

# Graphs Demux

31 January 2022 13:28

=====

==

## Question List

Lec	Q.No	Q.name	Link	Remarks
1	200	Number of islands	<a href="https://leetcode.com/problems/number-of-islands/">https://leetcode.com/problems/number-of-islands/</a>	
		Clone Graph **		
		Number of provinces	<a href="https://leetcode.com/problems/number-of-provinces/">https://leetcode.com/problems/number-of-provinces/</a>	
	207	Leetcode 207. Course Schedule ::		

## Lec-1

### 1) Leetcode : Clone graph

The screenshot shows the LeetCode problem page for "133. Clone Graph". The URL is [leetcode.com/problems/clone-graph/](https://leetcode.com/problems/clone-graph/). The page includes navigation links for Apps, Building Modern Web, Guide to write an ar..., Workspace, Log in | Innovation..., Problem, Explore, Problems, Interview, Contest, Discuss, Store. Below the navigation is a search bar with the text "LeetCode". The main content area shows the problem title "133. Clone Graph" in bold, followed by "Medium", "4607" solves, "2139" likes, "Add to List", and "Share". A note says "Given a reference of a node in a connected undirected graph." and asks to "Return a deep copy (clone) of the graph." It also states that "Each node in the graph contains a value (int) and a list (List<Node>) of its neighbors." A code snippet for the Node class is provided:

```
class Node {
    public int val;
    public List<Node> neighbors;
}
```

#### Test case format:

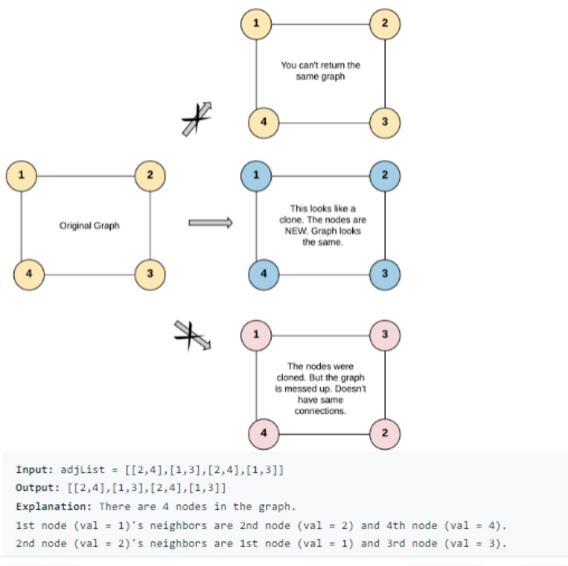
For simplicity, each node's value is the same as the node's index (1-indexed). For example, the first node with `val == 1`, the second node with `val == 2`, and so on. The graph is represented in the test case using an adjacency list.

An adjacency list is a collection of unordered lists used to represent a finite graph. Each list describes the set of neighbors of a node in the graph.

The given node will always be the first node with `val = 1`. You must return the **copy of the given node** as a reference to the cloned graph.

#### Example 1:

#### Example 1:



#### Example 2:

1

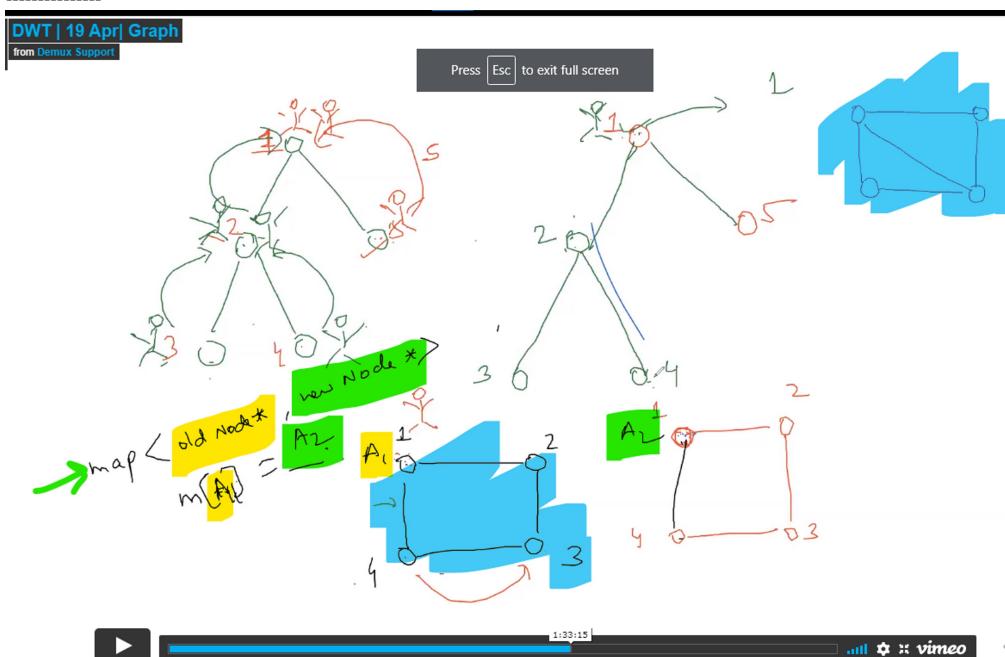
**Input:** adjList = [[]]  
**Output:** [[]]  
**Explanation:** Note that the input contains one empty list. The graph consists of only one node with val = 1 and it does not have any neighbors.

#### Example 3:

**Input:** adjList = []  
**Output:** []  
**Explanation:** This an empty graph, it does not have any nodes.

#### Constraints:

- The number of nodes in the graph is in the range [0, 100].
- $1 \leq \text{Node.val} \leq 100$
- $\text{Node.val}$  is unique for each node.
- There are no repeated edges and no self-loops in the graph.
- The Graph is connected and all nodes can be visited starting from the given node.



Clone will have issue in case of cyclic graphs if we just do normal dfs approach, we

have

## Sir ka soln::

The screenshot shows a LeetCode problem titled "DWT | 19 Apr | Graph". The code is written in C++ and implements a depth-first search (DFS) to clone a graph. The class `Node` has a value and a list of neighbors. The `Solution` class contains a `dfs` function that creates a new node for each visited node and adds it to a map. It then explores the neighbors of each node. The `cloneGraph` function returns a reference to the cloned node. The code includes comments explaining the logic.

```
10     }
11     Node(int _val) {
12         val = _val;
13         neighbors = _neighbors;
14     }
15     Node(int _val, vector<Node*> _neighbors) {
16         val = _val;
17         neighbors = _neighbors;
18     }
19 }
20 */
21
22 class Solution {
23 public:
24     void dfs(Node* start, unordered_map<Node*, Node*>& m) {
25         Node* new_node = new Node(start->val);
26         m[start] = new_node;
27
28         // Visit all the neighbors.
29         for (auto& nbr: start->neighbors) {
30             if (m.find(nbr) != m.end()) {
31                 new_node->neighbors.push_back(m[nbr]);
32             }
33             else {
34                 dfs(nbr, m);
35                 new_node->neighbors.push_back(m[nbr]);
36             }
37         }
38     }
39     Node* cloneGraph(Node* node) {
40         if (!node) return NULL;
41
42         unordered_map<Node*, Node*> m;
43
44         dfs(node, m);
45
46         return m[node];
47     }
48 }
```

Test case format:  
For simplicity sake, each node's value is the same as the node's index (1-indexed). For example, the first node with `val = 1`, the second node with `val = 2`, and so on. The graph is represented in the test case using an adjacency list.  
Adjacency list is a collection of unordered lists used to represent a finite graph. Each list describes the set of neighbors of a node in the graph.  
The given node will always be the first node with `val = 1`. You must return the **copy** of the given node as a reference to the cloned graph.

Example 1:

## My soln (same as above but with comments)

This is a copy of the solution provided by the teacher, with additional explanatory comments added to the original code.

```
1+ /*
2+ // Definition for a Node.
3+ class Node {
4+ public:
5+     int val;
6+     vector<Node*> neighbors;
7+     Node() {
8+         val = 0;
9+         neighbors = vector<Node*>();
10    }
11    Node(int _val) {
12        val = _val;
13        neighbors = vector<Node*>();
14    }
15    Node(int _val, vector<Node*> _neighbors) {
16        val = _val;
17        neighbors = _neighbors;
18    }
19 };
20 */
21
22 class Solution {
23 public:
24     void dfs(Node* cur,unordered_map<Node*,Node*>& m){
25         Node* newNode=new Node(cur->val);
26         m[cur]=newNode;//map old node ka address to newNode address
27
28         for(auto& nbr:cur->neighbors){
29             if(m.find(nbr)!=m.end()){
30                 //neighbour has been visited before & space has been allocated to its clone node
31                 newNode->neighbors.push_back(m[nbr]);
32             }
33         }
34     }
35
36     class Solution {
37 public:
38     void dfs(Node* cur,unordered_map<Node*,Node*>& m){
39         Node* newNode=new Node(cur->val);
40         m[cur]=newNode;//map old node ka address to newNode address
41
42         for(auto& nbr:cur->neighbors){
43             if(m.find(nbr)!=m.end()){
44                 //neighbour has been visited before & space has been allocated to its clone node
45                 newNode->neighbors.push_back(m[nbr]);
46             }
47             else{
48                 dfs(nbr,m);// explore that neighbour if its not visited
49                 //as only after visiting it once is the clone of that node is formed
50                 newNode->neighbors.push_back(m[nbr]);//then push that cloned node to newNode's neighbour vector
51             }
52         }
53     }
54
55     Node* cloneGraph(Node* node) {
56         if(node==nullptr) return nullptr;
57         unordered_map<Node*,Node*> m; //oldNode_address, newNode_address
58         // this map also tells us if a node is visited or not
59         dfs(node,m);
60
61         return m[node];//return the clone node for the given node
62     }
63 }
```

2)Leetcode : Number of provinces

leetcode.com/problems/number-of-provinces/submissions/

Apps Building Modern Web Guide to write an ar... Workspace Log in | Innovation... Problem setting gui... Apply | CodeChef cs50.harvard.edu Algo Muse

LeetCode Premium

Description Solution Discuss (999+) Submissions

**Success** Details >

Runtime: 43 ms, faster than 29.53% of C++ online submissions for Number of Provinces.

Memory Usage: 13.8 MB, less than 62.09% of C++ online submissions for Number of Provinces.

Next challenges:

- Number of Connected Components in an Undirected Graph
- Robot Return to Origin Sentence Similarity Sentence Similarity II
- The Earliest Moment When Everyone Become Friends
- Detonate the Maximum Bombs

Show off your acceptance: [Facebook](#) [Twitter](#) [LinkedIn](#)

Time Submitted	Status	Runtime	Memory	Language
02/02/2022 20:10	Accepted	43 ms	13.8 MB	cpp

```

1+ class Solution {
2 public:
3     void dfs(int node,vector<int>& visited,vector<vector<int>> &isConnected){
4         visited[node]=1;/mark visited
5         int n=visited.size();
6         for(int i=0;i<n;i++){
7             if(isConnected[node][i]==1 && i!=node && !visited[i]){
8                 // connected node & i!=node & not visited[i] then perform dfs on it
9                 dfs(i,visited,isConnected);
10            }
11        }
12    }
13    int findCircleNum(vector<vector<int>>& isConnected) {
14        int n=isConnected.size(),i,provinceCount=0;
15        //visited k live array
16        vector<int> visited(n,0);
17
18        for(i=0;i<n;i++){
19            if(!visited[i]){
20                dfs(i,visited,isConnected);
21                provinceCount++;
22            }
23        }
24    }
25
26    };

```

### 3) Leetcode : 547. Number of Provinces ::

<https://leetcode.com/problems/number-of-provinces/>

#### 547. Number of Provinces

Medium 4568 220 Add to List Share

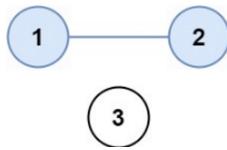
There are  $n$  cities. Some of them are connected, while some are not. If city  $a$  is connected directly with city  $b$ , and city  $b$  is connected directly with city  $c$ , then city  $a$  is connected indirectly with city  $c$ .

A **province** is a group of directly or indirectly connected cities and no other cities outside of the group.

You are given an  $n \times n$  matrix `isConnected` where `isConnected[i][j] = 1` if the  $i^{th}$  city and the  $j^{th}$  city are directly connected, and `isConnected[i][j] = 0` otherwise.

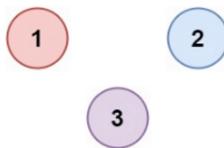
Return the total number of **provinces**.

#### Example 1:



Input: `isConnected = [[1,1,0],[1,1,0],[0,0,1]]`  
Output: 2

#### Example 2:



Input: `isConnected = [[1,0,0],[0,1,0],[0,0,1]]`  
Output: 3

#### Constraints:

- $1 \leq n \leq 200$
- $n == \text{isConnected.length}$
- $n == \text{isConnected}[i].length$
- $\text{isConnected}[i][j]$  is 1 or 0.
- $\text{isConnected}[i][i] == 1$
- $\text{isConnected}[i][j] == \text{isConnected}[j][i]$

```

1 class Solution {
2     public:
3         void dfs(int node, vector<int>& visited, vector<vector<int>> &isConnected) {
4             visited[node] = 1; // mark visited
5             int n = visited.size();
6             for (int i = 0; i < n; i++) {
7                 if (isConnected[node][i] == 1 && i != node && !visited[i]) {
8                     // connected node & i != node & not visited[i] then perform dfs on it
9                     dfs(i, visited, isConnected);
10                }
11            }
12        }
13        int findCircleNum(vector<vector<int>>& isConnected) {
14            int n = isConnected.size(), i, provinceCount = 0;
15            // visited k lie array
16            vector<int> visited(n, 0);
17            for (i = 0; i < n; i++) {
18                if (!visited[i]) {
19                    dfs(i, visited, isConnected);
20                    provinceCount++;
21                }
22            }
23        }
24        return provinceCount;
25    }
26 }
```

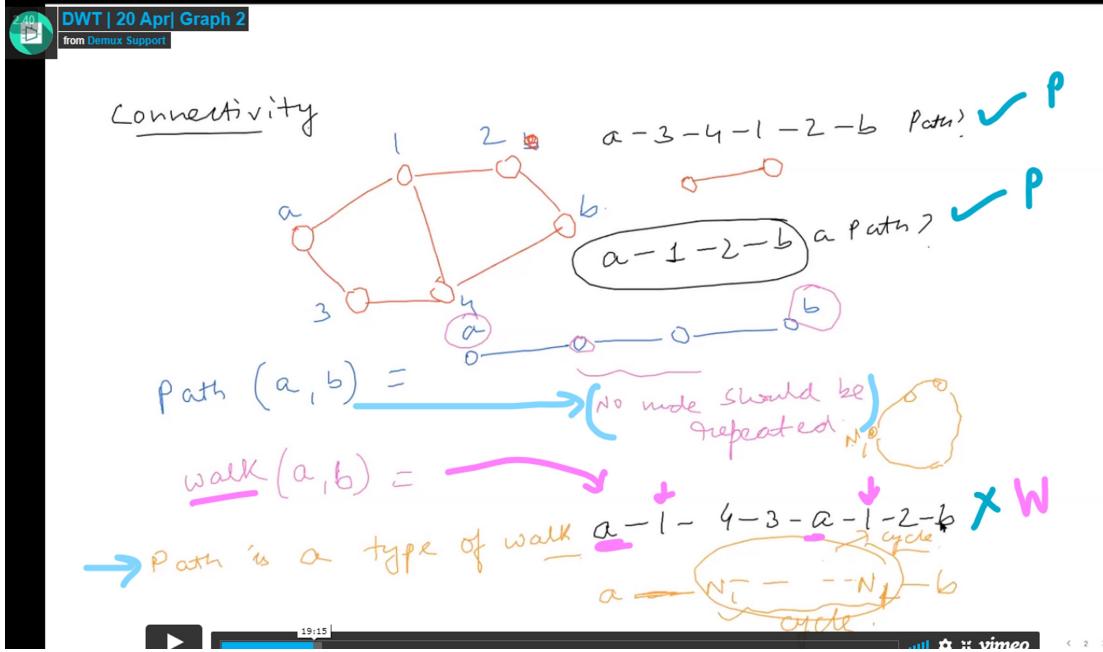
-----  
4) Leetcode :

=====

=====

=====

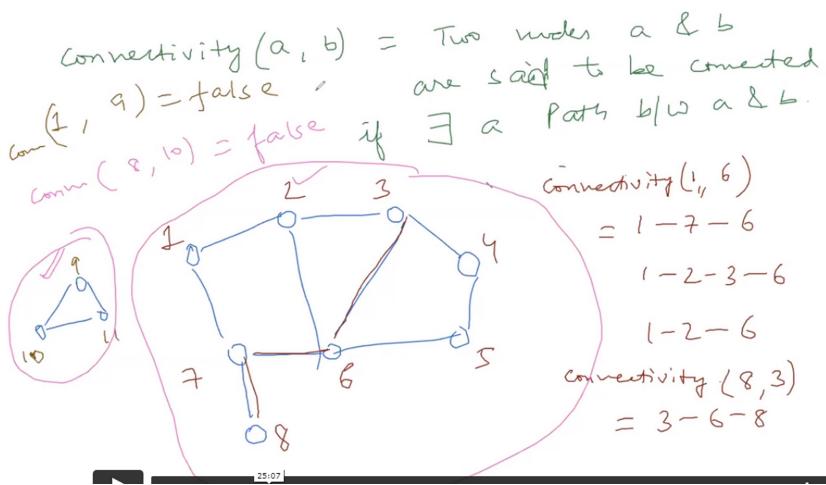
## Lec-2



- Path(a,b) - no nodes are repeated => path contains no cycles
- Walk(a,b) - any sequence of nodes that start at a and end at b. Walks can contain cycles

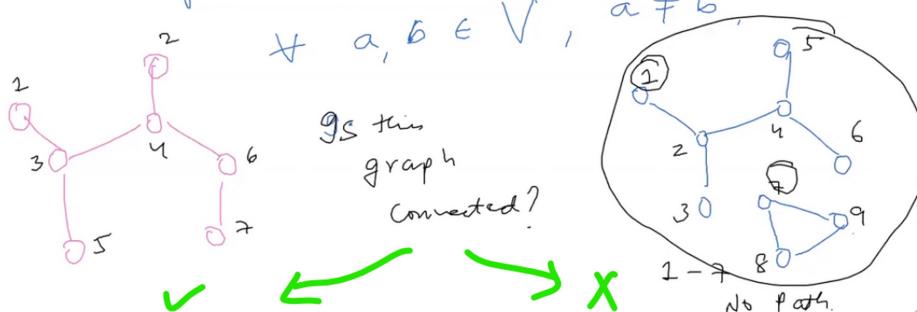
Path is a special type of walk!!

