matrix:}$$
$$A = \begin{bmatrix} 0 & 2 & 1 & \infty \\ 1 & 0 & 3 & \infty \\ \infty & 3 & 0 & \infty \\ 3 & 5 & 4 & 0 \end{bmatrix}$$
$$\xrightarrow{\text{Floyd Warshall}}$$
$$\text{Resultant Matrix:}$$
$$A = \begin{bmatrix} 0 & 2 & \infty & \infty \\ 1 & 0 & 3 & \infty \\ \infty & \infty & 0 & 0 \\ 3 & 5 & 4 & 0 \end{bmatrix}$$

$$\{1\} \{2\} \rightarrow \{1\} \{0\}$$

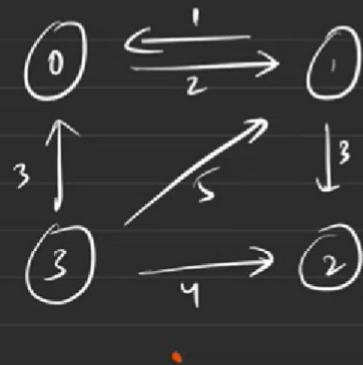
$$\{2\} \{3\} \rightarrow \{2\} \underline{\{0\}}$$

$$\{2\} \{3\} \rightarrow \{2\} \{0\}$$



⇒ G-42. Floyd Warshall Algorithm

2.20



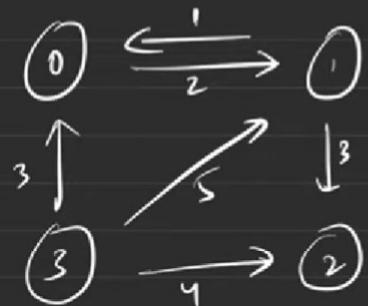
$$\text{Initial Matrix: } \begin{bmatrix} 0 & 1 & 2 & 3 \\ 0 & 0 & 2 & 1 & \infty \\ 1 & \infty & 0 & 3 & \infty \\ 2 & \infty & \infty & 0 & 1 \\ 3 & 3 & 5 & 4 & 0 \end{bmatrix}$$

wst

$$\xrightarrow{\text{Step 1}} \begin{bmatrix} 0 & 1 & 2 & 3 \\ 0 & 0 & 2 & \infty & \infty \\ 1 & \infty & 0 & 3 & \infty \\ 2 & \infty & \infty & 0 & 1 \\ 3 & 3 & 5 & 4 & 0 \end{bmatrix}$$

$$\begin{matrix} 9 & 3 & 3 & 9 & 1 & 3 \\ \rightarrow & & & & & \\ & 9 & 3 & 7 & 8 & 0 & 3 & + & 9 & 0 & 3 & 9 & 1 & 3 \\ & 3 & & & & & 2 \\ & & & & & = & 5 \end{matrix}$$





$$\begin{array}{c}
 0 \xleftrightarrow[2]{\quad} 1 \\
 3 \uparrow \quad \downarrow 3 \\
 3 \xrightarrow[4]{\quad} 2
 \end{array}
 \xrightarrow{\text{wst}}
 \begin{bmatrix}
 0 & 1 & 2 & 3 \\
 0 & 2 & 1 & 1 \\
 1 & 0 & 3 & 1 \\
 2 & 1 & 2 & 1 \\
 3 & 5 & 4 & 0
 \end{bmatrix}
 \xrightarrow{\text{wst}}
 \begin{bmatrix}
 0 & 1 & 2 & 3 \\
 0 & 2 & 1 & 1 \\
 1 & 0 & 3 & 1 \\
 2 & 1 & 2 & 1 \\
 3 & 5 & 4 & 0
 \end{bmatrix}$$

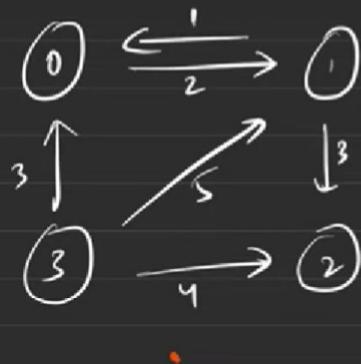
$$\begin{array}{l}
 \dots \\
 \{5\}921 \rightarrow \textcircled{73}\{903} + \textcircled{10}\{921}
 \end{array}$$



TUF

► G-42. Floyd Warshall Algorithm

2.20



$$\text{Initial Weight Matrix: } \begin{bmatrix} 0 & 2 & 1 & \infty \\ 1 & 0 & 3 & \infty \\ \infty & 2 & 0 & \infty \\ 3 & 5 & 4 & 0 \end{bmatrix}$$

wst

$$\text{After 1st iteration (via 1): } \begin{bmatrix} 0 & 2 & \infty & \infty \\ 1 & 0 & 3 & \infty \\ \infty & 0 & 0 & \infty \\ 3 & 5 & 4 & 0 \end{bmatrix}$$



16:57 / 30:12



G-42. Floyd Warshall Algorithm.



2.20

$$\begin{array}{c}
 \text{0} \left[\begin{array}{cccc} 0 & 2 & 1 & \alpha \\ 1 & 0 & 3 & \alpha \\ 2 & \alpha & 0 & \alpha \\ 3 & 3 & 5 & 0 \end{array} \right] \xrightarrow{\text{0}} \\
 \text{1} \left[\begin{array}{cccc} 0 & 2 & \alpha & \alpha \\ 1 & 0 & 3 & \alpha \\ 2 & \alpha & 0 & \alpha \\ 3 & 3 & 5 & 0 \end{array} \right] \\
 \text{2} \left[\begin{array}{cccc} 0 & 2 & \alpha & \alpha \\ 1 & 0 & 3 & \alpha \\ 2 & \alpha & 0 & \alpha \\ 3 & 3 & 5 & 0 \end{array} \right] \\
 \text{3} \left[\begin{array}{cccc} 0 & 2 & \alpha & \alpha \\ 1 & 0 & 3 & \alpha \\ 2 & \alpha & 0 & \alpha \\ 3 & 3 & 5 & 0 \end{array} \right]
 \end{array}$$

wst

$$\begin{aligned}
 \{0\} \{2\} &\rightarrow \{0\} \{1\} + \{1\} \{2\} \\
 2 &+ 3 \\
 \{0\} \{3\} &\rightarrow \{0\} \{1\} + \{1\} \{3\}
 \end{aligned}$$

$$\begin{array}{c}
 \text{0} \left[\begin{array}{ccc} 0 & 2 & 3 \\ 1 & 0 & 3 \\ 2 & \alpha & 0 \\ 3 & 5 & 0 \end{array} \right] \xrightarrow{\text{via } 1} \\
 \text{1} \left[\begin{array}{ccc} 0 & 2 & 3 \\ 1 & 0 & 3 \\ 2 & \alpha & 0 \\ 3 & 5 & 0 \end{array} \right] \\
 \text{2} \left[\begin{array}{ccc} 0 & 2 & 3 \\ 1 & 0 & 3 \\ 2 & \alpha & 0 \\ 3 & 5 & 0 \end{array} \right] \\
 \text{3} \left[\begin{array}{ccc} 0 & 2 & 3 \\ 1 & 0 & 3 \\ 2 & \alpha & 0 \\ 3 & 5 & 0 \end{array} \right]
 \end{array}$$



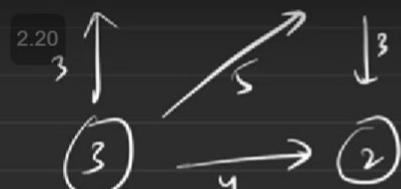
48 of 56



17:50 / 30:12



G-42. Floyd Warshall Algorithm



$$\begin{array}{c}
 \text{2.20} \\
 \begin{array}{ccc}
 3 & \nearrow 5 & \downarrow 3 \\
 \textcircled{3} & \xrightarrow{4} & \textcircled{2}
 \end{array}
 \end{array}
 \quad
 \begin{array}{c}
 1 \\
 2 \\
 3
 \end{array}
 \left[\begin{array}{ccccc}
 0 & 2 & \cancel{4} & \cancel{4} \\
 1 & 0 & 3 & \cancel{4} \\
 \cancel{4} & \cancel{4} & 0 & \cancel{4} \\
 3 & 5 & 4 & 0
 \end{array} \right]
 \xrightarrow{0}
 \begin{array}{c}
 0 \\
 1 \\
 2 \\
 3
 \end{array}
 \left[\begin{array}{ccccc}
 0 & 2 & \cancel{4} & \cancel{4} \\
 1 & 0 & 3 & \cancel{4} \\
 \cancel{4} & \cancel{4} & 0 & \cancel{4} \\
 \underline{3} & \underline{5} & \underline{4} & 0
 \end{array} \right]$$

wst

$$\{2\} \{5\} \rightarrow \{2\} \{1\} + \{1\} \{5\}$$

$$\{2\} \{3\} \rightarrow \{2\} \{1\} + \{1\} \{3\}$$

$$\begin{array}{l}
 \{3\} \{5\} \rightarrow \{3\} \{1\} + \{1\} \{5\} \\
 \quad \quad \quad 5 + 1 = \underline{\underline{6}}
 \end{array}$$

$$\begin{array}{l}
 \{3\} \{2\} = \{3\} \{1\} + \{1\} \{2\} \\
 \quad \quad \quad 5 + 3 = \underline{\underline{8}}
 \end{array}$$

$$\begin{array}{c}
 \text{via } 1 \\
 0 \\
 1 \\
 2 \\
 3
 \end{array}
 \left[\begin{array}{ccccc}
 0 & 2 & \cancel{4} & \cancel{4} \\
 0 & 0 & 5 & \cancel{4} \\
 1 & 0 & 3 & \cancel{4} \\
 \cancel{4} & \cancel{4} & 0 & - \\
 3 & 5 & - & 0
 \end{array} \right]$$



19:30 / 30:12



G-42. Floyd Warshall Algorithm



$$\begin{array}{c} \text{2.20} \\ 3 \uparrow \quad \downarrow 3 \\ \textcircled{3} \rightarrow \textcircled{2} \\ . \end{array} \quad \begin{matrix} & 0 & 2 & 1 & 1 \\ 1 & 0 & 3 & \cancel{1} & \cancel{1} \\ 2 & \cancel{1} & 0 & 0 & \cancel{1} \\ 3 & 3 & 5 & 4 & 0 \end{matrix} \xrightarrow{0} \begin{matrix} 0 & 2 & \cancel{1} & \cancel{1} \\ 1 & 0 & 3 & \cancel{1} \\ 2 & \cancel{1} & 0 & 0 \\ 3 & 3 & 5 & 4 & 0 \end{matrix}$$

wst

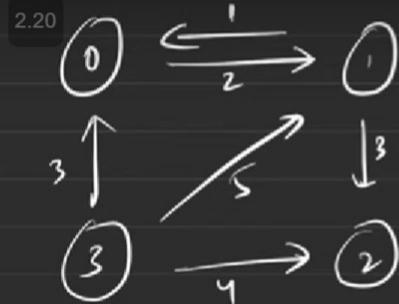
$$\left[\begin{array}{c} \checkmark \\ \checkmark \\ \checkmark \end{array} \right] \xleftarrow{\text{via } 3} \left[\begin{array}{c} - \\ - \\ - \end{array} \right] \xleftarrow{\text{via } 2} \begin{matrix} 0 & 2 & 5 & \cancel{1} \\ 1 & 0 & 3 & \cancel{1} \\ 2 & \cancel{1} & 0 & - \\ 3 & 3 & 5 & 4 & 0 \end{matrix}$$

via 1

→ shortest Path



⇒ G-42. Floyd Warshall Algorithm



$$wst = \begin{bmatrix} 0 & 1 & 2 & 3 \\ 0 & 0 & 2 & 1 & \infty \\ 1 & 1 & 0 & 3 & \infty \\ 2 & \infty & 2 & 0 & \infty \\ 3 & 3 & 5 & 4 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 1 & 2 & 3 \\ 0 & 0 & 2 & 1 & \infty \\ 1 & 1 & 0 & 3 & \infty \\ 2 & \infty & 2 & 0 & \infty \\ 3 & 3 & 5 & 4 & 0 \end{bmatrix}$$

via 1

\downarrow
 \downarrow
 \downarrow

$$\text{fun}(i=0 ; i < n ; i++)$$

$$\text{fun}(j = 0 ; j < n ; j++)$$

$$\begin{bmatrix} 0 & 1 & 2 & 3 \\ 0 & 0 & 2 & 5 & \infty \\ 1 & 1 & 0 & 3 & \infty \\ 2 & \infty & 2 & 0 & - \\ 3 & 3 & 5 & 4 & 0 \end{bmatrix}$$

$$wst[s_i][s_j] = \min \{ wst[s_i][s_j], \\ cost[s_i][s_j] + cost[s_{i+1}][s_j] \})$$



21:58 / 30:12



G-42. Floyd Warshall Algorithm

2.20 $(3) \xrightarrow{4} (2)$

$\{$
 $\text{fun}(\text{via} = 0; \text{via} < n; \text{via}++)$
 $\{$
 $\text{fun}(i = 0; i < n; i++)$
 $\{$
 $\text{fun}(j = 0; j < n; j++)$
 $\}$

$$\text{cost}[s_i][s_j] = \min\{\text{cost}[s_i][s_j], \text{cost}[s_i][\text{via}] + \text{cost}[\text{via}][s_j]\})$$

$\}$
 $\}$
 $\}$

$3 [3 \ 5 \ 4 \ 0]$

$3 [3 \ 5 \ 4 \ 0]$

$3 [3 \ 5 \ 4 \ 0]$

$0 [0 \ 2 \ 5 \ 4]$
 $1 [0 \ 3 \ 4]$
 $2 [4 \ 4 \ 0 \ -]$
 $3 [5 \ 4 \ 0 \ 0]$



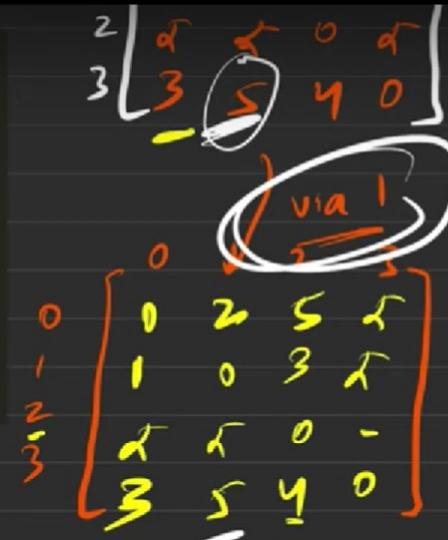
≡ G-42. Floyd Warshall Algorithm

^{2.20}The algorithm is not much intuitive as the other ones. It is more of a brute force, where all combination of paths have been tried to get the shortest paths.

Nothing to be panic much on the intuition, it is a simple brute on all paths. Focus on the three for loops.

```
for(j = 0; j < n; j++)
```

$$\underline{\text{cost } S_i \cup S_j} = \min \{ \text{cost } S_i \cup S_j, \\ \text{cost } S_i \cup \{v_{i+1}^a\} + \text{cost } S_{i+1} \cup S_j \})$$



⇒ G-42. Floyd Warshall Algorithm

2.20

→ How to detect a negative cycle



$$\text{dst} = \boxed{-3}$$

$-2 + -3 = -3$



23:43 / 30:12



→ G-42. Floyd Warshall Algorithm

2.20

x 2
②

ost = (-3)

fun ($i = 0 \rightarrow n$)
}

if ($ost[s_i]s_i < 0$)

{ "negative cycle"

}

}



24:06 / 30:12



G-42. Floyd Warshall Algorithm

Problems Courses Get Hired Contests POTD

Practice

2.20 Problem Editorial Submissions Comments

The problem is to find the shortest distances between every pair of vertices in a given edge-weighted directed Graph. The Graph is represented as an adjacency matrix, and the matrix denotes the weight of the edges (if it exists) else -1.

Do it in-place.

Example 1:

```
Input: matrix = {{0,25},{-1,0}}
Output: {{0,25},{-1,0}}
Explanation: The shortest distance between every pair is already given(if it exists).
```

Example 2:

```
Input: matrix = {{0,1,43},{1,0,6},{-1,-1,0}}
Output: {{0,1,7},{1,0,6},{-1,-1,0}}
Explanation: We can reach 3 from 1 as 1->2->3 and the cost will be 1+6=7 which is less than 43.
```

Your Task:
You don't need to read, return or print anything. Your task is to complete the function `shortest_distance()` which takes the

C++ (g++ 5.4) Average Time: 15m

```
1: // } Driver Code Ends
8: //User function template for C++
9:
10: class Solution {
11: public:
12:     void shortest_distance(vector<vector<int>>&matrix){
13:         //Your code here
14:     }
15: };
16: // } Driver Code Ends
```

Custom Input Comp

CC

24:34 / 30:12

TUF

G-42. Floyd Warshall Algorithm

Problems Courses Get Hired Contests POTD

Practice

2.20 Problem Editorial Submissions Comments

The problem is to find the shortest distances between every pair of vertices in a given edge-weighted directed Graph. The Graph is represented as an adjacency matrix, and the matrix denotes the weight of the edges (if it exists) else -1.

Do it in-place.

Example 1:

```
Input: matrix = {{0,25},{-1,0}}
Output: {{0,25},{-1,0}}
Explanation: The shortest distance between every pair is already given(if it exists).
```

Output Window

Compilation Results Custom Input

```
Request Queued.
Evaluating
```

```
1 // Driver Code Ends
2 //User function template for C++
3
4 class Solution {
5 public:
6     void shortest_distance(vector<vector<int>>&matrix){
7         int n = matrix.size();
8         for(int i = 0;i<n;i++) {
9             for(int j = 0;j<n;j++) {
10                 if(matrix[i][j] == -1) {
11                     matrix[i][j] = 1e9;
12                 }
13                 if(i == j) matrix[i][j] = 0;
14             }
15         }
16         for(int k = 0;k<n;k++) {
17             for(int i = 0;i<n;i++) {
18                 for(int j = 0;j<n;j++) {
19                     matrix[i][j] = min(matrix[i][j],
20                         matrix[i][k] + matrix[k][j]);
21                 }
22             }
23         }
24     }
25 }
26
27
28
29
30
31
32
33
34
35     for(int i = 0;i<n;i++) {
36         for(int j = 0;j<n;j++) {
37             if(matrix[i][j] == 1e9) {
38                 matrix[i][j] = -1;
39             }
40         }
41     }
42 };
43
44 // Driver Code Ends
```

Custom Input Comp

◀ ▶ ⏪ ⏩ 27:05 / 30:12

CC ⚙ #



G-42. Floyd Warshall Algorithm

Problems Courses Get Hired Contests POTD

Practice

2.20 Problem Editorial Submissions Comments

The problem is to find the shortest distances between every pair of vertices in a given edge-weighted directed Graph. The Graph is represented as an adjacency matrix, and the matrix denotes the weight of the edges (if it exists) else -1.

Do it in-place.

Example 1:

Input: matrix = {{0,25},{-1,0}}
Output: {{0,25},{-1,0}}

Explanation: The shortest distance between every pair is already given(if it exists).

Example 2:

Input: matrix = {{0,1,43},{1,0,6},{-1,-1,0}}
Output: {{0,1,7},{1,0,6},{-1,-1,0}}

Explanation: We can reach 3 from 1 as 1->2->3 and the cost will be 1+6=7 which is less than 43.

Your Task:

You don't need to read, return or print anything. Your task is to complete the function `shortest_distance()` which takes the

```
C++ (g++ 5.4) ✓ Average Time: 15m
1 // Driver Code Ends
2 //User function template for C++
3
4 class Solution {
5 public:
6     void shortest_distance(vector<vector<int>>&matrix){
7         int n = matrix.size();
8         for(int i = 0;i<n;i++) {
9             for(int j = 0;j<n;j++) {
10                 if(matrix[i][j] == -1) {
11                     matrix[i][j] = 1e9;
12                 }
13                 if(i == j) matrix[i][j] = 0;
14             }
15         }
16
17         for(int k = 0;k<n;k++) {
18             for(int i = 0;i<n;i++) {
19                 for(int j = 0;j<n;j++) {
20                     matrix[i][j] = min(matrix[i][j],
21                                         matrix[i][k] + matrix[k][j]);
22                 }
23             }
24         }
25
26         for(int i = 0;i<n;i++) {
27             if(matrix[i][i] < 0) {
28                 }
29         }
30     }
31
32     for(int i = 0;i<n;i++) {
33         if(matrix[i][i] < 0) {
34             }
35         }
36     }
37
38
39     for(int i = 0;i<n;i++) {
40         for(int j = 0;j<n;j++) {
41             if(matrix[i][j] == 1e9) {
42                 matrix[i][j] = -1;
43             }
44         }
45     }
46 }
```

Custom Input Compile

◀ ▶ ⏪ ⏩ 27:43 / 30:12

CC ⚙ #

→ G-42. Floyd Warshall Algorithm

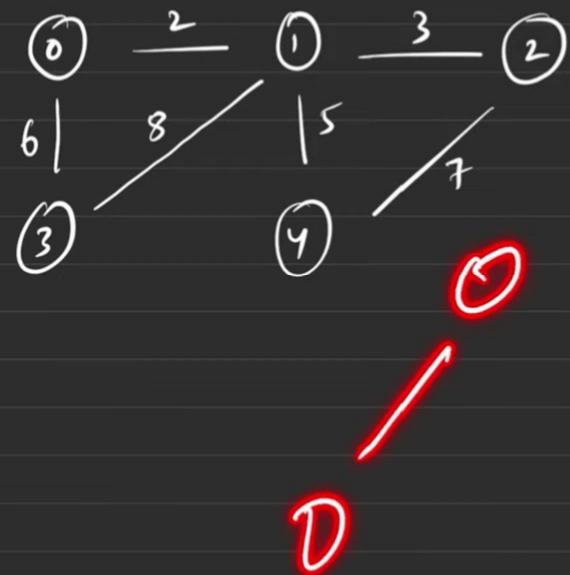
2.20



29:11 / 30:12



Minimum Spanning Tree → MST



$$\begin{array}{lcl} N \text{ nodes} & = & 5 \\ M \text{ edges} & = & 6 \end{array}$$

A tree in which we have
N nodes \geq N-1 edges
& all nodes are reachable
from each other .

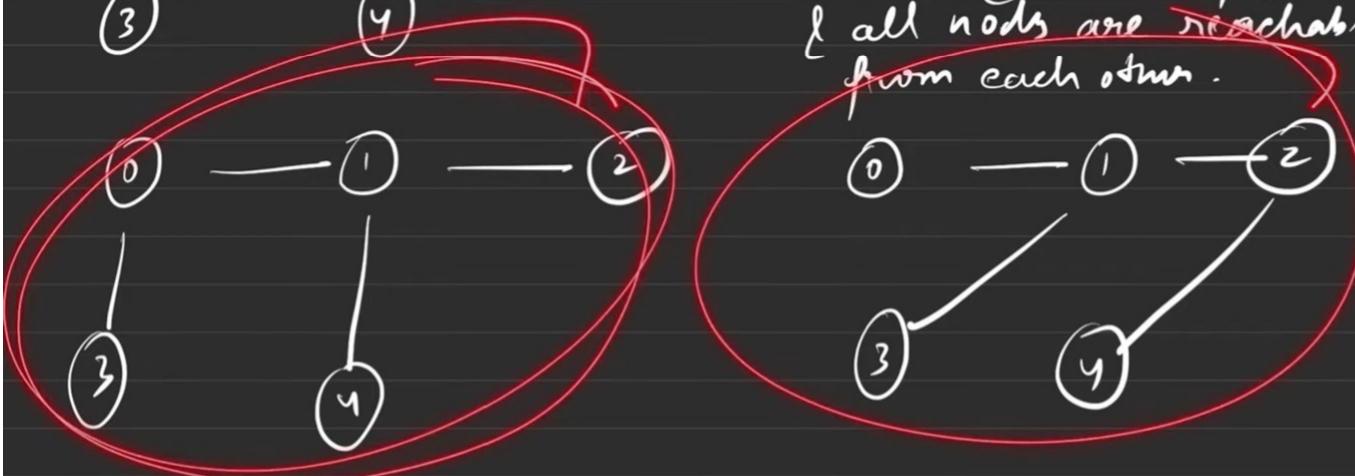


Minimum Spanning Tree \rightarrow MST



$$\begin{array}{lcl} \text{N nodes} & = 5 \\ \text{M edges} & = 6 \end{array}$$

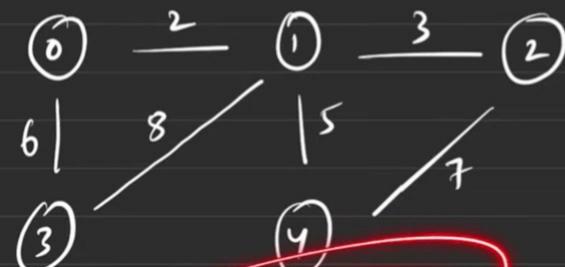
A tree in which we have
N nodes \geq N-1 edges
& all nodes are reachable
from each other.



⇒ G-44. Minimum Spanning Tree - Theory

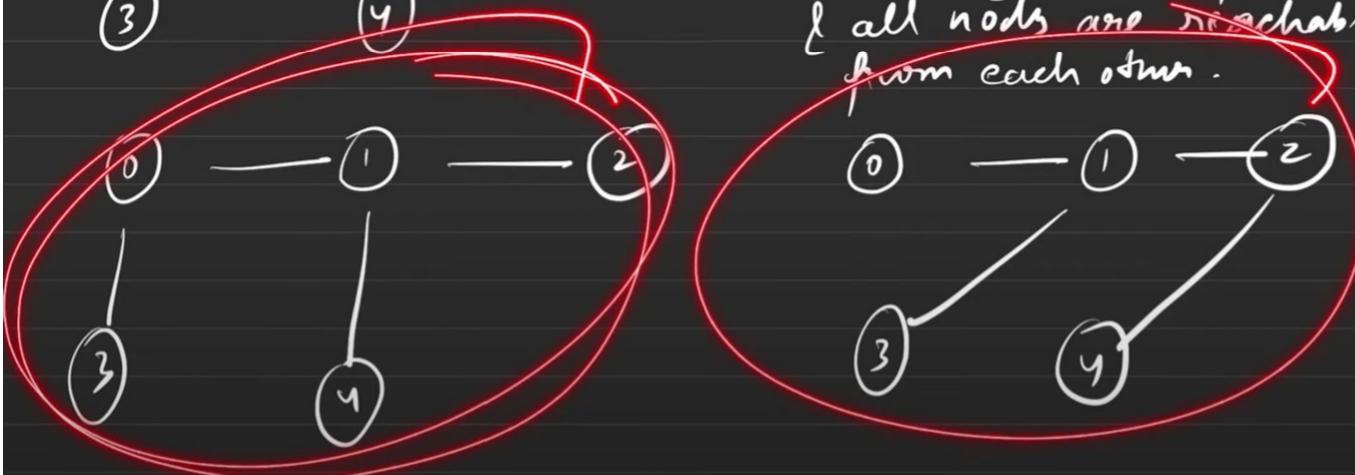
Minimum Spanning Tree → MST

1.75



$$\begin{array}{lcl} N \text{ nodes} & = 5 \\ M \text{ edges} & = 6 \end{array}$$

A tree in which we have
 N nodes $\geq N-1$ edges
& all nodes are ~~reachable~~ from each other.



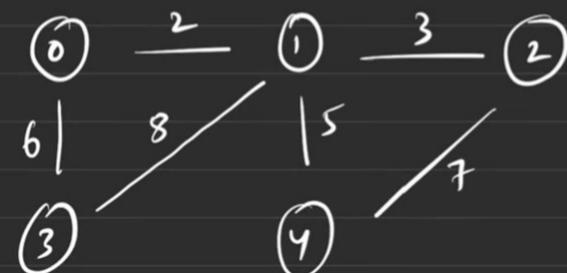
4:20 / 7:58



⇒ G-44. Minimum Spanning Tree - Theory

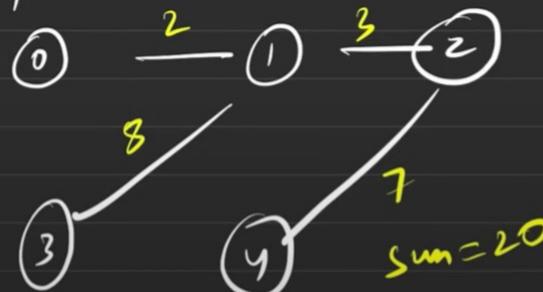
Minimum Spanning Tree → MST

1.75



$$\begin{array}{l} N \text{ nodes} \\ M \text{ edges} \end{array} = \begin{array}{l} 5 \\ 6 \end{array}$$

A tree in which we have
 N nodes & $N-1$ edges
& all nodes are reachable
from each other.



