Graphs Demux

31 January 2022 13:28

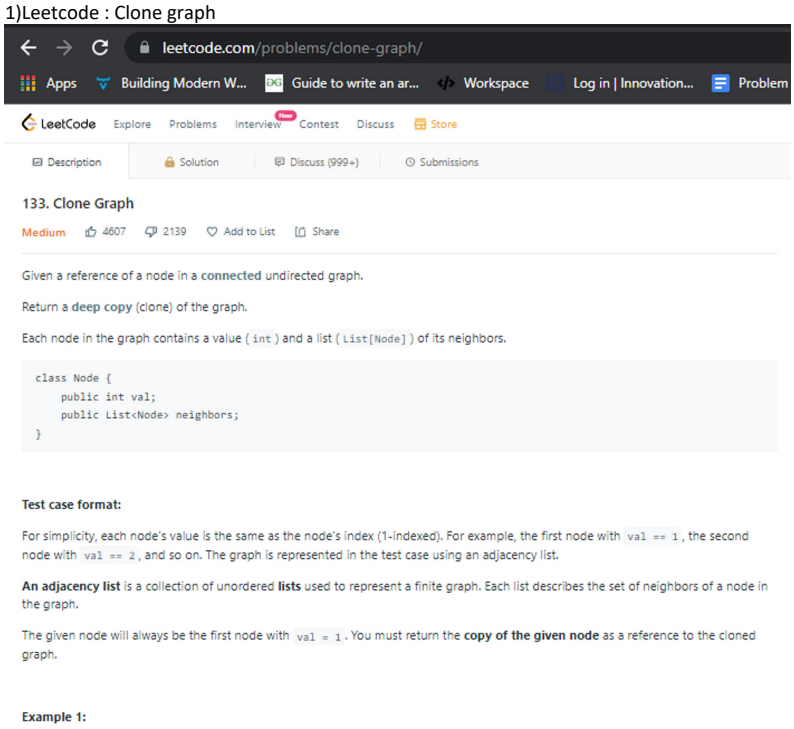
==========

Question List

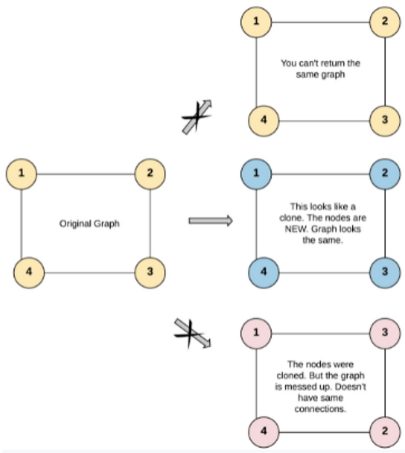
Lec	Q.No	Q.name	Link	Remarks
1	200	Number of islands	https://leetcode.com/problems/number-of-islands/	
		Clone Graph **		
		Number of provinces	https://leetcode.com/problems/number-of-provinces/	
	207	Leetcode 207. Course Schedule ::		

Lec-1

1)Leetcode : Clone graph



Example 1:




```
Input: adjList = [[2,4],[1,3],[2,4],[1,3]]
Output: [[2,4],[1,3],[2,4],[1,3]]
Explanation: There are 4 nodes in the graph.
1st node (val = 1)'s neighbors are 2nd node (val = 2) and 4th node (val = 4).
2nd node (val = 2)'s neighbors are 1st node (val = 1) and 3rd node (val = 3).
```

Example 2:

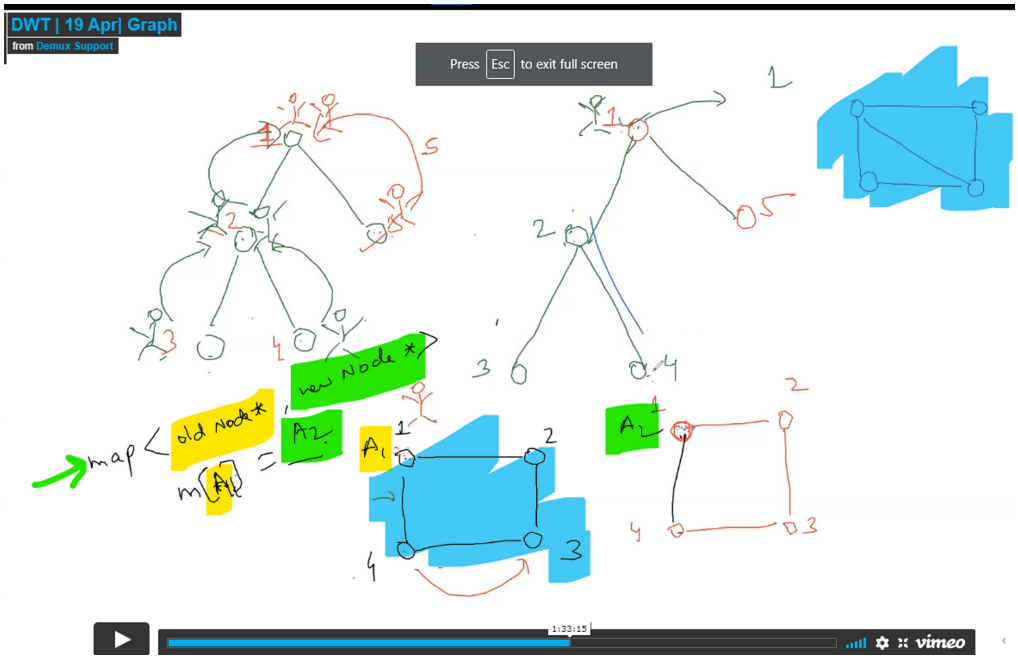
```
Input: adjList = [[]]
Output: [[]]
Explanation: Note that the input contains one empty list. The graph consists of only one node with
val = 1 and it does not have any neighbors.
```