- Connectivity (a,b) : if there exists a path bw a & b



connectivity of a graph

A graph  $G$  is said to be connected if  $\exists$  a path bw a & b &  $a, b \in V$ ,  $a \neq b$ .

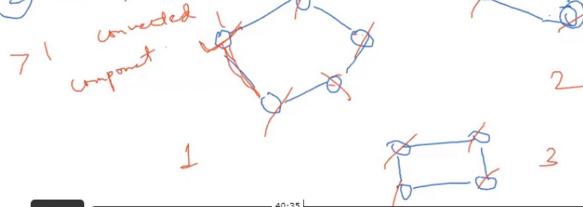


DWT | 20 Apr | Graph 2  
from Demux Support

① Given a graph, find out whether it is connected or not?  
1 connected component of a graph is connected then a traversal of the graph starting from any node will visit all the nodes of the graph.



② Components such that each component is connected.



Modelling graph problems

→ Explicitly given graph  
No graph explicitly given

① Define your graph for the problem → Trivial - Non-trivial

② Bucket the problem one of existing Problem → Connectivity

leetcode.com/problems/number-of-islands/

Description Solution Discuss (99...) Submissions C++ Autocomplete

**200. Number of Islands**

Medium 8213 241 Add to List Share

Given an  $m \times n$  2D binary grid  $\text{grid}$  which represents a map of '1' s (land) and '0' s (water), return the number of islands.

An island is surrounded by water and is formed by connecting adjacent lands horizontally or vertically. You may assume all four edges of the grid are all surrounded by water.

**Example 1:**

```

Input: grid = [
    ["1","1","1","1","0"],
    ["1","1","0","1","0"],
    ["1","1","0","0","0"],
    ["0","0","0","0","0"]
]
Output: 1

```

**Example 2:**

```

Input: grid = [
    ["1","1","0","0","0"],
    ["1","1","0","0","0"],
    ["0","0","1","0","0"],
    ["0","0","0","1","1"]
]

```

=====

(premium) Leetcode 694. Number of distinct islands ::

```

1+ class Solution {
2 public:
3     vector<vector<int>> nbrs = {{1, 0}, {0, 1}, {-1, 0}, {0, -1}};
4     void dfs(vector<vector<char>& grid, int i, int j, vector<vector<bool>& visited) {
5         visited[i][j] = true;
6
7         // Perform dfs on the neighbors.
8         for (auto& nbr: nbrs) {
9             int x = i + nbr[0];
10            int y = j + nbr[1];
11
12            // (x, y) is adjacent to (i, j)
13            if (x < 0 || y < 0 || x >= grid.size() || y >= grid[0].size())
14                || visited[x][y] || grid[x][y] == '0')
15                continue;
16
17            dfs(grid, x, y, visited);
18        }
19    }
20    int numIslands(vector<vector<char>& grid) {
21        int res = 0;
22        vector<vector<bool>> visited(grid.size(), vector<bool>(grid[0].size(), false));
23
24        for (int i = 0; i < grid.size(); i++) {
25            for (int j = 0; j < grid[0].size(); j++) {
26                if (grid[i][j] == '1' && !visited[i][j]) {
27                    dfs(grid, i, j, visited);
28                    res++;
29                }
30            }
31        }
32
33        return res;
34    }
35}

```

DWT | 20 Apr | Graph 2 | from Demux Support

LeetCode Explore Problems Mock Contest Discuss Store

Description Solution (391) Discuss (391) Submissions

C++ Autocomplete

```

1+ class Solution {
2 public:
3     int numDistinctIslands(vector<vector<int>>& grid) {
4
5     }
6 };

```

**694. Number of Distinct Islands**

Medium 1294 76 Add to List Share

Given a non-empty 2D array `grid` of 0's and 1's, an **island** is a group of 1's (representing land) connected 4-directionally (horizontal or vertical.) You may assume all four edges of the grid are surrounded by water.

Count the number of **distinct** islands. An island is considered to be the same as another if and only if one island can be translated (and not rotated or reflected) to equal the other.

**Example 1:**

```

11000
11000
00011
00011

```

Given the above grid map, return 1.

**Example 2:**

```

11011
10000
00001
11011

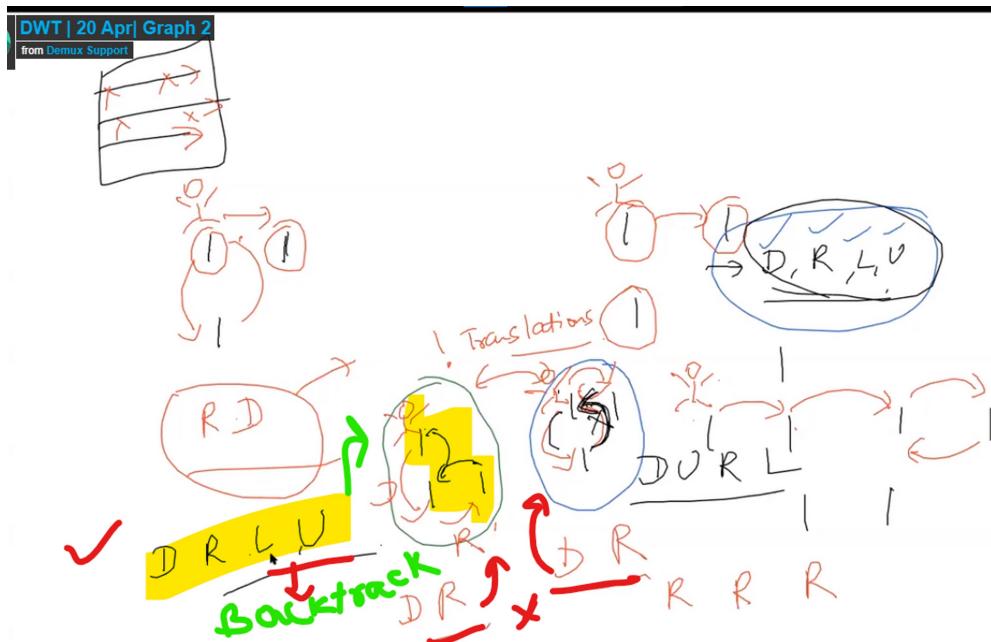
```

Given the above grid map, return 3.

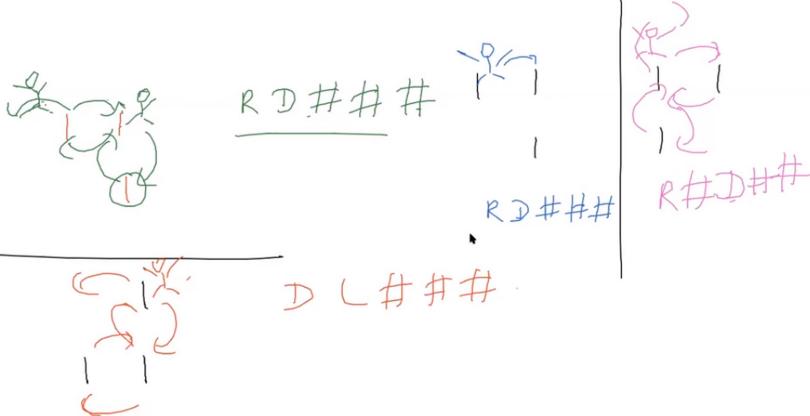
Notice that:

Your previous code was restored from your local storage. [Reset to default](#)

11 1:43:14 Problems Vimeo Submit



- To store the islands we are storing the direction of visiting/traversing the island, including the backtracking wala path add '#' to act as divider/ distinguish between different maps of same string but different structure(this is important)



- Reason to add hash can be understood from above examples

Sir's soln:

#### 694. Number of Distinct Islands

Medium 1294 76 Add to List Share

Given a non-empty 2D array `grid` of 0's and 1's, an **island** is a group of 1's (representing land) connected 4-directionally (horizontal or vertical.) You may assume all four edges of the grid are surrounded by water.

Count the number of **distinct** islands. An island is considered to be the same as another if and only if one island can be translated (and not rotated or reflected) to equal the other.

**Example 1:**

```
11000
11000
00011
00011
```

Given the above grid map, return 1.

**Example 2:**

```
11011
10000
00001
11011
```

```
1+ class Solution {
2+ public:
3+     vector<vector<int>> nbrs = {{1, 0}, {0, 1}, {-1, 0}, {0, -1}};
4+     vector<char> dirs = {'R', 'U', 'L', 'D'};
5+     void dfs(vector<vector<int>>& grid, int i, int j, vector<vector<bool>>& visited, string& island) {
6+         visited[i][j] = true;
7+
8+         for (int k = 0; k < nbrs.size(); k++) {
9+             int x = i + nbrs[k][0];
10+            int y = j + nbrs[k][1];
11+
12+            if (x < 0 || y < 0 || x >= grid.size() || y >= grid[0].size() || visited[x][y]
13+                || grid[x][y] == 0)
14+                continue;
15+
16+            island += dirs[k];
17+            dfs(grid, x, y, visited, island);
18+            island += '#';
19+        }
20+    }
21+    int numDistinctIslands(vector<vector<int>>& grid) {
22+        int m = grid.size(), n = grid[0].size(), i, j;
23+        unordered_set<string> islands;
24+        vector<vector<bool>> visited(m, vector<bool>(n, false));
25+
26+        for(i = 0; i < m; i++) {
27+            for(j = 0; j < n; j++) {
28+                if (grid[i][j] == 1 && !visited[i][j]) {
29+                    string island_pattern = "";
30+                    dfs(grid, i, j, visited, island_pattern);
31+                    if (islands.find(island_pattern) == islands.end())
32+                        islands.insert(island_pattern);
33+                }
34+            }
35+        }
36+    }
37+}
```

-----

#### 694. Number of Distinct Islands

Medium 1294 76 Add to List Share

Given a non-empty 2D array `grid` of 0's and 1's, an **island** is a group of 1's (representing land) connected 4-directionally (horizontal or vertical.) You may assume all four edges of the grid are surrounded by water.

Count the number of **distinct** islands. An island is considered to be the same as another if and only if one island can be translated (and not rotated or reflected) to equal the other.

**Example 1:**

```
11000
11000
00011
00011
```

Given the above grid map, return 1.

**Example 2:**

```
11011
10000
00001
11011
```

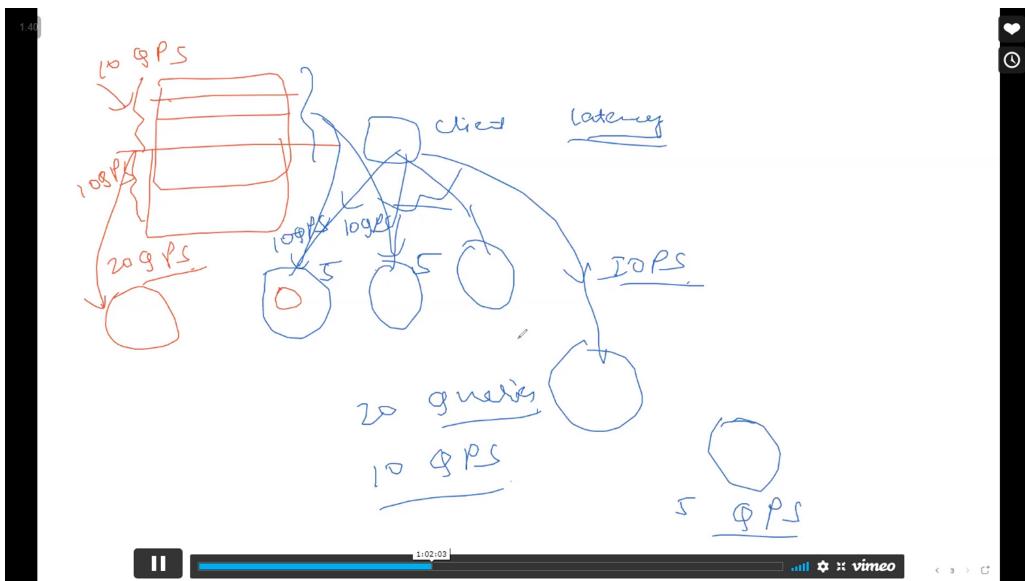
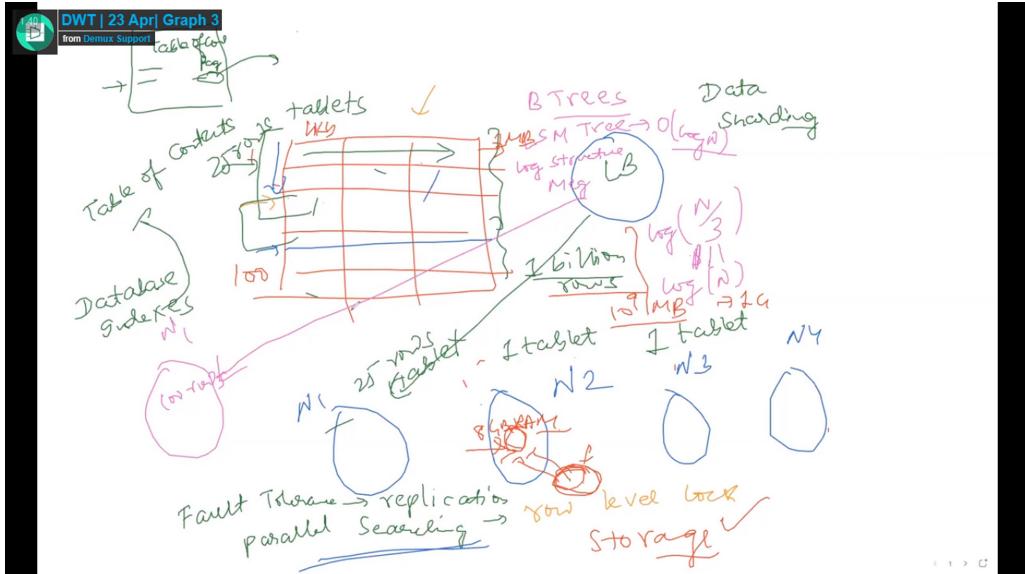
Given the above grid map, return 3.