⇒ G-44. Minimum Spanning Tree - Theory

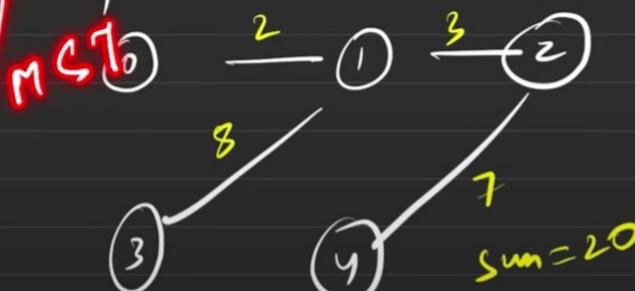
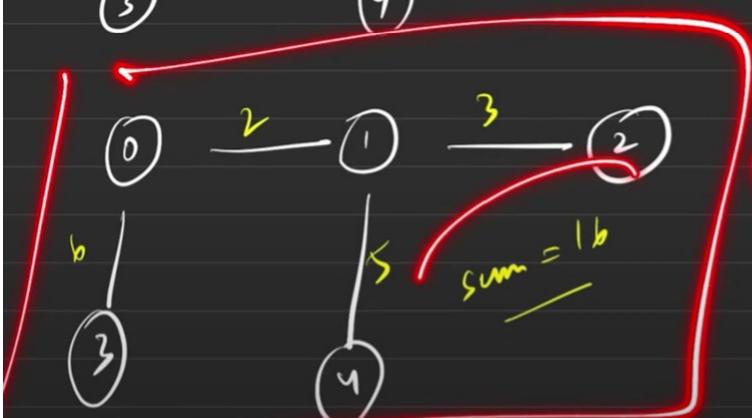
Minimum Spanning Tree → MST

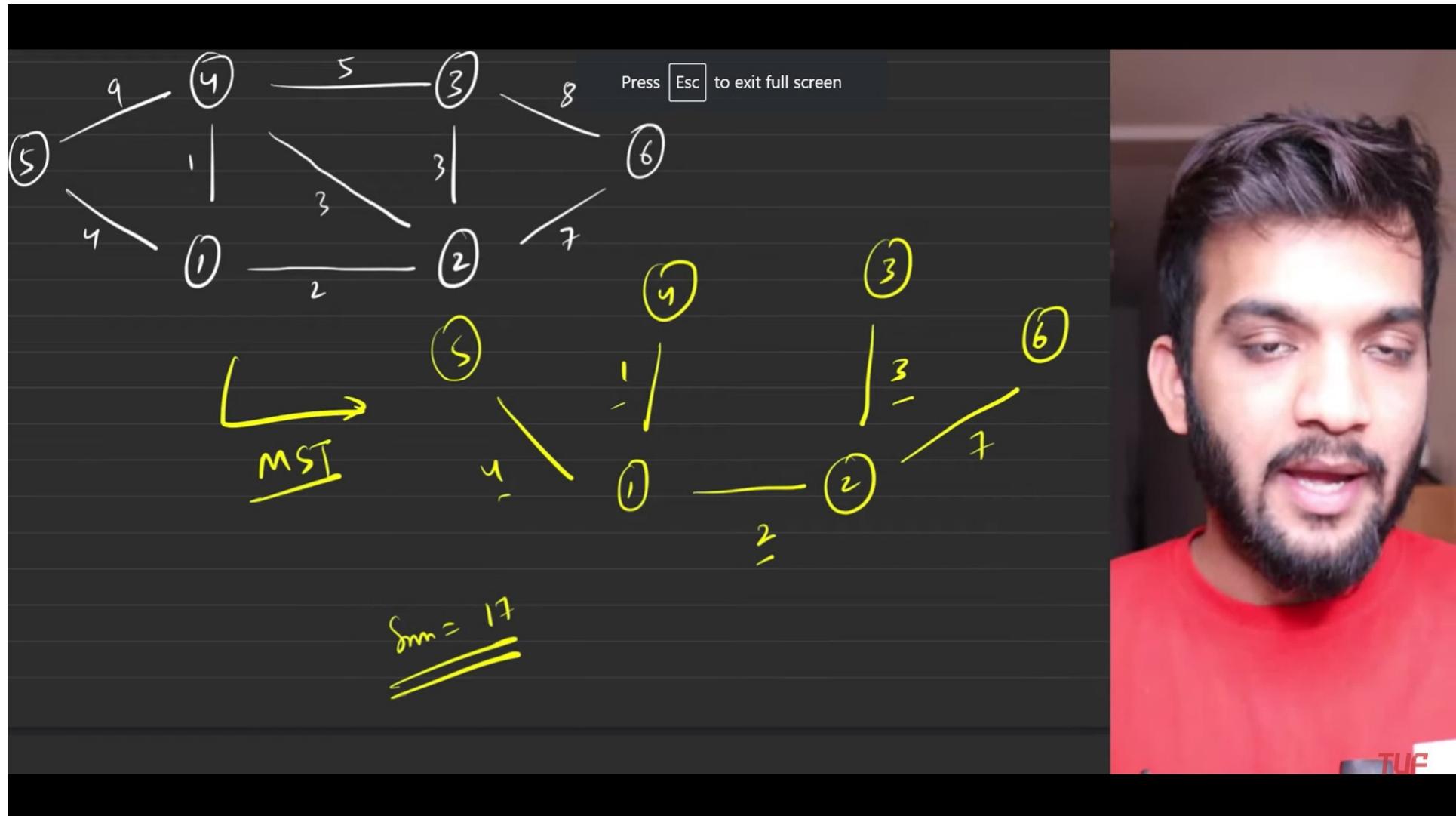
1.75

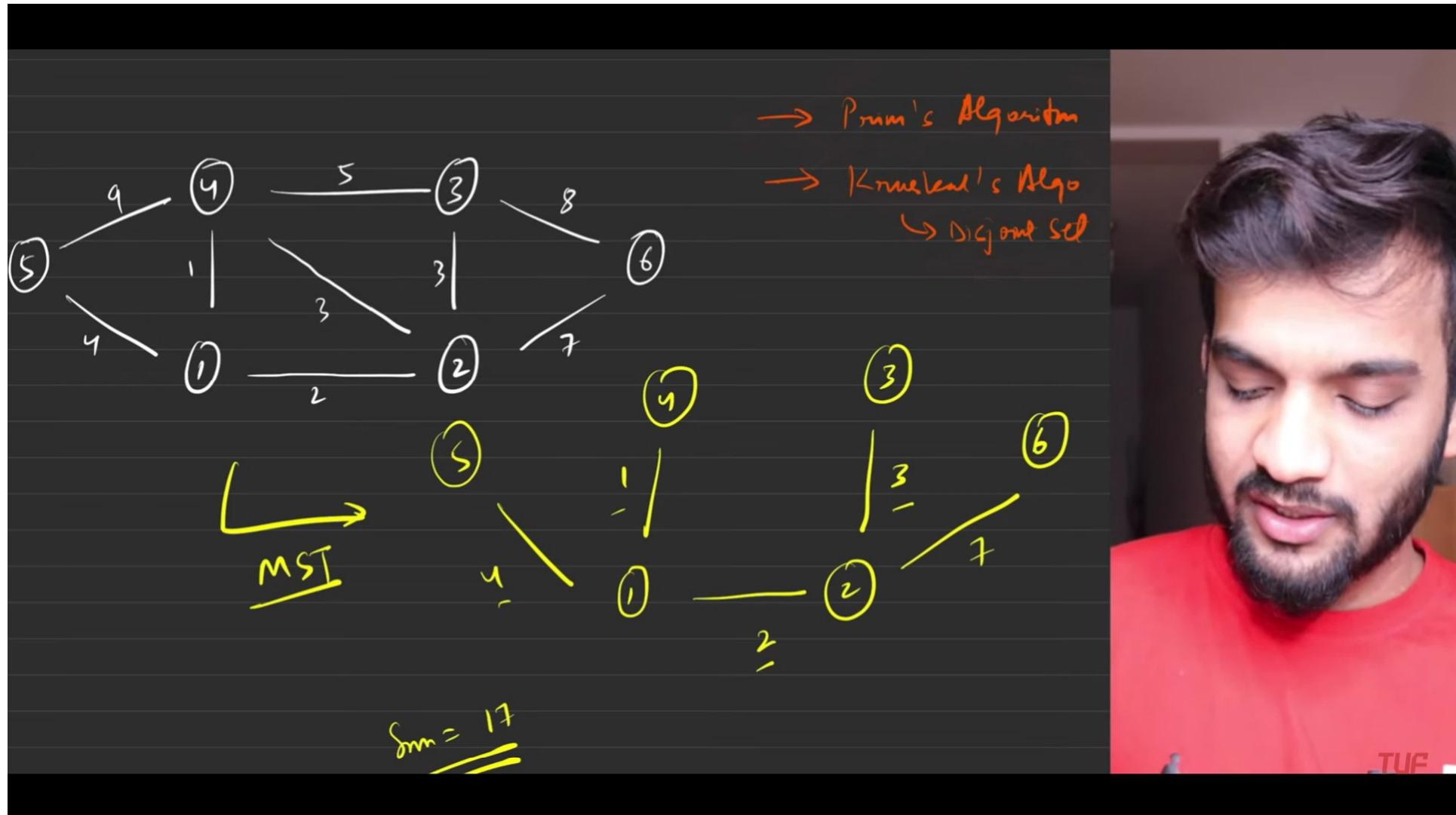


$$\begin{array}{l} N \text{ nodes} \\ M \text{ edges} \end{array} = 5$$
$$= 6$$

A tree in which we have
N nodes & N-1 edges
& all nodes are reachable
from each other.







LFU Cache - (Microsoft) : Explanation + Live Coding

2.05

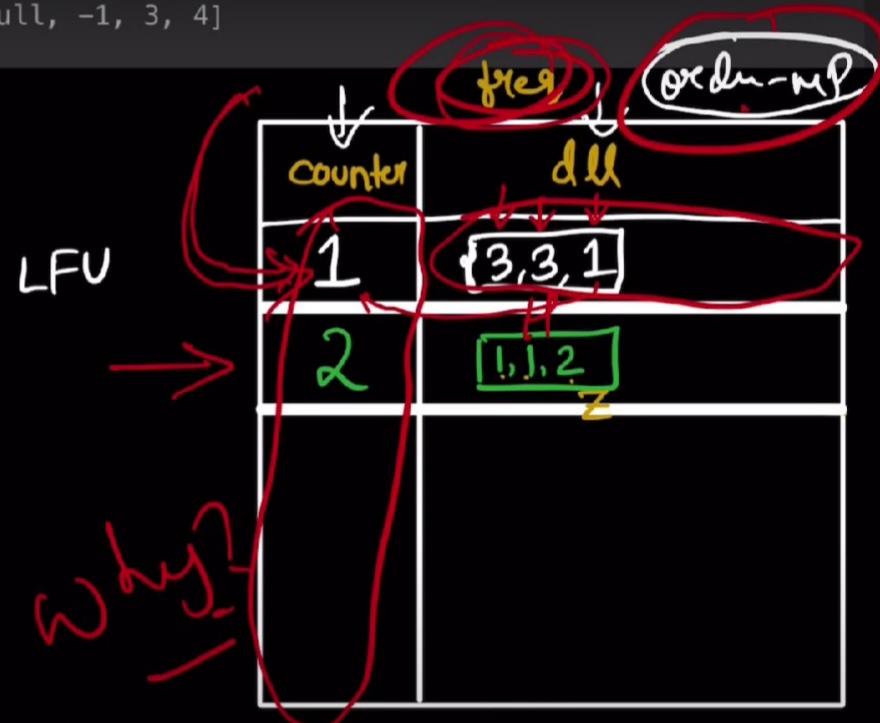
[null, null, null, 1, null, -1, 3, null, -1, 3, 4]

Key = 3
val = 3

Output:-

size = 2

Key	Address
1	Z
2	Y
3	H



30:52 / 49:28



LFU Cache - (Microsoft) : Explanation + Live Coding

2.05

freq[1] push-front ({ 3, 3, 1 });

dll

list



30:33 / 49:28



Sprinklr Placement Opportunity 2 Microsoft Teams - loading

teams.microsoft.com/_#/modern-calling/

VideoDownloadCo... Gmail YouTube Maps Classes Mentor and Group...

01:30 Chat People Raise React View More Camera Mic Share Leave

Your mic has been disabled
You can no longer unmute.

Technologies/Tech-Stack Used

Databases	Middleware	CI/Deployment	Frontend	AI
cassandra	kafka	docker	TS	Keras
mongoDB	Spark	GitLab	NEXT.js	PYTORCH
redis	MQTT	MESOS	Redux	spaCy
SCYLLA	ZooKeeper	KEDA	webpack	Istio
MySQL	GraphQL	Gradle	BABEL	learun

© 2021 Sprinklr, Inc. All rights reserved.

Lokesh Saini

Shivani Sin... Participants

MG

32°C Mostly cloudy

Search

ENG IN

4:35 PM 7/12/2023 4

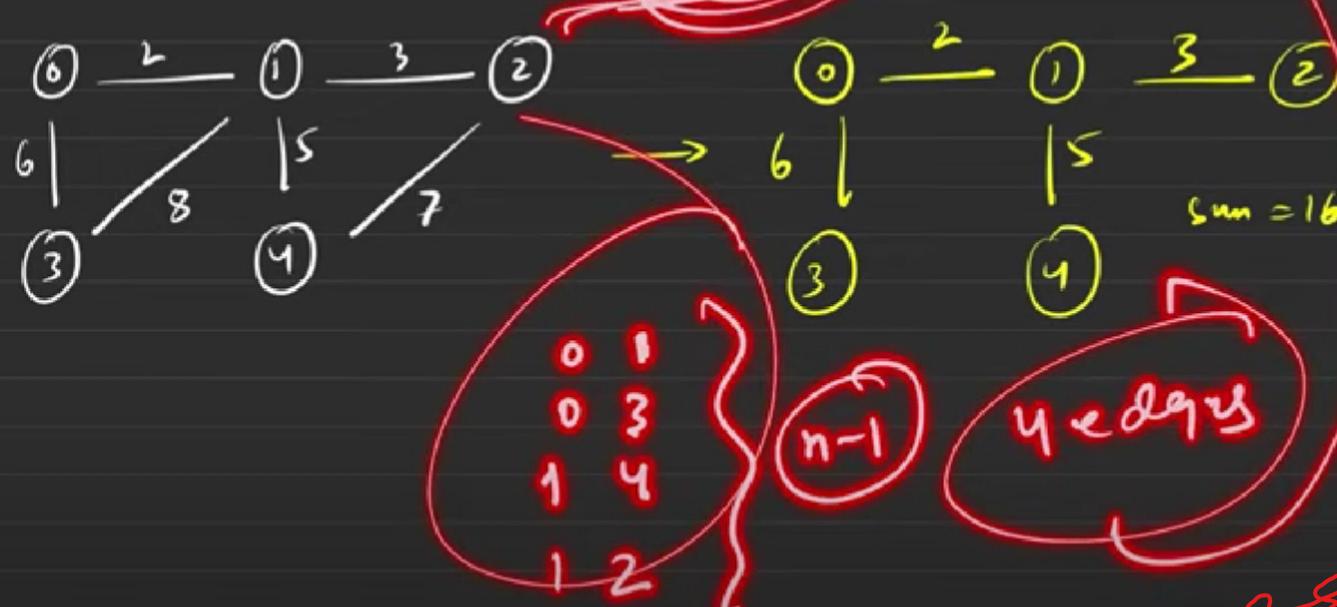
This screenshot captures a Microsoft Teams meeting in progress. The main focus is a presentation slide titled "Technologies/Tech-Stack Used", which is divided into five categories: Databases, Middleware, CI/Deployment, Frontend, and AI. Each category contains several logos of popular tools. A prominent message box at the top of the slide states, "Your mic has been disabled" and "You can no longer unmute." On the right side of the screen, the Microsoft Teams interface displays video feeds for four participants: Lokesh Saini, Shivani Sin..., MG, and another participant whose feed is partially visible. Below the video feeds, a participant list shows "Participants" with a count of "99+". The bottom of the screen features the Windows taskbar, displaying the date and time (4:35 PM, 7/12/2023), system notifications (4), and various pinned application icons. The overall environment suggests a technical or professional discussion about software development and infrastructure.

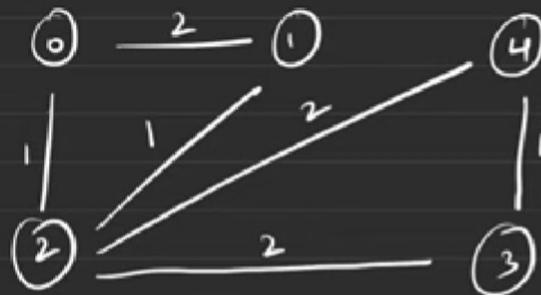
⇒ G-45. Prim's Algorithm - Minimum Spanning Tree - C++ and Java

1.60

Prim's Algorithm →

MST

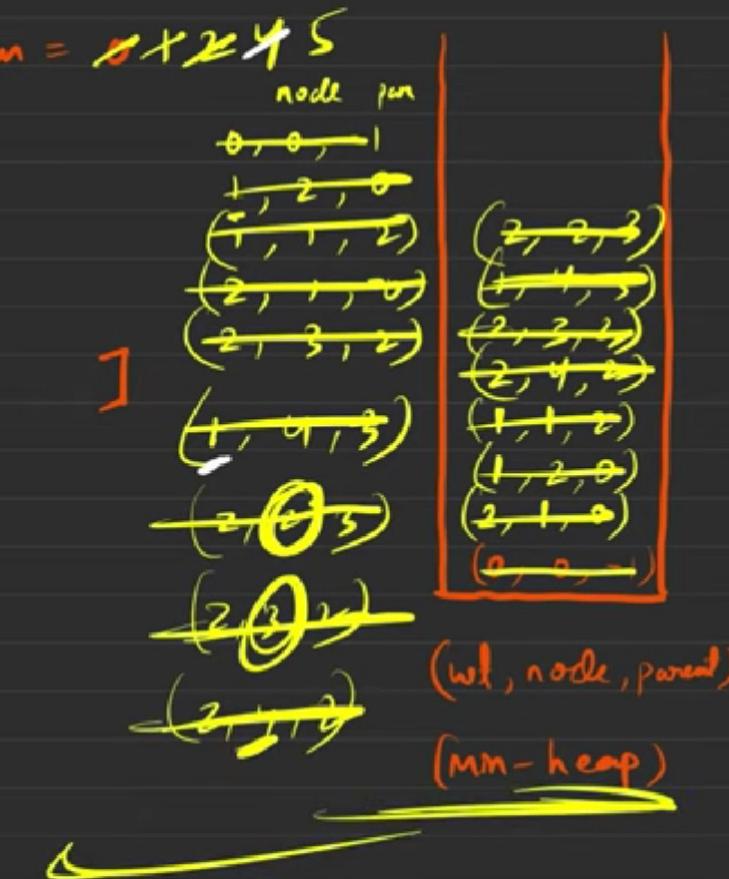




$$\text{sum} = 0 + 2 + 4 + 5$$

node sum

$$\text{MST} \rightarrow \{(0,2)(1,2)(2,3)(3,4)\}$$



$\text{sum} = 0 + 2 + 4 + 5$
 node sum

$\text{MST} \rightarrow \{(0,2), (1,2), (2,3), (3,4)\}$

$\text{vis}[] = [0, 1, 1, 1, 1]$

A red bracket groups the edges from (0,2) to (3,4). To its right is a vertical red line. Below the red line is a list of 12 permutations of the numbers 0, 1, 2, 3, 4, each followed by a red arrow pointing right. Handwritten annotations include circled '0' and '3' under the first two permutations, and circled '2' and '3' under the next two. Below the list is the text '(wt, node, parent)' and '(min-heap)' with yellow arrows pointing to them.

A large yellow circle with the word 'empty' written inside it is at the bottom left.

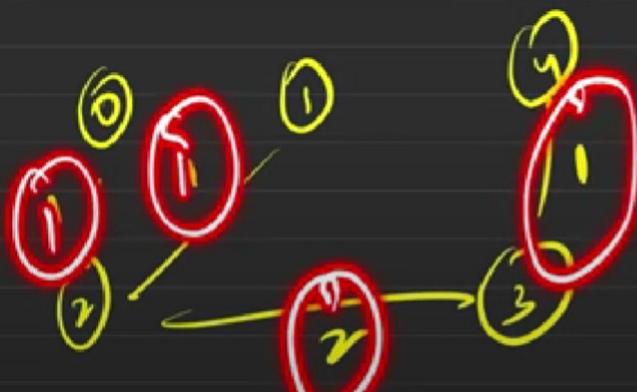
TUF



→ G-45. Prim's Algorithm - Minimum Spanning Tree - C++ and Java



MST $\rightarrow \underline{(0,2)} \underline{(1,2)} \underline{(2,3)} \underline{(3,4)}$



(min-heap)



Problems Courses Get Hired Contests POTD

Practice

C++ (g++ 5.4) • Average Time: 25m

Given a weighted, undirected and connected graph of V vertices and E edges. The task is to find the sum of weights of the edges of the Minimum Spanning Tree.

Example 1:

Input:

Output:

Explanation:

Problems Courses Get Hired Contests </POTD

Practice

C++ (g++ 5.4) • Average Time: 25m

Given a weighted, undirected and connected graph of V vertices and E edges. The task is to find the sum of weights of the edges of the Minimum Spanning Tree.

Example 1:

Input:

Output Window

Compilation Results Custom Input

Problem Solved Successfully ✓

You get marks only for the first correct submission if you solve the problem without viewing the full solution.

Test Cases Passed: 122 / 122 Your Total Score: 463

Total Time Taken: 2.41 Correct Submission Count:

```
1 // Driver Code Ends
2
3 class Solution
4 {
5     public:
6         //Function to Find sum of weights of edges of the Minimum Spanning Tree
7         int spanningTree(int V, vector<vector<int>> adj[])
8         {
9             priority_queue<pair<int,int>, vector<pair<int,int>>, greater<pair<int,int>> pq;
10            vector<int> vis(V, 0);
11            // [wt, node]
12            pq.push({0, 0});
13            int sum = 0;
14            while(!pq.empty())
15            {
16                auto it = pq.top();
17                pq.pop();
18                int node = it.second;
19                int wt = it.first;
20
21                if(vis[node] == 1) continue;
22                // add it to the mst
23                vis[node] = 1;
24                sum += wt;
25                for(auto it: adj[node])
26                {
27                    int adjNode = it[0];
28                    int edW = it[1];
29                    if(!vis[adjNode])
30                    {
31                        pq.push({edW, adjNode});
32                    }
33                }
34            }
35        }
36    }
37    return sum;
38}
39
40 // Driver Code Ends
```

Custom Input

G-45. Prim's Algorithm - Minimum Spanning Tree - C++ and Java

Problems Courses Get Hired Contests POTD Practice

1.90 Problem Editorial Submissions Comments

Given a weighted, undirected and connected graph of V vertices and E edges. The task is to find the sum of weights of the edges of the Minimum Spanning Tree.

Example 1:

Input:

Output Window

Compilation Results Custom Input

Test Cases Processed: 0 / 122

```
C++ (g++ 5.4) // } Driver Code Ends
1 class Solution
2 {
3     public:
4         //Function to find sum of weights of edges of the Minimum Spanning Tree
5         int spanningTree(int V, vector<vector<int>> adj[])
6         {
7             priority_queue<pair<int,int>, vector<pair<int,int>>, greater<pair<int,int>> pq;
8
9             vector<int> vis(V, 0);
10            // {wt, node}
11            pq.push({0, 0});
12            int sum = 0;
13            while(!pq.empty())
14            {
15                auto it = pq.top();
16                pq.pop();
17                int node = it.second;
18                int wt = it.first;
19
20                if(vis[node] == 1) continue;
21                // add it to the mst
22                vis[node] = 1;
23                sum += wt;
24                for(auto it: adj[node])
25                {
26                    int adjNode = it[0];
27                    int edW = it[1];
28                    if(!vis[adjNode])
29                    {
30                        pq.push({edW, adjNode});
31                    }
32                }
33            }
34            return sum;
35        }
36    }
37 // } Driver Code Ends
```

Custom Input

14:56 / 19:09 • C Implementation >

Settings

G-45. Prim's Algorithm - Minimum Spanning Tree - C++ and Java

Problems Courses Get Hired Contests POTD Practice

1.90 / Problem Editorial Submissions Comments

Average Time: 25m

Example 1:

Input:

Output Window

```
int spanningTree(int V, vector<vector<int>> adj[])
{
    priority_queue<pair<int,int>, vector<pair<int,int>>, greater<pair<int,int>> pq;
    vector<int> vis(V, 0);
    // {wt, node}
    pq.push({0, 0});
    int sum = 0;
    // E log E + E log E
    while(!pq.empty()) {
        // log E
        auto it = pq.top();
        pq.pop();
        int node = it.second;
        int wt = it.first;

        if(vis[node] == 1) continue;
        // add it to the mst
        vis[node] = 1;
        sum += wt;
        // E log E
        for(auto it: adj[node]) {
            int adjNode = it[0];
            int edW = it[1];
            if(!vis[adjNode]) {
                pq.push({edW, adjNode});
            }
        }
    }
    return sum;
}
```

Compilation Results Custom Input

Problem Solved Successfully ✓

You get marks only for the first correct submission if you solve the problem without viewing the full solution.

Test Cases Passed: 122 / 122 Your Total Score: 463

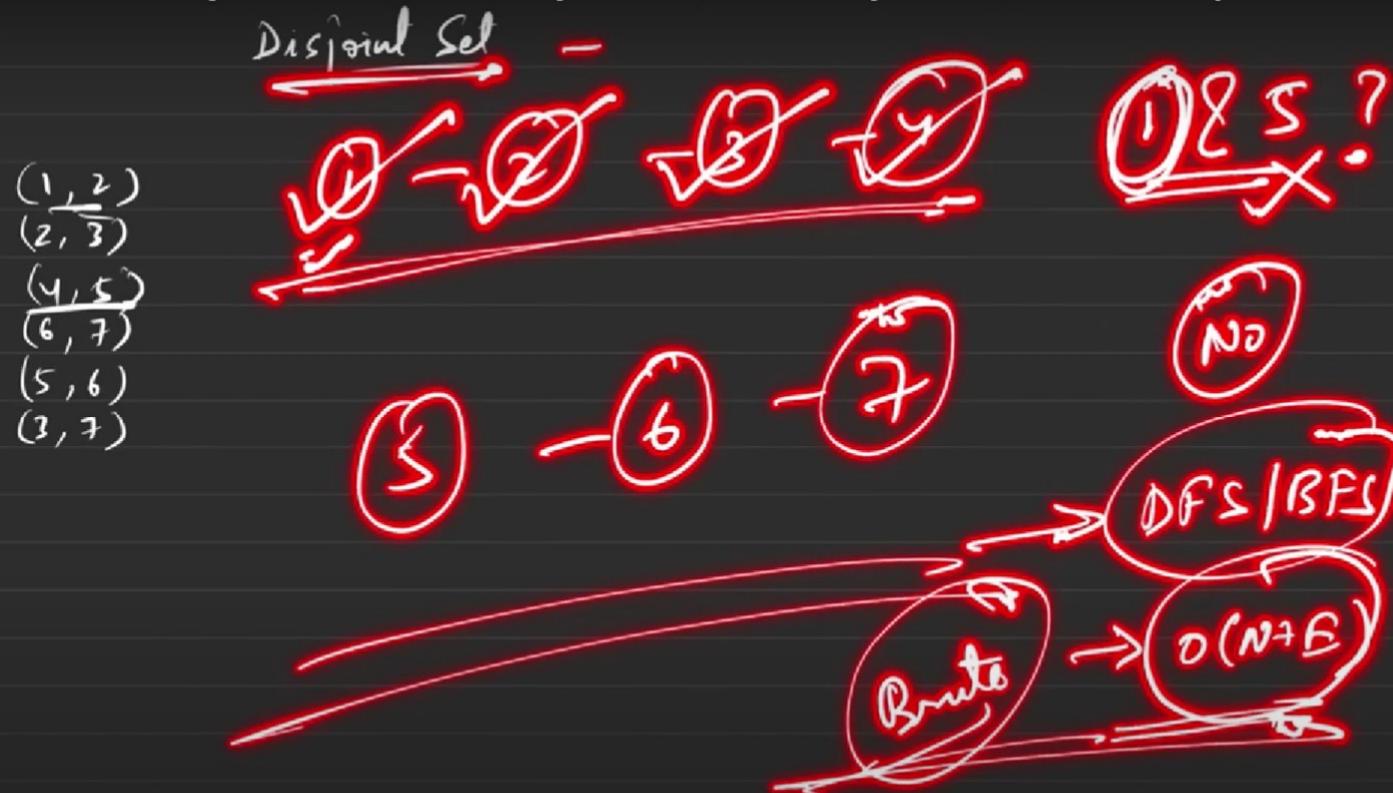
Custom Input Compile

◀ ▶ ⏪ ⏩ 18:19 / 19:09 • Intuition >

CC ⚙ #

→ G-46. Disjoint Set | Union by Rank | Union by Size | Path Compression

1.45



→ G-46. Disjoint Set | Union by Rank | Union by Size | Path Compression

1.45

Disjoint Set

constant

(1, 2)
(2, 3)
(4, 5)
(6, 7)
(5, 6)
(3, 7)

yes | no

1 2 5 ?



⇒ G-46. Disjoint Set | Union by Rank | Union by Size | Path Compression

1.45

Disjoint Set ↳ Dynamic array

(1, 2)
(2, 3)

(4, 5)
(6, 7)

(5, 6)
(3, 7)



2:05 / 42:14



→ G-46. Disjoint Set | Union by Rank | Union by Size | Path Compression

1.60

Disjoint Set

→ constant

Find_Par()

Union() → rank
Union() → size



② - ③

④ - ⑤

⑥ - ⑦

① ⑧ ↗ → ?