Example 3:

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

Constraints:

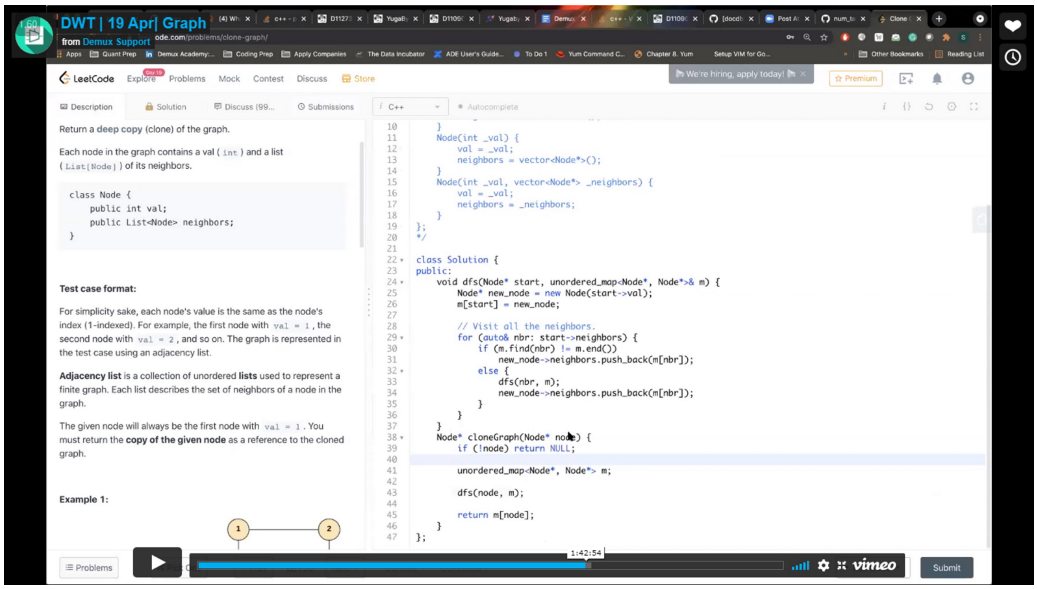
- The number of nodes in the graph is in the range `[0, 100]`.
- `1 <= Node.val <= 100`
- `Node.val` is unique for each node.
- There are no repeated edges and no self-loops in the graph.
- The Graph is connected and all nodes can be visited starting from the given node.



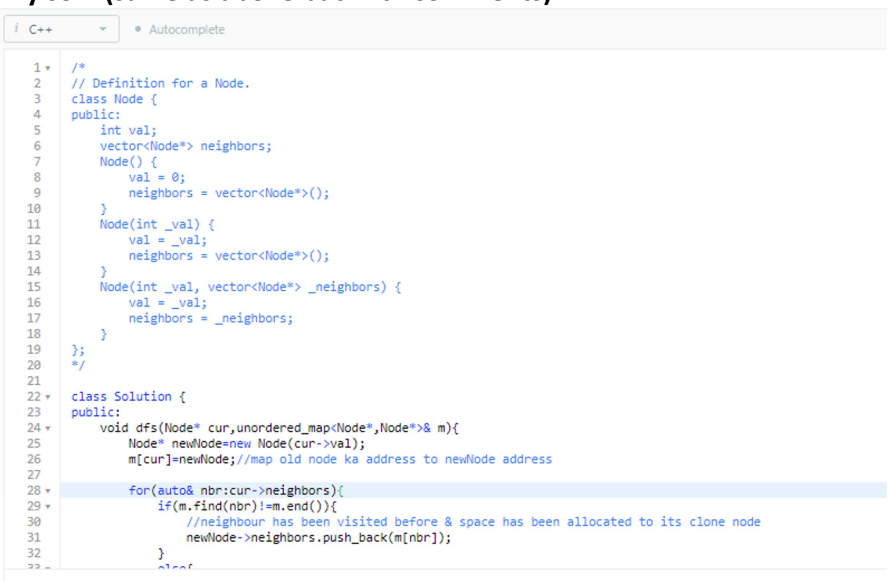
Clone will have issue in case of cyclic graphs if we just do normal dfs approach, we