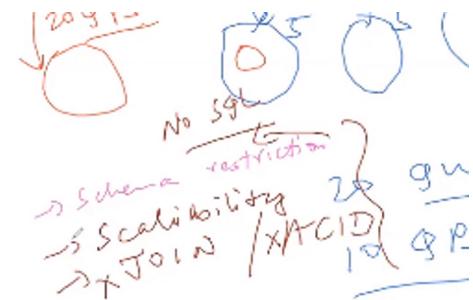
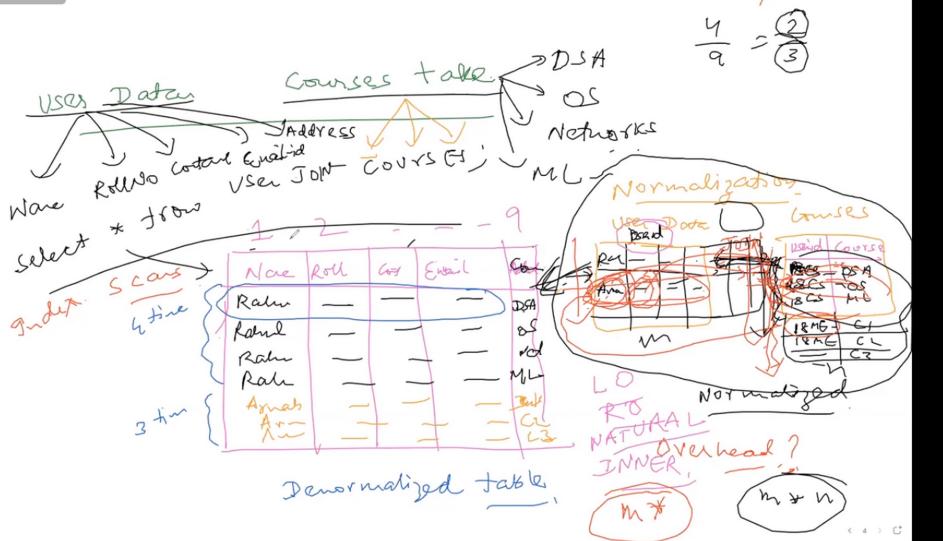
```
4+ vector<char> dirs = {'R', 'U', 'L', 'D'};
5+ void dfs(vector<vector<int>>& grid, int i, int j, vector<vector<bool>>& visited, string& island) {
6+     visited[i][j] = true;
7+
8+     for (int k = 0; k < nbrs.size(); k++) {
9+         int x = i + nbrs[k][0];
10+        int y = j + nbrs[k][1];
11+
12+        if (x < 0 || y < 0 || x >= grid.size() || y >= grid[0].size() || visited[x][y]
13+            || grid[x][y] == 0)
14+            continue;
15+
16+        island += dirs[k];
17+        dfs(grid, x, y, visited, island);
18+        island += '#';
19+    }
20+}
21+int numDistinctIslands(vector<vector<int>>& grid) {
22+    int m = grid.size(), n = grid[0].size(), i, j;
23+    unordered_set<string> islands;
24+    vector<vector<bool>> visited(m, vector<bool>(n, false));
25+
26+    for(i = 0; i < m; i++) {
27+        for(j = 0; j < n; j++) {
28+            if (grid[i][j] == 1 && !visited[i][j]) {
29+                string island_pattern = "";
30+                dfs(grid, i, j, visited, island_pattern);
31+                if (islands.find(island_pattern) == islands.end())
32+                    islands.insert(island_pattern);
33+            }
34+        }
35+    }
36+    return (int)islands.size();
37+};
```

Your previous code was restored from your local storage. Reset to default

Brief discussion on Load balancers, data sharding and its reasons:

- Fault tolerance -> replication of data to avoid data loss if one server gets destroyed
- Parallel searching : since indexes improve searching time
- Storage - by dividing data into blocks/tablets and storing them into different nodes, we are also increasing the cached memory for each node as RAM is limited , hence faster accesss to most frequent data
- Reduced Latency : if each node has power to process 5 QPS(queries per second) & we have 4 nodes then in 1 second we can perform 20 QPS as compared to 5QPS if we had 1 node only



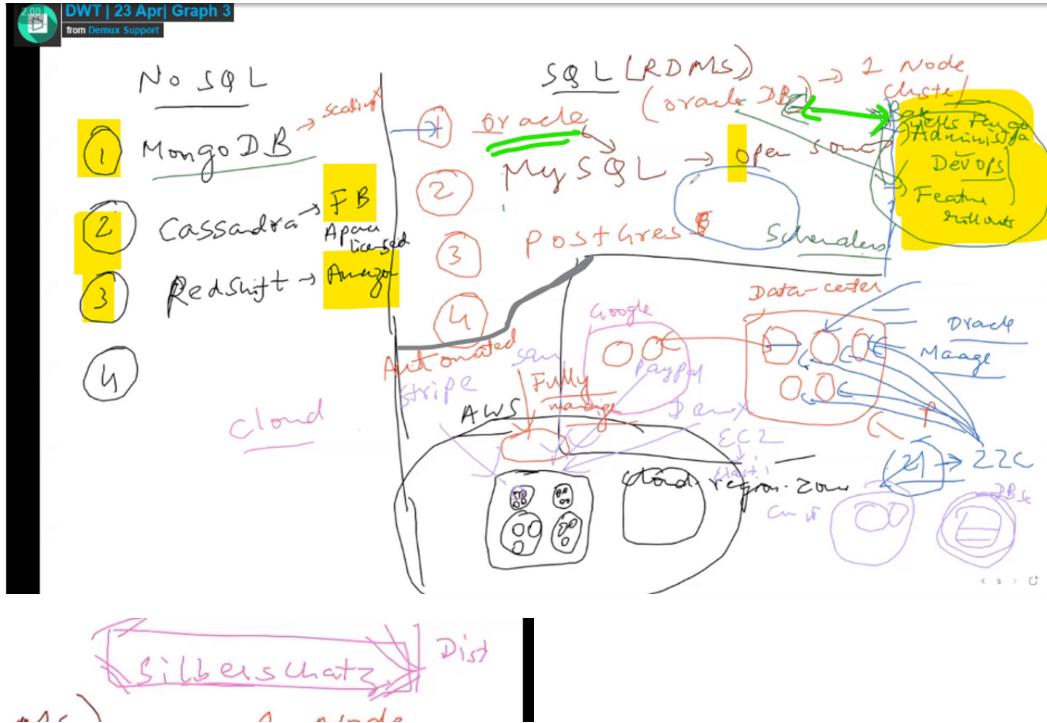


WHY need of JSON arised (when SQL was there)? Basically different btween SQL and NoSQL ::

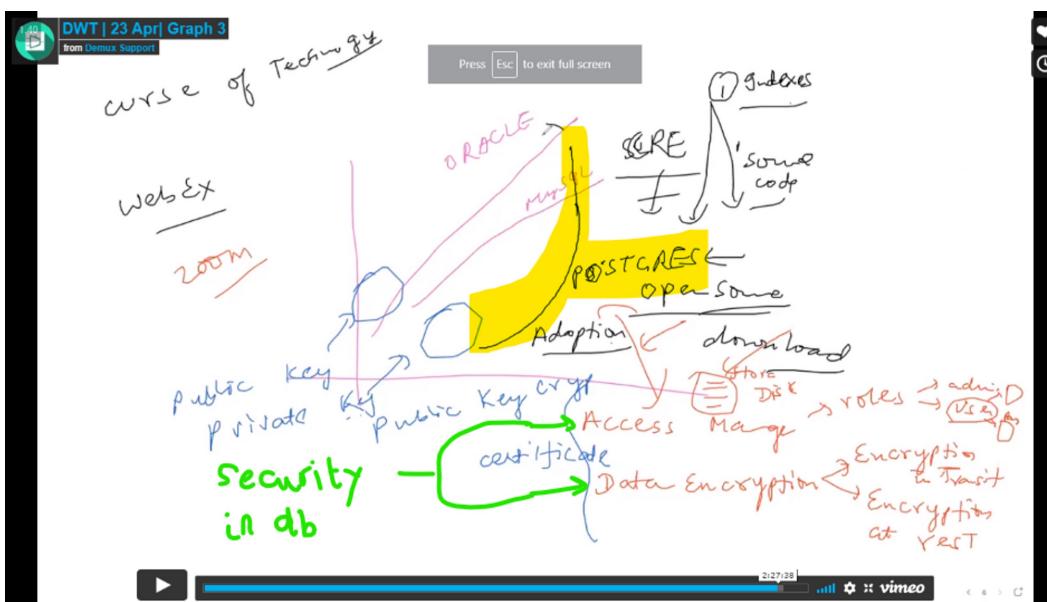
1. Get rid of joins as joins is a very painful/expensive task to perform,(index scan algorithm) as it takes linear time because data is present on the disk and not on the memory, and disks don't have any data structures as such so you have to linearly visit each one of them hence expensive lookup
2. strict schema
3. ACID properties : transactions(database lingo)
  - a. For any system/mission critical features you have to use SQL data bases as it provides ACID properties , eg: banking/payments services, retail & inventory buckets in e-Commerce, NASA / aircraft control , IOT devices

- b. banking Reason: for example in banking if during fund transfer money got out of your account and the receiving end wale ka server crashes at that point then you cannot afford to loose your money ,here ACID properties of SQL save the day
  - c. up to date data access of inventory items
  - d. Concurrent data access in NASA/ Aircraft systems
4. Need to know schema before hand in SQL in order to fill table otherwise you cannot fill the table at all
5. Scalability is very difficult in SQL (eg -joins in distributed SQL database, maintaining ACID properties in large database)

Very hard to guarantee ACID ka properties with a DISTRIBUTED SQL database .



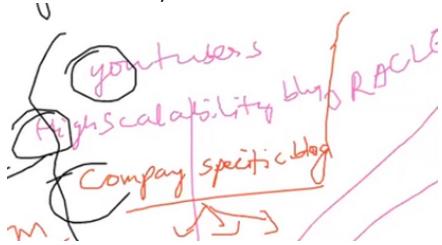
- Postgres is the most growing SQL database at present(see the adoption chart)



Companies are currently trying to make mixture of SQL + NoSQL data base to get best of both worlds, its an active research area as well.

- Yugabyte is also doing that only : its mission is to have one single database for any type of use: like you come say these are the features you want and you get it (like already defined in the model)

Resources to study from:



## LEC 4:

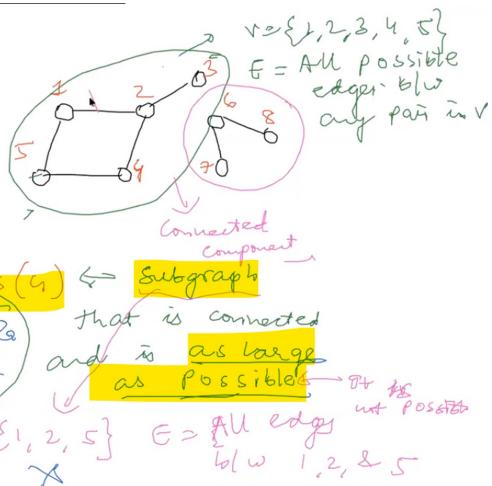
DWT | 24 Apr | Graph 4  
from Demux Support

Graph Cyclicity

Connectivity ( $a, b$ )

Connectivity ( $G$ )

Connected Components ( $G_1$ )



- Cyclicity

- Connectivity

DWT | 24 Apr | Graph 4  
from Demux Support

Connectivity ( $a, b$ )

( $G$ )

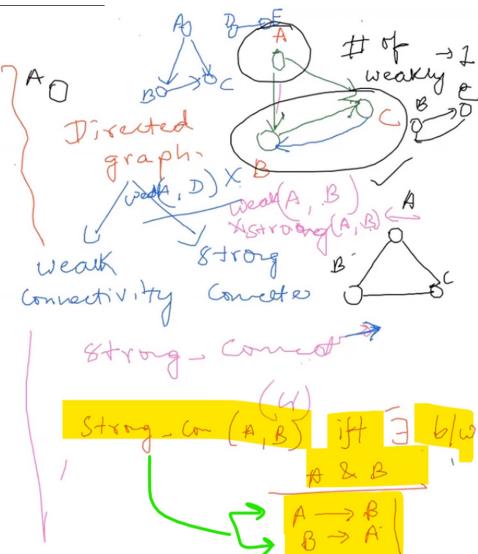
Connected Component ( $G$ )

undirected

weak-connectivity ( $a, b$ )

weak-con ( $G$ )

weak-CC



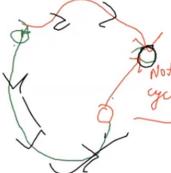
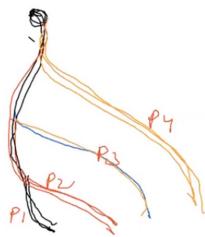
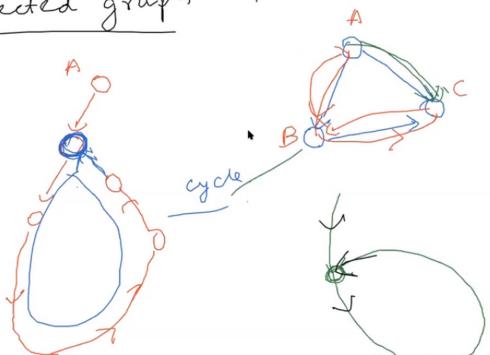
Cyclicity in undirected & directed graph:

- Undirected : if we visit a start node twice from 2 different nodes then we have a cycle
- Directed: if while visiting a path we revisit a already visited node then that

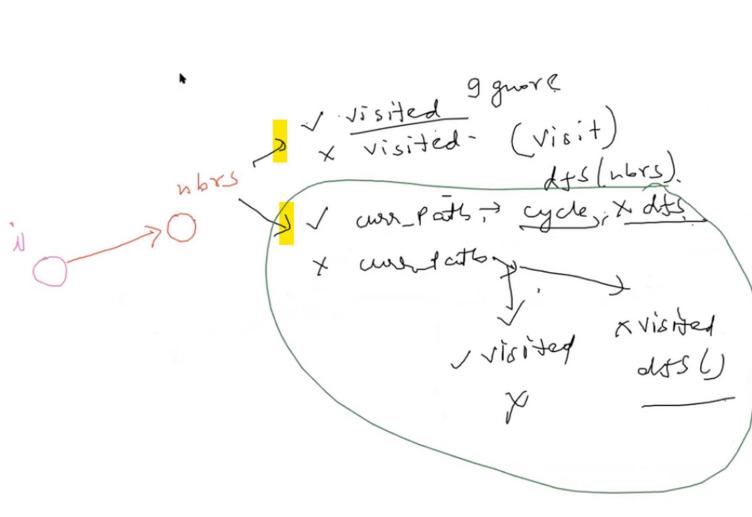
path has a cycle . Note : for this approach we have to mark nodes unvisited while backtracking

Q)  
// Leetcode 207. Course Schedule ::  
DWT | 24 April Graph 4  
from Demux Support

Cyclicity in  
Directed graph



59:13 | vimeo



Graphs  
Demux

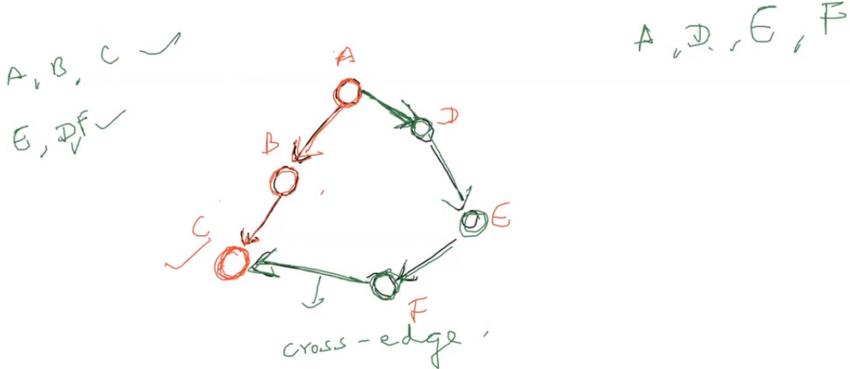
Audio recording started: 17:43 09 February 2022



Graphs  
Demux

Audio recording started: 17:46 09 February 2022

Audio recording started: 17:43 09 February 2022



- The above example is to protect our program from false positives of a cycle being present in the graph.

```

1+ class Solution {
2+ public:
3+     bool containsCycle(vector<vector<int>>& graph, vector<bool>& visited, vector<bool>& curr_path, int i) {
4+
5+         bool t = false;
6+         for (auto& nbrs: graph[i]) {
7+             if (curr_path[nbrs] == true)
8+                 t = true; I
9+
10+            if (!visited[nbrs]) {
11+                curr_path[nbrs] = true;
12+                visited[nbrs] = true;
13+                t = t || containsCycle(graph, visited, curr_path, nbrs);
14+                curr_path[nbrs] = false;
15+            }
16+
17+        }
18+
19+        return t;
20+    }
21+    bool canFinish(int numCourses, vector<vector<int>>& prerequisites) {
22+        // Construct the graph.
23+        // Adjacency list.
24+        vector<vector<int>> graph(numCourses);
25+
26+        for (auto& pre: prerequisites) {
27+            graph[pre[1]].push_back(pre[0]);
28+
29+            // Find out if this graph contains a cycle.
30+            vector<bool> visited(numCourses, false);
31+            vector<bool> curr_path(numCourses, false);
32+            bool t = false;
33+
34+            for (int i = 0; i < numCourses; i++) {
35+                if (!visited[i]) {
36+                    visited[i] = curr_path[i] = true;
37+                    t = t || containsCycle(graph, visited, curr_path, i);
38+                }
39+            }
40+
41+            return t;
42+        }
43+        return true;
44+    }
45+ };

```

**DW1 | 24 Apr | Graph 4**

from Demux Support odc.com/problems/course-schedule/

Press Esc to exit full screen

LeetCode Explore Problems Mock Contest Discuss Store

Description Solution Discuss (99...) Submissions

C++ Autocomplete

```

6+ for (auto& nbrs: graph[i]) {
7+     if (curr_path[nbrs] == true)
8+         t = true;
9+
10+    if (!visited[nbrs]) {
11+        curr_path[nbrs] = true;
12+        visited[nbrs] = true;
13+        t = t || containsCycle(graph, visited, curr_path, nbrs);
14+        curr_path[nbrs] = false;
15+    }
16+
17+ }
18+
19+ return t;
20+
21+ bool canFinish(int numCourses, vector<vector<int>>& prerequisites) {
22+     // Construct the graph.
23+     // Adjacency list.
24+     vector<vector<int>> graph(numCourses);
25+
26+     for (auto& pre: prerequisites) {
27+         graph[pre[1]].push_back(pre[0]);
28+
29+         // Find out if this graph contains a cycle.
30+         vector<bool> visited(numCourses, false);
31+         vector<bool> curr_path(numCourses, false);
32+         bool t = false;
33+
34+         for (int i = 0; i < numCourses; i++) {
35+             if (!visited[i]) {
36+                 visited[i] = curr_path[i] = true;
37+                 t = t || containsCycle(graph, visited, curr_path, i);
38+                 curr_path[i] = false; I
39+             }
40+
41+         }
42+
43+         return !t;
44+     }
45+ };

```

There are a total of numCourses courses you have to take, labeled from 0 to numCourses - 1. You are given an array prerequisites where prerequisites[i] = [ai, bi] indicates that you must take course bi first if you want to take course ai.

- For example, the pair [0, 1], indicates that to take course 0 you have to first take course 1.

Return true if you can finish all courses. Otherwise, return false.

**Example 1:**

**Input:** numCourses = 2, prerequisites = [[1,0]]  
**Output:** true  
**Explanation:** There are a total of 2 courses to take.  
To take course 1 you should have finished course 0. So it is possible.

**Example 2:**

**Input:** numCourses = 2, prerequisites = [[1,0], [0,1]]  
**Output:** false  
**Explanation:** There are a total of 2 courses to take.  
To take course 1 you should have finished course 0, and to take course 0 you should also have

Problems Pick One Prev 4/158 Next Console Contribute I Run Code Submit

leetcode.com/problems/course-schedule/submissions/

Success Details >

Runtime: 27 ms, faster than 55.74% of C++ online submissions for Course Schedule.