No



4:30 / 42:14

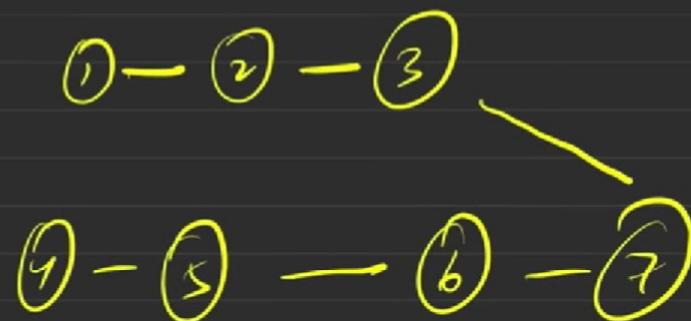


→ G-46. Disjoint Set | Union by Rank | Union by Size | Path Compression

1.60

Disjoint Set

$(1, 2)$ ✓
 $(2, 3)$ ✓
 $(4, 5)$ ✓
 $(6, 7)$ ✓
 $(5, 6)$ ✓
 $(3, 7)$ ✓



Find_Parent()
Union() → rank
Union() → size

1 2 3 → 4 5 6 7



⇒ G-46. Disjoint Set | Union by Rank | Union by Size | Path Compression

1.60

Disjoint Set

(1, 2)
(2, 3)

(4, 5)
(6, 7)

(5, 6)
(3, 7)

rank

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

parent

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

findPar()

Union() → rank
→ size

①² ②² ③² ④² ⑤² ⑥² ⑦²



TUF



6:55 / 42:14



⇒ G-46. Disjoint Set | Union by Rank | Union by Size | Path Compression

1.60

(1) (2) (3) (4) (5) (6) (7)

Union (u, v)

1. find ultimate parent of $u \& v$, p_u, p_v
2. find rank of p_u, p_v .
3. connect smaller rank to larger rank always



7:55 / 42:14



⇒ G-46. Disjoint Set | Union by Rank | Union by Size | Path Compression

1.60

$(1, 2)$ ✓
 $(2, 3)$

rank

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

$(4, 5)$
 $(6, 7)$
 $(5, 6)$
 $(3, 7)$

parent

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

find Parent()

Union(u, v) → rank

union(u, v)

$\textcircled{1}^2 \quad \textcircled{2}^2 \quad \textcircled{3}^2 \quad \textcircled{4}^2 \quad \textcircled{5}^2 \quad \textcircled{6}^2 \quad \textcircled{7}^2$

$$pu = 1 \quad n=0$$

$$pv = 2 \quad n=0$$

Union(u, v)

1. find ultimate parent of $u \& v$, pu, pv
2. find rank of pu, pv .
3. connect smaller to larger.



8:48 / 42:14



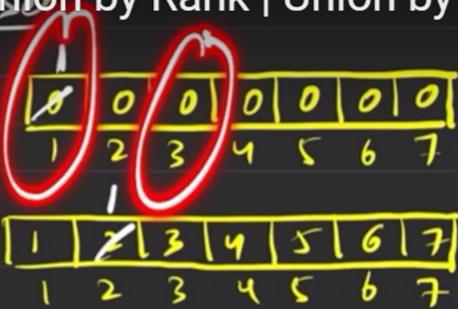
→ G-46. Disjoint Set | Union by Rank | Union by Size | Path Compression

1.60

$(1, 2)$ ✓
 $(2, 3)$ ✓
 $(4, 5)$
 $(6, 7)$
 $(5, 6)$
 $(3, 7)$

rank

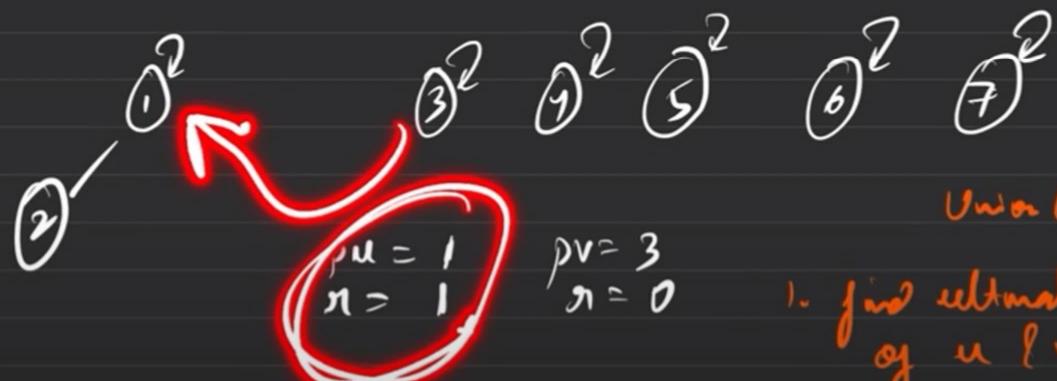
parent



findParent()

Union(u) → rank
Union(u) → size

union(u, v)



Union(u, v)

1. find ultimate parent of u & v, pu, pv.
2. find rank of pu, pv.
3. connect smaller to larger.



⇒ G-46. Disjoint Set | Union by Rank | Union by Size | Path Compression

1.60

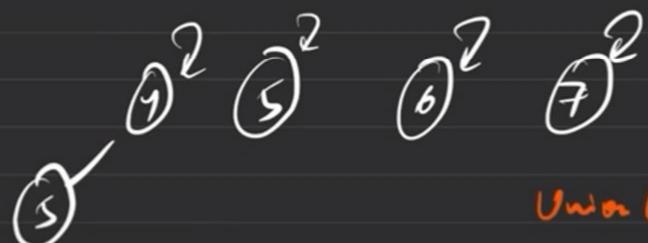
$(1, 2)$ ✓
 $(2, 3)$ ✓
 $(4, 5)$ ✓
 $(6, 7)$
 $(5, 6)$
 $(3, 7)$

rank

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

parent

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



$$pu = 4 \quad pv = 5 \\ r = 0 \quad h = 0$$

find Par()

Union() → rank

union(u, v)



Union(u, v)

1. find ultimate parent
of u & v, pu, pv

2. find rank of pu, pv.

3. connect smaller

to larger

◀ ▶ □ 🔍 10:37 / 42:14

▼

CC ⚙ #



⇒ G-46. Disjoint Set | Union by Rank | Union by Size | Path Compression

1.60

$(1, 2)$ ✓
 $(2, 3)$ ✓
 $(4, 5)$ ✓
 $(6, 7)$
 $(5, 6)$
 $(3, 7)$

rank

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

parent

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



$$p_u = 4 \quad p_v = 5 \\ r_u = 0 \quad r_v = 0$$

1. find ultimate parent of u & v , p_u, p_v
2. find rank of p_u, p_v .
3. connect smaller rank to larger.

Union (u, v)



TUP

◀ ▶ 🔍 10:40 / 42:14

▼

CC ⚙ #

disjoint sets

rank
 $(1, 2)$ ✓
 $(2, 3)$ ✓
 $(4, 5)$ ✓
 $(6, 7)$
 $(5, 6)$
 $(3, 7)$

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

parent

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



find Par()

Union() → rank

union(u, v)



Union(u, v)

- $m = 0$
- 1. find ultimate parent of u & v; pu, pv.
 - 2. find ranks of pu, pv.
 - 3. connect smaller to larger.



$(1, 2)$ ✓
 $(2, 3)$ ✓
 $(4, 5)$ ✓
 $(6, 7)$ ✓
 $(5, 6)$
 $(3, 7)$

disjoint sets

rank

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

parent

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



$$\rho u = 6 \quad \rho v = 7 \\ \sigma = 0 \quad \sigma = 0$$



$\text{Union}(u, v)$

1. find ultimate parent of $u \& v$, $\rho u, \rho v$
2. find rank of $\rho u, \rho v$.
3. connect smaller to larger.

find Par()

$\text{Union}(u) \rightarrow \text{rank}$

$\text{union}(u, v)$



disjoint sets

rank
 $(1, 2)$ ✓
 $(2, 3)$ ✓
 $(4, 5)$ ✓
 $(6, 7)$ ✓
 $(5, 6)$
 $(3, 7)$

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

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

find Par()

Union() → rank

union(u, v)



$$pu = 6 \quad pv = 7$$

$$m=0 \quad n=0$$

Union(u, v)

1. find ultimate parent of $u \& v$, pu, pv

2. find rank of pu, pv .

3. connect smaller to larger.



disjoint sets

rank
 $(1, 2)$ ✓
 $(2, 3)$ ✓
 $(4, 5)$ ✓
 $(6, 7)$ ✓
 $(5, 6)$
 $(3, 7)$

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

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

find Par()

Union() → rank

union(u, v)



Union(u, v)

1. find ultimate parent of u & v, pu, pv.
2. find ranks of pu, pv.
3. connect smaller to larger.



→ G-46. Disjoint Set | Union by Rank | Union by Size | Path Compression

1.60

$(1, 2)$ ✓
 $(2, 3)$ ✓
 $(4, 5)$ ✓
 $(6, 7)$ ✓
 $(5, 6)$ ✓
 $(3, 7)$

rank

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

parent

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

find Par()

Union() → rank

union(u, v)



Union(u, v)

find ultimate parent
of u & v, pu, pv

2. find rank of pu, pv.

3. connect smaller



⇒ G-46. Disjoint Set | Union by Rank | Union by Size | Path Compression

1.60

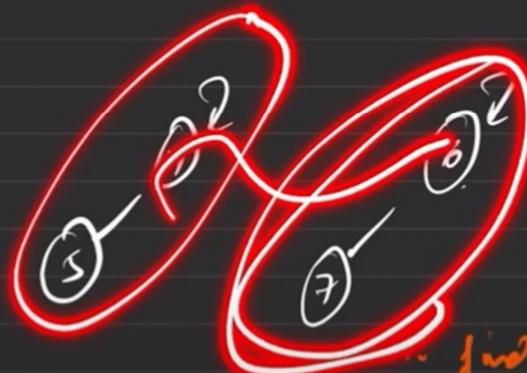
$(1, 2)$ ✓
 $(2, 3)$ ✓
 $(4, 5)$ ✓
 $(6, 7)$ ✓
 $(5, 6)$ ✓
 $(3, 7)$

rank

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

parent

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



$$\rho u = 4 \quad \rho v = 6$$

$n = 1$

$$\rho u = 1$$

find ultimate parent
of $u \& v$; $\rho u, \rho v$

2. find rank of $\rho u, \rho v$.

3. connect smaller

to larger

find Par()

Union() → rank

union(u, v)



⇒ G-46. Disjoint Set | Union by Rank | Union by Size | Path Compression

1.60

$(1, 2)$ ✓
 $(2, 3)$ ✓
 $(4, 5)$ ✓
 $(6, 7)$ ✓
 $(5, 6)$ ✓
 $(3, 7)$

rank

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

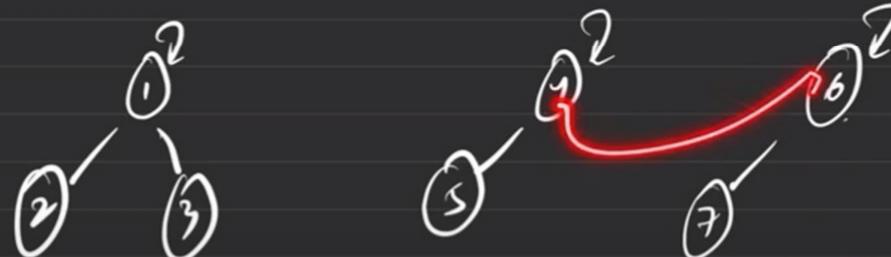
parent

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

find Par()

Union() → rank

union(u, v)



Union(u, v)

1. find ultimate parent
of u & v, pu, pv

2. find rank of pu, pv.

3. connect smaller



12:06 / 42:14



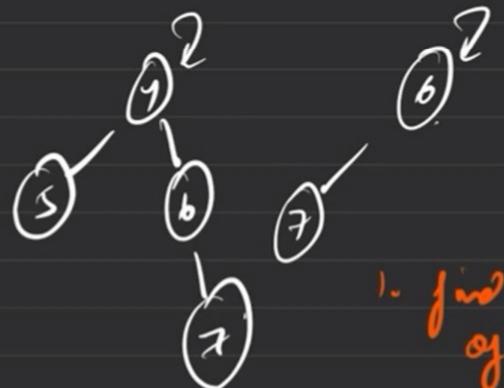
disjoint sets

rank
 $(1, 2)$ ✓
 $(2, 3)$ ✓
 $(4, 5)$ ✓
 $(6, 7)$ ✓
 $(5, 6)$ ✓
 $(3, 7)$

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

parent

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



find Par()

Union() → rank

union(u, v)

1. find ultimate parent of u & v, pu, pv.
2. find ranks of pu, pv.
3. connect smaller to larger.



TUF

disjoint sets

rank
 $(1, 2)$ ✓
 $(2, 3)$ ✓
 $(4, 5)$ ✓
 $(6, 7)$ ✓
 $(5, 6)$ ✓
 $(3, 7)$

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

parent

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



find Par()

Union() → rank

union(u, v)

Union(u, v)

1. find ultimate parent of u & v, pu, pv.
2. find ranks of pu, pv.
3. connect smaller to larger.



Disjoint set

$(1, 2)$ ✓
 $(2, 3)$ ✓

$(4, 5)$ ✓
 $(6, 7)$ ✓
 $(5, 1)$ ✓
 $(3, 7)$

rank

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

	1	2	3	4	5	6	7
parent	1	1	3	4	4	6	7



find Par()

Union() → rank

union(u, v)

Union(u, v)

1. find ultimate parent of $u \& v$, p_u, p_v
2. find rank of p_u, p_v .
3. connect smaller to larger.



(3, 7)



Union(u, v)

1. find ultimate parent of u & v, pu, pv.
2. find rank of pu, pr.
3. connect smaller rank to larger rank always

does 1 8 7 belong to same comp?

Ans → No



⇒ G-46. Disjoint Set | Union by Rank | Union by Size | Path Compression

1.60

- $(1, 2)$ ✓ rank
- $(2, 3)$ ✓ parent
- $(4, 5)$ ✓
- $(6, 7)$ ✓
- $(5, 6)$ ✓
- $(3, 7)$

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

Index

Union(l, r) \rightarrow rank

union(u, v)



does 1 8 7 belong to same compo?

Ans → No

1. find ultimate parent of u l v, pu, pv.
2. find rank of pu, pv.
3. connect smaller ranks to larger ranks.



⇒ G-46. Disjoint Set | Union by Rank | Union by Size | Path Compression

1.60

rank ✓

$(1, 2)$ ✓

$(2, 3)$ ✓

$(4, 5)$ ✓ parent

$(6, 7)$ ✓

$(5, 6)$ ✓

$(3, 7)$

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

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

Index ↗ rank

Union (l, r) ↗ rank

union (u, v)



does 1? 7 belong to same compo?
Ans → No

Union (u, v)

1. find ultimate parent of $u \& v$, pu, pv

2. find rank of pu, pv .

3. connect smaller rank to larger rank.



14:37 / 42:14



→ G-46. Disjoint Set | Union by Rank | Union by Size | Path Compression

1.60

rank

$(1, 2)$ ✓

$(2, 3)$ ✓

$(4, 5)$ ✓ parent

$(6, 7)$ ✓

$(5, 6)$ ✓

$(3, 7)$

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

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



does 1 8 7 belong to same compo?

Ans → No

Union () → rank

union (u, v)

log N

Union (u, v)

1. find ultimate parent of u & v, pu, pv

2. find rank of pu, pv.

3. connect smaller rank to larger rank.



⇒ G-46. Disjoint Set | Union by Rank | Union by Size | Path Compression

1.60

rank ✓

$(1, 2)$ ✓

$(2, 3)$ ✓

$(4, 5)$ ✓ parent

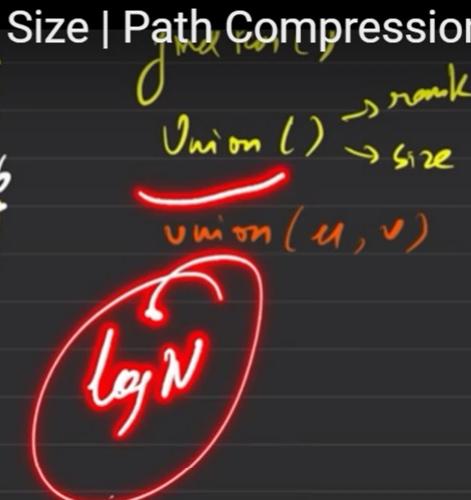
$(6, 7)$ ✓

$(5, 6)$ ✓

$(3, 7)$

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

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



Union (u, v)

1. find ultimate parent of u & v, pu, pv

2. find rank of pu, pv.

3. connect smaller rank to larger rank.

does 1 8 7 belong to same compo?

Ans → No

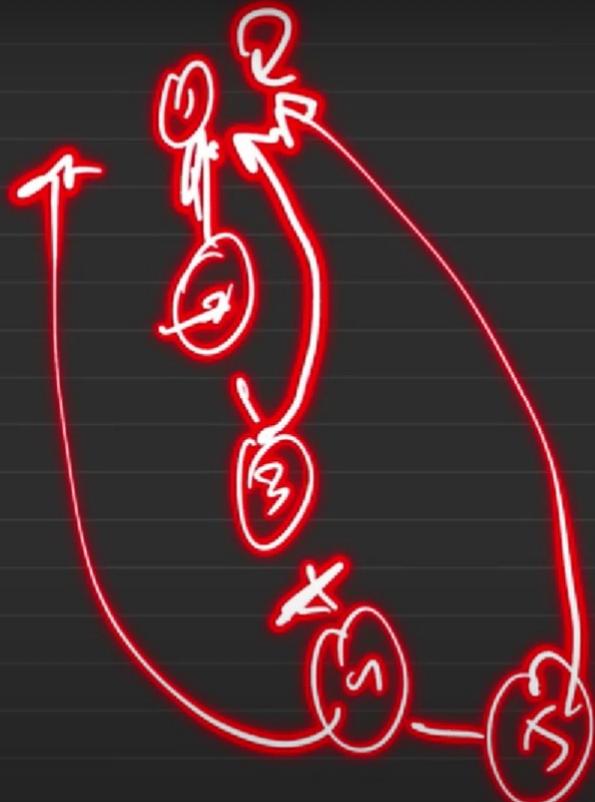


15:10 / 42:14



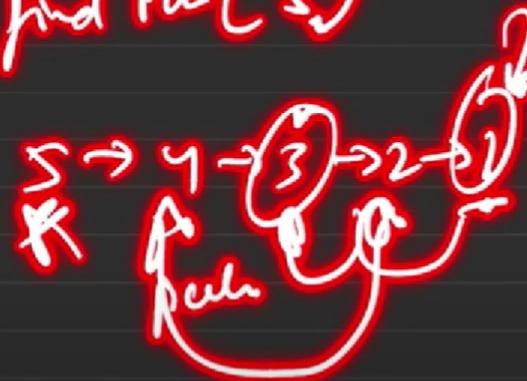
→ G-46. Disjoint Set | Union by Rank | Union by Size | Path Compression

1.60



$7 \rightarrow 6 \rightarrow 1$

Find $\text{P}(5)$



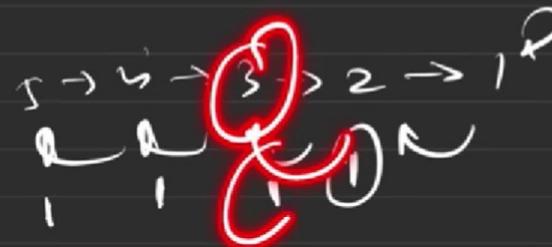
► G-46. Disjoint Set | Union by Rank | Union by Size | Path Compression

Press Esc to exit full screen

1.60



$7 \rightarrow 6 \rightarrow 1$



18:20 / 42:14



→ G-46. Disjoint Set | Union by Rank | Union by Size | Path Compression

1.60



(2)

7 → 6 → 1

5 → 4 → 3 → 2 → 1

1 2 3 4 5



18:39 / 42:14



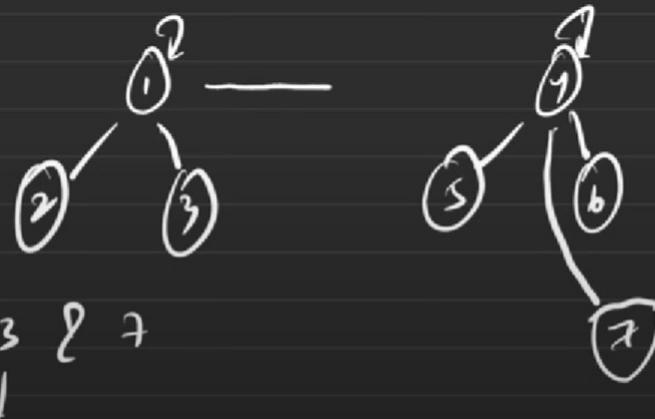
→ G-46. Disjoint Set | Union by Rank | Union by Size | Path Compression

1.60

Disjoint Set							
	1	2	3	4	5	6	7
rank	0	0	0	0	0	0	0
(1, 2)	✓						
(2, 3)	✓						
(4, 5)	✓						
(6, 7)	✓						
(5, 6)	✓						
(3, 7)							

parent

	1	2	3	4	5	6	7
parent	1	2	3	4	5	6	7



find Par()

Union() → rank

Union(u, v) → size

Union(u, v)

1. find ultimate parent of u & v, pu, pv
2. find rank of pu, pv.



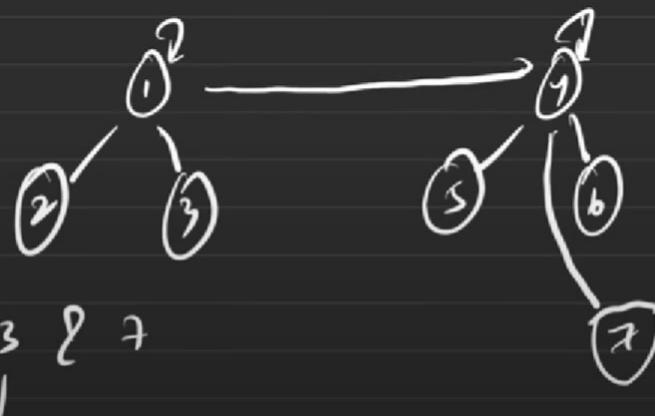
pu=1 pv=4

TUF

⇒ G-46. Disjoint Set | Union by Rank | Union by Size | Path Compression

1.60

Disjoint Set							
rank	1	2	3	4	5	6	7
(1, 2)	✓	0	0	0	0	0	0
(2, 3)	✓	1	1	4	4	4	4
(4, 5)	✓	1	2	3	4	2	1
(6, 7)	✓	1	2	3	4	5	6
(5, 6)	✓	1	2	3	4	5	7
(3, 7)		1	2	3	4	5	7



findPar()

Union() → rank

Union(u, v) → size

Union(u, v)



$(1, 2)$ ✓

$(2, 3)$ ✓

$(4, 5)$ ✓

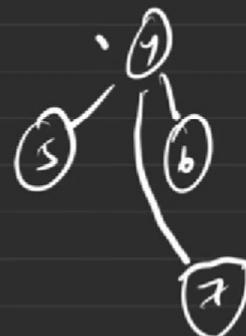
$(6, 7)$ ✓

$(5, 6)$ ✓

$(3, 7)$

parent

1	2	3	4	5	6	7	9
4	1	1	4	4	6	9	



Union() \rightarrow size

union(u, v)

1. find ultimate parent of $u \& v$, pu, pv
2. find rank of pu, pv .
3. connect smaller rank to larger rank always



$(1, 2)$ ✓

$(2, 3)$ ✓

$(4, 5)$ ✓

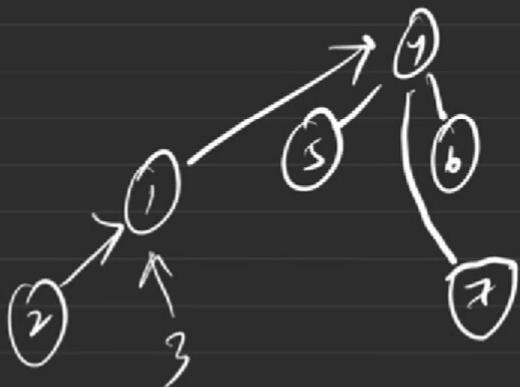
$(6, 7)$ ✓

$(5, 6)$ ✓

$(3, 7)$

parent

1	2	3	4	5	6	7	8
4	1	1	4	4	6	9	9



Union() \rightarrow size

union(u, v)

Union(u, v)

1. find ultimate parent of $u \& v$, p_u, p_v
2. find rank of p_u, p_v .
3. connect smaller rank to larger rank always



Disjoint Set

rank

$(1, 2)$ ✓

$(2, 3)$ ✓

$(4, 5)$ ✓

$(6, 7)$ ✓

$(5, 6)$ ✓

$(3, 7)$

parent

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

findPar()

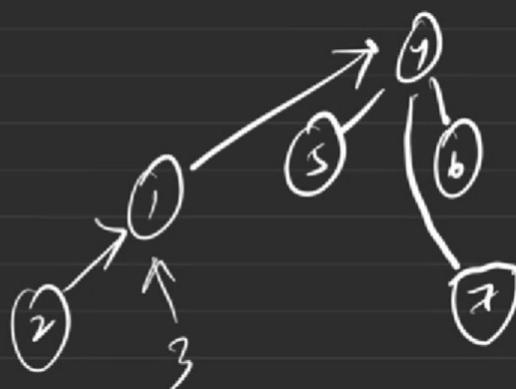
Union() \rightarrow rank

union(u, v)

Union(u, v)

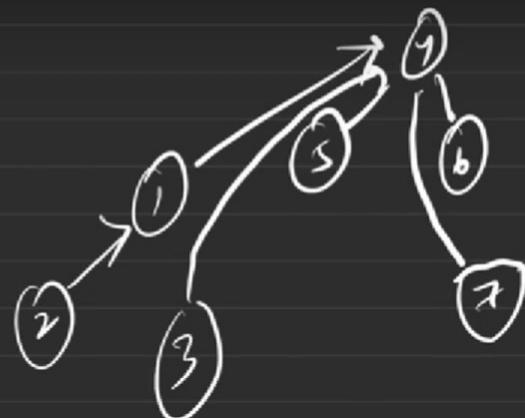
1. find ultimate parent
of u & v, pu, pv

2. find rank of pu, pv.