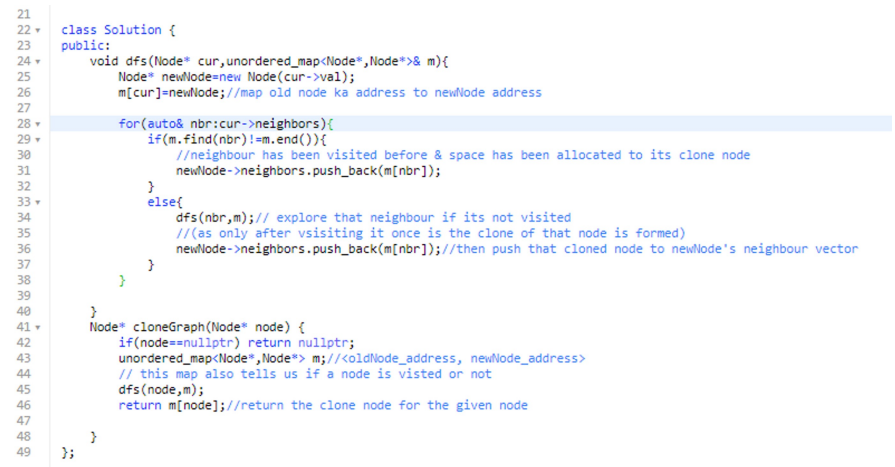
have

Sir ka soln::



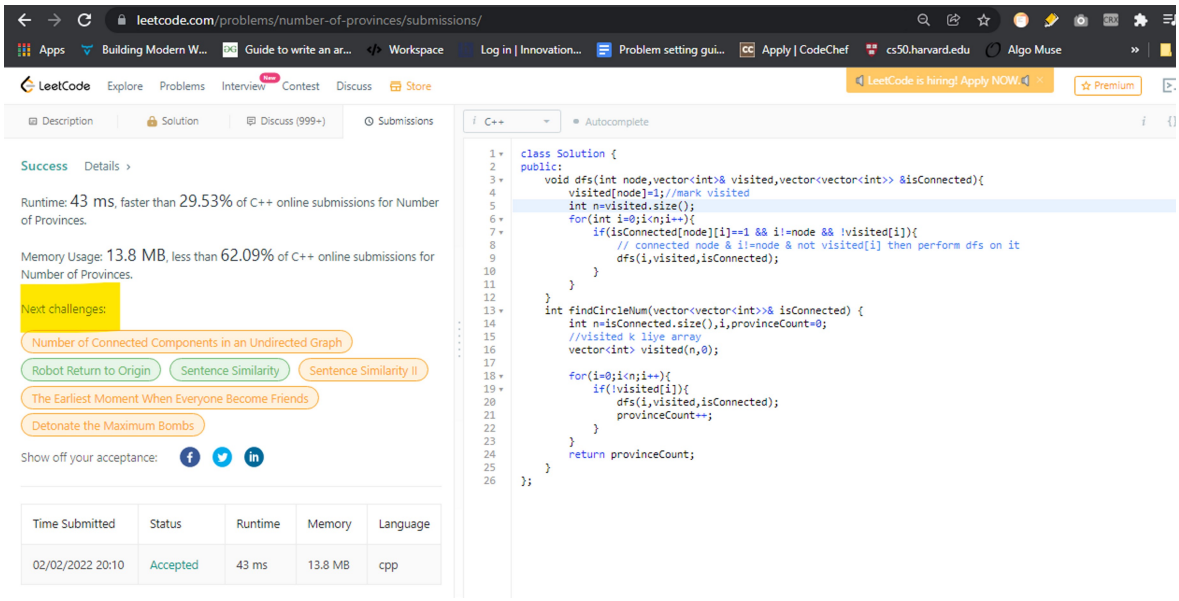
My soln (same as above but with comments)





---

2)Leetcode : Number of provinces



---

3) Leetcode : 547. Number of Provinces :: https://leetcode.com/problems/number-of-provinces/

547. Number of Provinces

Medium 4568 220 Add to List Share

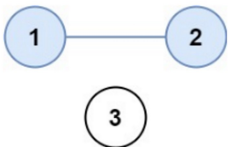
There are `n` cities. Some of them are connected, while some are not. If city `a` is connected directly with city `b`, and city `b` is connected directly with city `c`, then city `a` is connected indirectly with city `c`.

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

You are given an `n x 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.

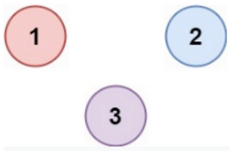
Return *the total number of **provinces***.