Memory Usage: 14.5 MB, less than 36.69% of C++ online submissions for Course Schedule.

Next challenges:

- [Course Schedule II](#)
- [Graph Valid Tree](#)
- [Minimum Height Trees](#)
- [Course Schedule III](#)

Show off your acceptance:

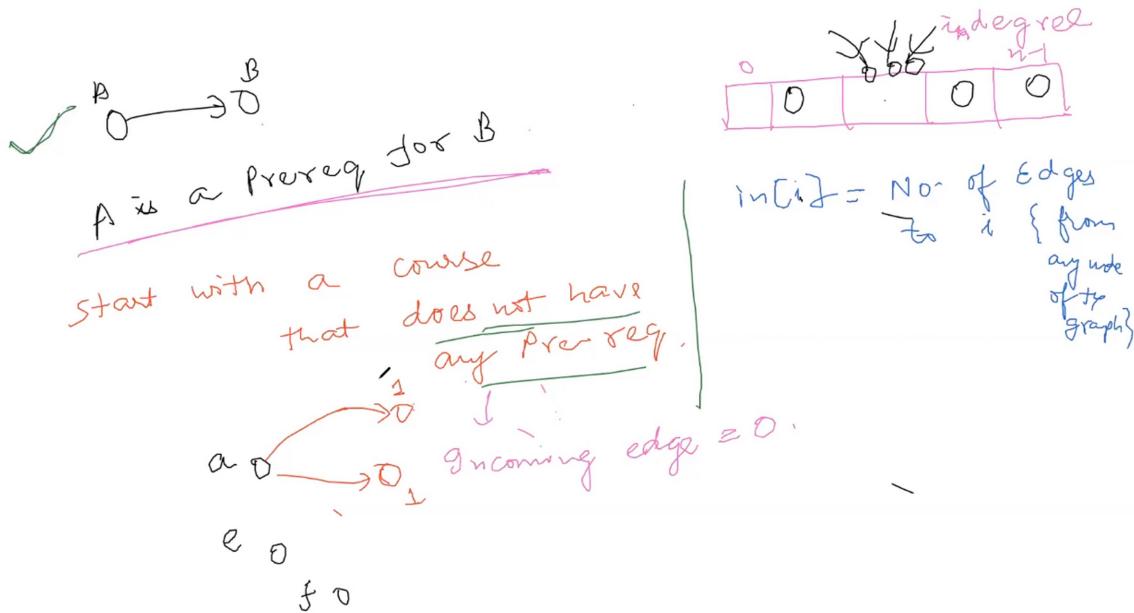
Time Submitted	Status	Runtime	Memory	Language
02/09/2022 21:46	Accepted	27 ms	14.5 MB	cpp
02/09/2022 21:43	Time Limit Exceeded	N/A	N/A	cpp
02/09/2022 20:48	Runtime Error	N/A	N/A	cpp
02/04/2022 20:36	Wrong Answer	N/A	N/A	cnn

```

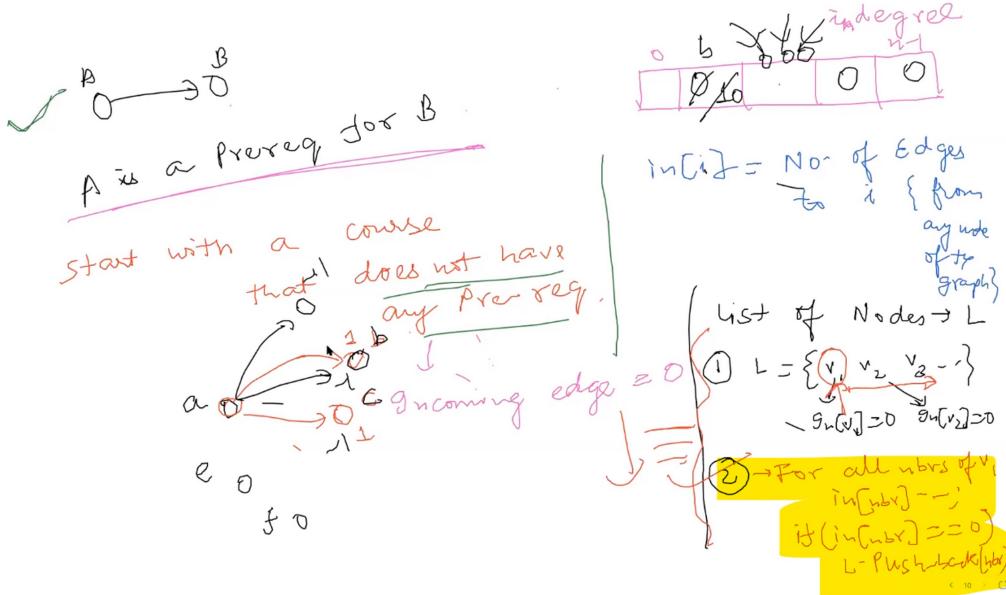
1* class Solution {
2* public:
3*     bool checkCycle(int node,vector<int>& visited,vector<bool> &cur_path,vector<vector<int>> &adj){
4*         if(!visited[node]) {
5*             visited[node]=1;
6*             bool cycle=false;
7*             for(auto& nbr:adj[node]){
8*                 if((cur_path[nbr]==true) return cycle=true;//cycle present
9* 
10*                 if(!visited[nbr])//not visited
11*                     cur_path[nbr]=true;
12*                     visited[nbr]=1;
13*                     cycle= cycle || checkCycle(nbr,visited,cur_path,adj);
14*                     cur_path[nbr]=false;
15*             }
16*             }return cycle;
17*     }
18*     bool canFinish(int numCourses, vector<vector<int>>& prerequisites) {
19*         int i,n=prerequisites.size();
20*         vector<int> visited(numCourses,0);
21*         vector<vector<int>> adj(numCourses);
22*         for(auto& x:prerequisites){
23*             adj[x[0]].push_back(x[1]);
24*         }
25*         for(i=0;i<numCourses;i++){
26*             if(!visited[i]){
27*                 vector<bool> cur_path(numCourses,false);
28*                 visited[i]=1;
29*                 cur_path[i]=true;
30*                 if(checkCycle(i,visited,cur_path,adj)){
31*                     return false;//if cycle present => course cant be finished as no starting point
32*                 }
33*                 cur_path[i] = false;
34*             }
35*         }
36*         return true;//no cycle => course can be finished
37*     }
}

```

Course schedule ii:



Visit in increasing order of indegree



DWT | 24 Apr | Graph 4

from Demux Support [ode.com/problems/course-schedule-II/](https://ode.com/problems/course-schedule-II/)

LeetCode Explore Problems Mock Contest Discuss Store

Description Solution Discuss (99...) Submissions

210. Course Schedule II

Medium 3619 166 Add to List Share

There are a total of  $n$  courses you have to take labelled from 0 to  $n - 1$ .

Some courses may have prerequisites, for example, if  $\text{prerequisites}[i] = [a_i, b_i]$  this means you must take the course  $b_i$  before the course  $a_i$ .

Given the total number of courses  $\text{numCourses}$  and a list of the prerequisite pairs, return the ordering of courses you should take to finish all courses.

If there are many valid answers, return **any** of them. If it is impossible to finish all courses, return **an empty array**.

**Example 1:**

```

Input: numCourses = 2, prerequisites = [[1,0]]
Output: [0,1]
Explanation: There are a total of 2 courses to take. To take course 1 you should have finished course 0. So the correct course order is [0,1].

```

**Example 2:**

```

Input: numCourses = 4, prerequisites = [[1,0], [2,0], [3,1], [3,2]]
Output: [0,1,2,3]
Explanation: There are a total of 4 courses to take. To take course 1 you should have finished course 0. To take course 2 you should have finished course 0. To take course 3 you should have finished courses 1 and 2. So the correct course order is [0,1,2,3].

```

You can also restore code from 4/24/2021 9:37 or Reset to default

2:08:54

```

vector<int> findOrder(int numCourses, vector<vector<int>>& prerequisites) {
    // Populate the graph and the indegree vector.
    vector<vector<int>> graph(numCourses);
    vector<int> in(numCourses, 0);
    vector<int> curr_set;
    int idx = 0;
    vector<int> res;

    for(auto& pre: prerequisites) {
        // Edge From 1 to 0.
        graph[pre[1]].push_back(pre[0]);
        in[pre[0]]++;
    }

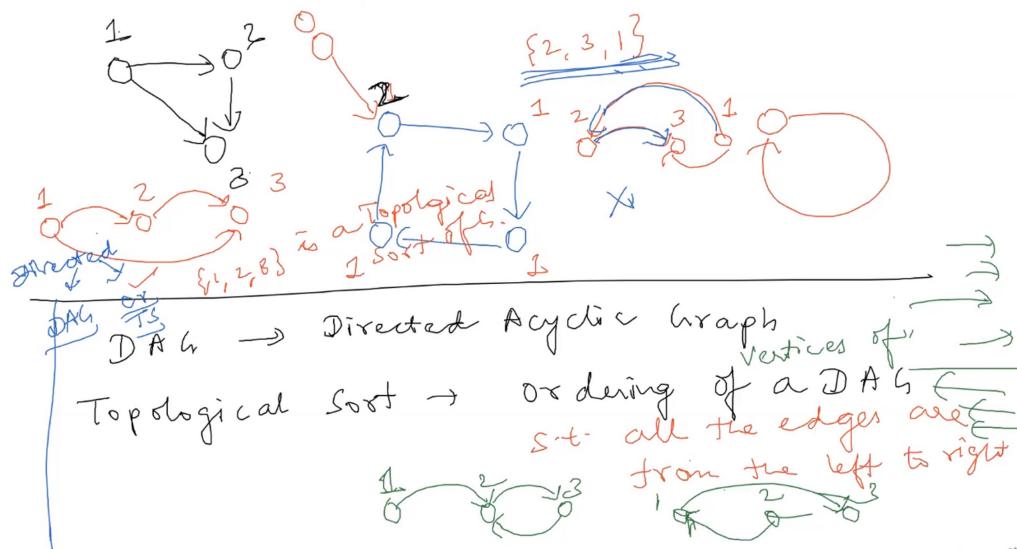
    // Initialize the initial list to traverse.
    for(int i = 0; i < numCourses; i++) {
        if(in[i] == 0)
            curr_set.push_back(i);
    }

    // Traverse this list: I
    while(idx < curr_set.size()) {
        int curr_node = curr_set[idx];
        res.push_back(curr_node);

        // Decrease the indegree by 1 for all the nbrs of idx
        for (auto& nbr: graph[curr_node]) {
            in[nbr]--;
            if (in[nbr] == 0)
                curr_set.push_back(nbr);
        }
        idx++;
    }

    return (int)res.size() == numCourses? res: vector<int>();
}

```

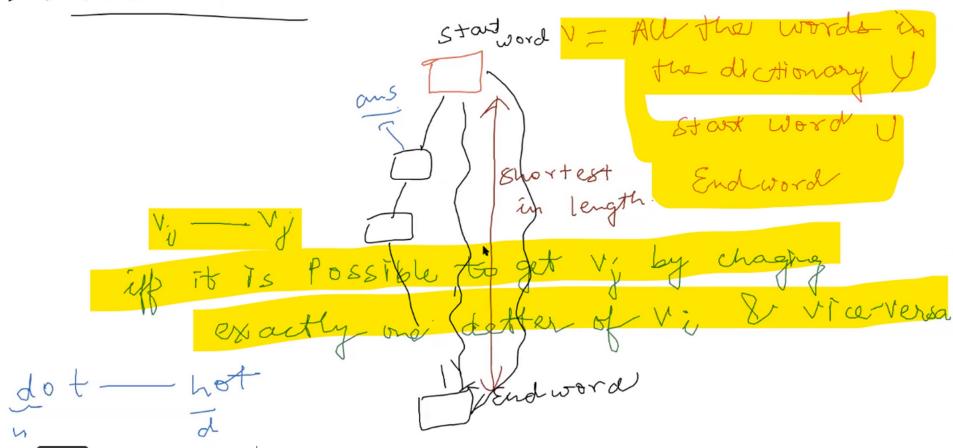


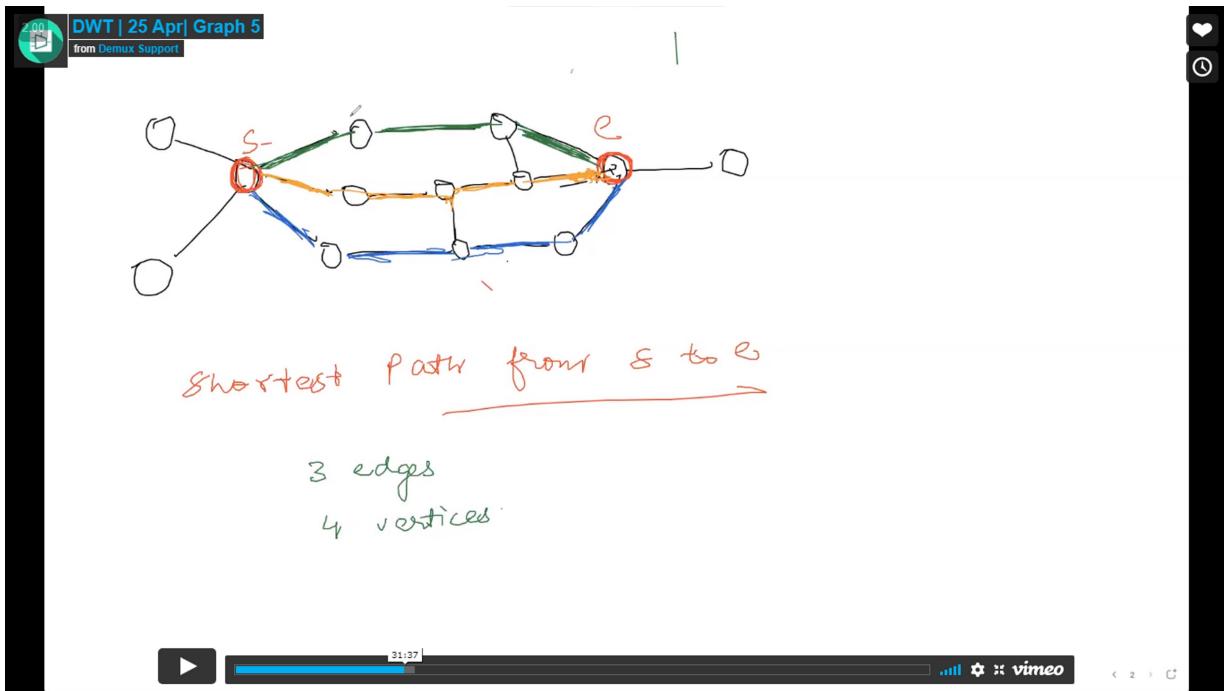
- Topological sort:: ordering of a DAG such that all the edges are in one direction (eg all edges are from either left to right/ all edges are from right to left)
- Eg in above image for

Lec 5:

Q) Leetcode :Word Ladder

### Shortest Path





DWT | 25 Apr | Graph 5  
from Demux Support code.com/problems/word-ladder/

LeetCode Explore Problems Mock Contest Discuss Store Press Esc to exit full screen

We're hiring, apply today! Premium

```

127. Word Ladder
Hard 4931 1416 Add to List Share

A transformation sequence from word beginWord to word endWord using a dictionary wordList is a sequence of words beginWord -> s1 -> s2 -> ... -> sk such that:
    • Every adjacent pair of words differs by a single letter.
    • Every si for 1 <= i <= k is in wordList. Note that beginWord does not need to be in wordList.
    • sk == endWord

Given two words, beginWord and endWord, and a dictionary wordList, return the number of words in the shortest transformation sequence from beginWord to endWord, or 0 if no such sequence exists.

Example 1:
Input: beginWord = "hit", endWord = "cog",
wordList = ["hot","dot","dog","lot","log","cog"]
Output: 5
Explanation: One shortest transformation sequence is "hit" -> "hot" -> "dot" -> "dog" -> "cog", which is 5 words long.

Example 2:
Input: beginWord = "hit", endWord = "cog"
Output: 0
    
```

My soln : same logic as sir's just that in my I get only the level of the wanted level , unlike sir's where he keeps distance/level of each word in the graph from beginWord

leetcode.com/problems/word-ladder/

Apps Building Modern Web... Guide to write an ar... Workspace Log in | Innovation... Problem setting gui... Apply | CodeChef cs50.harvard.edu Algo Muse Other bookmark

Description Solution Discuss (999+) Submissions C++ Autocomplete

**127. Word Ladder**

Hard 7440 1615 Add to List Share

A **transformation sequence** from word `beginWord` to word `endWord` using a dictionary `wordList` is a sequence of words `beginWord -> s1 -> s2 -> ... -> sk` such that:

- Every adjacent pair of words differs by a single letter.
- Every `si` for  $1 \leq i \leq k$  is in `wordList`. Note that `beginWord` does not need to be in `wordList`.
- $s_k == endWord$

Given two words, `beginWord` and `endWord`, and a dictionary `wordList`, return the **number of words** in the **shortest transformation sequence** from `beginWord` to `endWord`, or 0 if no such sequence exists.