⇒ G-46, Disjoint Set | Union by Rank | Union by Size | Path Compression

1.60

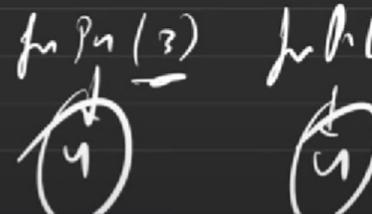


Union(u, v)

1. find ultimate parent of $u \& v$, p_u, p_v
2. find rank of p_u, p_v .

connect smaller
rank to larger
rank always

3 9 6 → ? ✓



Path compression



22:51 / 42:14



parent 4

6

→ connect smaller Urank
to larger rank

(Path compression)

Disjoint Set \rightarrow

rank

(1, 2) ✓

(2, 3) ✓

(4, 5) ✓

(6, 7) ✓

(5, 6) ✓

(3, 7) ✓

1	0	0	0	0	0	0	0
4	1	1	1	4	4	4	7
1	2	3	4	5	6	7	9

parent

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

find Par() $\rightarrow \Theta(n)$

Union() \rightarrow rank

union(u, v)



$\Theta(n^2) \approx \Theta(n \alpha)$

Union(u, v)

1. find ultimate parent
 $u = u \cup v \cdot \text{par}(u, v)$





G-46. Disjoint Set | Union by Rank | Union by Size | Path Compression

*connected smaller tree
to larger rank*

1.60

(Path compression)

Disjoint Set $\rightarrow \underline{\underline{o(4n)}}$

rank

1	0	0	0	0	0	0	0
4	1	1	1	4	4	4	4

$1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8$

$(1, 2) \checkmark$
 $(2, 3) \checkmark$
 $(4, 5) \checkmark$
 $(6, 7) \checkmark$
 $(5, 6) \checkmark$
 $(3, 7) \checkmark$

parent

1	2	3	4	5	6	7	8
1	1	1	4	4	4	4	4

find Parent $\rightarrow \underline{\underline{o(4n)}}$
Union $(u, v) \rightarrow$ rank
union (u, v)
 \Downarrow
 $\underline{\underline{o(4n)}} \approx \underline{\underline{o(n \log n)}}$

Union (u, v)

1. find ultimate parent
 $a = u \text{ or } b = u \text{ or } u = v$



23:31 / 42:14



→ G-46. Disjoint Set | Union by Rank | Union by Size | Path Compression

1.60

(1)

↑

(2)

↑

(3)

↑

(4)

↑ (3)

fun $\text{Par}(u)$

{
 if ($u == \text{par}[u]$)
 return u ;

return

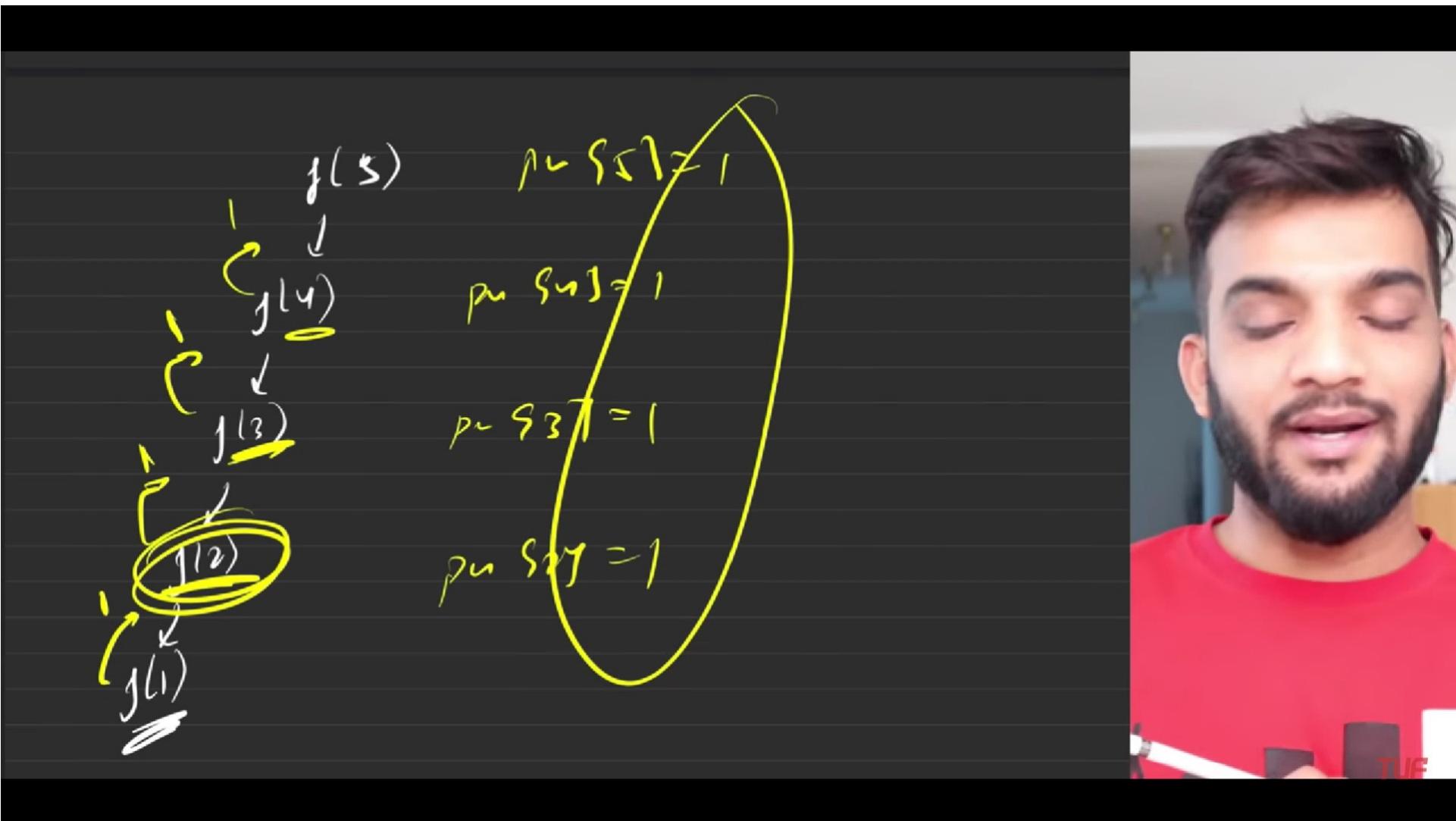
$\text{findPar}(\text{par}[u])$;

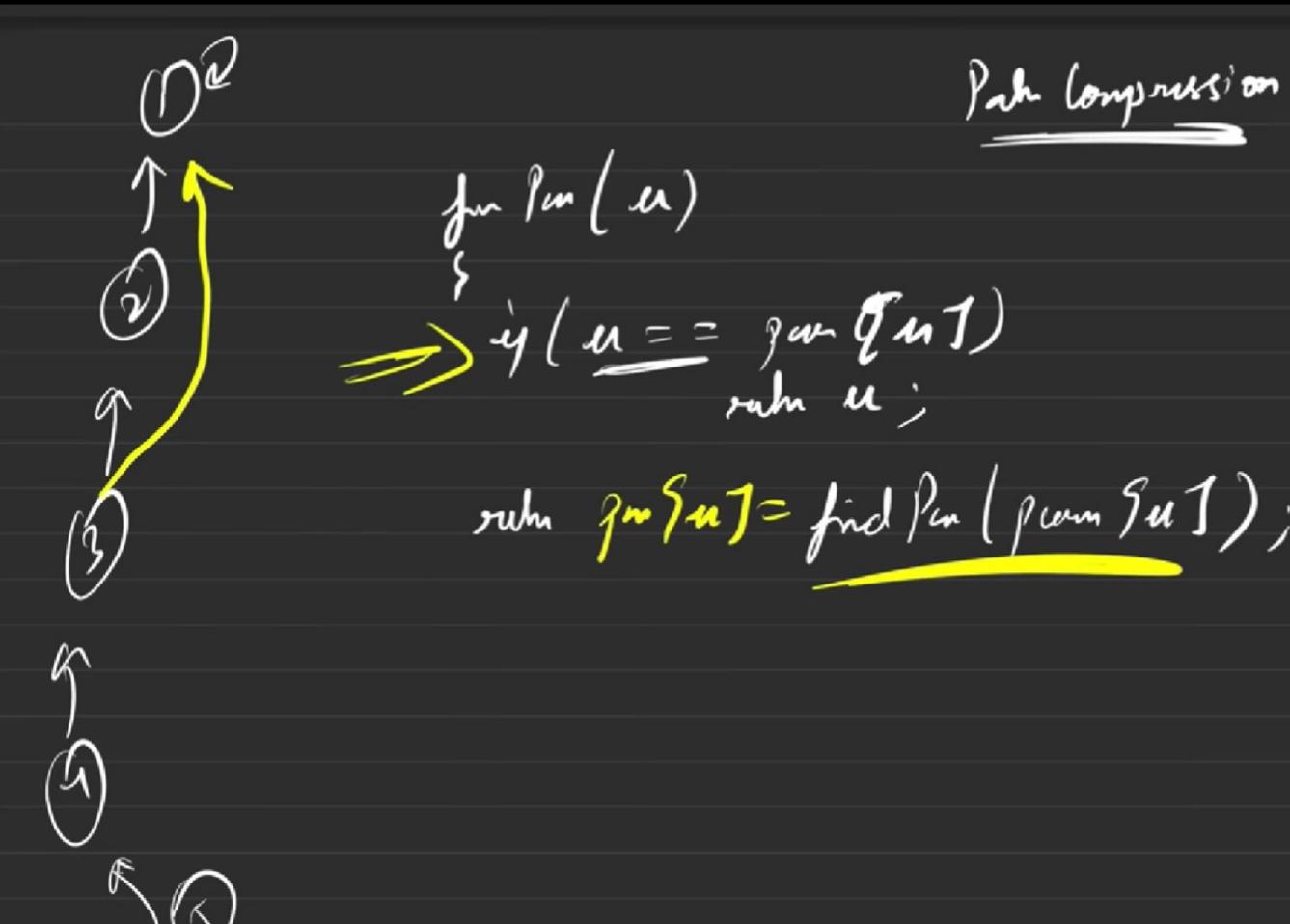
Path compression



24:57 / 42:14

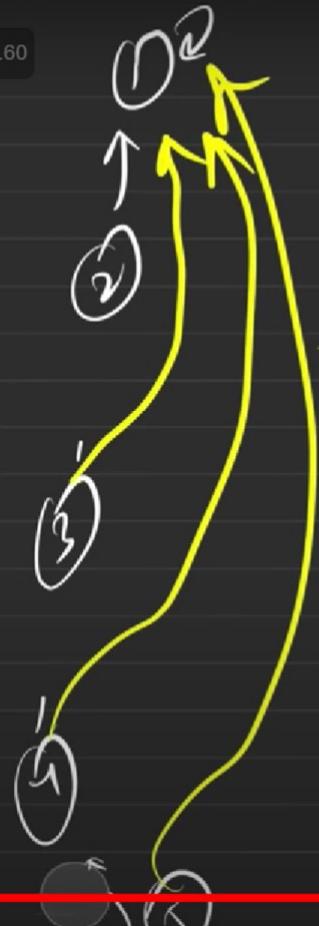






→ G-46. Disjoint Set | Union by Rank | Union by Size | Path Compression

1.60



fun $\text{Pm}(u)$

Path compression

$\Rightarrow \text{y}(u = \text{par}[u])$
when $\text{par}[u] = \text{findPm}(\text{par}[u]);$

when $\text{par}[u] = \text{findPm}(\text{par}[\text{par}[u]]);$





```

1 #include <bits/stdc++.h>
2 using namespace std;
3 class DisjointSet {
4     vector<int> rank, parent;
5 public:
6     DisjointSet(int n) {
7         rank.resize(n+1, 0);
8         parent.resize(n+1);
9         for(int i = 0;i<=n;i++) {
10             parent[i] = i;
11         }
12     }
13     int findUPar(int node) {
14         if(node == parent[node])
15             return node;
16         return parent[node] = findUPar(parent[node]);
17     }
18     void unionByRank(int u, int v)
19 };
20 int main() {
21 #ifndef ONLINE_JUDGE
22     freopen("input.txt", "r", stdin);
23     freopen("output.txt", "w", stdout);
24 #endif
25
26     return 0;
27 }
28
29
30
31 import java.io.*;
32 import java.util.*;
33 class DisjointSet {
34     List<Integer> rank = new ArrayList<>();
35     List<Integer> parent = new ArrayList<>();
36     public DisjointSet(int n) {
37         for(int i = 0;i<n;i++) {
38             rank.add(0);
39             parent.add(i);
40         }
41     }
42     public int findUPar(int node) {
43         if(node == parent.get(node)) {
44             return node;
45         }
46         int ulp = findUPar(parent.get(node));
47         parent.set(node, ulp);
48         return parent.get(node);
49     }
50     public void unionByRank(int u, int v) {
51         int ulp_u = findUPar(u);
52         int ulp_v = findUPar(v);
53         if(ulp_u == ulp_v) return;
54         if(rank.get(ulp_u) < rank.get(ulp_v)) {
55             parent.set(ulp_u, ulp_v);
56         } else if(rank.get(ulp_v) < rank.get(ulp_u)) {
57             parent.set(ulp_v, ulp_u);
58         } else {
59             parent.set(ulp_v, ulp_u);
60             int rankU = rank.get(ulp_u);
61             rank.set(ulp_u, rankU + 1);
62         }
63     }
64 }
65 class Main {
66     public static void main (String[] args) {
67         DisjointSet ds = new DisjointSet(7);
68         ds.unionByRank(1, 2);
69         ds.unionByRank(2, 3);
70         ds.unionByRank(4, 5);
71         ds.unionByRank(6, 7);
72         ds.unionByRank(5, 6);
73
74         // if 3 and 7 same or not
75         if(ds.findUPar(3) == ds.findUPar(7)) {
76             System.out.println("Same");
77         } else {
78             System.out.println("Not Same");
79         }
80         ds.unionByRank(3, 7);
81         if(ds.findUPar(3) == ds.findUPar(7)) {
82             System.out.println("Same");
83         } else {
84             System.out.println("Not Same");
85         }
86     }
87 }

```

TUF

► G-46. Disjoint Set | Union by Rank |Union by Size | Path Compression

1.60

```
5  public:
6      DisjointSet(int n) {
7          rank.resize(n+1, 0);
8          parent.resize(n+1);
9          for(int i = 0;i<=n;i++) {
10              parent[i] = i;
11          }
12      }
13
14      int findUPar(int node) {
15          if(node == parent[node])
16              return node;
17          return parent[node] = findUPar(parent[node]);
18      }
19
20      void unionByRank(int u, int v) {
21          int ulp_u = findUPar(u);
22          int ulp_v = findUPar(v);
23          if(ulp_u == ulp_v) return;
24          if(rank[ulp_u] < rank[ulp_v]) {
25              parent[ulp_u] = ulp_v;
26          }
27          else if(rank[ulp_v] < rank[ulp_u]) {
28              parent[ulp_v] = ulp_u;
29          }
30          else{
31              parent[ulp_v] = ulp_u;
32              rank[ulp_u]++;
33          }
34      };
35      int main() {
36 #ifndef ONLINE_JUDGE
37      freopen("input.txt", "r", stdin);
38      freopen("output.txt", "w", stdout);
39 #endif
40      return 0;
41  }
```

```
4  import java.util.*;
5  class DisjointSet {
6      List<Integer> rank = new ArrayList<>();
7      List<Integer> parent = new ArrayList<>();
8      public DisjointSet(int n) {
9          for(int i = 0;i<n;i++) {
10              rank.add(0);
11              parent.add(i);
12          }
13      }
14
15      public int findUPar(int node) {
16          if(node == parent.get(node)) {
17              return node;
18          }
19          int ulp_u = findUPar(parent.get(node));
20          parent.set(node, ulp_u);
21          return parent.get(node);
22      }
23
24      public void unionByRank(int u, int v) {
25          int ulp_u = findUPar(u);
26          int ulp_v = findUPar(v);
27          if(ulp_u == ulp_v) return;
28          if(rank.get(ulp_u) < rank.get(ulp_v)) {
29              parent.set(ulp_u, ulp_v);
30          }
31          else if(rank.get(ulp_v) < rank.get(ulp_u)) {
32              parent.set(ulp_v, ulp_u);
33          }
34          else{
35              parent.set(ulp_v, ulp_u);
36              rank.set(ulp_u, rank.get(ulp_u) + 1);
37          }
38      }
39
40  }
41  class Main {
42      public static void main (String[] args) {
43          DisjointSet ds = new DisjointSet(7);
44          ds.unionByRank(1, 2);
45          ds.unionByRank(2, 3);
46          ds.unionByRank(4, 5);
47          ds.unionByRank(6, 7);
48          ds.unionByRank(5, 6);
49
50          // If 3 and 7 same or not
51          if(ds.findUPar(3) == ds.findUPar(7)) {
52              System.out.println("Same");
53          }
54          else
55              System.out.println("Not Same");
56
57          ds.unionByRank(3, 7);
58          if(ds.findUPar(3) == ds.findUPar(7)) {
59              System.out.println("Same");
60          }
61          else
62              System.out.println("Not Same");
63      }
64  }
```

TUF



30:07 / 42:14





A screenshot of a computer monitor showing a terminal window with two code files and their execution results.

The terminal window has tabs for "code.cpp", "raj.html", "rajU.html", "A2ZDSACourseSheet.html", and "scriptFullCode.js".

The left pane shows `code.cpp` content:

```
16     return node;
17     return parent[node] = findUPar(parent[node]);
18 }
19
20 void unionByRank(int u, int v) {
21     int ulp_u = findUPar(u);
22     int ulp_v = findUPar(v);
23     if(ulp_u == ulp_v) return;
24     if(rank[ulp_u] < rank[ulp_v]) {
25         parent[ulp_u] = ulp_v;
26     }
27     else if(rank[ulp_v] < rank[ulp_u]) {
28         parent[ulp_v] = ulp_u;
29     }
30     else {
31         parent[ulp_v] = ulp_u;
32         rank[ulp_u]++;
33     }
34 }
35
36 int main() {
37 #ifndef ONLINE_JUDGE
38     freopen("input.txt", "r", stdin);
39     freopen("output.txt", "w", stdout);
40 #endif
41     DisjointSet ds(7);
42     ds.unionByRank(1, 2);
43     ds.unionByRank(2, 3);
44     ds.unionByRank(4, 5);
45     ds.unionByRank(6, 7);
46     ds.unionByRank(5, 6);
47     // if 3 and 7 same or not
48     if(ds.findUPar(3) == ds.findUPar(7)) {
49         cout << "Same\n";
50     }
51     else cout << "Not same\n";
52
53     ds.unionByRank(3, 7);
54     return 0;
55 }
56
```

The right pane shows `scriptFullCode.js` content:

```
3 import java.io.*;
4 import java.util.*;
5 class DisjointSet {
6     List<Integer> rank = new ArrayList<>();
7     List<Integer> parent = new ArrayList<>();
8     public DisjointSet(int n) {
9         for(int i = 0;i<n;i++) {
10             rank.add(0);
11             parent.add(i);
12         }
13     }
14
15     public int findUPar(int node) {
16         if(node == parent.get(node)) {
17             return node;
18         }
19         int ulp_u = findUPar(parent.get(node));
20         parent.set(node, ulp_u);
21         return parent.get(node);
22     }
23
24     public void unionByRank(int u, int v) {
25         int ulp_u = findUPar(u);
26         int ulp_v = findUPar(v);
27         if(ulp_u == ulp_v) return;
28         if(rank.get(ulp_u) < rank.get(ulp_v)) {
29             parent.set(ulp_u, ulp_v);
30         }
31         else if(rank.get(ulp_v) < rank.get(ulp_u)) {
32             parent.set(ulp_v, ulp_u);
33         }
34         else {
35             parent.set(ulp_v, ulp_u);
36             int rankU = rank.get(ulp_u);
37             rank.set(ulp_u, rankU + 1);
38         }
39     }
40 }
41 class Main {
42     public static void main (String[] args) {
43         DisjointSet ds = new DisjointSet(7);
44         ds.unionByRank(1, 2);
45         ds.unionByRank(2, 3);
46         ds.unionByRank(4, 5);
47         ds.unionByRank(6, 7);
48         ds.unionByRank(5, 6);
49
50         // if 3 and 7 same or not
51         if(ds.findUPar(3) == ds.findUPar(7)) {
52             System.out.println("Same");
53         }
54         else {
55             System.out.println("Not Same");
56         }
57
58         ds.unionByRank(3, 7);
59         if(ds.findUPar(3) == ds.findUPar(7)) {
60             System.out.println("Same");
61         }
62         else {
63             System.out.println("Not Same");
64         }
65     }
66 }
```

The terminal also displays the input file `input.txt` and output file `output.txt`:

`input.txt` content:

```
1 3 3 3
2 1 2 3
3 2 1 2
4 3 1 3
```

`output.txt` content:

```
1
```

► G-46. Disjoint Set | Union by Rank | Union by Size | Path Compression



```
1.60
37 #ifndef ONLINE_JUDGE
38 freopen("input.txt", "r", stdin);
39 freopen("output.txt", "w", stdout);
40 #endif
41 DisjointSet ds(7);
42 ds.unionByRank(1, 2);
43 ds.unionByRank(2, 3);
44 ds.unionByRank(4, 5);
45 ds.unionByRank(6, 7);
46 ds.unionByRank(5, 6);
47 // if 3 and 7 same or not
48 if(ds.findUPar(3) == ds.findUPar(7)) {
49     cout << "Same\n";
50 } else cout << "Not same\n";
51
52 ds.unionByRank(3, 7);
53 return 0;
54 }
55
56
57
58
59
60
61
62
63
64
65
66
67
[Finished in 1.3s]
```

```
1 3 3 3
2 1 2 3
3 2 1 2
4 3 1 3
1 Not same
2
```

```
48 DisjointSet {
49     List<Integer> rank = new ArrayList<>();
50     List<Integer> parent = new ArrayList<>();
51     public DisjointSet(int n) {
52         for(int i = 0;i<n;i++) {
53             rank.add(0);
54             parent.add(i);
55         }
56     }
57
58     public int findUPar(int node) {
59         if(node == parent.get(node)) {
60             return node;
61         }
62         int ulp_u = findUPar(parent.get(node));
63         parent.set(node, ulp_u);
64         return parent.get(node);
65     }
66
67     public void unionByRank(int u, int v) {
68         int ulp_u = findUPar(u);
69         int ulp_v = findUPar(v);
70         if(ulp_u == ulp_v) return;
71         if(rank.get(ulp_u) < rank.get(ulp_v)) {
72             parent.set(ulp_u, ulp_v);
73         } else if(rank.get(ulp_v) < rank.get(ulp_u)) {
74             parent.set(ulp_v, ulp_u);
75         } else {
76             parent.set(ulp_v, ulp_u);
77             int rankU = rank.get(ulp_u);
78             rank.set(ulp_u, rankU + 1);
79         }
80     }
81
82     class Main {
83         public static void main (String[] args) {
84             DisjointSet ds = new DisjointSet(7);
85             ds.unionByRank(1, 2);
86             ds.unionByRank(2, 3);
87             ds.unionByRank(4, 5);
88             ds.unionByRank(6, 7);
89             ds.unionByRank(5, 6);
90
91             // if 3 and 7 same or not
92             if(ds.findUPar(3) == ds.findUPar(7)) {
93                 System.out.println("Same");
94             } else
95                 System.out.println("Not Same");
96
97             ds.unionByRank(3, 7);
98             if(ds.findUPar(3) == ds.findUPar(7)) {
99                 System.out.println("Same");
100            } else
101                System.out.println("Not Same");
102        }
103    }
104 }
```

TUF



A screenshot of a computer screen showing a terminal window and a code editor. The terminal window on the right displays the contents of 'Input.txt' and 'output.txt'. The code editor on the left contains C++ and Java code for a Disjoint Set Union (DSU) implementation.

Input.txt

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

output.txt

1	Not same
2	Same
3	

code.cpp

```
32         rank[ulp_u]++;
33     }
34 }
35 };
36 int main() {
37 #ifndef ONLINE_JUDGE
38     freopen("input.txt", "r", stdin);
39     freopen("output.txt", "w", stdout);
40 #endif
41     DisjointSet ds(7);
42     ds.unionByRank(1, 2);
43     ds.unionByRank(2, 3);
44     ds.unionByRank(4, 5);
45     ds.unionByRank(6, 7);
46     ds.unionByRank(5, 6);
47     // if 3 and 7 same or not
48     if(ds.findUPar(3) == ds.findUPar(7)) {
49         cout << "Same\n";
50     }
51     else cout << "Not same\n";
52
53     ds.unionByRank(3, 7);
54
55     if(ds.findUPar(3) == ds.findUPar(7)) {
56         cout << "Same\n";
57     }
58     else cout << "Not same\n";
59     return 0;
60 }
61
62
[Finished in 0.9s]
```

scriptFullCode.js

```
3 import java.io.*;
4 import java.util.*;
5 class DisjointSet {
6     List<Integer> rank = new ArrayList<>();
7     List<Integer> parent = new ArrayList<>();
8     public DisjointSet(int n) {
9         for(int i = 0;i<n;i++) {
10             rank.add(0);
11             parent.add(i);
12         }
13     }
14
15     public int findUPar(int node) {
16         if(node == parent.get(node)) {
17             return node;
18         }
19         int ulp_u = findUPar(parent.get(node));
20         parent.set(node, ulp_u);
21         return parent.get(node);
22     }
23
24     public void unionByRank(int u, int v) {
25         int ulp_u = findUPar(u);
26         int ulp_v = findUPar(v);
27         if(ulp_u == ulp_v) return;
28         if(rank.get(ulp_u) < rank.get(ulp_v)) {
29             parent.set(ulp_u, ulp_v);
30         }
31         else if(rank.get(ulp_v) < rank.get(ulp_u)) {
32             parent.set(ulp_v, ulp_u);
33         }
34         else {
35             parent.set(ulp_v, ulp_u);
36             int rankU = rank.get(ulp_u);
37             rank.set(ulp_u, rankU + 1);
38         }
39     }
40 }
41 class Main {
42     public static void main (String[] args) {
43         DisjointSet ds = new DisjointSet(7);
44         ds.unionByRank(1, 2);
45         ds.unionByRank(2, 3);
46         ds.unionByRank(4, 5);
47         ds.unionByRank(6, 7);
48         ds.unionByRank(5, 6);
49
50         // if 3 and 7 same or not
51         if(ds.findUPar(3) == ds.findUPar(7)) {
52             System.out.println("Same");
53         }
54         else {
55             System.out.println("Not Same");
56         }
57         ds.unionByRank(3, 7);
58         if(ds.findUPar(3) == ds.findUPar(7)) {
59             System.out.println("Same");
60         }
61         else {
62             System.out.println("Not Same");
63         }
64     }
}
```