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 <= n <= 200`
- `n == isConnected.length`
- `n == isConnected[i].length`
- `isConnected[i][j]` is `1` or `0`.
- `isConnected[i][i] == 1`
- `isConnected[i][j] == isConnected[j][i]`

```cpp
class Solution {
public:
    void dfs(int node,vector<int>& visited,vector<vector<int>> &isConnected){
        visited[node]=1;//mark visited
        int n=visited.size();
        for(int i=0;i<n;i++){
            if(isConnected[node][i]==1 && i!=node && !visited[i]){
                // connected node & i!=node & not visited[i] then perform dfs on it
                dfs(i,visited,isConnected);
            }
        }
    }
    int findCircleNum(vector<vector<int>>& isConnected) {
        int n=isConnected.size(),i,provinceCount=0;
        //visited k liye array
        vector<int> visited(n,0);

        for(i=0;i<n;i++){
            if(!visited[i]){
                dfs(i,visited,isConnected);
                provinceCount++;
            }
        }
        return provinceCount;
    }
};
```

---

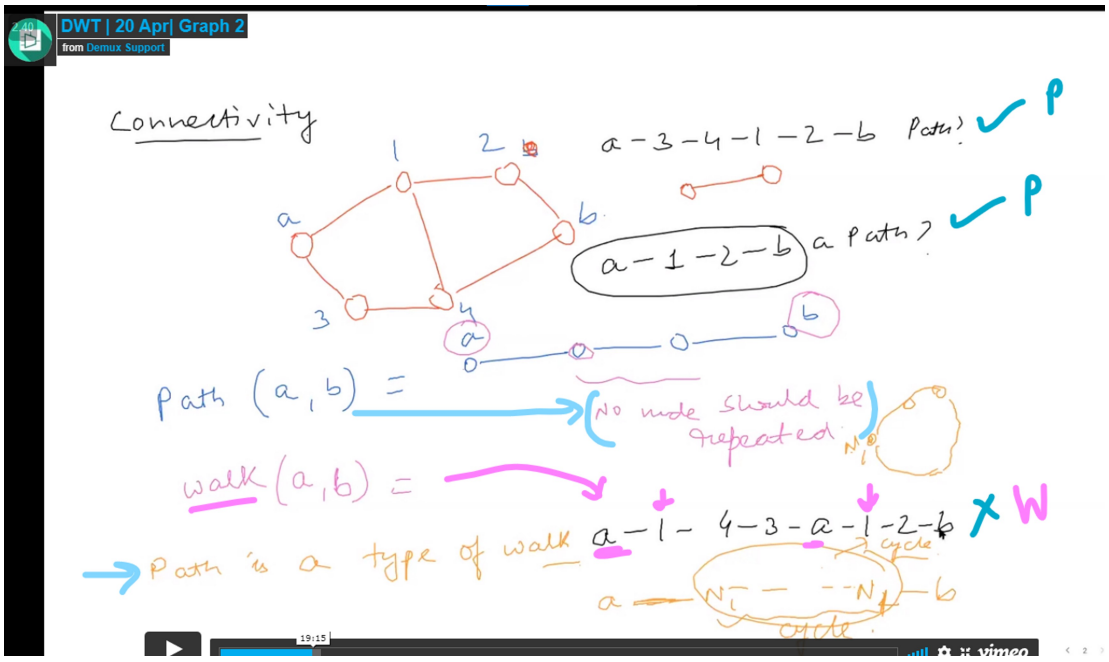
4) Leetcode :