**Example 1:**

```
Input: beginWord = "hit", endWord = "cog", wordList = ["hot", "dot", "dog", "lot", "log", "cog"]
Output: 5
Explanation: One shortest transformation sequence is "hit" -> "hot" -> "dot" -> "dog" -> "cog", which is 5 words long.
```

**Example 2:**

```
Input: beginWord = "hit", endWord = "cog", wordList = ["hot", "dot", "dog", "lot", "log", "cog"]
Output: 0
Explanation: No transformation sequence exists.
```

```
1 class Solution {
2 public:
3     //1. model of graph : nodes as wordlist elements & beginWord (undirected edges bw elements at distance 1)
4     //2. find shortest distance between beginWord & endWord nodes : use level order traversal for it(bfs)
5     int ladderLength(string beginWord, string endWord, vector<string>& wordList) {
6         int ans=0,i,j,level=0;
7         unordered_set<string> visited,dict;
8         queue<string> curLevel;
9
10        for(auto& word:wordList){
11            dict.insert(word);
12        }
13        curLevel.push(beginWord); //start with the beginWord
14        visited.insert(beginWord); //mark it as visited
15
16        while(!curLevel.empty()){
17            int len=curLevel.size();
18            level++; // increment level
19            while(len>0){
20                len--;
21                string cur,temp;
22                cur=temp=curLevel.front();
23                curLevel.pop();
24
25                for(i=0;i<cur.size();i++){
26                    for(j=0;j<26;j+=26){
27                        cur[i] = 'a'+j; // change the i'th character in current word to all possible letters 'a' to 'z'
28                        // if(cur==endWord) return level+1;
29                        if(dict.find(cur)!=dict.end() && visited.find(cur)==visited.end()){
30                            curLevel.push(cur);
31                            visited.insert(cur); //mark visited
32                            if(cur==endWord) return level+1;
33                        }
34                    }
35                }
36            }
37        }
38    }
39    return 0;
40 }
```

N=length of word

5000=5000 length of dictionary

If we traverse the dictionary( $O(5000 * n)$ ) it then it would be more expensive as compared to if we find every possible word at distance 1 from current word( $O(n * 26)$ )

- Also in general in real world scenarios its advised not to traverse/iterate over the dictionary as they can be too big(length in millions...)

=====

=====

Q) Leetcode 542. 01 Matrix ::

leetcode.com/problems/01-matrix/

Apps Building Modern Web... Guide to write an ar... Workspace Log in | Innovation...

Description Solution Discuss (999+) Submissions

**542. 01 Matrix**

Medium 4088 198 Add to List Share

Given an  $m \times n$  binary matrix `mat`, return the **distance of the nearest 0** for each cell.

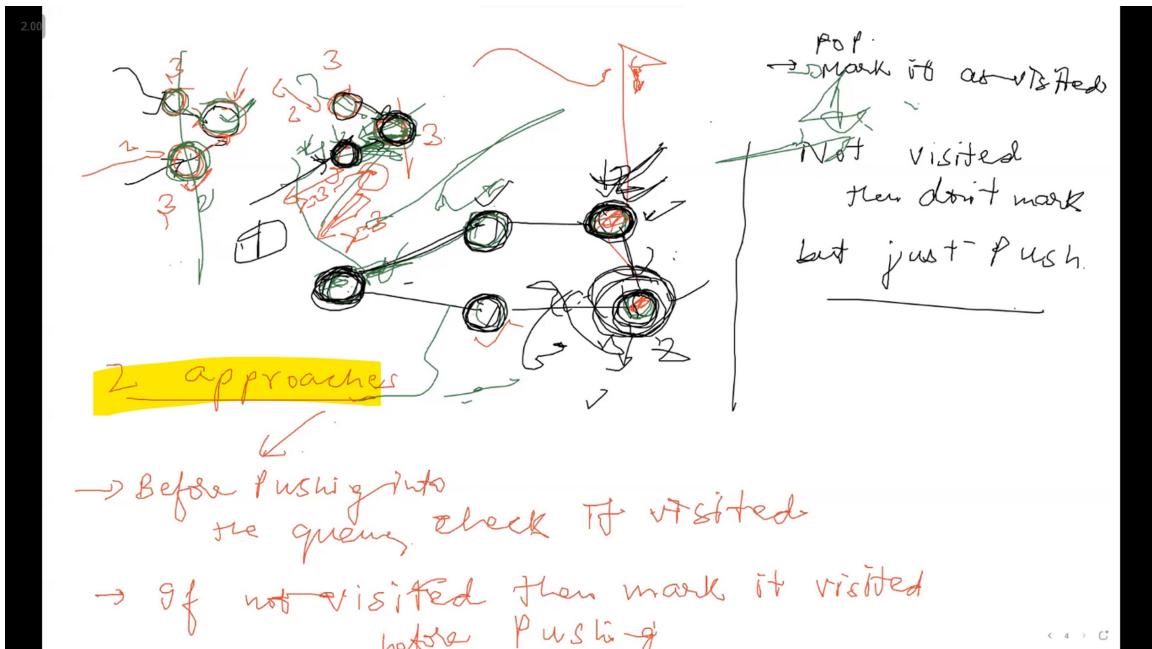
The distance between two adjacent cells is 1.

**Example 1:**

0	0	0
0	1	0
0	0	0

```
Input: mat = [[0,0,0],[0,1,0],[0,0,0]]
Output: [[0,0,0],[0,1,0],[0,0,0]]
```

**Example 2:**



My soln::

leetcode.com/problems/01-matrix/submissions/

Success Details >

Runtime: 138 ms, faster than 31.97% of C++ online submissions for 01 Matrix.

Memory Usage: 30.2 MB, less than 49.67% of C++ online submissions for 01 Matrix.

Next challenges:

[Shortest Path to Get Food](#)

[Minimum Operations to Remove Adjacent Ones in Matrix](#)

Show off your acceptance:

Time Submitted	Status	Runtime	Memory	Language
02/15/2022 20:48	Accepted	138 ms	30.2 MB	cpp

```

1 * class Solution {
2     public:
3         int dx[4]={-1,1,0,0};
4         int dy[4]={0,0,-1,1};
5         bool inRange(int i,int j,int n,int m){// kinda like a inRange() function with modified aim to find distance
6             // int r=dist.size(),c=dist[0].size(),x=1e5;
7             if(i<0 || j<0 || i>n || j>m) return false;
8             return true;
9         }
10
11         vector<vector<int>> updateMatrix(vector<vector<int>>& mat) {
12             int n=mat.size(),m=mat[0].size(),i,j,distance=0;
13             queue<pair<int,int>> curset;// current set to be iterated upon
14             vector<vector<int>> visited(n,vector<int>(m,-1));
15             for(i=0;i<n;i++){
16                 for(j=0;j<m;j++){
17                     if(mat[i][j]==0){
18                         visited[i][j]=distance;//initial distance =0 for all 0's in th matrix
19                         curset.push({i,j});
20                     }
21                 }
22             }
23             while(!curset.empty()){
24                 int len=curset.size();
25                 distance++; //increment the level for level order traversal
26                 while(len>0){
27                     len--;
28                     i=curset.front().first,j=curset.front().second;
29                     curset.pop();
30                     for(int x=0;x<4;x++){
31                         int a=i+dx[x],b=j+dy[x];
32                         if(inRange(a,b,n,m) && visited[a][b]==-1){
33                             //inrange and unvisited
34                             visited[a][b]=distance;
35                             curset.push({a,b});
36                         }
37                     }
38                 }
39             }
40         }
41     }
42 }
```

Success Details >

Runtime: 138 ms, faster than 31.97% of C++ online submissions for 01 Matrix.

Memory Usage: 30.2 MB, less than 49.67% of C++ online submissions for 01 Matrix.

Next challenges:

[Shortest Path to Get Food](#)

[Minimum Operations to Remove Adjacent Ones in Matrix](#)

Show off your acceptance: [f](#) [t](#) [in](#)

Time Submitted	Status	Runtime	Memory	Language
02/15/2022 20:48	Accepted	138 ms	30.2 MB	cpp

```
o
9         return true;
10    }
11
12    vector<vector<int>> updateMatrix(vector<vector<int>>& mat) {
13        int m=mat.size(),n=mat[0].size(),i,j,distance=0;
14        queue<pair<int,int>> curset;// current set to be iterated upon
15        vector<vector<int>> visited(n,vector<int>(m,-1));
16        for(i=0;i<n;i++){
17            for(j=0;j<m;j++){
18                if(mat[i][j]==0){
19                    visited[i][j]=distance;//initial distance =0 for all 0's in th matrix
20                    curset.push({i,j});
21                }
22            }
23        }
24        while(!curset.empty()){
25            int len=curset.size();
26            distance++; //increment the level for level order traversal
27            while(len>0){
28                len--;
29                i=curset.front().first,j=curset.front().second;
30                curset.pop();
31                for(int x=0;x<4;x++){
32                    int a=i+dx[x],b=j+dy[x];
33                    if(inRange(a,b,n,m) && visited[a][b]==-1){
34                        //inrange and unvisited
35                        visited[a][b]=distance;
36                        curset.push({a,b});
37                    }
38                }
39            }
40        }
41        return visited;
42    }
43
44};
```

### Sir's soln: Approach 1:

During(/before) pushing into queue mark it visited (I did the same above)

The screenshot shows a browser window with the URL <https://leetcode.com/problems/01-matrix/>. The page displays the problem statement and the solution code in C++.

**Problem Statement:** Given an  $m \times n$  binary matrix  $\text{mat}$ , return the distance of the nearest 0 for each cell.

**Code:**

```
public:
    vector<vector<int>> offset = {{0, 1}, {1, 0}, {-1, 0}, {0, -1}};
    vector<vector<int>> updateMatrix(vector<vector<int>>& mat) {
        // q[0] = row idx.
        // q[1] = col idx.
        // q[2] = distance.
        int m = mat.size(), n = mat[0].size();
        queue<vector<int>> q;
        vector<vector<int>> res(m, vector<int>(n, -1));

        // Initialize the q and visited array (res).
        for (int i = 0; i < m; i++) {
```

The screenshot shows a browser window with the URL <https://leetcode.com/problems/01-matrix/>. The page displays the problem statement and the solution code in C++.

**Problem Statement:** Given an  $m \times n$  binary matrix  $\text{mat}$ , return the distance of the nearest 0 for each cell.

**Code:**

```
// q[0] = row idx.
// q[1] = col idx.
// q[2] = distance.
int m = mat.size(), n = mat[0].size();
queue<vector<int>> q;
vector<vector<int>> res(m, vector<int>(n, -1));

// Initialize the q and visited array (res).
for (int i = 0; i < m; i++) {
    for (int j = 0; j < n; j++) {
        if (mat[i][j] == 0) {
            q.push({i, j, 0});
            res[i][j] = 0;
        }
    }
}

// Level-order traversal.
while (!q.empty()) {
    vector<int> curr = q.front();
    q.pop();

    // Go through the nbrs.
    for (auto& entry : offset) {
        int x = curr[0] + entry[0];
        int y = curr[1] + entry[1];

        if (x < 0 || y < 0 || x >= m || y >= n)
            continue;

        if (res[x][y] == -1) {
            res[x][y] = curr[2] + 1;
            q.push({x, y, curr[2] + 1});
        }
    }
}

return res;
```

**Example 1:**

0	0	0
0	1	0
0	0	0

**Input:** mat = [[0,0,0],[0,1,0],[0,0,0]]  
**Output:** [[0,0,0],[0,1,0],[0,0,0]]

**Example 2:**

-	-	-	-
-	-	-	-
-	-	-	-

### Approach 2:

This approach is used in djikstra's algorithm

DWT | 25 Apr| Graph 5 from Demux Support

LeetCode Explore Problems Mock Contest Discuss Store

Description Solution Discuss (612) Submissions C++ Autocomplete

### 542. 01 Matrix

Medium 2279 125 Add to List Share

Given an  $m \times n$  binary matrix  $\text{mat}$ , return the distance of the nearest 0 for each cell.

The distance between two adjacent cells is 1.

Example 1:

0	0	0
0	1	0
0	0	0

```
public:
    vector<vector<int>> offset = {{0, 1}, {1, 0}, {-1, 0}, {0, -1}};

    vector<vector<int>> updateMatrix(vector<vector<int>>& mat) {
        // q[0] = row idx.
        // q[1] = col idx.
        // q[2] = distance.
        int m = mat.size(), n = mat[0].size();
        queue<vector<int>> q;
        vector<vector<int>> res(m, vector<int>(n, -1));

        // Initialize the q and visited array (res).
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                if (mat[i][j] == 0) {
                    q.push({i, j, 0});
                }
            }
        }

        // Level-order traversal.
        while (!q.empty()) {
            vector<int> curr = q.front();
            q.pop();

            if (res[curr[0]][curr[1]] != -1)
                continue;

            res[curr[0]][curr[1]] = curr[2];

            // Go through the nbrs.
            for (auto entry : offset) {
                int x = curr[0] + entry[0];
                int y = curr[1] + entry[1];

                if (x < 0 || y < 0 || x >= m || y >= n)
                    continue;

                if (res[x][y] == -1) {
                    q.push({x, y, curr[2] + 1});
                }
            }
        }
    }
};
```

DWT | 25 Apr| Graph 5 from Demux Support

LeetCode Explore Problems Mock Contest Discuss Store

Description Solution Discuss (612) Submissions C++ Autocomplete

### 542. 01 Matrix

Medium 2279 125 Add to List Share

Given an  $m \times n$  binary matrix  $\text{mat}$ , return the distance of the nearest 0 for each cell.

The distance between two adjacent cells is 1.

Example 1:

0	0	0
0	1	0
0	0	0

Input: mat = [[0,0,0],[0,1,0],[0,0,0]]  
Output: [[0,0,0],[0,1,0],[0,0,0]]

Example 2:

-	-	-
---	---	---

```
// q[0] = row idx.
// q[1] = col idx.
// q[2] = distance.
int m = mat.size(), n = mat[0].size();
queue<vector<int>> q;
vector<vector<int>> res(m, vector<int>(n, -1));
[docdb] Master: Remove infinite retry on CreateTablet: issue #1378 · yugabyte/yugabyte-db
// Initialize the q and visited array (res).
for (int i = 0; i < m; i++) {
    for (int j = 0; j < n; j++) {
        if (mat[i][j] == 0) {
            q.push({i, j, 0});
        }
    }
}

// Level-order traversal.
while (!q.empty()) {
    vector<int> curr = q.front();
    q.pop();

    if (res[curr[0]][curr[1]] != -1)
        continue;

    res[curr[0]][curr[1]] = curr[2];

    // Go through the nbrs.
    for (auto entry : offset) {
        int x = curr[0] + entry[0];
        int y = curr[1] + entry[1];

        if (x < 0 || y < 0 || x >= m || y >= n)
            continue;

        if (res[x][y] == -1) {
            q.push({x, y, curr[2] + 1});
        }
    }
}
```

Problems Pick One < Prev Next > Console Contribute Run Code ^ Submit

=====

Lec 6:

Q) leetcode 269. alien dictionary

DWT | 27 Apr| Graph 6 from Demux Support [ode.com/problems/alien-dictionary/](https://ode.com/problems/alien-dictionary/)

LeetCode Explore Problems Mock Contest Discuss Store

Press Esc to exit full screen We're hiring, apply today!

Description Solution Discuss (926) Submissions

Return a string of the unique letters in the new alien language sorted in **lexicographically increasing order** by the new language's rules. If there is no solution, return "" . If there are multiple solutions, return **any of them**.

A string  $s$  is **lexicographically smaller** than a string  $t$  if at the first letter where they differ, the letter in  $s$  comes before the letter in  $t$  in the alien language. If the first  $\min(s.length, t.length)$  letters are the same, then  $s$  is smaller if and only if  $s.length < t.length$ .

**Example 1:**  
**Input:** words = ["wrt", "wrf", "er", "ett", "rftt"]  
**Output:** "wertf"

**Example 2:**  
**Input:** words = ["z", "x"]  
**Output:** "zx"

**Example 3:**  
**Input:** words = ["z", "x", "z"]  
**Output:** ""  
**Explanation:** The order is invalid, so return "".

**Constraints:**

1 ≤ words.length ≤ 300  
 Each word in words has length in range [1, 10].  
 words[i] consists of only lowercase English letters.

1:09:39

Testcase Run Code Result Debugger

You are not signed in. Sign in to run or submit your code.

Problems  Vimeo Submit

```

class Solution {
public:
    string alienOrder(vector<string>& words) {
        // Construct the graph.
        // Nodes are the letters.
        // a->b iff a < b.
        // Adjacency list.
        unordered_map<char, unordered_set<char>> graph;
        int n = words.size();
        unordered_map<char, int> indegree;

        for (int i = 0; i < n; i++) {
            for (int k = 0; k < words[i].size(); k++) {
                if (graph.find(words[i][k]) == graph.end()) {
                    graph[words[i][k]] = {};
                    indegree[words[i][k]] = 0;
                }
            }
        }

        for (int i = 0; i < n; i++) {
            for (int j = i+1; j < n; j++) {
                if (words[i] == words[j])
                    continue;

                int k = 0, l = 0;
                while(k < words[i].size() && l < words[j].size() && words[i][k] == words[j][l]) {
                    k++;
                    l++;
                }

                if (words[j] is a prefix of words[i])
                    if (l == words[j].size())
                        return "";

                if (l == words[i].size())
                    continue;

                if (graph[words[i][k]].find(words[j][l]) == graph[words[i][k]].end())
                    graph[words[i][k]].insert(words[j][l]);
                    indegree[words[j][l]]++;
            }
        }
    }
}

```

Your previous code was restored from your local storage. [Reset to default](#)

DWT | 27 Apr| Graph 6 from Demux Support [ode.com/problems/alien-dictionary/](https://ode.com/problems/alien-dictionary/)

LeetCode Explore Problems Mock Contest Discuss Store

Day 27 We're hiring, apply today!

Description Solution Discuss Submissions

269. Alien Dictionary Hard 2483 479 Add to List

There is a new alien language that uses the English alphabet. However, the order among the letters is unknown to you.

You are given a list of strings words from the alien language's dictionary, where the strings in words are **sorted lexicographically** by the rules of this new language.