TUF

⇒ G-46. Disjoint Set | Union by Rank | Union by Size | Path Compression

1.60

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 class DisjointSet {
4     vector<int> rank, parent;
5 public:
6     DisjointSet(int n) {
7         rank.resize(n+1, 0);
8         parent.resize(n+1);
9         for(int i = 0; i <= n; i++) {
10             parent[i] = i;
11         }
12     }
13     int findUPar(int node) {
14         if(node == parent[node])
15             return node;
16         return parent[node] = findUPar(parent[node]);
17     }
18     void unionByRank(int u, int v) {
19         int ulp_u = findUPar(u);
20         int ulp_v = findUPar(v);
21         if(ulp_u == ulp_v) return;
22         if(rank[ulp_u] < rank[ulp_v]) {
23             parent[ulp_u] = ulp_v;
24         }
25         else if(rank[ulp_v] < rank[ulp_u]) {
26             parent[ulp_v] = ulp_u;
27         }
28         else {
29             parent[ulp_v] = ulp_u;
30         }
31     }
} [Finished in 0.9s]
```



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

output.txt
1 Not same
2 Same
3

TUF

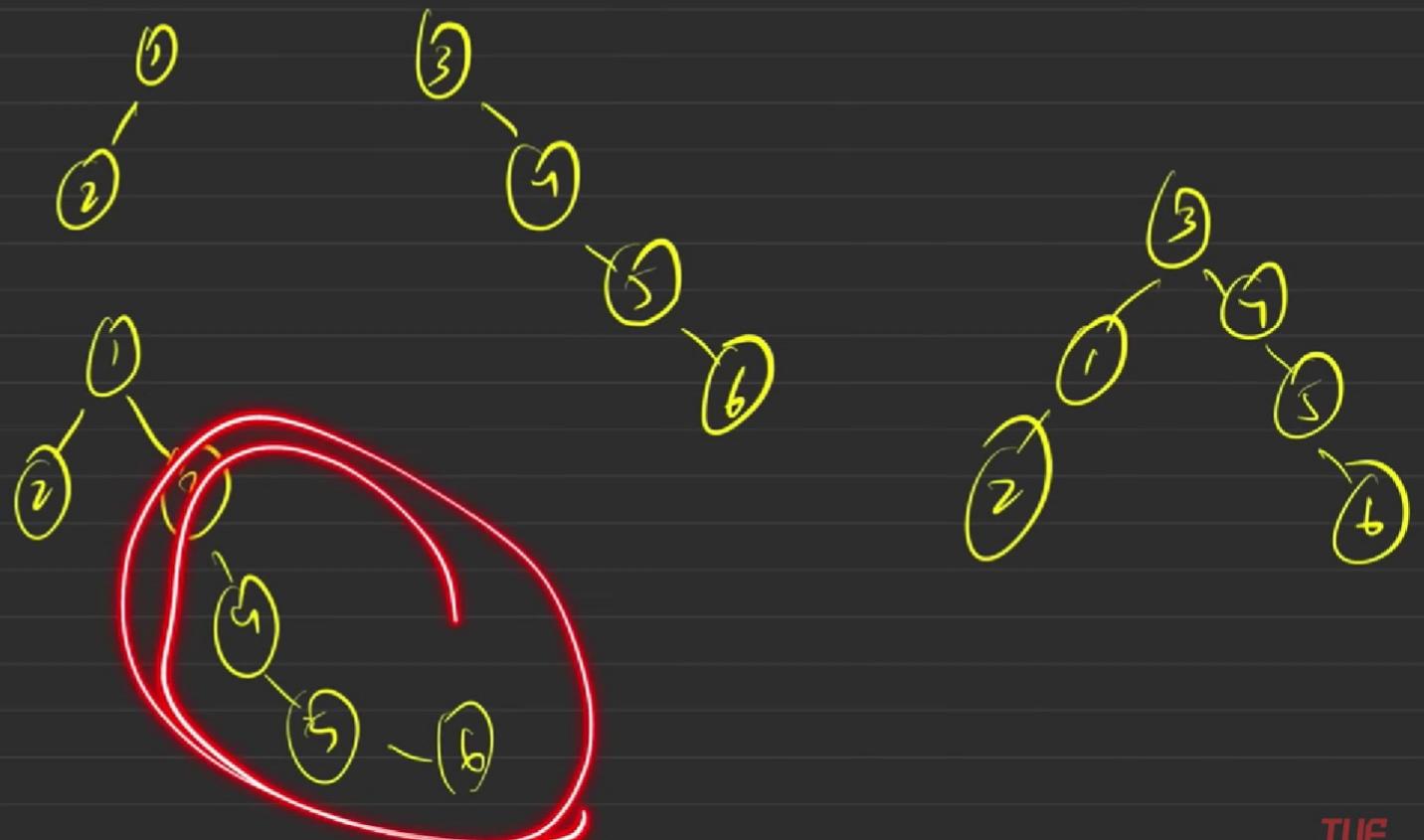


32:25 / 42:14





Why connect smaller to larger?



☰ G-46. Disjoint Set | Union by Rank | Union by Size | Path Compression



Union By Size

TUF



(1, 2)
(2, 3)
(4, 5)
(6, 7)
(5, 6)
(3, 7)

Union By Size

size →

$\boxed{1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1}$
1 2 3 4 5 6 7

parent →

$\boxed{1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7}$
1 2 3 4 5 6 7

0[?] 1[?] 2[?] 3[?] 4[?] 5[?] 6[?] 7[?]



(1, 2) ✓
(2, 3) ✓
(4, 5)
(6, 7)
(5, 6)
(3, 7)

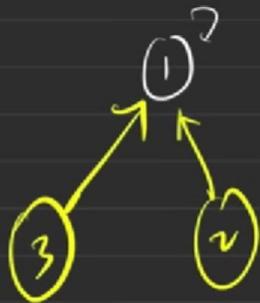
Union By Size

size →

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

parent →

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

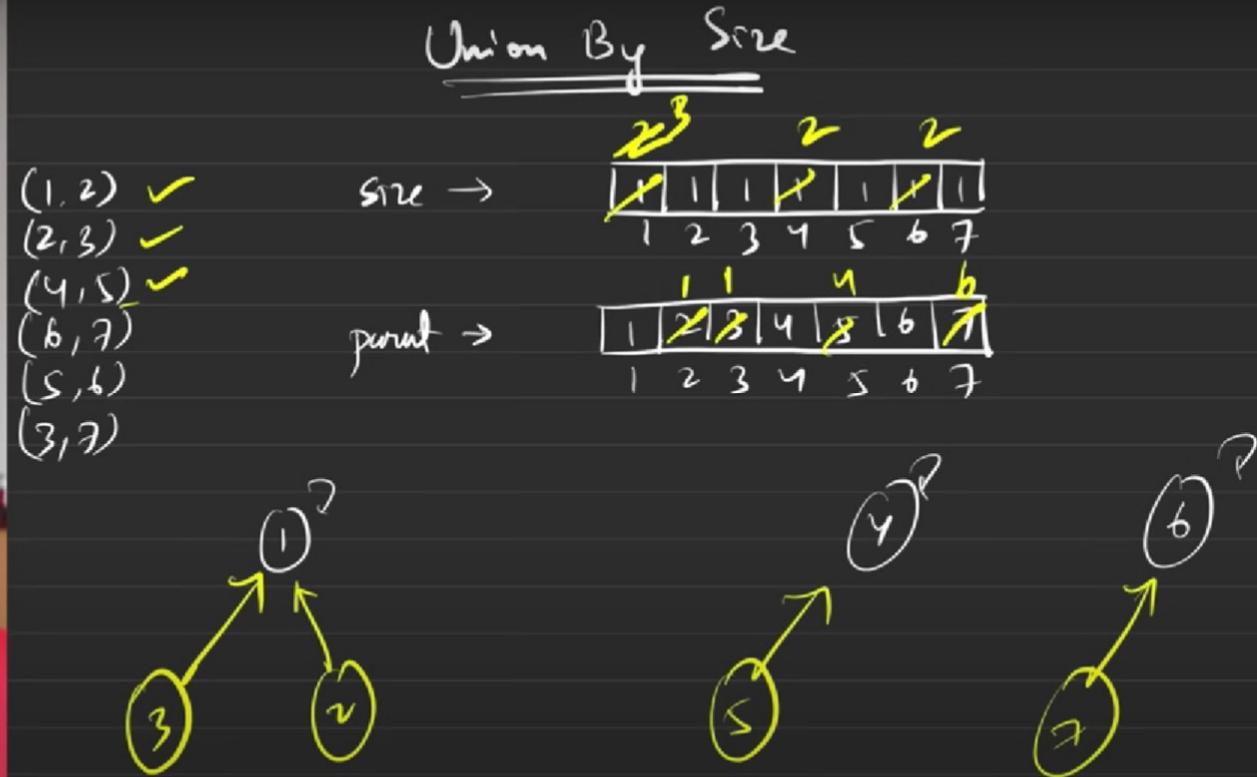


3[?] 4[?] 5[?] 6[?] 7[?]

⇒ G-46. Disjoint Set | Union by Rank | Union by Size | Path Compression



1.60



37:03 / 42:14



⇒ G-46. Disjoint Set | Union by Rank | Union by Size | Path Compression



1.60



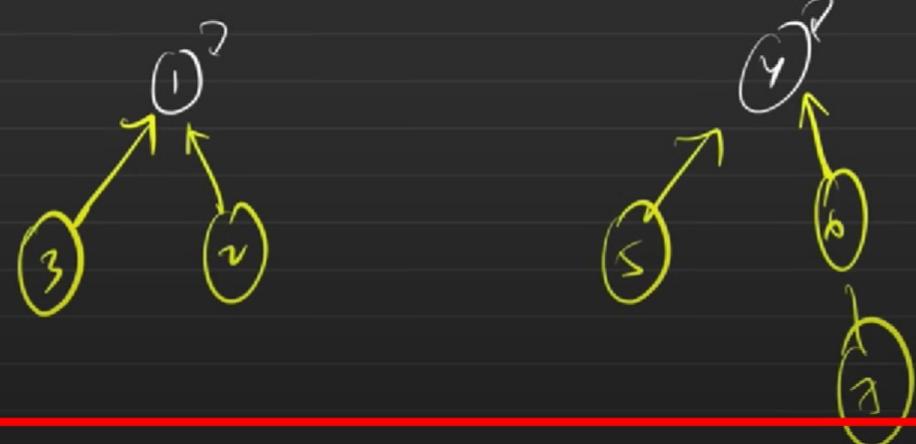
Union By Rank

size →

(1, 2)	✓	1	2	3	4	5	6	7
(2, 3)	✓	1	2	3	4	5	6	7
(4, 5)	✓	1	2	3	4	5	6	7
(6, 7)	✓	1	2	3	4	5	6	7
(5, 6)	✓	1	2	3	4	5	6	7
(3, 7)		1	2	3	4	5	6	7

parent →

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



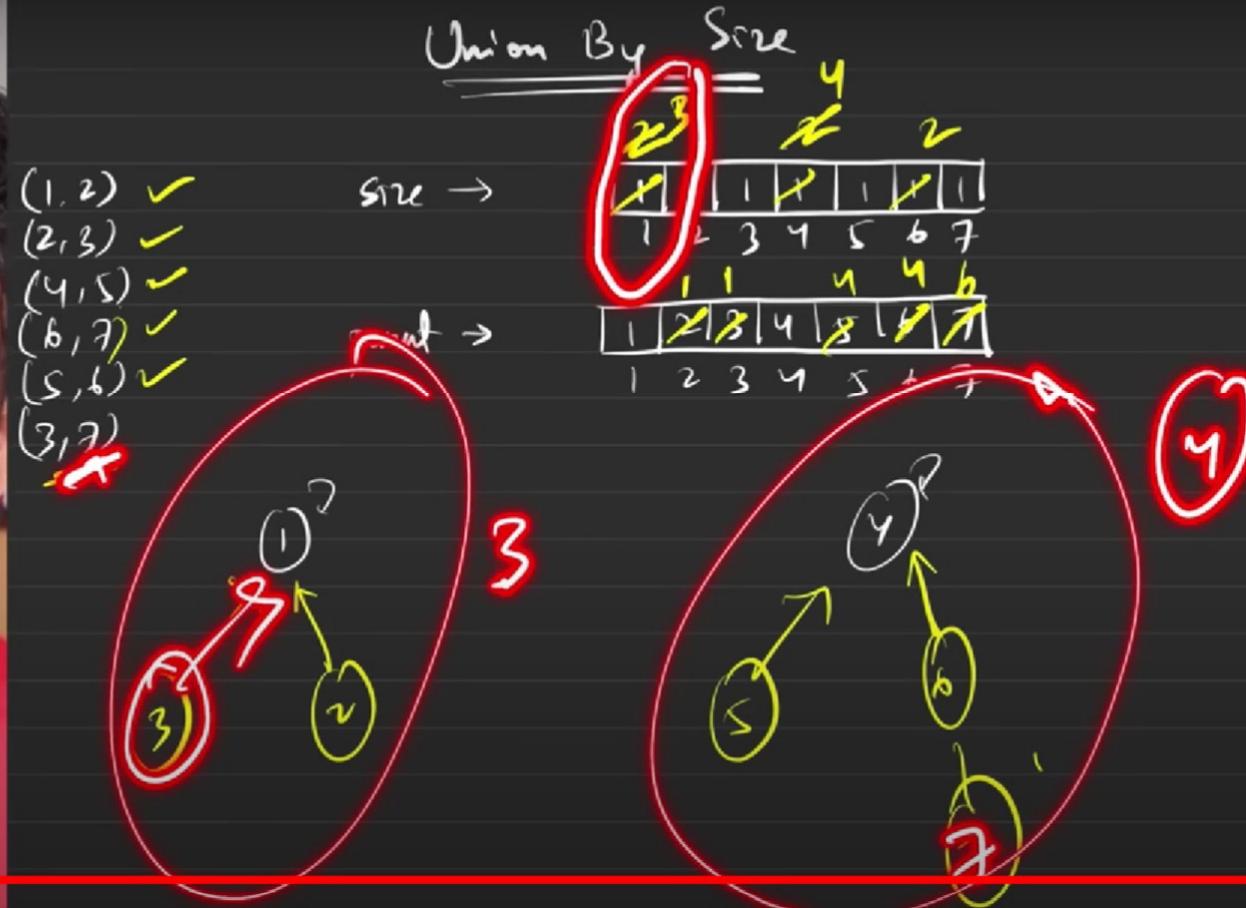
TUF



37:35 / 42:14



☰ G-46. Disjoint Set | Union by Rank | Union by Size | Path Compression

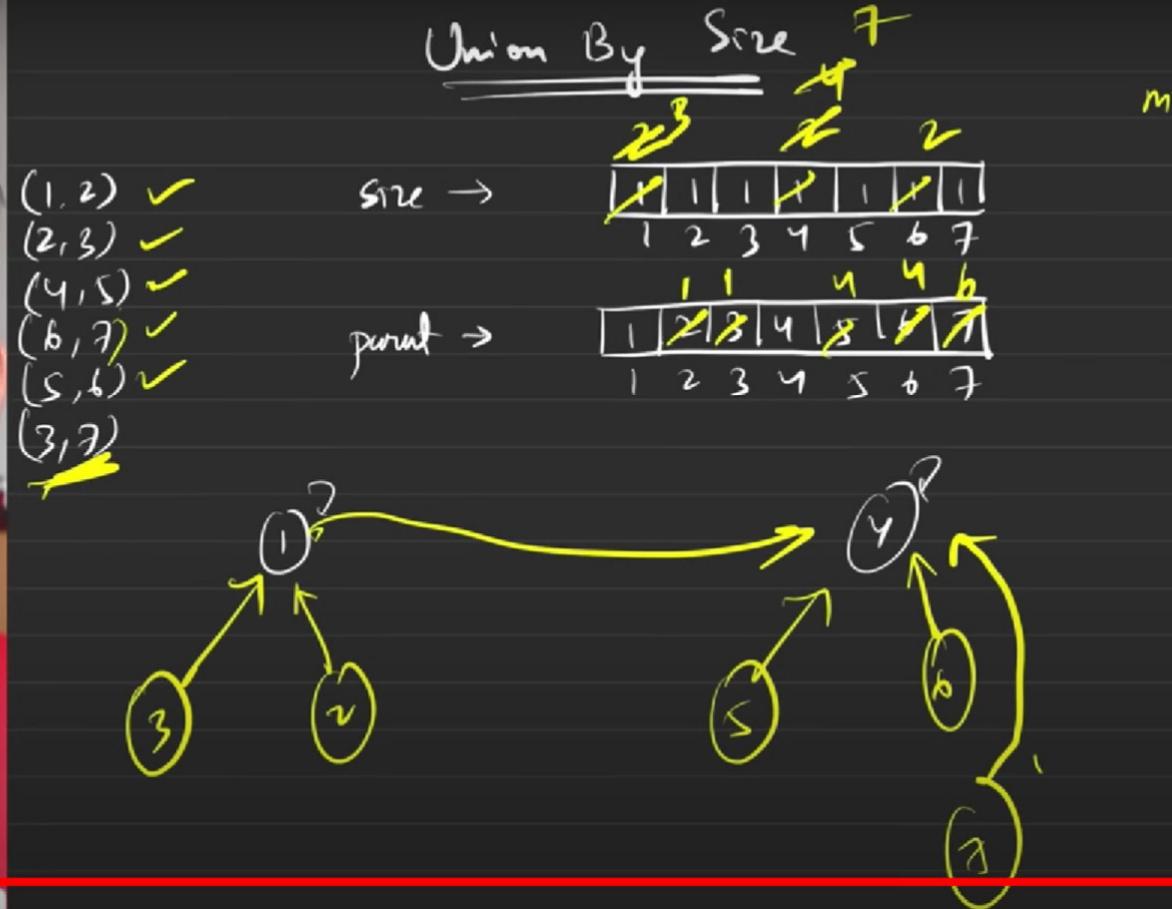


TUF

→ G-46. Disjoint Set | Union by Rank | Union by Size | Path Compression



1.60



TUF



38:17 / 42:14



► G-46. Disjoint Set | Union by Rank | Union by Size | Path Compression

1.60

```
31     parent[ulp_v] = ulp_u;
32     rank[ulp_u]++;
33 }
34 }
35
36 void unionBySize(int u, int v) {
37     int ulp_u = findUPar(u);
38     int ulp_v = findUPar(v);
39     if(ulp_u == ulp_v) return;
40     if(size[ulp_u] < size[ulp_v]) {
41         parent[ulp_u] = ulp_v;
42         size[ulp_v] += size[ulp_u];
43     }
44     else {
45         parent[ulp_v] = ulp_u;
46         size[ulp_u] += size[ulp_v];
47     }
48 }
49
50 int main() {
51 #ifndef ONLINE_JUDGE
52 freopen("input.txt", "r", stdin);
53 freopen("output.txt", "w", stdout);
54 #endif
55     DisjointSet ds(7),
56     ds.unionByRank(1, 2);
57     ds.unionByRank(2, 3);
58     ds.unionByRank(4, 5);
59     ds.unionByRank(6, 7);
60     ds.unionByRank(5, 6);
61     // if 3 and 7 same or not
62     if(ds.findUPar(3) == ds.findUPar(7)) {
63 [Finished in 0.9s]
```

```
5 class DisjointSet {
6     List<Integer> rank = new ArrayList<>();
7     List<Integer> parent = new ArrayList<>();
8     List<Integer> size = new ArrayList<>();
9
10    public DisjointSet(int n) {
11        for(int i = 0; i < n; i++) {
12            rank.add(0);
13            parent.add(i);
14            size.add(1);
15        }
16    }
17
18    public int findUPar(int node) {
19        if(node == parent.get(node)) {
20            return node;
21        }
22        int ulp_u = findUPar(parent.get(node));
23        parent.set(node, ulp_u);
24        return parent.get(node);
25    }
26
27    public void unionByRank(int u, int v) {
28    }
29
30    public void unionBySize(int u, int v) {
31        int ulp_u = findUPar(u);
32        int ulp_v = findUPar(v);
33        if(ulp_u == ulp_v) return;
34        if(size.get(ulp_u) < size.get(ulp_v)) {
35            parent.set(ulp_u, ulp_v);
36            size.set(ulp_v, size.get(ulp_v) + size.get(ulp_u));
37        }
38        else {
39            parent.set(ulp_v, ulp_u);
40            size.set(ulp_u, size.get(ulp_u) + size.get(ulp_v));
41        }
42    }
43
44    class Main {
45        public static void main (String[] args) {
46            DisjointSet ds = new DisjointSet(7);
47            ds.unionByRank(1, 2);
48            ds.unionByRank(2, 3);
49            ds.unionByRank(4, 5);
50            ds.unionByRank(6, 7);
51            ds.unionByRank(5, 6);
52            // if 3 and 7 same or not
53            if(ds.findUPar(3) == ds.findUPar(7)) {
54                System.out.println("Same");
55            }
56            else
57                System.out.println("Not Same");
58
59            ds.unionByRank(3, 7);
60            if(ds.findUPar(3) == ds.findUPar(7)) {
61                System.out.println("Same");
62            }
63            else
64                System.out.println("Not Same");
65        }
66    }
67
68
69
70
71
72
73
74
75
76
77
78
79
```



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

input.txt
1 Not same
2 Same
3

output.txt

TUF



39:52 / 42:14



► G-46. Disjoint Set | Union by Rank | Union by Size | Path Compression

1.60

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 class DisjointSet {
4     vector<int> rank, parent, size;
5 public:
6     DisjointSet(int n) {
7         rank.resize(n+1, 0);
8         parent.resize(n+1);
9         size.resize(n+1);
10        for(int i = 0;i<=n;i++) {
11            parent[i] = i;
12            size[i] = 1;
13        }
14    }
15
16    int findUPar(int node) {
17        if(node == parent[node])
18            return node;
19        return parent[node] = findUPar(parent[node]);
20    }
21
22    void unionByRank(int u, int v) {
23        int ulp_u = findUPar(u);
24        int ulp_v = findUPar(v);
25        if(ulp_u == ulp_v) return;
26        if(rank[ulp_u] < rank[ulp_v]) {
27            parent[ulp_u] = ulp_v;
28        }
29        else if(rank[ulp_v] < rank[ulp_u]) {
30            parent[ulp_v] = ulp_u;
31        }
32    }
33
34    [Finished in 0.9s]
```



Input.txt

```
1 3 3 3
2 1 2 3
3 2 1 2
4 3 1 3
```



output.txt

```
1 Not same
2 Same
3
```

TUF



40:16 / 42:14



► G-46. Disjoint Set | Union by Rank | Union by Size | Path Compression

1.60

```
43     parent[ulp_u] = ulp_v;
44     size[ulp_v] += size[ulp_u];
45   } else {
46     parent[ulp_v] = ulp_vu;
47     size[ulp_u] += size[ulp_v];
48   }
49 }
50 };
51 int main() {
52 #ifndef ONLINE_JUDGE
53 freopen("input.txt", "r", stdin);
54 freopen("output.txt", "w", stdout);
55 #endif
56   DisjointSet ds(7),
57   ds.unionBySize(1, 2);
58   ds.unionBySize(2, 3);
59   ds.unionBySize(4, 5);
60   ds.unionBySize(6, 7);
61   ds.unionBySize(5, 6);
62 // if 3 and 7 same or not
63 if(ds.findUPar(3) == ds.findUPar(7)) {
64   cout << "Same\n";
65 }
66 else cout << "Not same\n";
67
68   ds.unionBySize(3, 7);
69
70   if(ds.findUPar(3) == ds.findUPar(7)) {
71     cout << "Same\n";
72   }
73 }
```

[Finished in 0.9s]

```
5 class DisjointSet {
6   List<Integer> rank = new ArrayList<>();
7   List<Integer> parent = new ArrayList<>();
8   List<Integer> size = new ArrayList<>();
9   public DisjointSet(int n) {
10    for(int i = 0; i < n; i++) {
11      rank.add(0);
12      parent.add(i);
13      size.add(1);
14    }
15  }
16
17  public int findUPar(int node) {
18    if(node == parent.get(node)) {
19      return node;
20    }
21    int ulp = findUPar(parent.get(node));
22    parent.set(node, ulp);
23    return parent.get(node);
24  }
25
26  public void unionByRank(int u, int v) {
27  }
28
29  public void unionBySize(int u, int v) {
30    int ulp_u = findUPar(u);
31    int ulp_v = findUPar(v);
32    if(ulp_u == ulp_v) return;
33    if(size.get(ulp_u) < size.get(ulp_v)) {
34      parent.set(ulp_u, ulp_v);
35      size.set(ulp_v, size.get(ulp_v) + size.get(ulp_u));
36    }
37    else {
38      parent.set(ulp_v, ulp_u);
39      size.set(ulp_u, size.get(ulp_u) + size.get(ulp_v));
40    }
41  }
42
43  class Main {
44    public static void main (String[] args) {
45      DisjointSet ds = new DisjointSet(7);
46      ds.unionByRank(1, 2);
47      ds.unionByRank(2, 3);
48      ds.unionByRank(4, 5);
49      ds.unionByRank(6, 7);
50      ds.unionByRank(5, 6);
51
52 // if 3 and 7 same or not
53 if(ds.findUPar(3) == ds.findUPar(7)) {
54   System.out.println("Same");
55 }
56 else
57   System.out.println("Not Same");
58
59   ds.unionBySize(3, 7);
60   if(ds.findUPar(3) == ds.findUPar(7)) {
61     System.out.println("Same");
62   }
63 else
64   System.out.println("Not Same");
65  }
66 }
```



Input.txt

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

output.txt

1	Not same
2	Same
3	

TUF



40:19 / 42:14



► G-46. Disjoint Set | Union by Rank | Union by Size | Path Compression



```
1.60
22 void unionByRank(int u, int v) {
23     int ulp_u = findUPar(u);
24     int ulp_v = findUPar(v);
25     if(ulp_u == ulp_v) return;
26     if(rank[ulp_u] < rank[ulp_v]) {
27         parent[ulp_u] = ulp_v;
28     }
29     else if(rank[ulp_v] < rank[ulp_u]) {
30         parent[ulp_v] = ulp_u;
31     }
32     else {
33         parent[ulp_v] = ulp_u;
34         rank[ulp_u]++;
35     }
36 }
37
38 void unionBySize(int u, int v) {
39     int ulp_u = findUPar(u);
40     int ulp_v = findUPar(v);
41     if(ulp_u == ulp_v) return;
42     if(size[ulp_u] < size[ulp_v]) {
43         parent[ulp_u] = ulp_v;
44         size[ulp_v] += size[ulp_u];
45     }
46     else {
47         parent[ulp_v] = ulp_u;
48         size[ulp_u] += size[ulp_v];
49     }
50 }
51 }
52 int main() {
[Finished in 0.9s]
```

```
5 class DisjointSet {
6     List<Integer> rank = new ArrayList<>();
7     List<Integer> parent = new ArrayList<>();
8     List<Integer> size = new ArrayList<>();
9
10    public DisjointSet(int n) {
11        for(int i = 0; i < n; i++) {
12            rank.add(0);
13            parent.add(i);
14            size.add(1);
15        }
16    }
17
18    public int findUPar(int node) {
19        if(node == parent.get(node)) {
20            return node;
21        }
22        int ulp = findUPar(parent.get(node));
23        parent.set(node, ulp);
24        return parent.get(node);
25    }
26
27    public void unionByRank(int u, int v) {
28    }
29
30    public void unionBySize(int u, int v) {
31        int ulp_u = findUPar(u);
32        int ulp_v = findUPar(v);
33        if(ulp_u == ulp_v) return;
34        if(size.get(ulp_u) < size.get(ulp_v)) {
35            parent.set(ulp_u, ulp_v);
36            size.set(ulp_v, size.get(ulp_v) + size.get(ulp_u));
37        }
38        else {
39            parent.set(ulp_v, ulp_u);
40            size.set(ulp_u, size.get(ulp_u) + size.get(ulp_v));
41        }
42    }
43
44    class Main {
45        public static void main (String[] args) {
46            DisjointSet ds = new DisjointSet(7);
47            ds.unionByRank(1, 2);
48            ds.unionByRank(2, 3);
49            ds.unionByRank(4, 5);
50            ds.unionByRank(6, 7);
51            ds.unionByRank(5, 8);
52
53            // If 3 and 7 same or not
54            if(ds.findUPar(3) == ds.findUPar(7)) {
55                System.out.println("Same");
56            }
57            else {
58                System.out.println("Not Same");
59            }
60            ds.unionByRank(3, 7);
61            if(ds.findUPar(3) == ds.findUPar(7)) {
62                System.out.println("Same");
63            }
64            else {
65                System.out.println("Not Same");
66            }
67        }
68    }
69 }
```



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

1	Not same
2	Same
3	

TUF



40:17 / 42:14



► G-46. Disjoint Set | Union by Rank | Union by Size | Path Compression

1.60

```
31     }
32     else {
33         parent[ulp_v] = ulp_u;
34         rank[ulp_u]++;
35     }
36 }
37
38 void unionBySize(int u, int v) {
39     int ulp_u = findUPar(u);
40     int ulp_v = findUPar(v);
41     if(ulp_u == ulp_v) return;
42     if(size[ulp_u] < size[ulp_v]) {
43         parent[ulp_u] = ulp_v;
44         size[ulp_v] += size[ulp_u];
45     }
46     else {
47         parent[ulp_v] = ulp_u;
48         size[ulp_u] += size[ulp_v];
49     }
50 }
51
52 int main() {
53 #ifndef ONLINE_JUDGE
54     freopen("input.txt", "r", stdin);
55     freopen("output.txt", "w", stdout);
56 #endif
57     DisjointSet ds(7);
58     ds.unionBySize(1, 2);
59     ds.unionBySize(2, 3);
60     ds.unionBySize(4, 5);
61     ds.unionBySize(6, 7);
62     // [1, 2, 3, 4, 5, 6, 7]
63
64     [Finished in 0.8s]
```

```
5 class DisjointSet {
6     List<Integer> rank = new ArrayList<>();
7     List<Integer> parent = new ArrayList<>();
8     List<Integer> size = new ArrayList<>();
9
10    public DisjointSet(int n) {
11        for(int i = 0; i < n; i++) {
12            rank.add(0);
13            parent.add(i);
14            size.add(1);
15        }
16    }
17
18    public int findUPar(int node) {
19        if(node == parent.get(node)) {
20            return node;
21        }
22        int ulp = findUPar(parent.get(node));
23        parent.set(node, ulp);
24        return parent.get(node);
25    }
26
27    public void unionByRank(int u, int v) {
28    }
29
30    public void unionBySize(int u, int v) {
31        int ulp_u = findUPar(u);
32        int ulp_v = findUPar(v);
33        if(ulp_u == ulp_v) return;
34        if(size.get(ulp_u) < size.get(ulp_v)) {
35            parent.set(ulp_u, ulp_v);
36            size.set(ulp_v, size.get(ulp_v) + size.get(ulp_u));
37        }
38        else {
39            parent.set(ulp_v, ulp_u);
40            size.set(ulp_u, size.get(ulp_u) + size.get(ulp_v));
41        }
42    }
43
44    class Main {
45        public static void main (String[] args) {
46            DisjointSet ds = new DisjointSet(7);
47            ds.unionByRank(1, 2);
48            ds.unionByRank(2, 3);
49            ds.unionByRank(4, 5);
50            ds.unionByRank(6, 7);
51            ds.unionByRank(5, 8);
52
53            // If 3 and 7 same or not
54            if(ds.findUPar(3) == ds.findUPar(7)) {
55                System.out.println("Same");
56            }
57            else {
58                System.out.println("Not Same");
59            }
60            ds.unionByRank(3, 7);
61            if(ds.findUPar(3) == ds.findUPar(7)) {
62                System.out.println("Same");
63            }
64            else {
65                System.out.println("Not Same");
66            }
67        }
68    }
69
70 }
```



Input.txt

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

output.txt

1	Not same
2	Same
3	

TUF



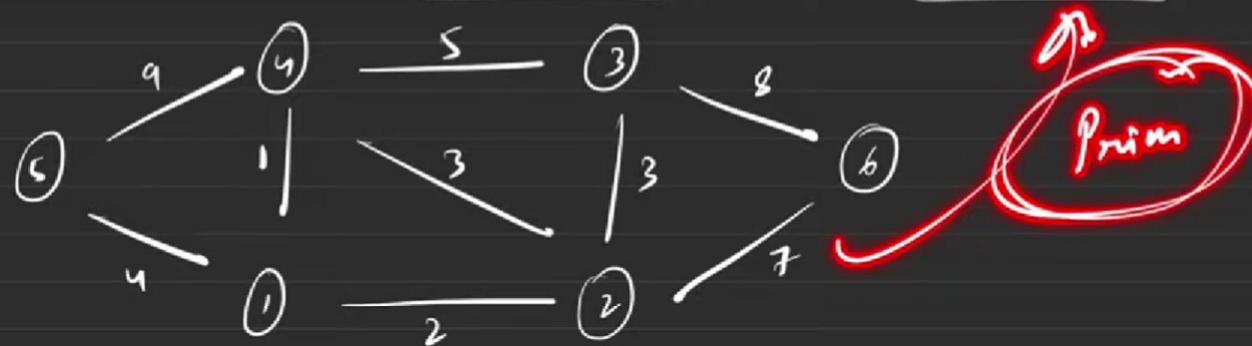
40:53 / 42:14



► G-47. Kruskal's Algorithm - Minimum Spanning Tree - C++ and Java

1.90

Kruskal's Algorithm → find MST



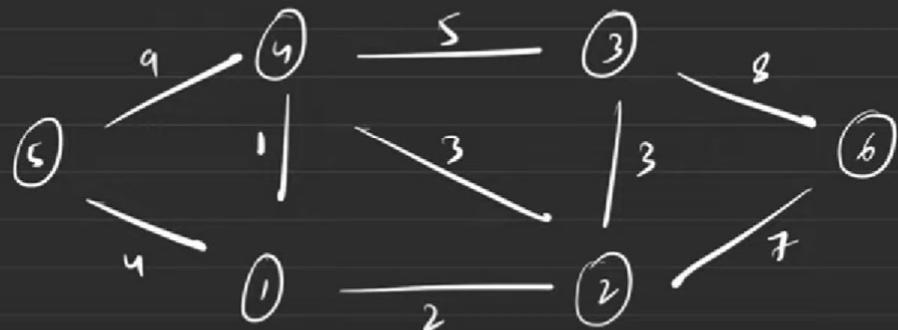
► G-47. Kruskal's Algorithm - Minimum Spanning Tree - C++ and Java



1.90

↳ Find Union↳ Union By RankDisjoint Set

Kruskal's Algorithm → Find MST



↳ sort all true
edges vector wth.
(w, u, v)