==========

==========

========

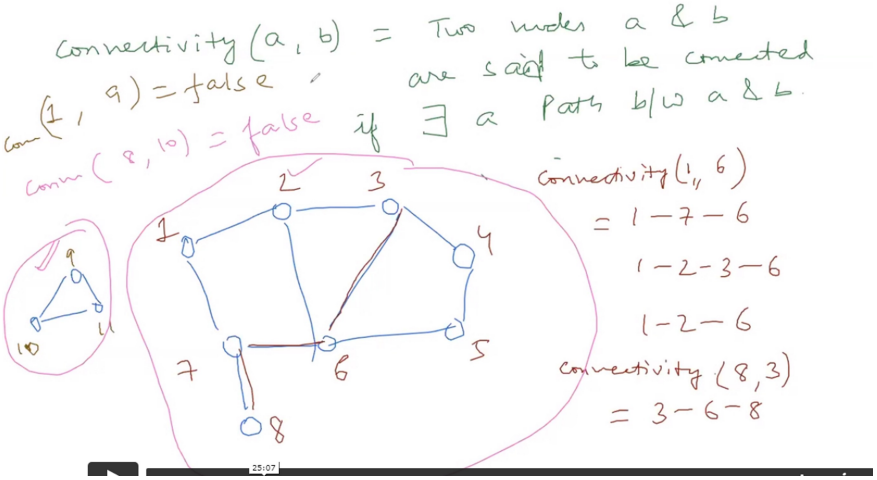
Lec-2

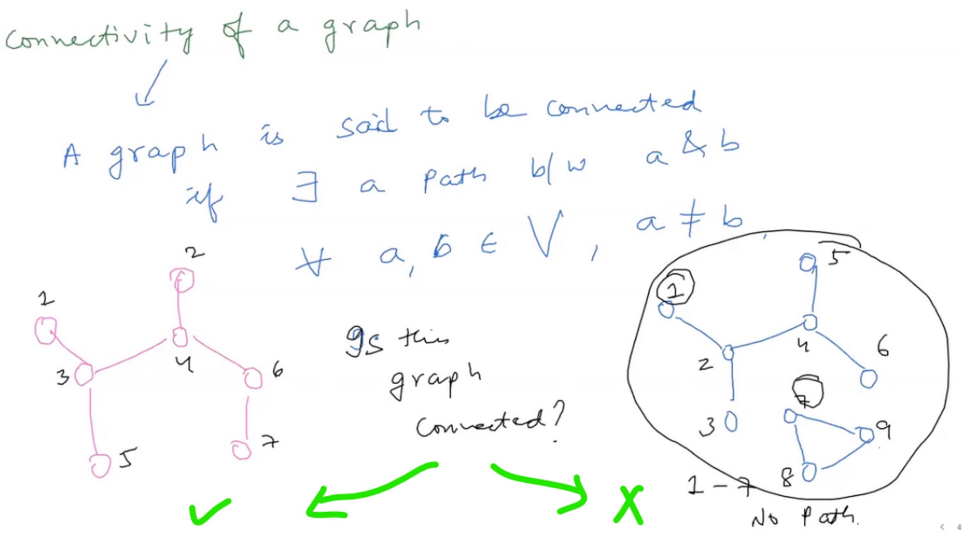


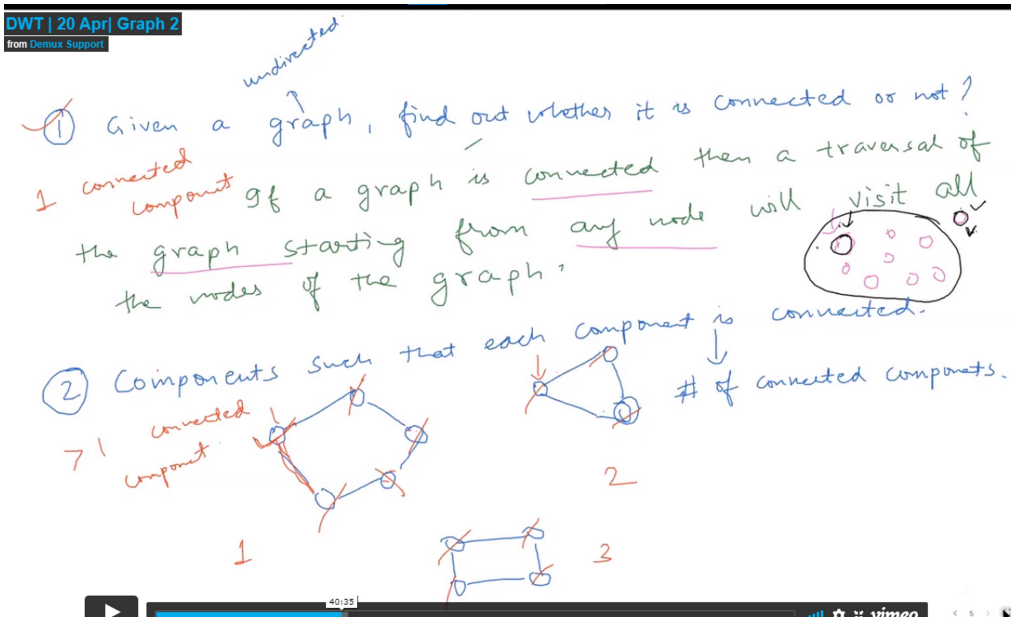
- Path(a,b) - no nodes are repeated => path contains no cycles
- Walk(a,b) - any sequence of nodes that start at a and end at b. Walks can contain cycles

Path is a special type of walk!!