Return a string of the unique letters in the new alien language sorted in **lexicographically increasing order** by the new language's rules. If there is no solution, return "" . If there are multiple solutions, return **any of them**.

A string  $s$  is **lexicographically smaller** than a string  $t$  if at the first letter where they differ, the letter in  $s$  comes before the letter in  $t$  in the alien language. If the first  $\min(s.length, t.length)$  letters are the same, then  $s$  is smaller if and only if  $s.length < t.length$ .

**Example 1:**  
**Input:** words = ["wrt", "wrf", "er", "ett", "rftt"]  
**Output:** "wertf"

**Example 2:**  
**Input:** words = ["z", "x"]  
**Output:** "zx"

**Example 3:**  
**Input:** words = ["z", "x", "z"]  
**Output:** ""  
**Explanation:** The order is invalid, so return "".

**Constraints:**

1 ≤ words.length ≤ 300  
 Each word in words has length in range [1, 10].  
 words[i] consists of only lowercase English letters.

1:09:39

Testcase Run Code Result Debugger

You are not signed in. Sign in to run or submit your code.

Problems  Vimeo Submit

```

// Nodes are the letters.
// a->b iff a < b.
// Adjacency list.
unordered_map<char, unordered_set<char>> graph;
int n = words.size();
unordered_map<char, int> indegree;

for (int i = 0; i < n; i++) {
    for (int k = 0; k < words[i].size(); k++) {
        if (graph.find(words[i][k]) == graph.end()) {
            graph[words[i][k]] = {};
            indegree[words[i][k]] = 0;
        }
    }
}

for (int i = 0; i < n; i++) {
    for (int j = i+1; j < n; j++) {
        if (words[i] == words[j])
            continue;

        int k = 0, l = 0;
        while(k < words[i].size() && l < words[j].size() && words[i][k] == words[j][l]) {
            k++;
            l++;
        }

        if (words[j] is a prefix of words[i])
            if (l == words[j].size())
                return "";

        if (l == words[i].size())
            continue;

        if (graph[words[i][k]].find(words[j][l]) == graph[words[i][k]].end())
            graph[words[i][k]].insert(words[j][l]);
            indegree[words[j][l]]++;
    }
}

```

Your previous code was restored from your local storage. [Reset to default](#)

DWT | 27 Apr| Graph 6

from Demux Support [ode.com/problems/alien-dictionary/](https://ode.com/problems/alien-dictionary/)

Apps Quant Prep Demux Academy... Coding Prep Apply Companies The Data Incubator ADE User's Guide... To Do 1 Yum Command C... Chapter 8. Yum

**269. Alien Dictionary**

Hard 2483 479 Add to List

There is a new alien language that uses the English alphabet. However, the order among the letters is unknown to you.

You are given a list of strings words from the alien language's dictionary, where the strings in words are sorted lexicographically by the rules of this new language.

Return a string of the unique letters in the new alien language sorted in lexicographically increasing order by the new language's rules. If there is no solution, return "" . If there are multiple solutions, return any of them.

A string s is lexicographically smaller than a string t if at the first letter where they differ, the letter in s comes before the letter in t in the alien language. If the first  $\min(s.length, t.length)$  letters are the same, then s is smaller if and only if  $s.length < t.length$ .

**Example 1:**

```
Input: words = ["wrt","wrf","er","ett","rftt"]
Output: "wertf"
```

**Example 2:**

```
Input: words = ["z","x"]
Output: "zx"
```

Your previous code was restored from your local storage. [Reset to default](#)

1:10:39 |

### Optimized code for above:

- Here we capture only the relations between adjacent words only(as the relationship is transitive so those will be automatically captured in the graph itself, so we don't need to populate the relations between all the words rather just adjacent words would also work)

DWT | 27 Apr| Graph 6

from Demux Support [ode.com/problems/alien-dictionary/](https://ode.com/problems/alien-dictionary/)

Apps Quant Prep Demux Academy... Coding Prep Apply Companies The Data Incubator ADE User's Guide... To Do 1 Yum Command C... Chapter 8. Yum Setup VIM for Go... Other Bookmarks

We're hiring, apply today!

**269. Alien Dictionary**

Hard 2483 479 Add to List

There is a new alien language that uses the English alphabet. However, the order among the letters is unknown to you.

You are given a list of strings words from the alien language's dictionary, where the strings in words are sorted lexicographically by the rules of this new language.

Return a string of the unique letters in the new alien language sorted in lexicographically increasing order by the new language's rules. If there is no solution, return "" . If there are multiple solutions, return any of them.

A string s is lexicographically smaller than a string t if at the first letter where they differ, the letter in s comes before the letter in t in the alien language. If the first  $\min(s.length, t.length)$  letters are the same, then s is smaller if and only if  $s.length < t.length$ .

**Example 1:**

```
for (int i = 0; i < n; i++) {
    for (int k = 0; k < words[i].size(); k++) {
        if (graph.find(words[i][k]) == graph.end()) {
            graph[words[i][k]] = {};
            indegree[words[i][k]] = 0;
        }
    }
}

for (int i = 0; i < n-1; i++) {
    int j = i+1;
    if (words[i] == words[j])
        continue;

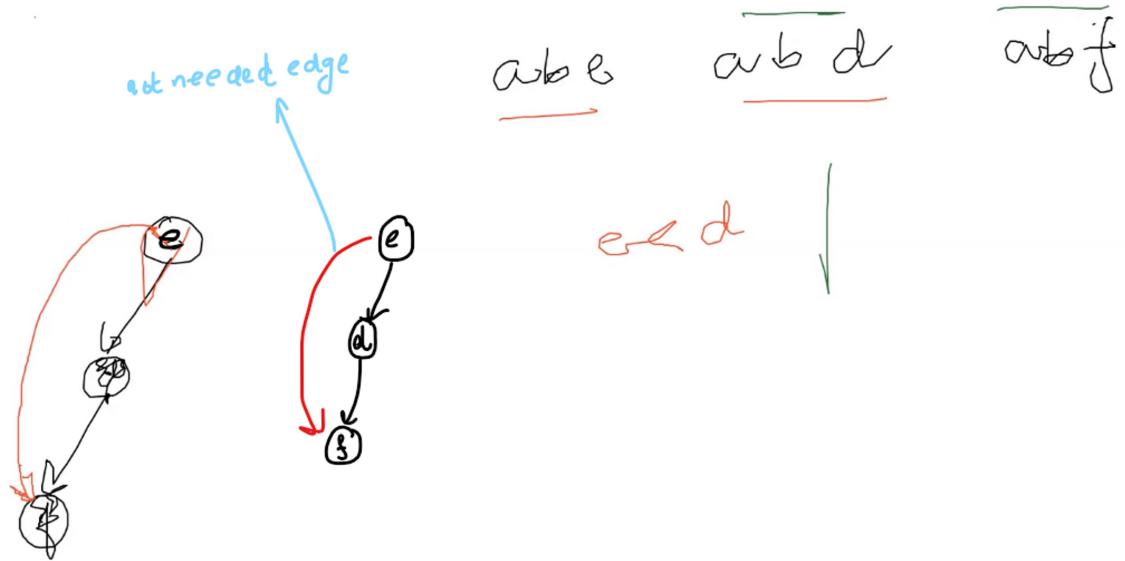
    int k = 0, l = 0;
    while(k < words[i].size() && l < words[j].size() && words[i][k] == words[j][l]) {
        k++;
        l++;
    }

    // words[j] is a prefix of words[i]
    if (j == words[i].size())
        return "";

    if (graph[words[i][k]].find(words[j][l]) == graph[words[i][k]].end()) {
        graph[words[i][k]].insert(words[j][l]);
        indegree[words[j][l]]++;
    }
}
```

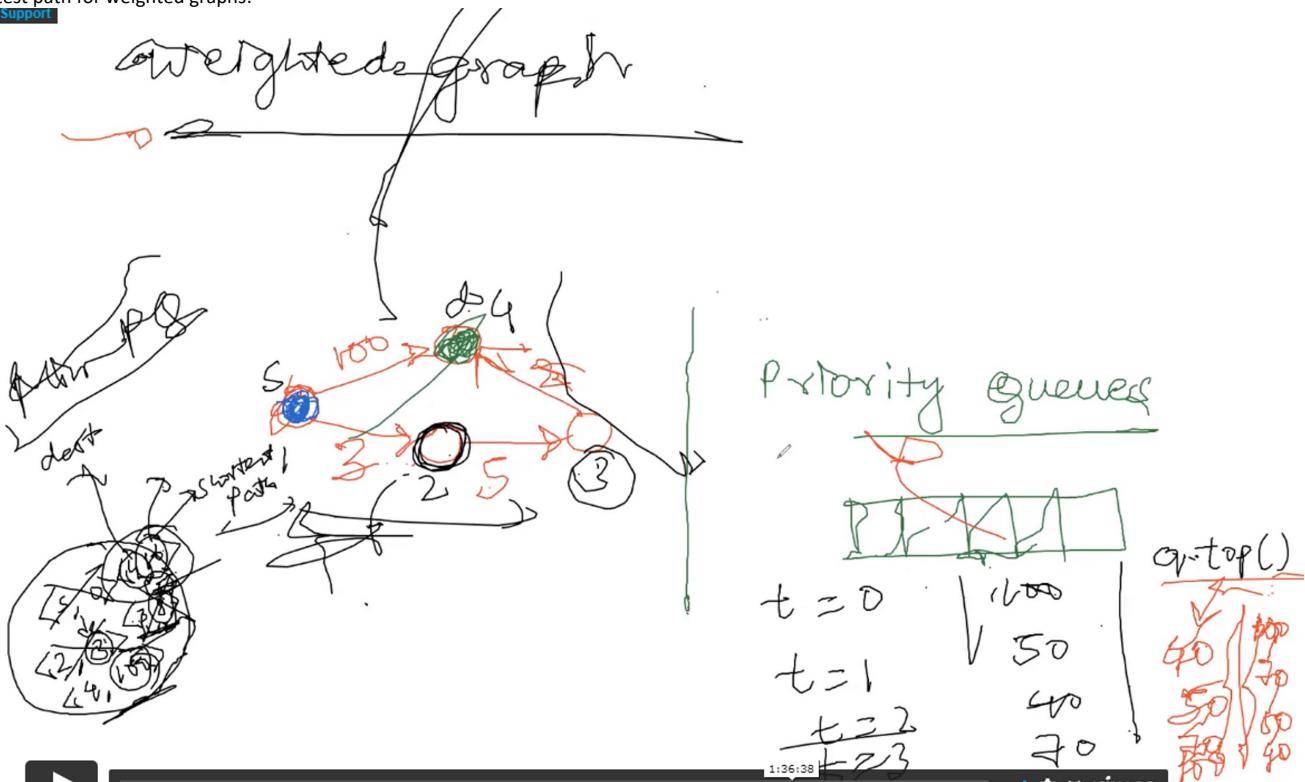
Your previous code was restored from your local storage. [Reset to default](#)

1:13:12 |



Shortest path for weighted graphs:

[emux support](#)



Q)  
Sir ka solnm:

DWT | 27 Apr| Graph 6 from Demux Support

[LeetCode](#) Explore Problems Mock Contest Discuss Store

We're hiring, apply today!

Descript... Solution Discuss... Submis...

### 743. Network Delay Time

Medium 2517 246 Add to List Share

You are given a network of  $n$  nodes, labeled from 1 to  $n$ . You are also given  $\text{times}$ , a list of travel times as directed edges  $\text{times}[i] = (u_i, v_i, w_i)$ , where  $u_i$  is the source node,  $v_i$  is the target node, and  $w_i$  is the time it takes for a signal to travel from source to target.

We will send a signal from a given node  $k$ . Return the time it takes for all the  $n$  nodes to receive the signal. If it is impossible for all the  $n$  nodes to receive the signal, return  $-1$ .

**Example 1:**

```

int networkDelayTime(vector<vector<int>>& times, int n, int k) {
    // Construct the graph.
    vector<vector<pair<int, int>> graph(n+1);
    vector<bool> visited(n+1, false);
    int res = 0, count = 0;

    for (const auto& edge : times) {
        graph[edge[0]].push_back({edge[2], edge[1]});
    }

    // Weighted BFS.
    // Declare a min Priority Queue.
    priority_queue<pair<int, int>, vector<pair<int, int>*, greater<pair<int, int>> pq;
    pq.push({0, k});

    while(!pq.empty()) {
        pair<int, int> t = pq.top();
        pq.pop();

        if (visited[t.second])
            continue;

        visited[t.second] = true;
        count++;

        res = max(res, t.first);

        // Push all the nbrs
        for (const auto& nbrs : graph[t.second]) {
            pq.push({t.first+nbrs.first, nbrs.second});
        }
    }

    return count == n ? res : -1;
}

```

Run Code ▾ Submit

Input: times = [[2,1,1],[2,3,1],[3,4,1]], n = 4, k = 2  
Output: 2

#### Example 2:

Input: times = [[1,2,1]], n = 2, k = 1  
Output: 1

#### Example 3:

Input: times = [[1,2,1]], n = 2, k = 2  
Output: -1

#### Constraints:

- $1 \leq k \leq n \leq 100$
- $1 \leq \text{times.length} \leq 6000$
- $\text{times}[i].length == 3$
- $1 \leq u_i, v_i \leq n$
- $u_i \neq v_i$
- $0 \leq w_i \leq 100$

My soln:

leetcode.com/problems/network-delay-time/submissions/

Building Modern Web... Guide to write an ar... Workspace Log in | Innovation... Problem setting gui... Apply | CodeChef cs50.harvard.edu Algo Muse Topcoder Other b...

Success Details >

Runtime: 187 ms, faster than 47.49% of C++ or Network Delay Time.

Memory Usage: 41.5 MB, less than 48.50% of Network Delay Time.

Next challenges:

- [The Time When the Network Becomes Idle](#)
- [Second Minimum Time to Reach Destination](#)

Show off your acceptance: [f](#) [t](#) [in](#)

Time Submitted	Status	Run
02/18/2022 18:34	Accepted	187
02/18/2022 17:54	Accepted	198

743/2174 Run Code Sub

```

1 class Solution {
2 public:
3     int networkDelayTime(vector<vector<int>>& times, int n, int k) {
4         // Level order traversal : bfs -> but since its weighted we maintain a *priority queue*
5         // maintain running max for time & visited set also
6         // at last check if visited set size == n, if not then return -1 else return running max
7         unordered_set<int> visited; //we could use array of size n also for this(more effective)
8         // Weighted BFS(weighted level order traversal) = priority queue
9         priority_queue<pair<int,int>, vector<pair<int,int>>, greater<pair<int,int>> pq; // <time,node> :sort by time
10        int i,j,time=0,tmax=0;
11
12        vector<vector<pair<int,int>> adj(n); // ui :>vi,weight
13        for(const auto& x:times){
14            adj[x[0]-1].push_back({x[1]-1,x[2]}); //building graph (0 : n-1)
15        }
16        k=k-1; //making 0 based nodes
17        pq.push({time,k}); //pushing initial node k into the pq
18        // Level order traversal
19        while(!pq.empty()){
20            int t,node;
21            t = pq.top().first, node = pq.top().second;
22            pq.pop();
23            visited.insert(node);
24            tmax = max(tmax,t);
25            if(visited.size()==n) return tmax; //this has to be inside the loop
26            //Push all neighbours
27            for(const auto& nbr:adj[node]){
28                if(visited.find(nbr.first)==visited.end()){ //unvisited nbr
29                    pq.push({t + nbr.second , nbr.first}); // {t+wt,nxt_node}
30                }
31            }
32        }
33        if(visited.size()==n) return tmax;
34        return -1;
35    }
}

```

=====  
=====  
LEC - 7

Q) <https://leetcode.ca/all/1102.html>

leetcode.ca/problems/path-with-maximum-minimum-value/

Apps Quant Prep Demux Academy Coding Prep Apply Companies The Data Incubator ADE User's Guide... To Do Yum Command C... Chapter 8. Yum Setup VIM for Go...

LeetCode Explore Problems Mock Contest Discuss Store We're hiring, apply today!

Description Solution Discuss (1...) Submissions C++ Autocomplete

### 1102. Path With Maximum Minimum Value

Medium 680 72 Add to List Share

Given a matrix of integers  $A$  with  $R$  rows and  $C$  columns, find the **maximum** score of a path starting at  $[0,0]$  and ending at  $[R-1,C-1]$ .

The score of a path is the **minimum** value in that path. For example, the value of the path  $8 \rightarrow 4 \rightarrow 5 \rightarrow 9$  is 4.

A path moves some number of times from one visited cell to any neighbouring unvisited cell in one of the 4 cardinal directions (north, east, west, south).