1, 1, 4
2, 1, 2
3, 2, 3
3, 2, 4
4, 1, 5
5, 3, 4
7, 2, 6
8, 3, 6
9, 4, 5

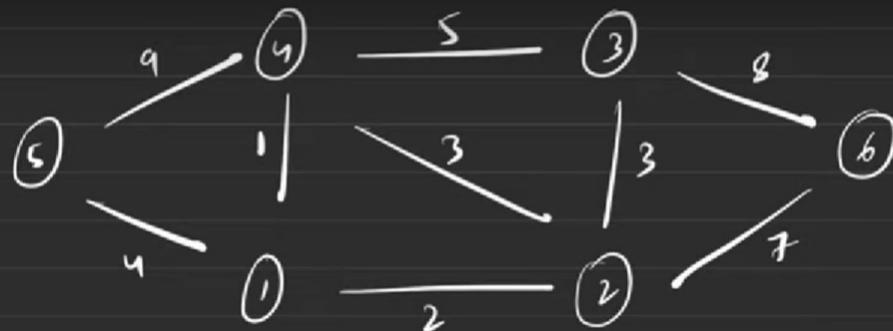


2:27 / 13:10



► G-47. Kruskal's Algorithm-- Minimum Spanning Tree - C++ and Java

2.05



(*) Sort all the edges vector w.r.t.
(wh, u, v)

✓ → 1, 1, 4
✓ → 2, 1, 2
3, 2, 3
3, 2, 1
4, 1, 5
5, 3, 4
7, 2, 6
8, 3, 6
9, 4, 5

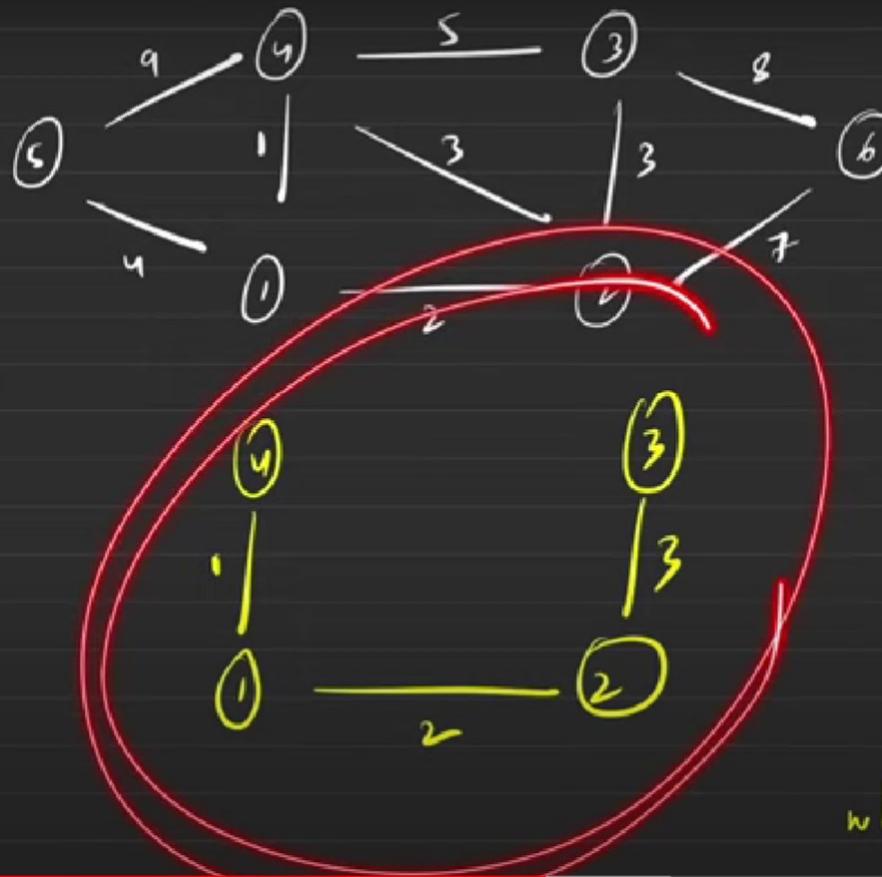


$$nk = 1 + 2$$



► G-47. Kruskal's Algorithm-- Minimum Spanning Tree - C++ and Java

2.05



(*) Sort all the edges vector w.r.t.
(wh, u, v)

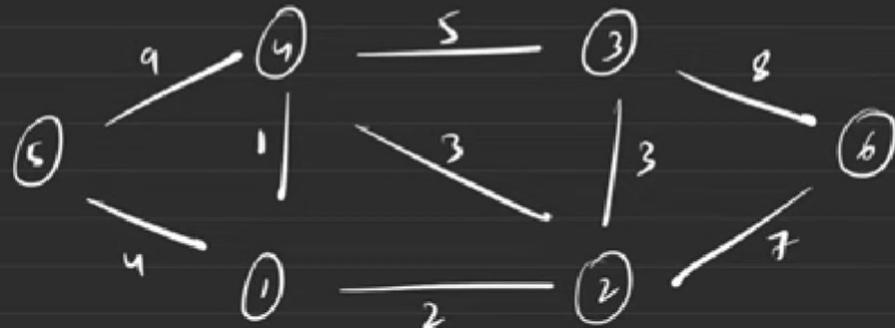
- ✓ → 1, 1, 4
- ✓ → 2, 1, 2
- ✓ → 3, 2, 3
- ✗ 3, 2, 4
- 4, 1, 5
- 5, 3, 4
- 7, 2, 6
- 8, 3, 6
- 9, 4, 5



► G-47. Kruskal's Algorithm - Minimum Spanning Tree - C++ and Java

2.05

Kruskal's Algorithm → find MST



(.) Sort all the edges acc to wt.
(wh, u, v)

- ✓ → 1, 1, 4
- ✓ → 2, 1, 2
- ✓ → 3, 2, 3
- X 3, 2, 4
- ✓ 4, 1, 5
- X 5, 3, 4
- 7, 2, 6
- 8, 3, 6
- 9, 4, 5



→ G-47. Kruskal's Algorithm - Minimum Spanning Tree - C++ and Java

(c) Sort all the

edges cect to wh.
(wh, u, v)

$\checkmark \rightarrow 1, 1, 4$

→ 2, 1, 2

1 → 2, 2, 3

✓ 3, 2, 3

X, Z, Y

✓ 4, 1, 5

X 5, 3, 4

7, 2, 6

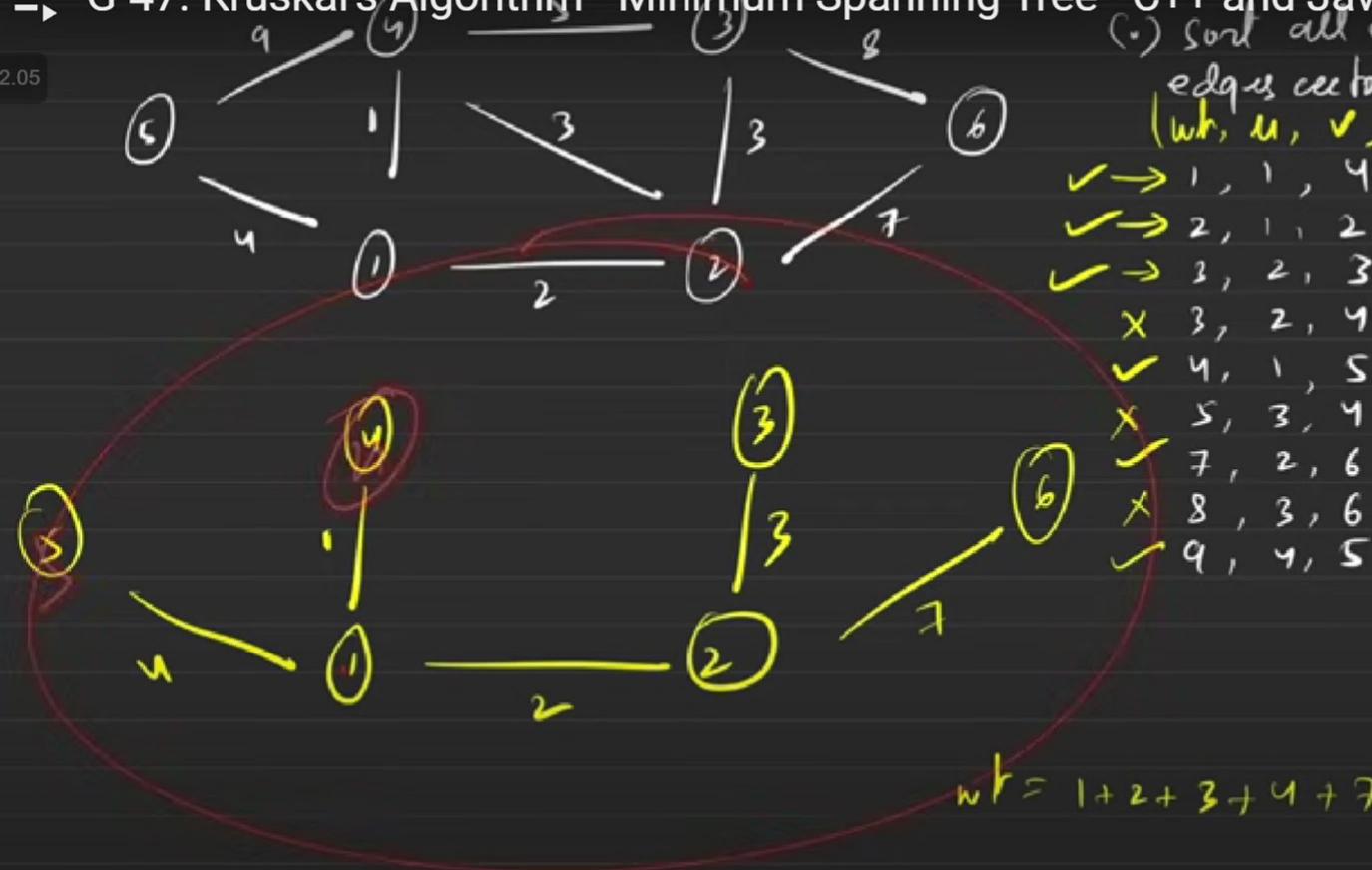
$$b) \quad \times \quad 8,3,6$$

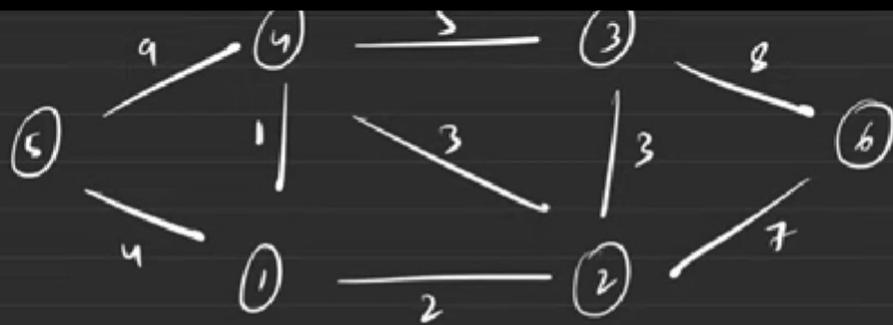
✓ 9, 9, 5

5

$$= 1 + 2 + 3 + 4 + 7$$

For more information about the study, please contact Dr. John D. Cacioppo at (773) 704-7895 or via e-mail at cacioppo@uic.edu.





(c) sort all the
edges according to weight.
(Wh, W, ✓)

- ✓ → 1, 1, 4
- ✓ → 2, 1, 2
- ✓ → 3, 2, 3
- X 3, 2, 4
- ✓ 4, 1, 5
- X 5, 3, 4
- ✓ 7, 2, 6
- X 8, 3, 6
- X 9, 4, 5



$$Wt = 1 + 2 + 3 + 4 + 7 \\ = 17$$



G-47. Kruskal's Algorithm - Minimum Spanning Tree - C++ and Java

Problems Courses Get Hired Contests CPOTD Practice

2.05 Problem Editorial Submissions Comments

Minimum Spanning Tree

Medium Accuracy: 49.39% Submissions: 63352 Points: 4

Geek Week 2022 is LIVE! Click Here to View All the Exciting Offers!

Given a weighted, undirected and connected graph of V vertices and E edges. The task is to find the sum of weights of the edges of the Minimum Spanning Tree.

Example 1:

Input:
3 3

Output Window

Compilation Results Custom Input

Problem Solved Successfully

You get marks only for the first correct submission if you solve the problem without viewing the full solution.

Test Cases Passed: 122 / 122 Your Total Score: 471

Total Time Taken: 2.42 Correct Submission Count: 6

C++ (g++ 5.4) Average Time: 26m

```
50     else {
51         parent[ulp_v] = ulp_u;
52         size[ulp_u] += size[ulp_v];
53     }
54 }
55 class Solution
56 {
57 public:
58 //Function to find sum of weights of edges of the Minimum Spanning Tree.
59 int spanningTree(int V, vector<vector<int>> adj[])
60 {
61     // 1 -> 2 wt = 5
62     // 1 -> (2, 5)
63     // 2 -> (1, 5)
64
65     // 5, 1, 2
66     // 5, 2, 1
67     vector<pair<int, pair<int, int>> edges;
68     // 0m []
69     for(int i = 0;i<V;i++) {
70         for(auto it : adj[i]) {
71             int adjNode = it[0];
72             int wt = it[1];
73             int node = i;
74
75             edges.push_back({wt, {node, adjNode}});
76         }
77     }
78     DisjointSet ds(V);
79     sort(edges.begin(), edges.end());
80     int mstWt = 0;
81     for(auto it : edges) {
82         int wt = it.first;
83         int u = it.second.first;
84         int v = it.second.second;
85
86         if(ds.findUPar(u) != ds.findUPar(v)) {
87             mstWt += wt;
88             ds.unionBySize(u, v);
89         }
90     }
91     return mstWt;
92 }
93 }
```

Custom Input

G-47. Kruskal's Algorithm - Minimum Spanning Tree - C++ and Java

Problems Courses Get Hired Contests CPOTD Practice

2.05 Problem Editorial Submissions Comments

C++ (g++ 5.4) Average Time: 26m

```
1: // Driver Code Ends
2:
3: class DisjointSet {
4:     vector<int> rank, parent, size;
5:
6: public:
7:     DisjointSet(int n) {
8:         rank.resize(n+1, 0);
9:         parent.resize(n+1);
10:        size.resize(n+1);
11:        for(int i = 0;i<n;i++) {
12:            parent[i] = i;
13:            size[i] = 1;
14:        }
15:    }
16:
17:    int findUPar(int node) {
18:        if(node == parent[node])
19:            return node;
20:        return parent[node] = findUPar(parent[node]);
21:    }
22:
23:    void unionByRank(int u, int v) {
24:        int ulp_u = findUPar(u);
25:        int ulp_v = findUPar(v);
26:        if(ulp_u == ulp_v) return;
27:        if(rank[ulp_u] < rank[ulp_v]) {
28:            parent[ulp_u] = ulp_v;
29:        }
30:        else if(rank[ulp_v] < rank[ulp_u]) {
31:            parent[ulp_v] = ulp_u;
32:        }
33:        else {
34:            parent[ulp_v] = ulp_u;
35:            rank[ulp_u]++;
36:        }
37:    }
38:
39:    void unionBySize(int u, int v) {
40:        int ulp_u = findUPar(u);
41:        int ulp_v = findUPar(v);
42:        if(ulp_u == ulp_v) return;
43:        if(size[ulp_u] < size[ulp_v]) {
44:            parent[ulp_u] = ulp_v;
45:            size[ulp_v] += size[ulp_u];
46:        }
47:    }
48:
49: };
50:
```

Custom Input

◀ ▶ ⏪ 10:34 / 13:10 ⏩

CC ⚙ #

G-47. Kruskal's Algorithm - Minimum Spanning Tree - C++ and Java

Problems Courses Get Hired Contests CPOTD Practice

2.05 Problem Editorial Submissions Comments C++ (g++ 5.4) Average Time: 26m

```
103 class Edge implements Comparable<Edge> {
104     int src, dest, weight;
105     Edge(int _src, int _dest, int _wt) {
106         this.src = _src; this.dest = _dest; this.weight = _wt;
107     }
108     // Comparator function used for
109     // sorting edges based on their weight
110     public int compareTo(Edge compareEdge)
111     {
112         return this.weight - compareEdge.weight;
113     }
114 }
115 class Solution
116 {
117     // Function to find sum of weights of edges of the Minimum Spanning Tree.
118     static int spanningTree(int V,
119     ArrayList<ArrayList<Integer>>> adj)
120     {
121         List<Edge> edges = new ArrayList<Edge>();
122         // O(N * E)
123         for(int i = 0;i<V;i++) {
124             for(int j = 0;j < adj.get(i).size();j++) {
125                 int adjNode = adj.get(i).get(j).get(0);
126                 int wt = adj.get(i).get(j).get(1);
127                 int node = i;
128                 Edge temp = new Edge(i, adjNode, wt);
129                 edges.add(temp);
130             }
131         }
132         DisjointSet ds = new DisjointSet(V);
133         // M log M
134         Collections.sort(edges);
135         int mstWt = 0;
136         // M x 4 x alpha x 2
137         for(int i = 0;i<edges.size();i++) {
138             int wt = edges.get(i).weight;
139             int u = edges.get(i).src;
140             int v = edges.get(i).dest;
141
142             if(ds.findUPar(u) != ds.findUPar(v)) {
143                 mstWt += wt;
144                 ds.unionBySize(u, v);
145             }
146         }
147         return mstWt;
148     }
149 }
```

63 // 1 -> {2, 5}
64 // 2 -> {1, 5}
65
66 // 5, 1, 2
67 // 5, 2, 1
68 vector<pair<int, pair<int, int>>> edges;
69 for(int i = 0;i<V;i++) {
70 for(auto it : adj[i]) {
71 int adjNode = it[0];
72 int wt = it[1];
73 int node = i;
74
75 edges.push_back({wt, {node, adjNode}});
76 }
77 }
78 DisjointSet ds(V);
79 sort(edges.begin(), edges.end());
80 int mstWt = 0;
81 for(auto it : edges) {
82 int wt = it.first;
83 int u = it.second.first;
84 int v = it.second.second;
85
86 if(ds.findUPar) {
87 mstWt += wt;
88 }
89 }
90 return mstWt;
91 }
92 }
93 };
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108

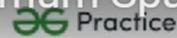
Custom Input

◀ ▶ ⏪ ⏩ ⏴ ⏵ 10:36 / 13:10

CC ⚙ #

G-47. Kruskal's Algorithm - Minimum Spanning Tree - C++ and Java

Problems Courses Get Hired Contests CPOTD



2.05 Problem

Editorial

Submissions

Comments

C++ (g++ 5.4) ▾

Average Time: 26m

```
103 class Edge implements Comparable<Edge> {
104     int src, dest, weight;
105     Edge(int _src, int _dest, int _wt) {
106         this.src = _src; this.dest = _dest; this.weight = _wt;
107     }
108     // Comparator function used for
109     // sorting edges based on their weight
110     public int compareTo(Edge compareEdge)
111     {
112         return this.weight - compareEdge.weight;
113     }
114 }
115 class Solution
116 {
117     // Function to find sum of weights of edges of the Minimum Spanning Tree.
118     static int spanningTree(int V,
119     ArrayList<ArrayList<Integer>>> adj)
120     {
121         List<Edge> edges = new ArrayList<Edge>();
122         // O(N * E)
123         for(int i = 0;i<V;i++) {
124             for(int j = 0;j < adj.get(i).size();j++) {
125                 int adjNode = adj.get(i).get(j).get(0);
126                 int wt = adj.get(i).get(j).get(1);
127                 int node = i;
128                 Edge temp = new Edge(i, adjNode, wt);
129                 edges.add(temp);
130             }
131         }
132         DisjointSet ds = new DisjointSet(V);
133         // M log M
134         Collections.sort(edges);
135         int mstWt = 0;
136         // M x 4 x alpha x 2
137         for(int i = 0;i<edges.size();i++) {
138             int wt = edges.get(i).weight;
139             int u = edges.get(i).src;
140             int v = edges.get(i).dest;
141
142             if(ds.findUPar(u) != ds.findUPar(v)) {
143                 mstWt += wt;
144                 ds.unionBySize(u, v);
145             }
146         }
147         return mstWt;
148     }
149 }
150 }
```



TUF



10:49 / 13:10



Custom Input



G-47. Kruskal's Algorithm - Minimum Spanning Tree - C++ and Java

Problems Courses Get Hired Contests POTD Practice

2.05 Problem Editorial Submissions Comments

Minimum Spanning Tree

Medium Accuracy: 49.39% Submissions: 63352 Points: 4

Geek Week 2022 is LIVE! Click Here to View All the Exciting Offers!

Given a weighted, undirected and connected graph of V vertices and E edges. The task is to find the sum of weights of the edges of the Minimum Spanning Tree.

Example 1:

Input:
3 3
0 1 5
1 2 3
0 2 1

Output:
4

Explanation:

```
C++ (g++ 5.4) Average Time: 28ms
47     parent[ulp_u] = ulp_v;
48     size[ulp_v] += size[ulp_u];
49   }
50   else {
51     parent[ulp_v] = ulp_u;
52     size[ulp_u] += size[ulp_v];
53   }
54 }
55 };
56 class Solution {
57 {
58 public:
59 //Function to find sum of weights of edges of the Minimum Spanning Tree.
60 int spanningTree(int V, vector<vector<int>> adj[])
61 {
62   // 1 -> 2 wt = 5
63   // 1 -> (2, 5)
64   // 2 -> (1, 5)
65   // 5, 1, 2
66   // 5, 2, 1
67   // 0(M)
68   vector<pair<int, pair<int,int>> edges;
69   // O(N + E)
70   for(int i = 0;i<V;i++) {
71     for(auto it : adj[i]) {
72       int adjNode = it[0];
73       int wt = it[1];
74       int node = i;
75
76       edges.push_back({wt, {node, adjNode}});
77     }
78   }
79   DisjointSet ds(V);
80   // M log M
81   sort(edges.begin(), edges.end());
82   int mstWt = 0;
83   // [M x A x alpha x 2]
84   for(auto it : edges) {
85     int wt = it.first;
86     int u = it.second.first;
87     int v = it.second.second;
88
89     if(ds.findUPar(u) != ds.findUPar(v)) {
90       mstWt += wt;
91       ds.unionBySize(u, v);
92     }
93   }
94   return mstWt;
95 }
```

Custom Input

◀ ▶ ⏪ ⏩ 12:45 / 13:10

CC ⚙ #

G-54. Strongly Connected Components - Kosaraju's Algorithm

Press Esc to exit full screen

1.90

Strongly Connected Components \rightarrow Kosaraju's Algorithm.



\rightarrow no. of SCC
 \rightarrow print the SCC



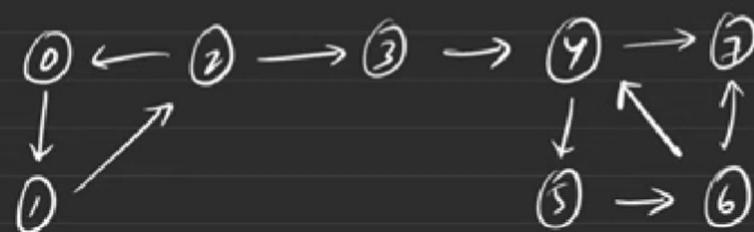
0:28 / 22:43



G-54. Strongly Connected Components - Kosaraju's Algorithm

1.90

Strongly Connected Components → Kosaraju's Algorithm.



→ Doubled Graph



⇒ G-54. Strongly Connected Components - Kosaraju's Algorithm

1.90

Strongly Connected Components → Kosaraju's Algorithm.



1:06 / 22:43



► G-54. Strongly Connected Components - Kosaraju's Algorithm

1.90

Strongly Connected Components → Kosaraju's Algorithm.



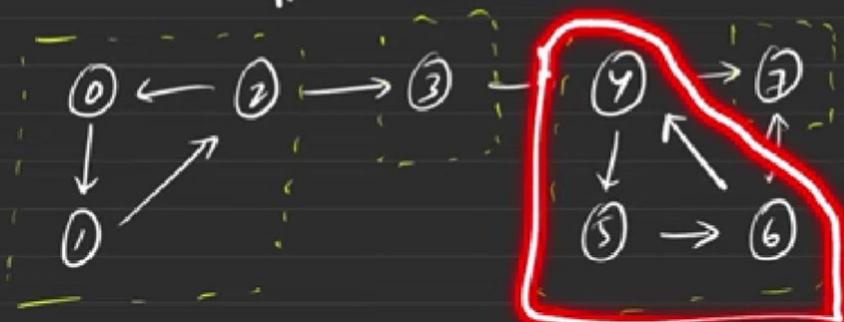
1:26 / 22:43



⇒ G-54. Strongly Connected Components - Kosaraju's Algorithm

1.90

Strongly Connected Components → Kosaraju's Algorithm.



0 1 2
3



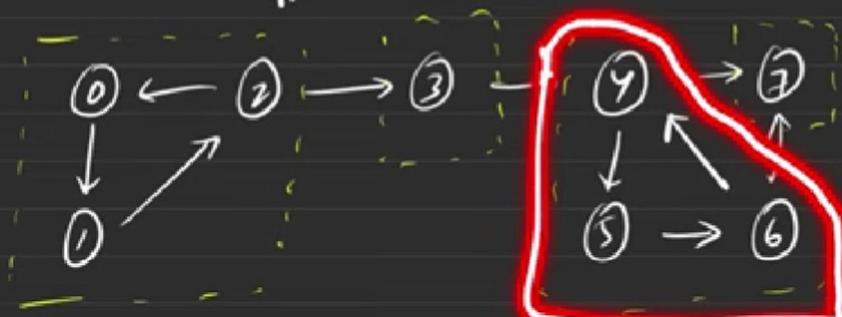
3:01 / 22:43



⇒ G-54. Strongly Connected Components - Kosaraju's Algorithm

1.90

Strongly Connected Components → Kosaraju's Algorithm.

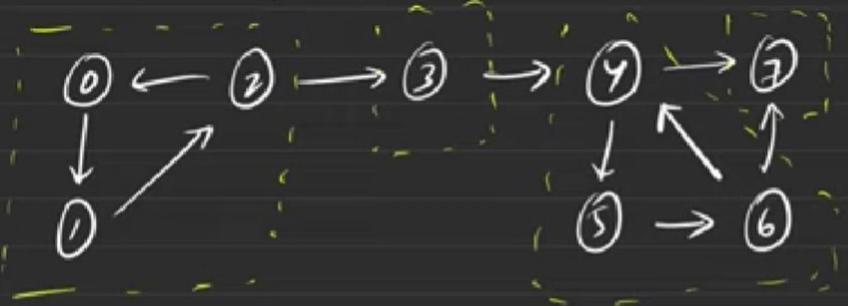


0 1 2
3



3:04 / 22:43





0 1 2
3 4 5 6
7

{ } 7 succ



► G-54. Strongly Connected Components - Kosaraju's Algorithm

Press Esc to exit full screen

1.00

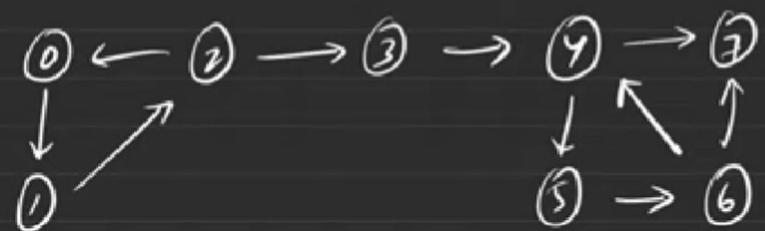
Strongly Connected Components \rightarrow Kosaraju's Algorithm.



→ no. of SCC
→ print the SCC



Strongly Connected Components → Kosaraju's Algorithm.



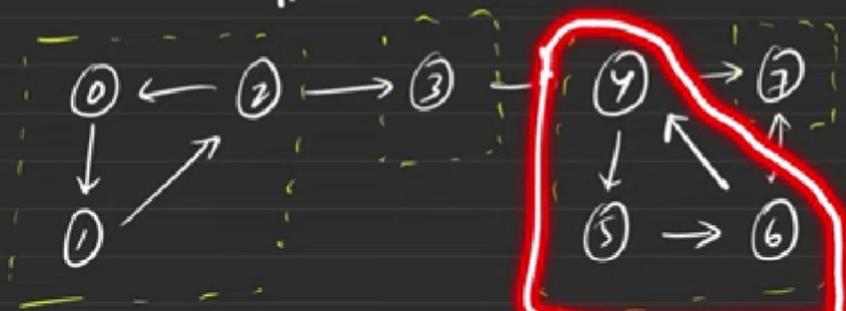
→ Doubled Graph



⇒ G-54. Strongly Connected Components - Kosaraju's Algorithm

1.60

Strongly Connected Components → Kosaraju's Algorithm.



0 1 2
3



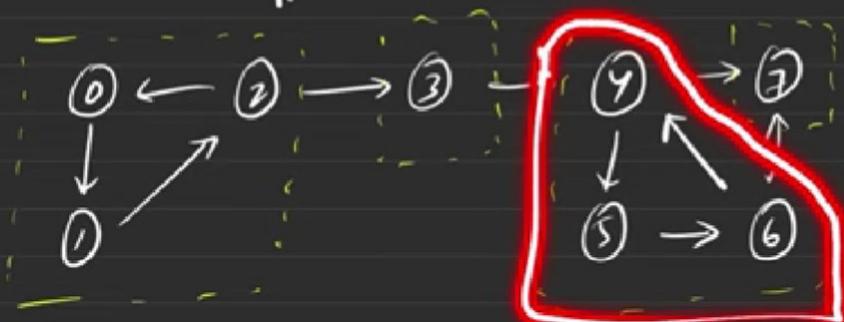
2:57 / 22:43



⇒ G-54. Strongly Connected Components - Kosaraju's Algorithm

1.60

Strongly Connected Components → Kosaraju's Algorithm.