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



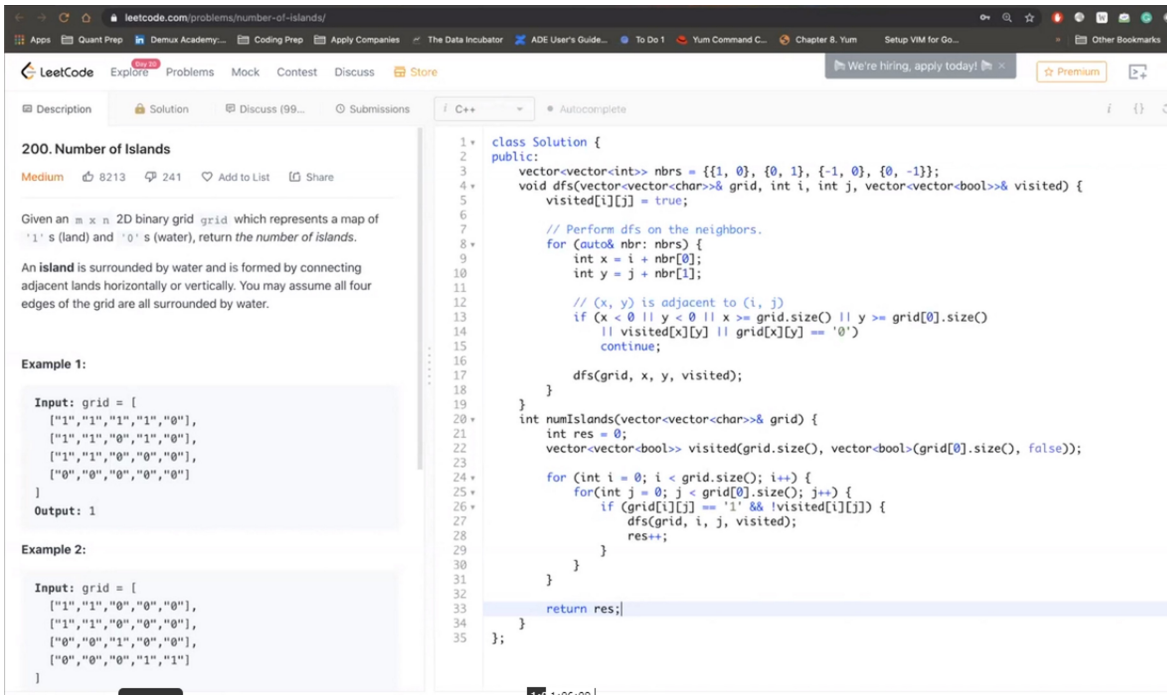




Modelling graph Problems

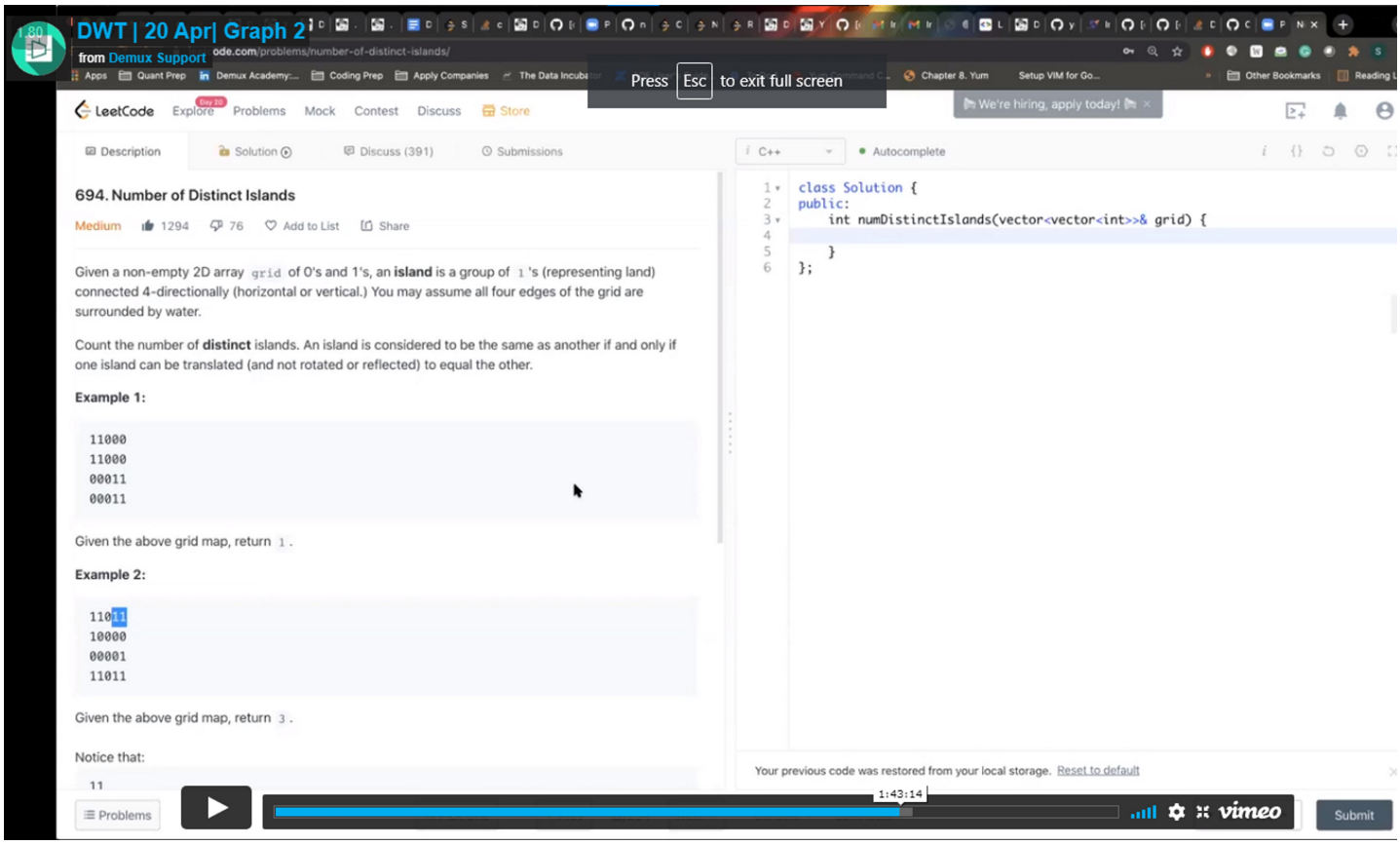
- Explicitly given graph
- No graph explicitly given