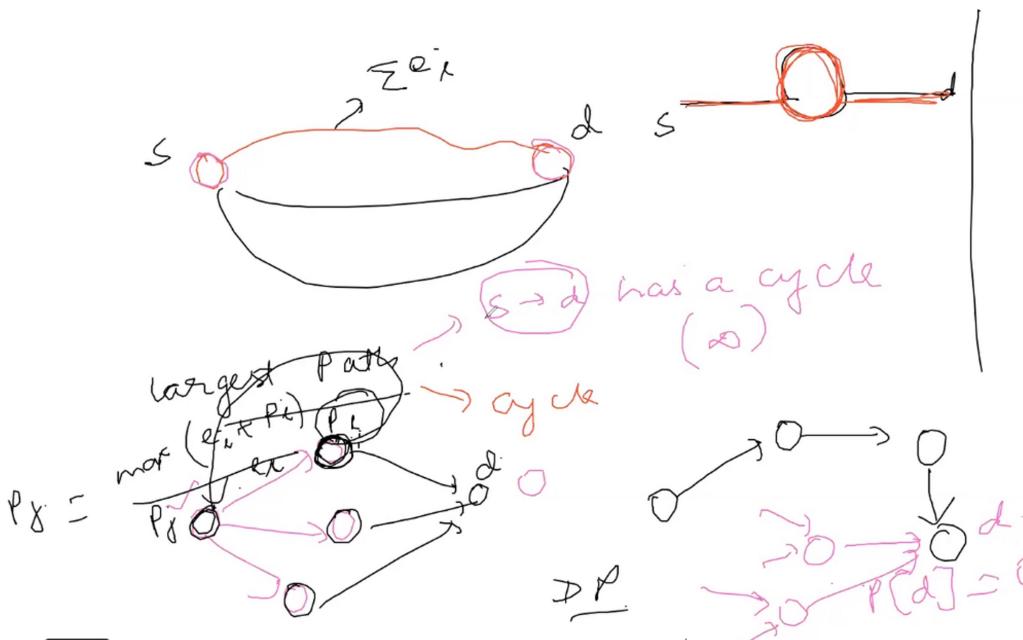
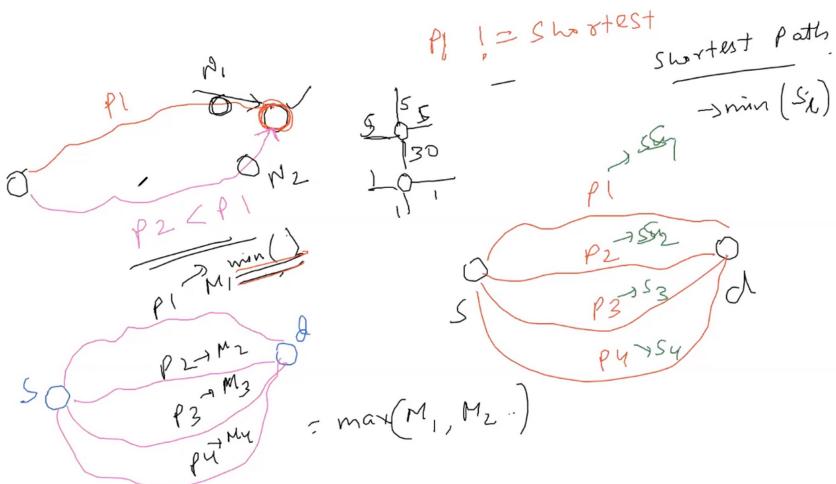
**Example 1:**

5	4	5
1	2	6
7	4	6

**Input:** [[5,4,5],[1,2,6],[7,4,6]]  
**Output:** 4  
**Explanation:** The path with the maximum score is highlighted in yellow.

**Example 2:**

2	2	1	2	2	2
1	1	3	3	1	3



Soln 1 : using pq ::

```
i C++ ▾ • Autocomplete
1 class Solution {
2 public:
3     vector<vector<int>> dirs = {{0, 1}, {1, 0}, {-1, 0}, {0, -1}};
4
5     int maximumMinimumPath(vector<vector<int>>& A) {
6         int m = A.size(), n = A[0].size(), i, j;
7         priority_queue<vector<int>, vector<vector<int>>, less<vector<int>> pq;
8         vector<vector<bool>> visited(m, vector<bool>(n, false));
9
10        pq.push({A[0][0], 0, 0});
11
12        while(!pq.empty()) {
13            vector<int> t = pq.top();
14            pq.pop();
15
16            if (visited[t[1]][t[2]])
```

DWT | 28 Apr| Graph 7

from Demux Support [ode.com/problems/path-with-maximum-minimum-value/](#)

Apps Quant Prep Demux Academy... Coding Prep Apply Companies The Data Incubator ADE User's Guide... To Do 1 Yum Command C... Chapter 8. Yum Setup VIM for Go...

LeetCode Explore Problems Mock Contest Discuss Store

We're hiring, apply today!

Description Solution Discuss (1...) Submissions Autocomplete C++

**1102. Path With Maximum Minimum Value**

Medium 680 72 Add to List Share

Given a matrix of integers  $A$  with  $R$  rows and  $C$  columns, find the **maximum** score of a path starting at  $[0,0]$  and ending at  $[R-1,C-1]$ .

The score of a path is the **minimum** value in that path. For example, the value of the path  $8 \rightarrow 4 \rightarrow 5 \rightarrow 9$  is 4.

A path moves some number of times from one visited cell to any neighbouring unvisited cell in one of the 4 cardinal directions (north, east, west, south).

**Example 1:**

5	4	5
1	2	6
7	4	6

**Input:** [[5,4,5],[1,2,6],[7,4,6]]  
**Output:** 4  
**Explanation:**  
The path with the maximum score is highlighted in yellow.

**Example 2:**

2	2	1	2	2	2
1	2	2	2	1	2

Console Contribute /

```

4
5 * int maximumMinimumPath(vector<vector<int>>& A) {
6     int m = A.size(), n = A[0].size(), i, j;
7     priority_queue<vector<int>, vector<vector<int>>, less<vector<int>> pq;
8     vector<vector<bool>> visited(m, vector<bool>(n, false));
9
10    pq.push({A[0][0], 0, 0});
11
12    while(!pq.empty()) {
13        vector<int> t = pq.top();
14        pq.pop();
15
16        if (visited[t[1]][t[2]])
17            continue;
18
19        visited[t[1]][t[2]] = true;
20
21        if (t[1] == m-1 && t[2] == n-1)
22            return t[0];
23
24        // Traverse the nbrs.
25        for (auto& dir : dirs) {
26            int x = t[1] + dir[0];
27            int y = t[2] + dir[1];
28
29            if (x < 0 || y < 0 || x >= m || y >= n || visited[x][y])
30                continue;
31
32            pq.push({min(t[0], A[x][y]), x, y});
33        }
34
35        // NF
36        return -1;
37    }
38
39 };

```

Your previous code was restored from your local storage. [Reset to default](#)

LeetCode Explore Problems Mock Contest Discuss Store

We're hiring, apply today!

Description Solution Discuss (1...) Submissions Autocomplete C++

**1102. Path With Maximum Minimum Value**

Medium 680 72 Add to List Share

Given a matrix of integers  $A$  with  $R$  rows and  $C$  columns, find the **maximum** score of a path starting at  $[0,0]$  and ending at  $[R-1,C-1]$ .

The score of a path is the **minimum** value in that path. For example, the value of the path  $8 \rightarrow 4 \rightarrow 5 \rightarrow 9$  is 4.

A path moves some number of times from one visited cell to any neighbouring unvisited cell in one of the 4 cardinal directions (north, east, west, south).

**Example 1:**

5	4	5
1	2	6
7	4	6

**Input:** [[5,4,5],[1,2,6],[7,4,6]]  
**Output:** 4  
**Explanation:**  
The path with the maximum score is highlighted in yellow.

**Example 2:**

2	2	1	2	2	2
1	2	2	2	1	2

Console Contribute /

```

3
4
5 * vector<vector<int>> dirs = {{0, 1}, {1, 0}, {-1, 0}, {0, -1}};
6
7 int maximumMinimumPath(vector<vector<int>>& A) {
8     int m = A.size(), n = A[0].size(), i, j;
9     priority_queue<vector<int>, vector<vector<int>>, less<vector<int>> pq;
10    vector<vector<bool>> visited(m, vector<bool>(n, false));
11
12    pq.push({A[0][0], 0, 0});
13
14    while(!pq.empty()) {
15        vector<int> t = pq.top();
16        pq.pop();
17
18        if (visited[t[1]][t[2]])
19            continue;
20
21        visited[t[1]][t[2]] = true;
22
23        if (t[1] == m-1 && t[2] == n-1)
24            return t[0];
25
26        // Traverse the nbrs.
27        for (auto& dir : dirs) {
28            int x = t[1] + dir[0];
29            int y = t[2] + dir[1];
30
31            if (x < 0 || y < 0 || x >= m || y >= n || visited[x][y])
32                continue;
33
34            pq.push({min(t[0], A[x][y]), x, y});
35        }
36
37        // NF
38        return -1;
39    }

```

Run Code ^

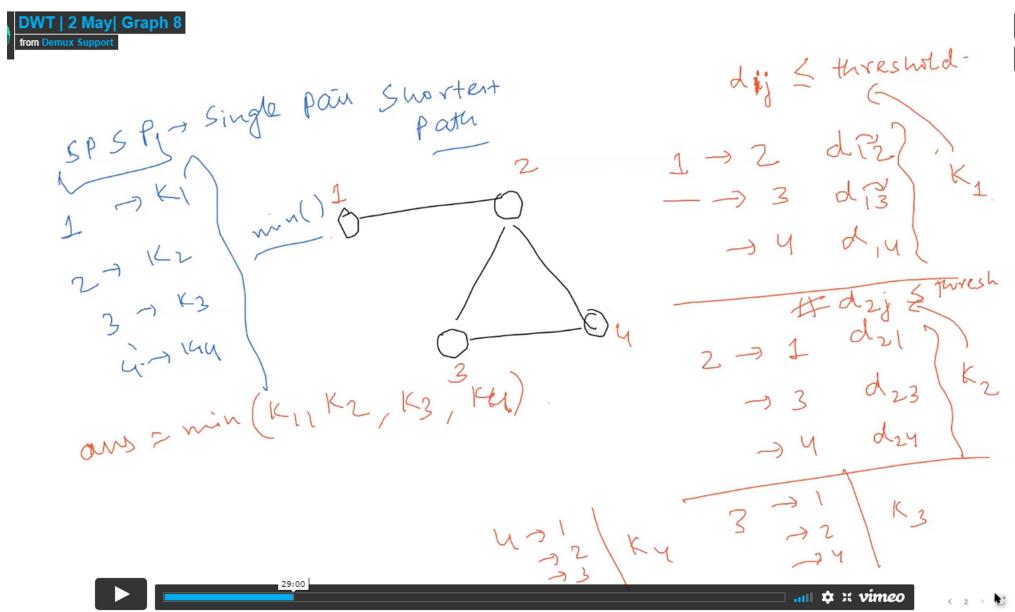
Rewatch this question lec 7 :: rewatch it::

=====

Lec 8 ::

Leetcode:: 1334. Find the City With the Smallest Number of Neighbors at a Threshold Distance

1334. Find the City With the Smallest Number of Neighbors at a Threshold Distance

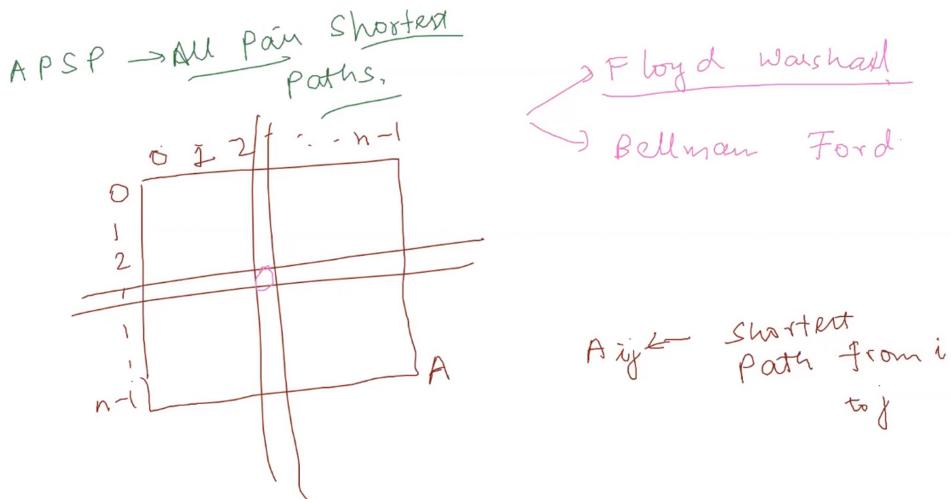


Brute force :

- single pair shortest path

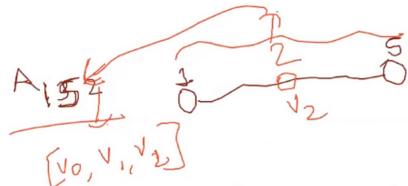
Efficient:

- All pair shortest path
  - o Floyd warshall(dp soln, but not that hard to think of )
  - o Bellman ford

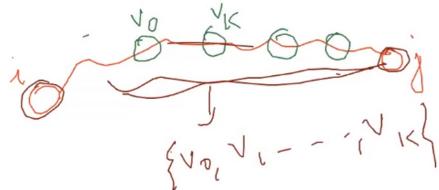


## Floyd Warshall

$0, 1, 2, \dots, n-1$



$A_{ijk} \leftarrow$  shortest path from  $i$  to  $j$  that contains  
only vertices from  $\{v_0, v_1, \dots, v_k\}$

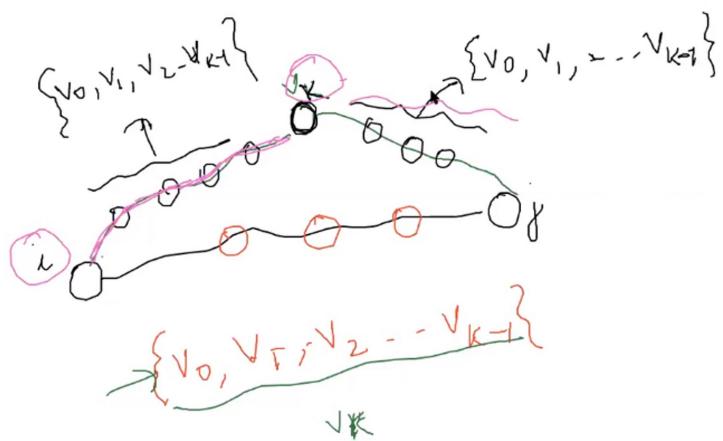


WT | 2 May | Graph 8

Demux Support

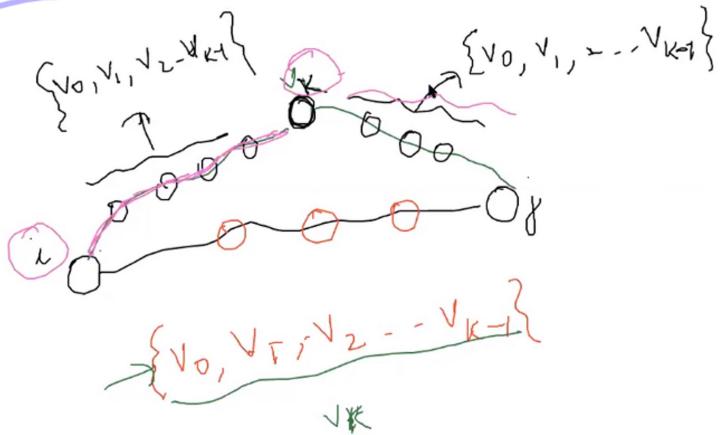
## Floyd Warshall

$$A_{ijk} = \min(A_{ijk-1}, A_{ikk-1} + A_{kjk-1})$$

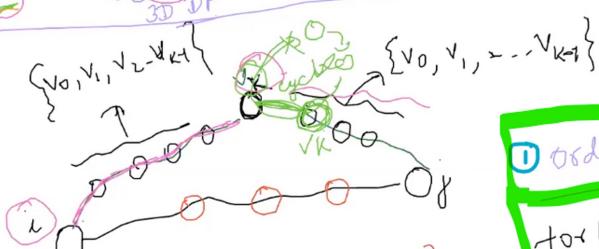
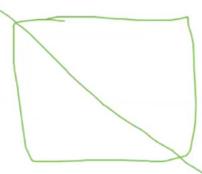


## Floyd Warshall

$$A_{ijk} = \min(A_{ijk-1}, A_{ikk-1} + A_{kkj-1})$$



$$\rightarrow A_{ijk} = \min(A_{ijk-1}, A_{ikk-1} + A_{kkj-1})$$

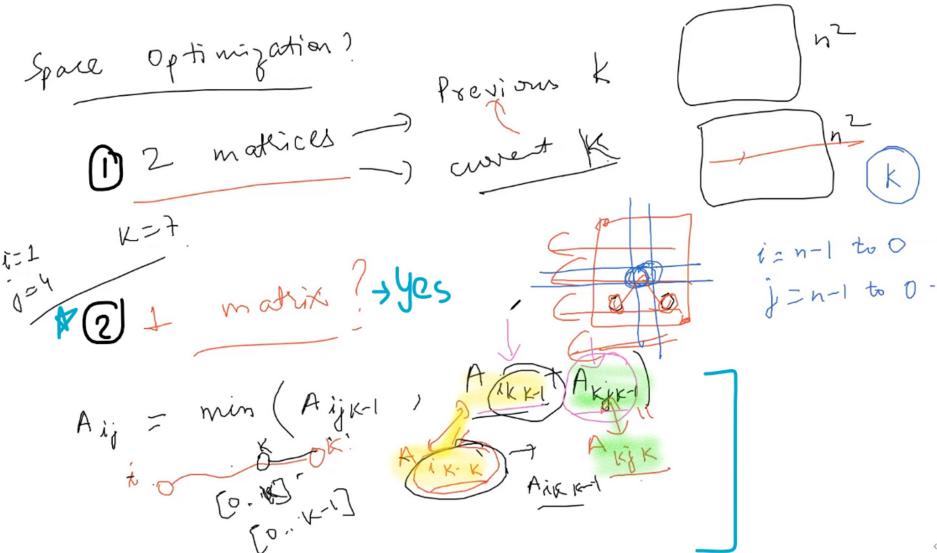


① Order of iteration

```
for (k=0; k < n; k++)
{
    for (i=0; i < n; i++)
        for (j=0; j < n; j++)
            {
                ...
            }
}
```

② Base Case

$A(i, i) = 0$   
 $\forall i \in V$



NOTE:

$A(i)(j)(k)$  = minimum metric (eg: path sum) to reach form node  $i$  to node  $j$ ; with intermediate nodes belonging to the set:  $\{v_0, v_1, \dots, v_k\}$

Things we need to consider:

1. Order of iteration (/order of filling the table):
  - a. Form the recursive relation it's clear we need to fill the minimum  $k$  first, i.e., since  $A(ijk)$  is dependent on  $A(ij(k-1))$  therefore we fill the table for  $k=0$  first then for  $k=1, \dots$  & so on
2. Base Case
  - a.  $dp(ij) = -1$ ; if no edge between nodes  $i$  &  $j$
  - b.  $dp(ij) = W_{ij}$ ; if there is a edge between node  $i$  &  $j$  and weight of the edge is  $W_{ij}$
3. Space Optimization:
  - a. Can it be done with 2 matrices?
    - i. Yes, one for previous  $k$ , and one for current  $k$
  - b. Can it be done with **just 1 matrix**?
    - i. YES, we know the matrix will be updated and that we cannot avoid, now we see that the updated values and previous values are same/not changed so it won't affect our ans, therefore using only 1 matrix will suffice. Another way to think about it is: the updated value is the same as the previous value so storing the previous value in another matrix is not needed.
    - ii. Recursion:  $A(i)(j)(k) = \min(A(i)(j)(k-1), A(i)(k)(k-1) + A(k)(j)(k-1))$
    - iii. We see that:  $A(i)(k)(k-1) = A(i)(k)(k)$ ; since going from  $i$  to  $k$  with intermediate nodes as  $v_0, v_1, \dots, v_{k-1}$  is same as going from  $i$  to  $k$  with intermediate nodes as  $v_0, v_1, \dots, v_k$ ; (otherwise it would imply that there is a self loop/cycle in the graph and making path sum infinite)
    - iv. Similar case for:  $A(k)(j)(k-1) = A(k)(j)(k)$

DWT | 2 May | Graph 8

from Demux Support

[LeetCode](#) Explore Problems Mock Contest Discuss Store

Description Solution Discuss (2...) Submissions Autocomplete

1334. Find the City With the Smallest Number of Neighbors at a Threshold Distance

Medium 540 Q 34 Add to List Share

There are  $n$  cities numbered from 0 to  $n-1$ . Given the array edges where  $\text{edges}[i] = [\text{from}_i, \text{to}_i, \text{weight}_i]$  represents a bidirectional and weighted edge between cities  $\text{from}_i$  and  $\text{to}_i$ , and given the integer  $\text{distanceThreshold}$ .