0 1 2
3



3:05 / 22:43



⇒ G-54. Strongly Connected Components - Kosaraju's Algorithm



0 1 2
3 4 5 6 } 9 SCC



3:30 / 22:43

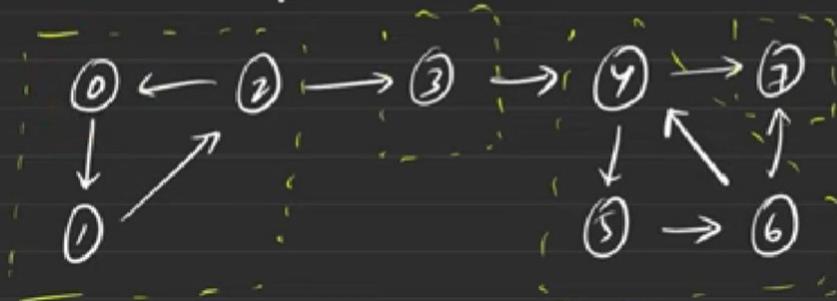


⇒ G-54. Strongly Connected Components - Kosaraju's Algorithm

Strongly Connected Components Kosaraju's Algorithm.

1.60

Press Esc to exit full screen



Thought Process

| scc1 | → | scc2 | → | scc3 | → | scc4 |



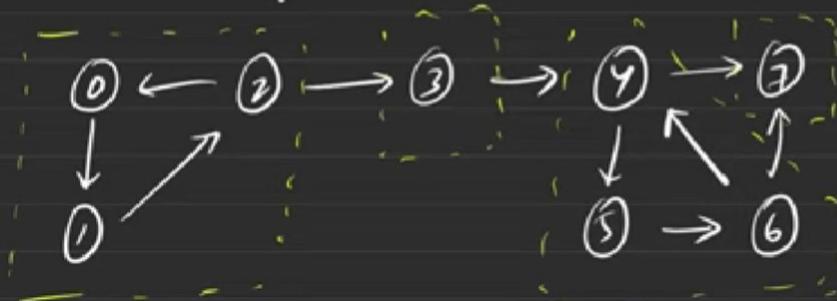
5:22 / 22:43



⇒ G-54. Strongly Connected Components - Kosaraju's Algorithm

Strongly Connected Components → Kosaraju's Algorithm.

1.60



Thought Process

[scc1] ↘ [scc2] ↘ [scc3] ↘ [scc4]



5:46 / 22:43



⇒ G-54. Strongly Connected Components - Kosaraju's Algorithm

Strongly Connected Components → Kosaraju's Algorithm.

1.60



Thought Process

| scc1 | ↘ | scc2 | ↘ | scc3 | ↘ | scc4 |



⇒ G-54. Strongly Connected Components - Kosaraju's Algorithm

Strongly Connected Components → Kosaraju's Algorithm.

1.60

Thought Process



SCC1 | SCC2 | SCC3 | SCC4



Strongly Connected Components \rightarrow Kosaraju's Algorithm.

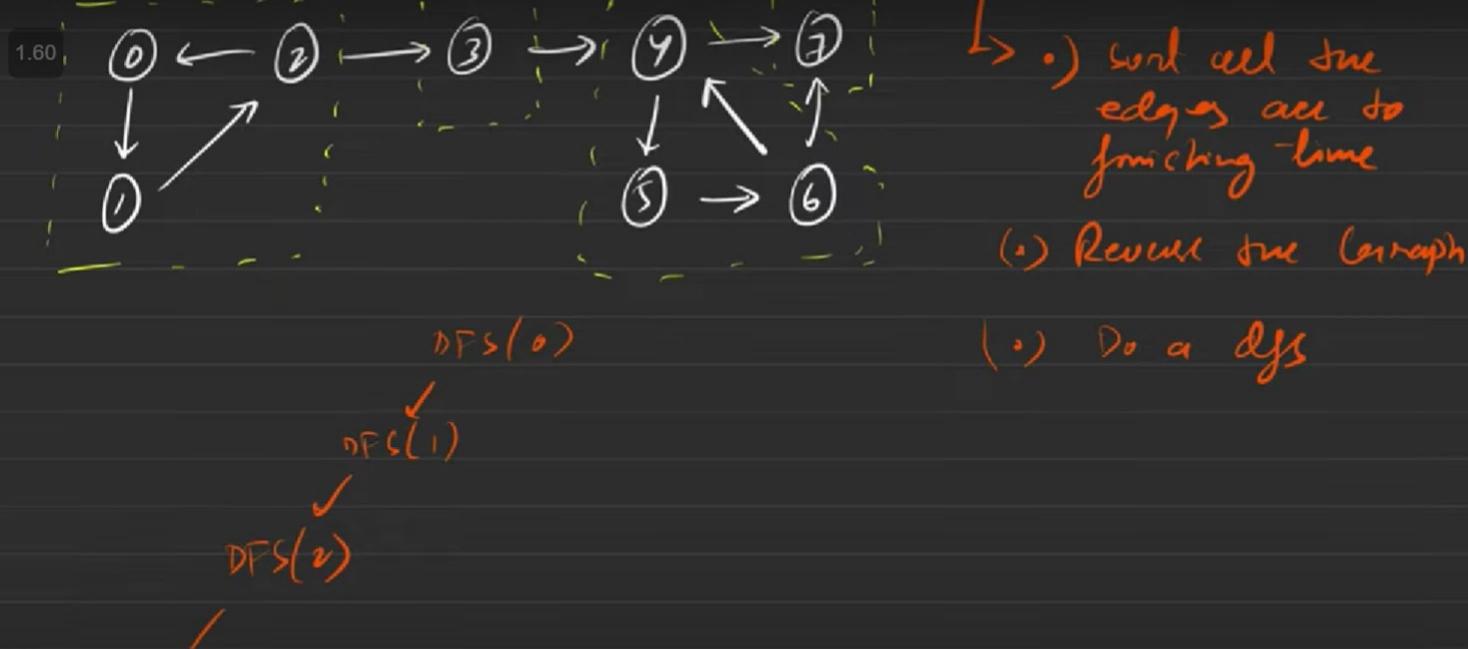


Thought Process



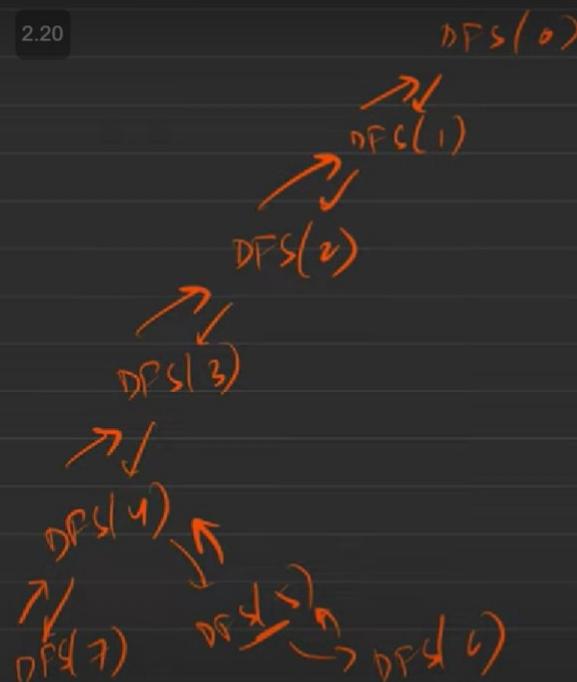
TUF

→ G-54. Strongly Connected Components - Kosaraju's Algorithm



G-54. Strongly Connected Components - Kosaraju's Algorithm

2.20



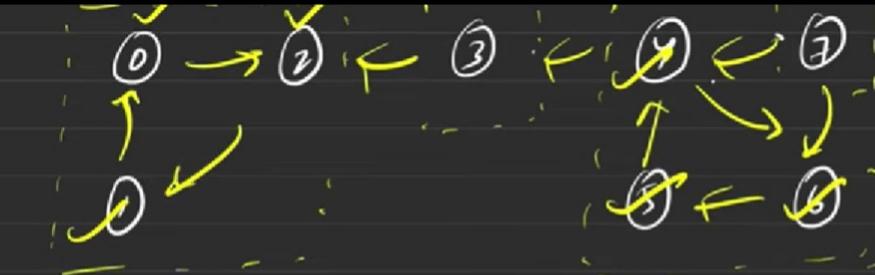
(.) Do a djs

0
1
2
3
4
5
6
7



11:15 / 22:43





Work all the edges are to finishing time

✓ Reverse the graph
 (+) Do a DFS



⇒ G-54. Strongly Connected Components - Kosaraju's Algorithm

2.20

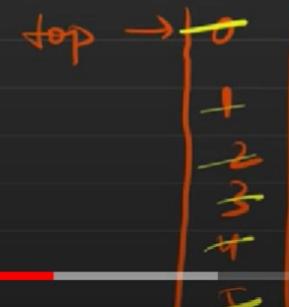
Strongly Connected Components → Kosaraju's Algorithm.



DFS(7)

0 2 1
3
4 6 5
7

- ↪ sort all the edges acc to finishing time
- ✓ Reverse the graph
- (+) Do a DFS



14:18 / 22:43



4

G-54. Strongly Connected Components - Kosaraju's Algorithm

Problems Courses Get Hired Contests </POTD

Practice Press Esc to exit full screen

C++ (g++ 5.4) • Average Time: 20m

```
class Solution
{
    void dfs(int node, int vis[], ArrayList<ArrayList<Integer>> adj,
             Stack<Integer> st) {
        vis[node] = 1;
        for(Integer it : adj.get(node)) {
            if(vis[it] == 0) {
                dfs(it, vis, adj, st);
            }
        }
        st.push(node);
    }

    private void dfs3(int node, int vis[], ArrayList<ArrayList<Integer>> adjT) {
        vis[node] = 1;
        for(Integer it : adjT.get(node)) {
            if(vis[it] == 0) {
                dfs3(it, vis, adjT);
            }
        }
    }

    //Function to find number of strongly connected components in the graph.
    public int kosaraju(int V, ArrayList<ArrayList<Integer>> adj)
    {
        int[] vis = new int[V];
        Stack<Integer> st = new Stack<Integer>();
        for(int i = 0;i<V;i++) {
            if(vis[i] == 0) {
                dfs(i, vis, adj, st);
            }
        }

        ArrayList<ArrayList<Integer>> adjT = new ArrayList<ArrayList<Integer>>();
        for(int i = 0;i<V;i++) {
            adjT.add(new ArrayList<Integer>());
        }
        for(int i = 0;i<V;i++) {
            vis[i] = 0;
            for(Integer it : adj.get(i)) {
                // i -> it
                // it -> i
                adjT.get(it).add(i);
            }
        }
        int scc = 0;
        while(!st.isEmpty()) {
            int node = st.peek();
            st.pop();
            if(vis[node] == 0) {
                scc++;
                dfs3(node, vis, adjT);
            }
        }
    }
}
```

class Solution

```
void dfs(int node, vector<int> &vis, vector<int> adj[],
         stack<int> &st) {
    vis[node] = 1;
    for(auto it : adj[node]) {
        if(!vis[it]) {
            dfs(it, vis, adj, st);
        }
    }
    st.push(node);
}

public:
//Function to find number of strongly connected components in the graph.
int kosaraju(int V, vector<int> adj[])
{
    vector<int> &vis(V, 0);
    stack<int> st;
    for(int i = 0;i<V;i++) {
        if(!vis[i]) {
            dfs(i, vis, adj, st);
        }
    }

    vector<int> adjT[V];
    for(int i = 0;i<V;i++) {
        for(auto it : adj[i]) {
            // i -> it
            // it -> i
            adjT[it].push_back(i);
        }
    }
}
```

Custom Input Compile

CC Settings Full Screen

◀ ▶ ⏪ ⏩ ⏴ 18:42 / 22:43



G-54. Strongly Connected Components - Kosaraju's Algorithm

Problems Courses Get Hired Contests </POTD

```
47 class Solution
48 {
49     public void dfs(int node, int[] vis, ArrayList<ArrayList<Integer>> adj,
50                     Stack<Integer> st) {
51         vis[node] = 1;
52         for(Integer it : adj.get(node)) {
53             if(vis[it] == 0) {
54                 dfs(it, vis, adj, st);
55             }
56         }
57         st.push(node);
58     }
59     private void dfs3(int node, int[] vis, ArrayList<ArrayList<Integer>> adjT) {
60         vis[node] = 1;
61         for(Integer it : adjT.get(node)) {
62             if(vis[it] == 0) {
63                 dfs3(it, vis, adjT);
64             }
65         }
66     }
67     //Function to find number of strongly connected components in the graph.
68     public int kosaraju(int V, ArrayList<ArrayList<Integer>> adj)
69     {
70         int[] vis = new int[V];
71         Stack<Integer> st = new Stack<Integer>();
72         for(int i = 0;i<V;i++) {
73             if(vis[i] == 0) {
74                 dfs(i, vis, adj, st);
75             }
76         }
77         ArrayList<ArrayList<Integer>> adjT = new ArrayList<ArrayList<Integer>>();
78         for(int i = 0;i<V;i++) {
79             adjT.add(new ArrayList<Integer>());
80         }
81         for(int i = 0;i<V;i++) {
82             vis[i] = 0;
83             for(Integer it : adj.get(i)) {
84                 // i -> it
85                 // it -> i
86                 adjT.get(it).add(i);
87             }
88         }
89         int scc = 0;
90         while(!st.isEmpty()) {
91             int node = st.peek();
92             st.pop();
93             if(vis[node] == 0) {
94                 scc++;
95                 dfs3(node, vis, adjT);
96             }
97         }
98     }
99 }
```

```
18         st.push(node);
19     }
20     public:
21     //Function to find number of strongly connected components in the graph.
22     int kosaraju(int V, vector<int> adj[])
23     {
24         vector<int> &vis(V, 0);
25         stack<int> st;
26         for(int i = 0;i<V;i++) {
27             if(!vis[i]) {
28                 dfs(i, vis, adj, st);
29             }
30         }
31         vector<int> adjT[V];
32         for(int i = 0;i<V;i++) {
33             vis[i] = 0;
34             for(auto it : adj[i]) {
35                 // i -> it
36                 // it -> i
37                 adjT[it].push_back(i);
38             }
39         }
40         int scc = 0;
41         while(!st.empty()) {
42             int node = st.top();
43             st.pop();
44             if(!vis[node]) {
45                 scc++;
46                 dfs3(node, vis, adjT);
47             }
48         }
49     }
50 }
51
52
53
54
55 };
```



19:47 / 22:43



Custom Input Compile



G-54. Strongly Connected Components - Kosaraju's Algorithm

Problems Courses Get Hired Contests < POTD

Practice



```
47 class Solution
48 {
49     void dfs(int node, int[] vis, ArrayList<ArrayList<Integer>> adj,
50             Stack<Integer> st) {
51         vis[node] = 1;
52         for(Integer it : adj.get(node)) {
53             if(vis[it] == 0) {
54                 dfs(it, vis, adj, st);
55             }
56         }
57         st.push(node);
58     }
59     private void dfs3(int node, int[] vis, ArrayList<ArrayList<Integer>> adjT) {
60         vis[node] = 1;
61         for(Integer it : adjT.get(node)) {
62             if(vis[it] == 0) {
63                 dfs3(it, vis, adjT);
64             }
65         }
66     }
67     //Function to find number of strongly connected components in the graph.
68     public int kosaraju(int V, ArrayList<ArrayList<Integer>> adj)
69     {
70         int[] vis = new int[V];
71         Stack<Integer> st = new Stack<Integer>();
72         for(int i = 0;i<V;i++) {
73             if(vis[i] == 0) {
74                 dfs(i, vis, adj, st);
75             }
76         }
77         ArrayList<ArrayList<Integer>> adjT = new ArrayList<ArrayList<Integer>>();
78         for(int i = 0;i<V;i++) {
79             adjT.add(new ArrayList<Integer>());
80         }
81         for(int i = 0;i<V;i++) {
82             vis[i] = 0;
83             for(Integer it : adj.get(i)) {
84                 // i -> it
85                 // it -> i
86                 adjT.get(it).add(i);
87             }
88         }
89     }
90     int scc = 0;
91     while(!st.isEmpty()) {
92         int node = st.peek();
93         st.pop();
94         if(vis[node] == 0) {
95             scc++;
96             dfs3(node, vis, adjT);
97         }
98     }
99 }
```

```
C++ (g++ 5.4) ▶ Driver Code Ends
1
2
3
4
5
6
7
8 class Solution
9 {
10     private:
11     void dfs(int node, vector<int> &vis, vector<int> adj[],
12             stack<int> &st) {
13         vis[node] = 1;
14         for(auto it: adj[node]) {
15             if(!vis[it]) {
16                 dfs(it, vis, adj, st);
17             }
18         }
19         st.push(node);
20     }
21     private:
22     void dfs3(int node, vector<int> &vis, vector<int> adjT[]) {
23         vis[node] = 1;
24         for(auto it: adjT[node]) {
25             if(!vis[it]) {
26                 dfs3(it, vis, adjT);
27             }
28         }
29     }
30 }
31 public:
32     //Function to find number of strongly connected components in the graph.
33     int kosaraju(int V, vector<int> adj[])
34     {
35         vector<int> &vis(V, 0);
36         stack<int> st;
37         for(int i = 0;i<V;i++) {
38             if(!vis[i]) {
39                 dfs(i, vis, adj, st);
40             }
41         }
42     }
43 }
```



20:22 / 22:43



Custom Input

Compile



G-54. Strongly Connected Components - Kosaraju's Algorithm

Given a Directed Graph with V vertices (Numbered from 0 to $V-1$) and E edges, Find the number of strongly connected components in the graph.

Example 1:

Input:

```
graph TD; 0 --> 1; 0 --> 2; 0 --> 3; 1 --> 2;
```

Output Window

Compilation Results

```
prog.cpp:27:11: error: adj was not declared in this scope
    dfs3(it, vis, adj);
            ^
prog.cpp: In member function int Solution::kosaraju(int,
std::vector<int>*):
prog.cpp:35:30: error: expression list treated as compound
expression in initializer [-fpermissive]
    vector<int> &vis(V, 0);
                           ^
prog.cpp:35:30:.....
```

Expected Output:

Code:

```
2.20 Problem
Editorial Submissions Comments
C++ (g++ 5.4) Average Time: 20m
5

29     }
30 }
31 public:
32 //Function to find number of strongly connected components in the graph.
33 int kosaraju(int V, vector<int> adj[])
34 {
35     vector<int> &vis(V, 0);
36     stack<int> st;
37     for(int i = 0;i<V;i++) {
38         if(!vis[i]) {
39             dfs(i, vis, adj, st);
40         }
41     }
42
43     vector<int> adjT[V];
44     for(int i = 0;i<V;i++) {
45         vis[i] = 0;
46         for(auto it : adj[i]) {
47             // i -> it
48             // it -> i
49             adjT[it].push_back(i);
50         }
51     }
52     int scc = 0;
53     while(!st.empty()) {
54         int node = st.top();
55         st.pop();
56         if(!vis[node]) {
57             scc++;
58             dfs3(node, vis, adjT);
59         }
60     }
61     return scc;
62 }
63
64
65 };
66 };
```

Player Controls: ▶️ ⏪ ⏩ 🔍 20:41 / 22:43

Video Player Controls: CC ⚙️ #

G-54. Strongly Connected Components - Kosaraju's Algorithm

Problems Courses Get Hired Contests POTD Practice

2.20 Problem Editorial Submissions Comments C++ (g++ 5.4) Average Time: 20m

Given a Directed Graph with V vertices (Numbered from 0 to $V-1$) and E edges, Find the number of strongly connected components in the graph.

Example 1:

Input:

Output Window

Compilation Results Custom Input

Compilation Completed

For Input: 5
5 5
1 0
0 2
2 1
0 3
3 4

Code:

```
15-     if(!vis[it]) {
16-         dfs(it, vis, adj, st);
17-     }
18- }
19-
20-     st.push(node);
21- }
22- private:
23- void dfs3(int node, vector<int> &vis, vector<int> adjT[]) {
24-     vis[node] = 1;
25-     for(auto it : adjT[node]) {
26-         if(!vis[it]) {
27-             dfs3(it, vis, adjT);
28-         }
29-     }
30- }
31- public:
32- //Function to find number of strongly connected components in the graph.
33- int kosaraju(int V, vector<int> adj[])
34- {
35-     vector<int> vis(V, 0);
36-     stack<int> st;
37-     for(int i = 0;i<V;i++) {
38-         if(!vis[i]) {
39-             dfs(i, vis, adj, st);
40-         }
41-     }
42-
43-     vector<int> adjT[V];
44-     for(int i = 0;i<V;i++) {
45-         vis[i] = 0;
46-         for(auto it : adj[i]) {
47-             // i -> it
48-             // it -> i
49-             adjT[it].push_back(i);
50-         }
51-     }
52-     int scc = 0;
```

Your Output: 2

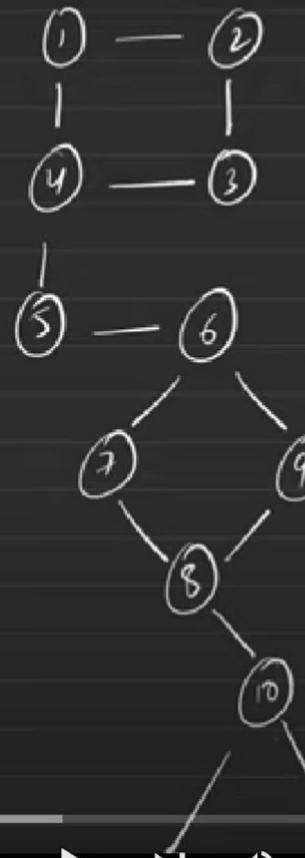
20:51 / 22:43

Custom Input Compile CC Settings Full Screen

► G-55. Bridges in Graph - Using Tarjan's Algorithm of time in and low time

Bridges in Graph

1.90



t_{in} → DFS time insertion

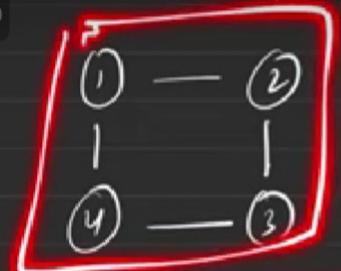
low → min lowest time insertion
of all adjacent nodes
apart from parent.



→ G-55. Bridges in Graph - Using Tarjan's Algorithm of time in and low time

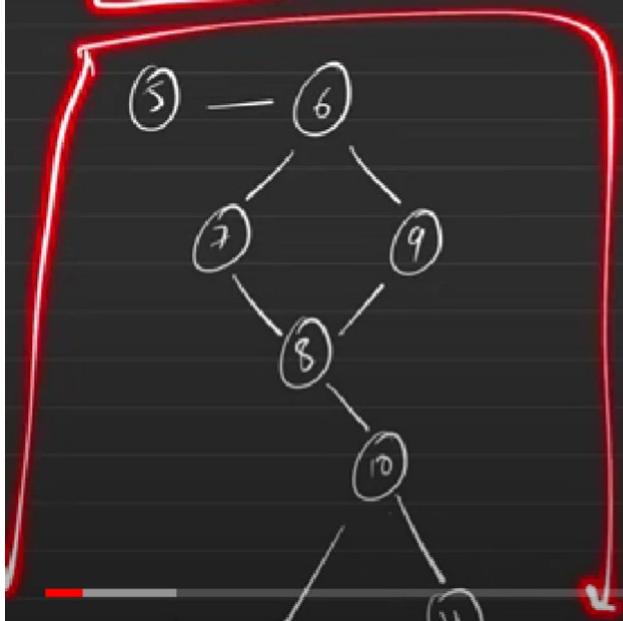
Bridges in Graph

1.90



t_{in} → DFS time insertion

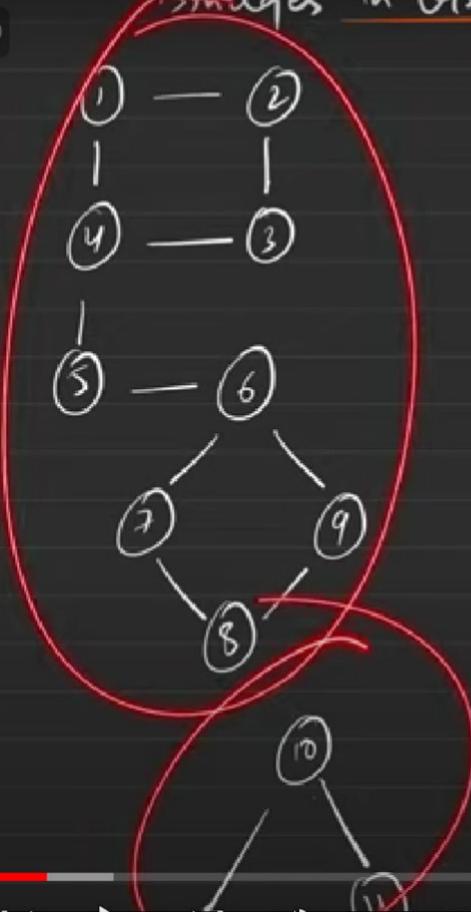
low → min lowest time insertion
of all adjacent nodes
apart from parent.



→ G-55. Bridges in Graph - Using Tarjan's Algorithm of time in and low time

1.90

Bridges in Graph



t_{inj} → DFS time insertion

low_{ij} → min lowest time insertion
of all adjacent nodes
apart from parent.



Bridges in Graph

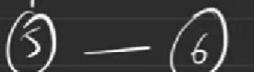
pre-neg \rightarrow DFS



|



|



—

9



—

11

time \rightarrow DFS time insertion

low \rightarrow min lowest time insertion
of all adjacent nodes
apart from parent.



⇒ G-55. Bridges in Graph - Using Tarjan's Algorithm of time in and low time

1.90
0 — 1

1 — 4 — 3

1 — 5 — 6

7 — 8 — 9

8 — 10 — 11

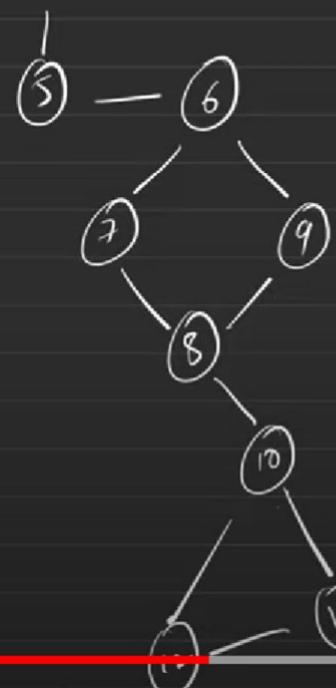
$t_{in}[j] \rightarrow$ DFS time insertion

$low[j] \rightarrow$ min lowest time insertion
of all adjacent nodes
apart from parent.

4 5
5 6
10 8



► G-55. Bridges in Graph - Using Tarjan's Algorithm of time in and low time



t_{inj} → DFS time insertion

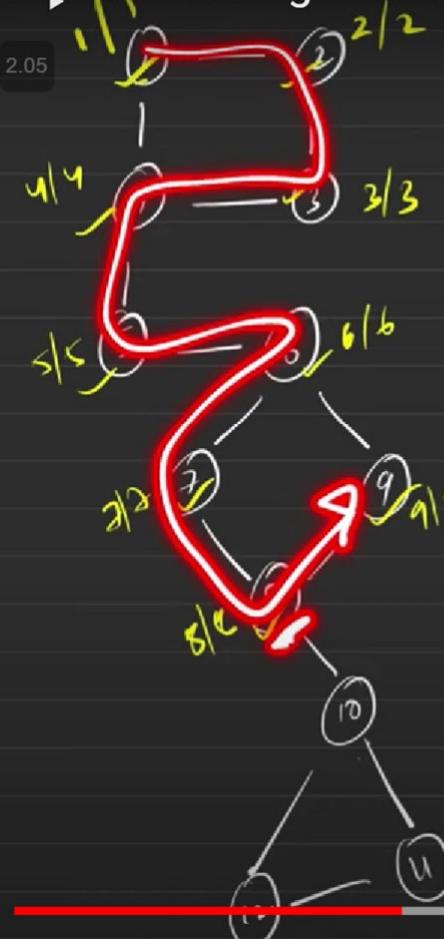
low_{inj} → min lowest time insertion
of all adjacent nodes
apart from parent.



3:30 / 23:24



⇒ G-55. Bridges in Graph - Using Tarjan's Algorithm of time in and low time



$ti[i]$ → DFS time insertion

$low[i]$ → min lowest time insertion
of all adjacent nodes
apart from parent.