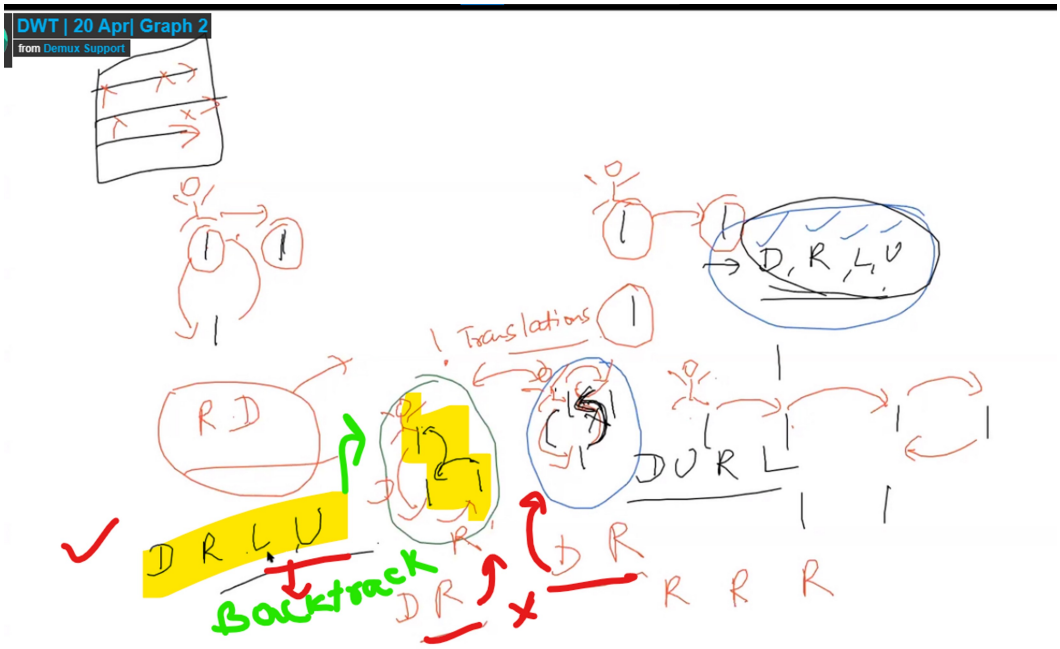
1. Define your graph for the problem
   - Trivial
   - Non-trivial
2. Bucket the Problem one of the existing Problem
   - Connectivity



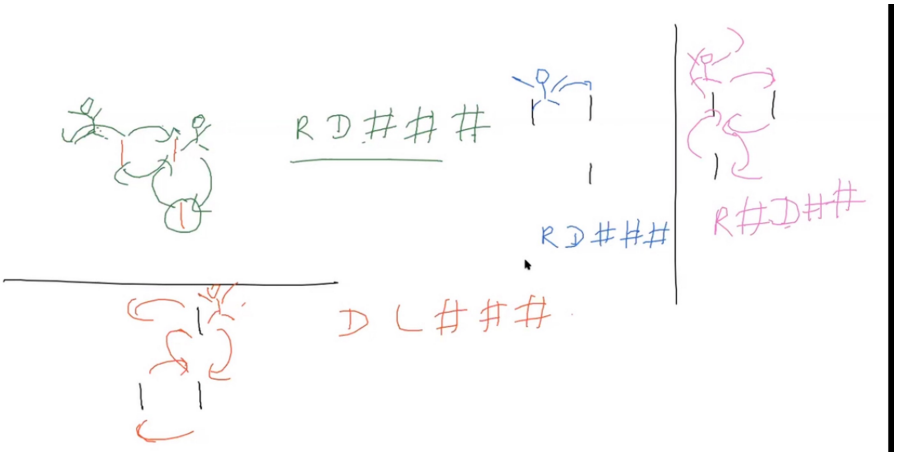
==========

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





- 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 form 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
```

```cpp
class Solution {
public:
    vector<vector<int>> nbrs = {{1, 0}, {0, 1}, {-1, 0}, {0, -1}};
    vector<char> dirs = {'R', 'U', 'L', 'D'};
    void dfs(vector<vector<int>>& grid, int i, int j, vector<vector<bool>>& visited, string& island) {
        visited[i][j] = true;

        for (int k = 0; k < nbrs.size(); k++) {
            int x = i + nbrs[k][0];
            int y = j + nbrs[k][1];

            if (x < 0 || y < 0 || x >= grid.size() || y >= grid[0].size() || visited[x][y]
                || grid[x][y] == 0)
                continue;

            island += dirs[k];
            dfs(grid, x, y, visited, island);
            island += '#';
        }
    }
    int numDistinctIslands(vector<vector<int>>& grid) {
        int m = grid.size(), n = grid[0].size(), i, j;
        unordered_set<string> islands;
        vector<vector<bool>> visited(m, vector<bool>(n, false));

        for(i = 0; i < m; i++) {
            for(j = 0; j < n; j++) {
                if (grid[i][j] == 1 && !visited[i][j]) {
                    string island_pattern = "";
                    dfs(grid, i, j, visited, island_pattern);
                    if (islands.find(island_pattern) == islands.end())
                        islands.insert(island_pattern);
                }
            }
        }
```

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`.