Return the city with the smallest number of cities that are reachable through some path and whose distance is at most  $\text{distanceThreshold}$ . If there are multiple such cities, return the city with the greatest number.

Notice that the distance of a path connecting cities  $i$  and  $j$  is equal to the sum of the edges' weights along that path.

**Example 1:**

```

1+ class Solution {
2+ public:
3+     int findTheCity(int n, vector<vector<int>>& edges, int distanceThreshold) {
4+         vector<vector<pair<int, int>> graph(n);
5+         vector<vector<int>> dp(n, vector<int>(n, INT_MAX));
6+         int i, j, k, count, res = INT_MAX, res_vertex;
7+
8+         for (const auto& edge : edges) {
9+             graph[edge[0]].push_back({edge[1], edge[2]});
10+            graph[edge[1]].push_back({edge[0], edge[2]});
11+        }
12+        // Base case
13+        dp[0][0] = edge[2];
14+        dp[0][0][edge[0]] = edge[2];
15+
16+        // Base case.
17+        for (int i = 0; i < n; i++) {
18+            dp[i][i] = 0;
19+
20+            for (k = 0; k < n; k++) {
21+                for (l = 0; l < n; l++) {
22+                    for (j = 0; j < n; j++) {
23+                        if (dp[l][k] != INT_MAX && dp[k][j] != INT_MAX)
24+                            dp[l][j] = min(dp[l][j], dp[l][k]+dp[k][j]);
25+                    }
26+                }
27+            }
28+
29+            for (i = 0; i < n; i++) {
30+                count = 0;
31+                for (j = 0; j < n; j++) {
32+                    if (dp[i][j] <= distanceThreshold)
33+                        count++;
34+                }
35+            }
36+            if (count < res)
37+                res = count;
38+        }
39+
40+    }
41+
42+    return res_vertex;
43+
44+ };
45+

```

Floyd Warshall  
 $(APS \rightarrow)$   
 $dp[i][j]$

We're hiring, apply today!

Medium 540 Q 34 Add to List Share

There are  $n$  cities numbered from 0 to  $n-1$ . Given the array edges where  $\text{edges}[i] = [\text{from}_i, \text{to}_i, \text{weight}_i]$  represents a bidirectional and weighted edge between cities  $\text{from}_i$  and  $\text{to}_i$ , and given the integer  $\text{distanceThreshold}$ .

Return the city with the smallest number of cities that are reachable through some path and whose distance is at most  $\text{distanceThreshold}$ . If there are multiple such cities, return the city with the greatest number.

Notice that the distance of a path connecting cities  $i$  and  $j$  is equal to the sum of the edges' weights along that path.

**Example 1:**

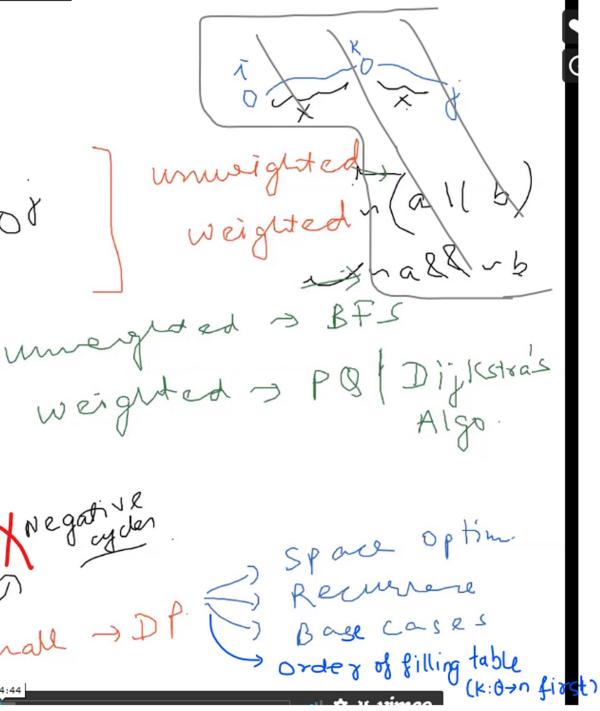
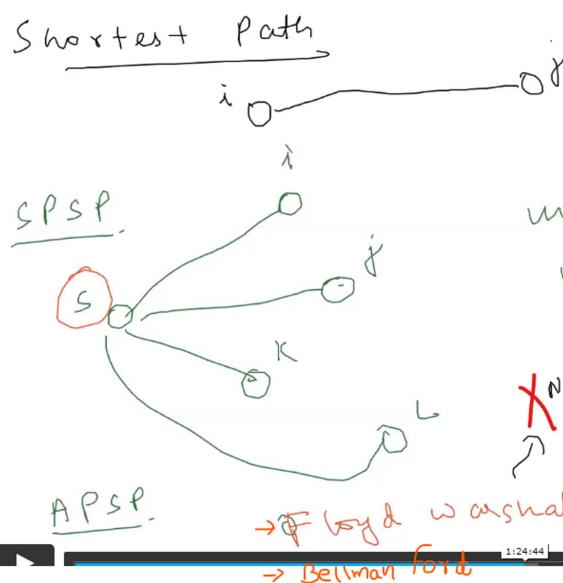
```

1+ class Solution {
2+ public:
3+     int findTheCity(int n, vector<vector<int>>& edges, int distanceThreshold) {
4+         graph[0].push_back({1, 3});
5+         graph[1].push_back({0, 2});
6+         graph[2].push_back({1, 3});
7+         graph[3].push_back({0, 1});
8+         // Base case.
9+         dp[0][0] = edge[2];
10+        dp[0][0][edge[0]] = edge[2];
11+
12+        // Base case.
13+        for (int i = 0; i < n; i++)
14+            dp[i][i] = 0;
15+
16+        for (k = 0; k < n; k++) {
17+            for (l = 0; l < n; l++) {
18+                for (j = 0; j < n; j++) {
19+                    if (dp[l][k] != INT_MAX && dp[k][j] != INT_MAX)
20+                        dp[l][j] = min(dp[l][j], dp[l][k]+dp[k][j]);
21+                }
22+            }
23+        }
24+
25+        for (i = 0; i < n; i++) {
26+            count = 0;
27+            for (j = 0; j < n; j++) {
28+                if (dp[i][j] <= distanceThreshold)
29+                    count++;
30+            }
31+            if (res >= count) {
32+                res = count;
33+                res_vertex = i;
34+            }
35+        }
36+
37+        return res_vertex;
38+
39+    }
40+
41+ };
42+

```

I  
for this on

Overview om shortest path:

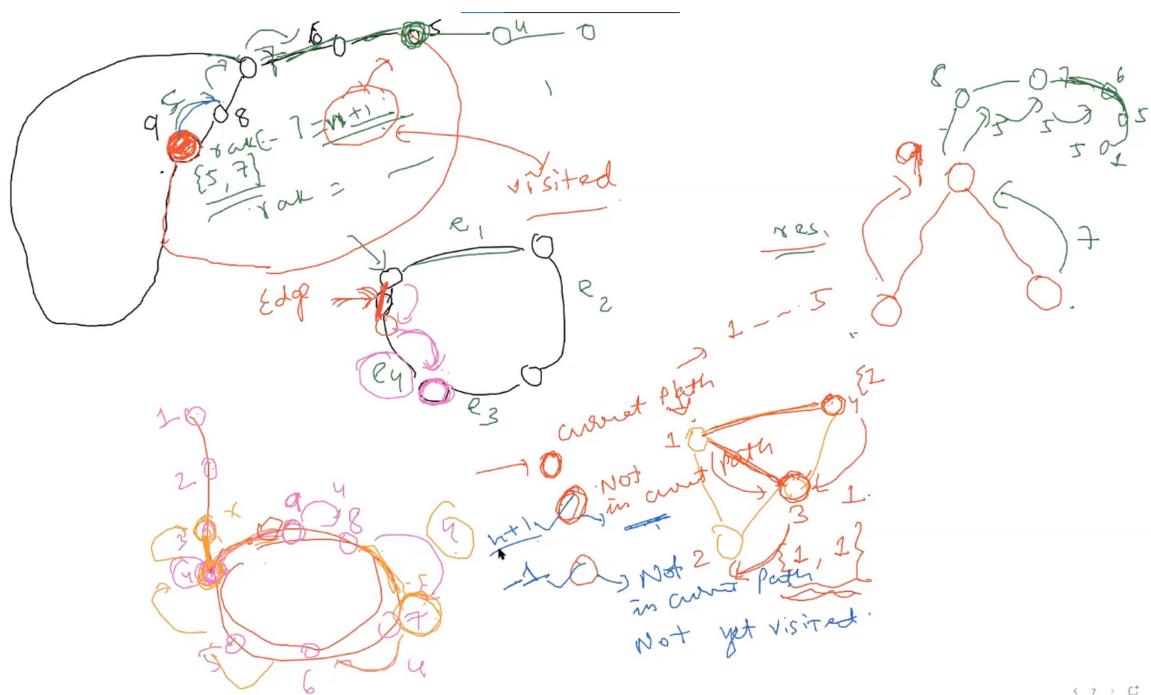


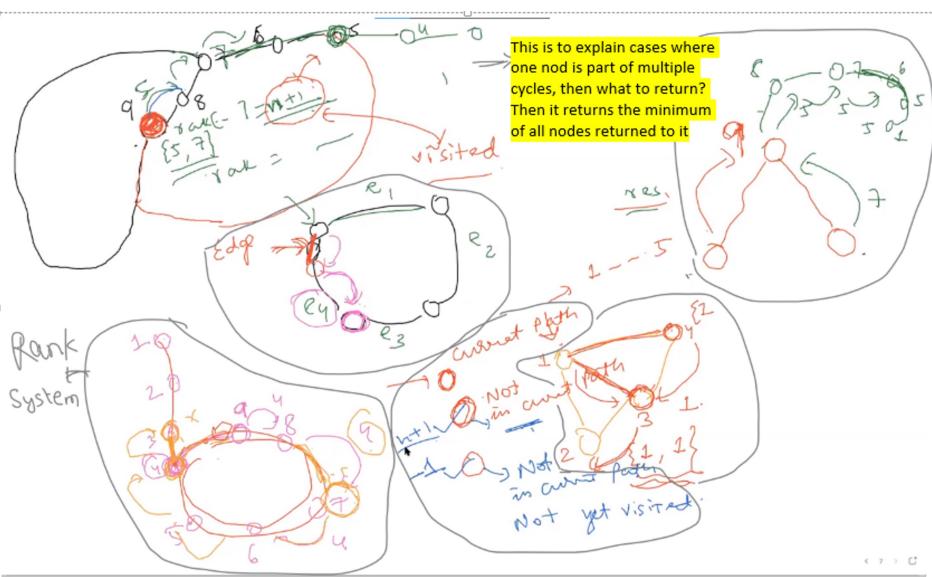
=====

### Leetcode 1192. Critical Connections in a Network

From <https://leetcode.com/problems/critical-connections-in-a-network/>

Here we have to tell bridges - any edge which is not a part of any cycle



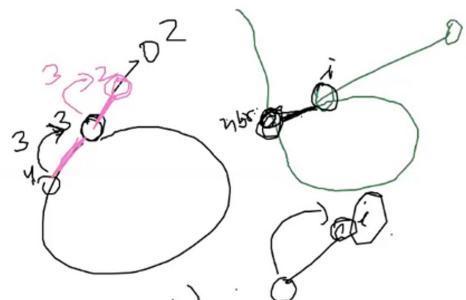


$\text{rank}[i] = -1$

```

int dts ( int i, int prev_rank )
{
    rank[i] = prev_rank + 1;
    for ( nbr ) {
        if ( rank[nbr] == -1 ) {
            if ( rank[nbr] == dts( nbr, prev_rank ) ) {
                rank[nbr] = prev_rank + 1;
            } else if ( rank[nbr] != prev_rank ) {
                continue;
            } else v = min( rank[nbr], prev_rank );
        }
        rank[i] = v;
    }
    return rank[i];
}

```



< 8 > C

**DWT | 2 May| Graph 8** (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)

**from Demux Support** [ode.com/problems/critical-connections-in-a-network/](https://ode.com/problems/critical-connections-in-a-network/)

**LeetCode** Explore Problems Mock Contest Discuss Store

We're hiring, apply today! Premium

**Description** **Solution** **Discuss (476)** **Submissions**

**1192. Critical Connections in a Network**

**Hard** 2431 121 Add to List Share

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:**

```

public:
    int dfs (vector<vector<int>>& graph, int i, vector<int>& rank, vector<vector<int>& res, int prev_rank, int n) {
        rank[i] = prev_rank + 1;
        int t = INT_MAX;

        for (const auto& nbr : graph[i]) {
            if (rank[nbr] == -1) {
                int k = dfs(graph, nbr, rank, res, prev_rank+1, n);
                if (k > rank[i])
                    res.push_back({i, nbr});
                t = min(t, k);
            }
            else if (rank[nbr] == n+1)
                continue;
            else if (rank[nbr] != prev_rank)
                t = min(t, rank[nbr]);
        }

        rank[i] = n + 1;
        return t;
    }

vector<vector<int>> criticalConnections(int n, vector<vector<int>& connections) {
    // Construct the graph.
    vector<vector<int>> graph(n);

    for (const auto& edge : connections) {
        graph[edge[0]].push_back(edge[1]);
        graph[edge[1]].push_back(edge[0]);
    }

    // Tarzan's Algorithm.
    vector<vector<int>> res;
    vector<int> rank(n, -1);

    int k = dfs(graph, 0, rank, res, 0, n);

    return res;
}

```

2:16:31

**1192. Critical Connections in a Network**

**Hard** 2431 121 Add to List Share

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:**

```

int n) {
    rank[i] = prev_rank + 1;
    int t = INT_MAX;

    for (const auto& nbr : graph[i]) {
        if (rank[nbr] == -1) {
            int k = dfs(graph, nbr, rank, res, prev_rank+1, n);
            if (k > rank[i])
                res.push_back({i, nbr});
            t = min(t, k);
        }
        else if (rank[nbr] == n+1)
            continue;
        else if (rank[nbr] != prev_rank)
            t = min(t, rank[nbr]);
    }

    rank[i] = n + 1;
    return t;
}

vector<vector<int>> criticalConnections(int n, vector<vector<int>& connections) {
    // Construct the graph.
    vector<vector<int>> graph(n);

    for (const auto& edge : connections) {
        graph[edge[0]].push_back(edge[1]);
        graph[edge[1]].push_back(edge[0]);
    }

    // Tarzan's Algorithm.
    vector<vector<int>> res;
    vector<int> rank(n, -1);

    int k = dfs(graph, 0, rank, res, 0, n);

    return res;
}

```

Console Contribute i Run Code Submit

- <https://leetcode.com/problems/find-the-city-with-the-smallest-number-of-neighbors-at-a-threshold-distance/>
  - <https://leetcode.com/problems/critical-connections-in-a-network/>
- // Leetcode 1334. Find the City With the Smallest Number of Neighbors at a Threshold Distance: <https://leetcode.com/problems/find-the-city-with-the-smallest-number-of-neighbors-at-a-threshold-distance/>

// Leetcode 1192. Critical Connections in a Network::  
<https://leetcode.com/problems/critical-connections-in-a-network/>