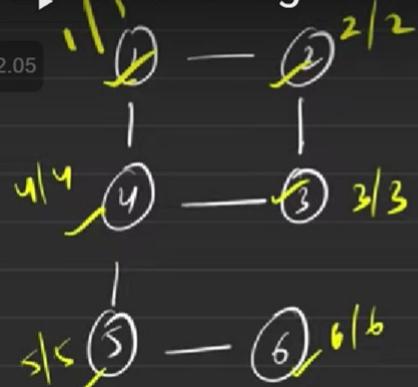


4:55 / 23:24



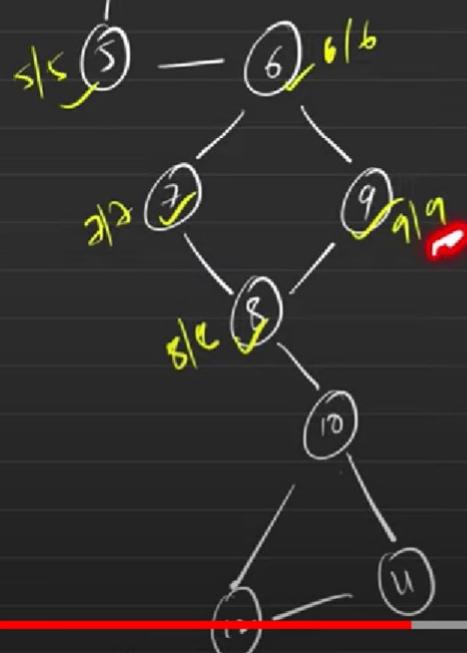
⇒ G-55. Bridges in Graph - Using Tarjan's Algorithm of time in and low time

2.05



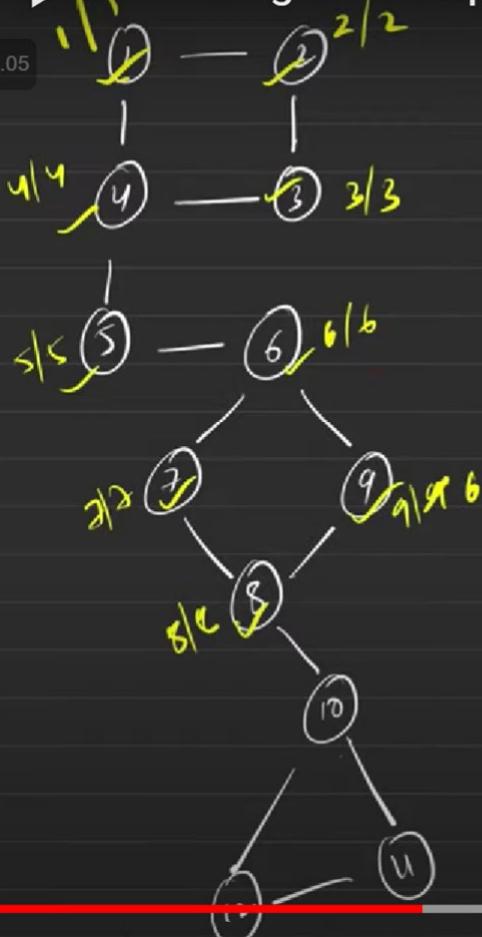
time → DFS time insertion

low → min lowest time insertion
of all adjacent nodes
apart from parent.



⇒ G-55. Bridges in Graph - Using Tarjan's Algorithm of time in and low time

2.05

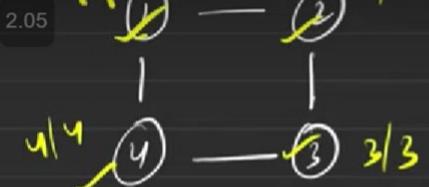


t_{in} → DFS time insertion

low_{in} → min lowest time insertion
of all adjacent nodes
apart from parent.



⇒ G-55. Bridges in Graph - Using Tarjan's Algorithm of time in and low time



t_{inj} → DFS time insertion

low_{inj} → min lowest time insertion
of all adjacent nodes
apart from parent.

DFS(8)
↑
DFS(9)



G-55. Bridges in Graph - Using Tarjan's Algorithm of time in and low time

LeetCode

1.45 Description Discussion Solutions Submissions / C++ Auto

1192. Critical Connections in a Network

Hint ⓘ

Hard ✓ 5K 167 ⚡ Companies

There are n servers numbered from 0 to $n - 1$ connected by undirected server-to-server connections forming a network where $\text{connections}[i] = [a_i, b_i]$ represents a connection between servers a_i and b_i . Any server can reach other servers directly or indirectly through the network.

A *critical connection* is a connection that, if removed, will make some servers unable to reach some other server.

Return all critical connections in the network in any order.

Example 1:

```
1 class Solution {
2 public:
3     vector<vector<int>> criticalConnections(int n, vector<vector<int>>& connections) {
4
5     }
6 };
```

16:31 / 23:24

CC ⚡ #

G-55. Bridges in Graph - Using Tarjan's Algorithm of time in and low time

The image shows a video player interface. On the left, there is a screenshot of a LeetCode problem page for "G-55. Bridges in Graph - Using Tarjan's Algorithm of time in and low time". The code editor displays two versions of the solution in C++:

```
1 class Solution {
2     private int timer = 1;
3     private void dfs(int node, int parent, int[] vis,
4                      ArrayList<ArrayList<Integer>> adj, int tin[], int low[],
5                      List<List<Integer>> bridges) {
6         vis[node] = 1;
7         tin[node] = low[node] = timer;
8         timer++;
9         for(Integer it: adj.get(node)) {
10             if(it == parent) continue;
11             if(vis[it] == 0) {
12                 dfs(it, node, vis, adj, tin, low, bridges);
13                 low[node] = Math.min(low[node], low[it]);
14                 // node --- it
15                 if(low[it] > tin[node]) {
16                     bridges.add(Arrays.asList(it, node));
17                 }
18             } else {
19                 low[node] = Math.min(low[node], low[it]);
20             }
21         }
22     }
23     public List<List<Integer>> criticalConnections(int n,
24             List<List<Integer>> connections) {
25         ArrayList<ArrayList<Integer>> adj =
26             new ArrayList<ArrayList<Integer>>();
27         for(int i = 0;i<n;i++) {
28             adj.add(new ArrayList<Integer>());
29         }
30         for(List<Integer> it : connections) {
31             int u = it.get(0); int v = it.get(1);
32             adj.get(u).add(v);
33             adj.get(v).add(u);
34         }
35         int[] vis = new int[n];
36         int[] tin = new int[n];
37         int[] low = new int[n];
38         List<List<Integer>> bridges = new ArrayList<List<Integer>>();
39         dfs(0, -1, vis, adj, tin, low, bridges);
40         return bridges;
41     }
42 }
```

```
1 class Solution {
2     private:
3     int timer = 1;
4     private:
5     void dfs(int node, int parent, vector<int> &vis,
6              vector<vector<int>> adj[], int tin[], int low[]) {
7         vis[node] = 1;
8         tin[node] = low[node] = timer;
9         timer++;
10        for(auto it: adj[node]) {
11            if(it == parent) continue;
12            if(vis[it] == 0) {
13                dfs(it, node, vis, adj, tin, low);
14                low[node] = min(low[node], low[it]);
15            }
16        }
17    }
18    public:
19    vector<vector<int>> criticalConnections(int n,
20        vector<vector<int>> &connections) {
21        vector<int> adj[n];
22        for(auto it : connections) {
23            adj[it[0]].push_back(it[1]);
24            adj[it[1]].push_back(it[0]);
25        }
26        vector<int> vis(n, 0);
27        int tin[n];
28        int low[n];
29    }
30 }
```

The input for the problem is: `n = 4, connections = [[0,1],[1,2],`

The video player controls at the bottom include: back, forward, volume, and a timestamp of 19:40 / 23:24. There are also icons for closed captions (CC), settings, and full screen.

A video feed of a man with a beard and short hair, wearing a red t-shirt with the letters "TUF" on it, is visible on the right side of the interface.



G-55. Bridges in Graph - Using Tarjan's Algorithm of time in and low time



1.60
Description Discussion Solutions Submissions C++ Auto

```
1 class Solution {
2     private int timer = 1;
3     private void dfs(int node, int parent, int[] vis,
4                      ArrayList<ArrayList<Integer>> adj, int tin[], int low[],
5                      List<List<Integer>> bridges) {
6         vis[node] = 1;
7         tin[node] = low[node] = timer;
8         timer++;
9         for(Integer it: adj.get(node)) {
10             if(it == parent) continue;
11             if(vis[it] == 0) {
12                 dfs(it, node, vis, adj, tin, low, bridges);
13                 low[node] = Math.min(low[node], low[it]);
14                 // node --- it
15                 if(low[it] > tin[node]) {
16                     bridges.add(Arrays.asList(it, node));
17                 }
18             } else {
19                 low[node] = Math.min(low[node], low[it]);
20             }
21         }
22     }
23     public List<List<Integer>> criticalConnections(int n,
24             List<List<Integer>> connections) {
25         ArrayList<ArrayList<Integer>> adj =
26             new ArrayList<ArrayList<Integer>>();
27         for(int i = 0;i<n;i++) {
28             adj.add(new ArrayList<Integer>());
29         }
30         for(List<Integer> it : connections) {
31             int u = it.get(0); int v = it.get(1);
32             adj.get(u).add(v);
33             adj.get(v).add(u);
34         }
35         int[] vis = new int[n];
36         int[] tin = new int[n];
37         int[] low = new int[n];
38         List<List<Integer>> bridges = new ArrayList<List<Integer>>();
39         dfs(0, -1, vis, adj, tin, low, bridges);
40         return bridges;
41     }
42 }
```

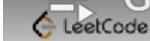
Input: n = 4, connections = [[0,1],[1,2],

```
1 class Solution {
2     private:
3     int timer = 1;
4     private:
5     void dfs(int node, int parent, vector<int> &vis,
6              vector<vector<int>> adj[], int tin[], int low[], vector<vector<int>> &bridges)
7     {
8         vis[node] = 1;
9         tin[node] = low[node] = timer;
10        timer++;
11        for(auto it: adj[node]) {
12            if(it == parent) continue;
13            if(vis[it] == 0) {
14                dfs(it, node, vis, adj, tin, low, bridges);
15                low[node] = min(low[node], low[it]);/* bridges
16                // node --- it
17                if(low[it] > tin[node]) {
18
19                }
20            } else {
21
22            }
23        }
24    }
25 public:
26     vector<vector<int>> criticalConnections(int n,
27             vector<vector<int>> &connections) {
28         vector<vector<int>> adj[n];
29         for(auto it : connections) {
30             adj[it[0]].push_back(it[1]);
31             adj[it[1]].push_back(it[0]);
32         }
33     }
34 }
```

◀ ▶ ⏪ ⏩ 20:33 / 23:24

CC ⚙ #

G-55. Bridges in Graph - Using Tarjan's Algorithm of time in and low time



1.60

Description Discussion Solutions Submissions C++ Auto

```
1 class Solution {
2     private int timer = 1;
3     private void dfs(int node, int parent, int[] vis,
4                      ArrayList<ArrayList<Integer>> adj, int tin[], int low[],
5                      List<List<Integer>> bridges) {
6         vis[node] = 1;
7         tin[node] = low[node] = timer;
8         timer++;
9         for(Integer it: adj.get(node)) {
10             if(it == parent) continue;
11             if(vis[it] == 0) {
12                 dfs(it, node, vis, adj, tin, low, bridges);
13                 low[node] = Math.min(low[node], low[it]);
14                 // node --- it
15                 if(low[it] > tin[node]) {
16                     bridges.add(Arrays.asList(it, node));
17                 }
18             } else {
19                 low[node] = Math.min(low[node], low[it]);
20             }
21         }
22     }
23     public List<List<Integer>> criticalConnections(int n,
24             List<List<Integer>> connections) {
25         ArrayList<ArrayList<Integer>> adj =
26             new ArrayList<ArrayList<Integer>>();
27         for(int i = 0;i<n;i++) {
28             adj.add(new ArrayList<Integer>());
29         }
30         for(List<Integer> it : connections) {
31             int u = it.get(0); int v = it.get(1);
32             adj.get(u).add(v);
33             adj.get(v).add(u);
34         }
35         int[] vis = new int[n];
36         int[] tin = new int[n];
37         int[] low = new int[n];
38         List<List<Integer>> bridges = new ArrayList<>();
39         dfs(0, -1, vis, adj, tin, low, bridges);
40         return bridges;
41     }
42 }
```

Input: n = 4, connections = [[0,1],[1,2],

```
1 class Solution {
2     private:
3     int timer = 1;
4     private:
5     void dfs(int node, int parent, vector<int> &vis,
6              vector<vector<int>> adj[], int tin[], int low[], vector<vector<int>> &bridges)
7     {
8         vis[node] = 1;
9         tin[node] = low[node] = timer;
10        timer++;
11        for(auto it: adj[node]) {
12            if(it == parent) continue;
13            if(vis[it] == 0) {
14                dfs(it, node, vis, adj, tin, low, bridges);
15                low[node] = min(low[node], low[it]);
16                // node --- it
17                if(low[it] > tin[node]) {
18                    bridges.push_back({it, node});
19                }
20            } else {
21                low[node] = min(low[node], low[it]);
22            }
23        }
24    }
25 public:
26     vector<vector<int>> criticalConnections(int n,
27             vector<vector<int>> &connections) {
28         vector<vector<int>> adj[n];
29         for(auto it : connections) {
30             adj[it[0]].push_back(it[1]);
31             adj[it[1]].push_back(it[0]);
32         }
33     }
34 }
```



21:27 / 23:24



G-55. Bridges in Graph - Using Tarjan's Algorithm of time in and low time



```
1.60 Description Discussion Solutions Submissions C++ Auto
1 class Solution {
2     private int timer = 1;
3     private void dfs(int node, int parent, int[] vis,
4             ArrayList<ArrayList<Integer>> adj, int tin[], int low[],
5             List<List<Integer>> bridges) {
6         vis[node] = 1;
7         tin[node] = low[node] = timer;
8         timer++;
9         for(Integer it: adj.get(node)) {
10             if(it == parent) continue;
11             if(vis[it] == 0) {
12                 dfs(it, node, vis, adj, tin, low, bridges);
13                 low[node] = Math.min(low[node], low[it]);
14                 // node --- it
15                 if(low[it] > tin[node]) {
16                     bridges.add(Arrays.asList(it, node));
17                 }
18             } else {
19                 low[node] = Math.min(low[node], low[it]);
20             }
21         }
22     }
23     public List<List<Integer>> criticalConnections(int n,
24             List<List<Integer>> connections) {
25         ArrayList<ArrayList<Integer>> adj =
26             new ArrayList<ArrayList<Integer>>();
27         for(int i = 0;i<n;i++) {
28             adj.add(new ArrayList<Integer>());
29         }
30         for(List<Integer> it : connections) {
31             int u = it.get(0); int v = it.get(1);
32             adj.get(u).add(v);
33             adj.get(v).add(u);
34         }
35         int[] vis = new int[n];
36         int[] tin = new int[n];
37         int[] low = new int[n];
38         List<List<Integer>> bridges = new ArrayList<List<Integer>>();
39         dfs(0, -1, vis, adj, tin, low, bridges);
40         return bridges;
41     }
42 }
43 
```

tin[node] = low[node] = timer;
timer++;
for(auto it: adj[node]) {
 if(it == parent) continue;
 if(vis[it] == 0) {
 dfs(it, node, vis, adj, tin, low, bridges);
 low[node] = min(low[node], low[it]);
 // node --- it
 if(low[it] > tin[node]) {
 bridges.push_back({it, node});
 }
 } else {
 low[node] = min(low[node], low[it]);
 }
}
}
public:
vector<vector<int>> criticalConnections(int n,
vector<vector<int>>& connections) {
 vector<int> adj[n];
 for(auto it : connections) {
 adj[it[0]].push_back(it[1]);
 adj[it[1]].push_back(it[0]);
 }
 vector<int> vis(n, 0);
 int tin[n];
 int low[n];
 vector<vector<int>> bridges;
 dfs(0, -1, vis, adj, tin, low, bridges);
 return bridges; |

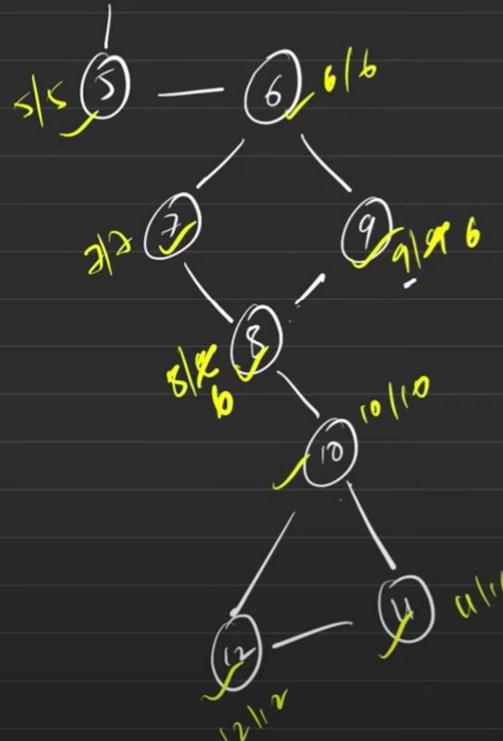
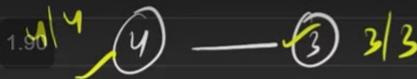
Input: n = 4, connections = [[0,1],[1,2],

21:52 / 23:24

CC G #

G-55. Bridges in Graph - Using Tarjan's Algorithm of time in and low time

*low[i] → min lowest time insertion
of all adjacent nodes
apart from parent.*

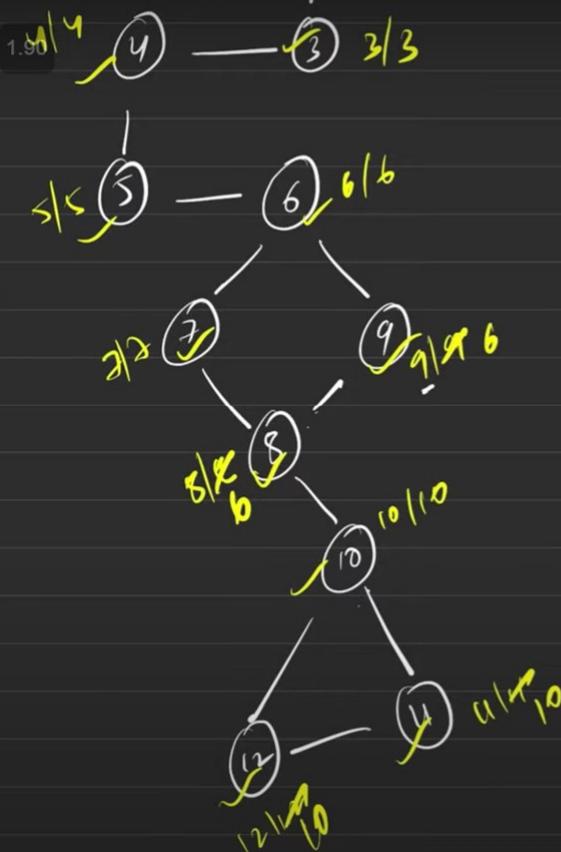


DPS(8)
/ \
DPS(9)



G-55. Bridges in Graph - Using Tarjan's Algorithm of time in and low time

*low[i] → min lowest time insertion
of all adjacent nodes
apart from parent.*



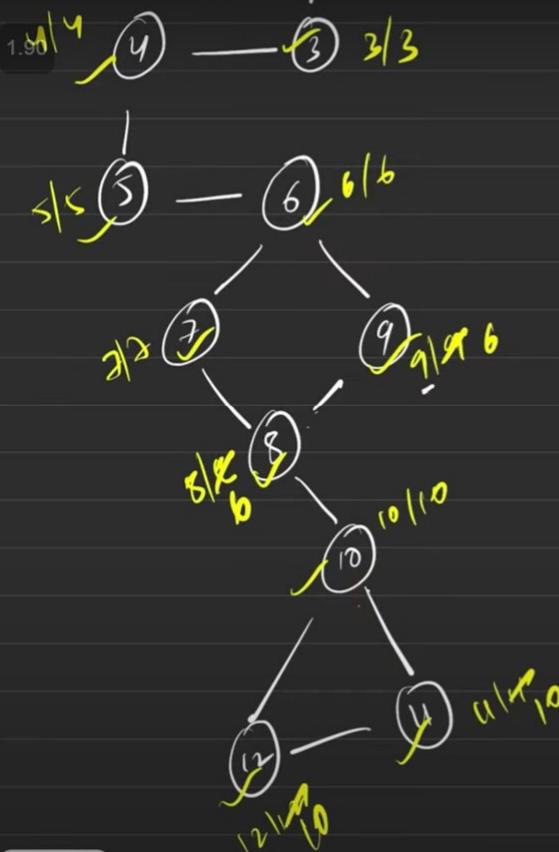
DFS(11)

↗
DFS(12)



G-55. Bridges in Graph - Using Tarjan's Algorithm of time in and low time

*low[i] → min lowest time insertion
of all adjacent nodes
apart from parent.*



DFS(10)
↑
DFS(11)



81 of 85

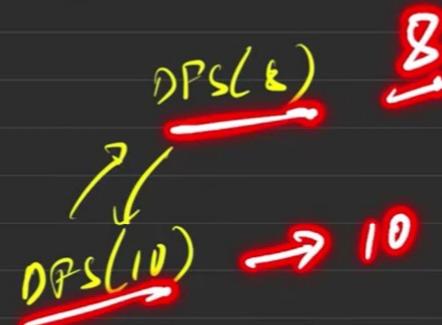
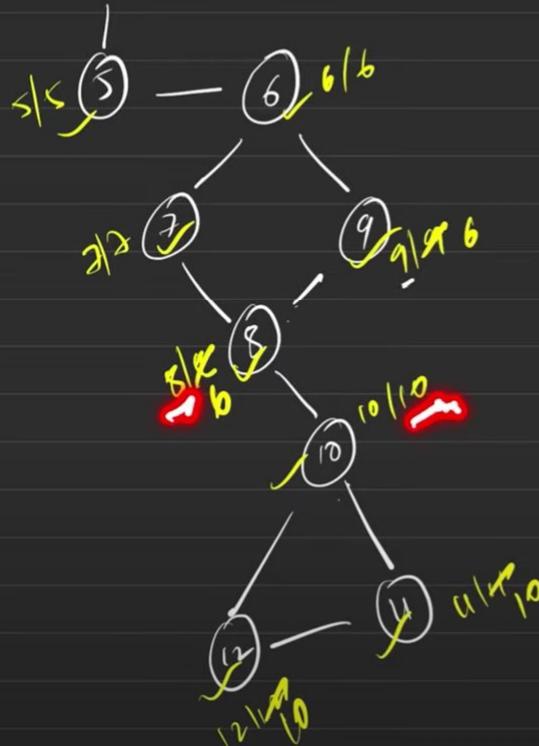
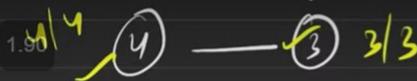


11:01 / 23:24



G-55. Bridges in Graph - Using Tarjan's Algorithm of time in and low time

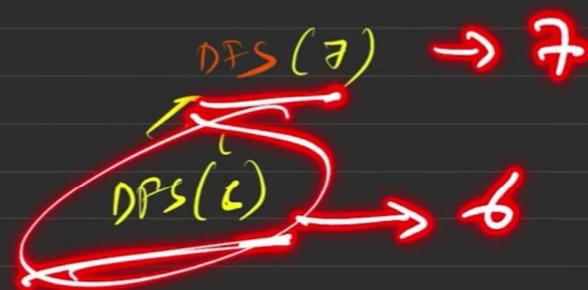
*[low] → min lowest time insertion
of all adjacent nodes
apart from parent.*



G-55. Bridges in Graph - Using Tarjan's Algorithm

low → min lowest time inclusion
of all adjacent nodes
apart from parent.

1.90 4 3/3



G-55. Bridges in Graph - Using Tarjan's Algorithm

1.90 u/l^4 y — v/b $3/3$



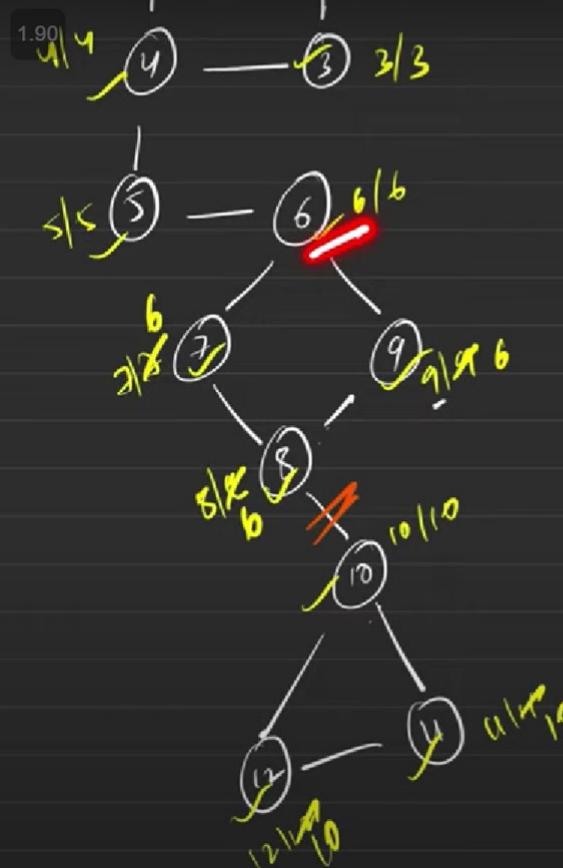
low → min lowest time among
all adjacent nodes
apart from parent.

$\text{DPS}(0) \rightarrow b$

$\text{DPS}(7) \Rightarrow b$



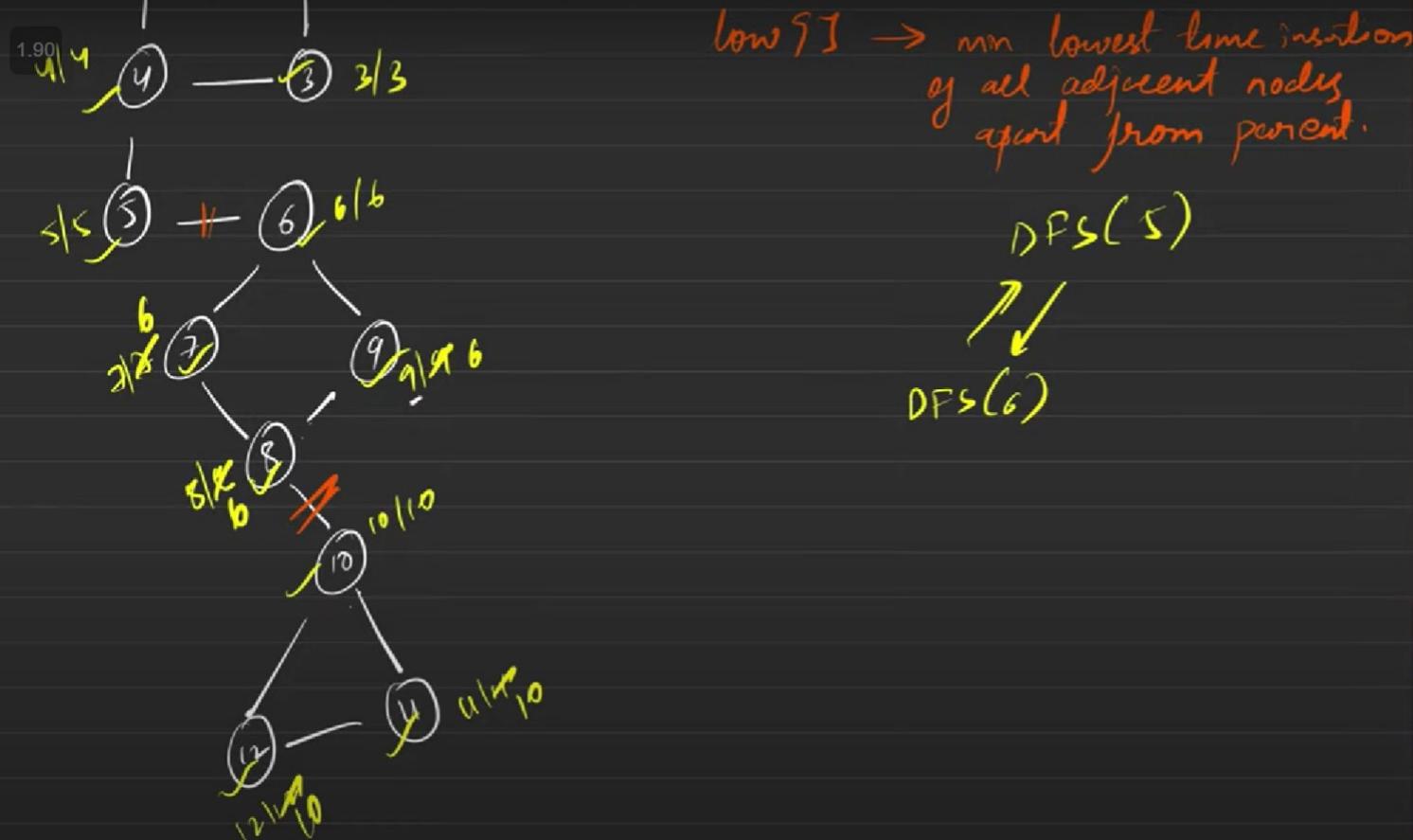
G-55. Bridges in Graph - Using Tarjan's Algorithm of time in and low time



$\text{low}[j] \rightarrow \min$ lowest time insertions
of all adjacent nodes
apart from parent.



G-55. Bridges in Graph - Using Tarjan's Algorithm of time in and low time



G-55. Bridges in Graph - Using Tarjan's Algorithm of time in and low time