```cpp
    vector<char> dirs = {'R', 'U', 'L', 'D'};
    void dfs(vector<vector<int>>& grid, int i, int j, vector<vector<bool>>& visited, string& island) {
        visited[i][j] = true;

        for (int k = 0; k < nbrs.size(); k++) {
            int x = i + nbrs[k][0];
            int y = j + nbrs[k][1];

            if (x < 0 || y < 0 || x >= grid.size() || y >= grid[0].size() || visited[x][y]
                || grid[x][y] == 0)
                continue;

            island += dirs[k];
            dfs(grid, x, y, visited, island);
            island += '#';
        }
    }
    int numDistinctIslands(vector<vector<int>>& grid) {
        int m = grid.size(), n = grid[0].size(), i, j;
        unordered_set<string> islands;
        vector<vector<bool>> visited(m, vector<bool>(n, false));

        for(i = 0; i < m; i++) {
            for(j = 0; j < n; j++) {
                if (grid[i][j] == 1 && !visited[i][j]) {
                    string island_pattern = "";
                    dfs(grid, i, j, visited, island_pattern);
                    if (islands.find(island_pattern) == islands.end())
                        islands.insert(island_pattern);
                }
            }
        }

        return (int)islands.size();
    }
};
```

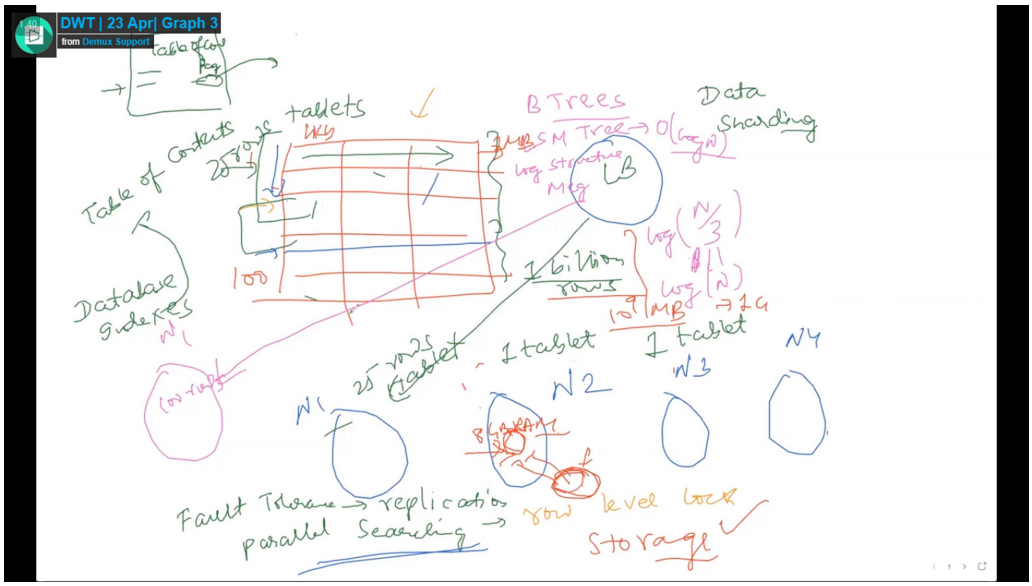
Your previous code was restored from your local storage. Reset to default

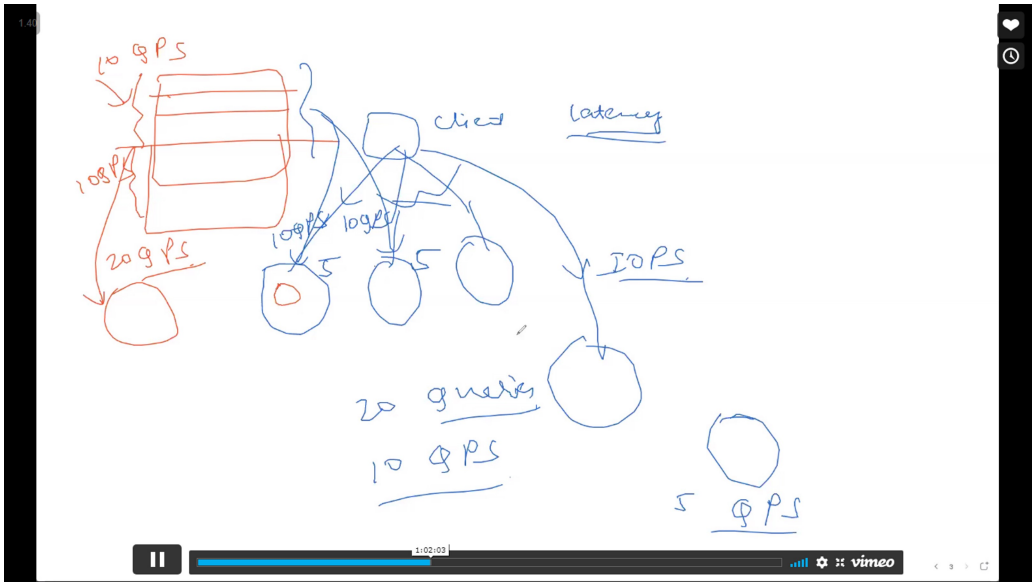
==========

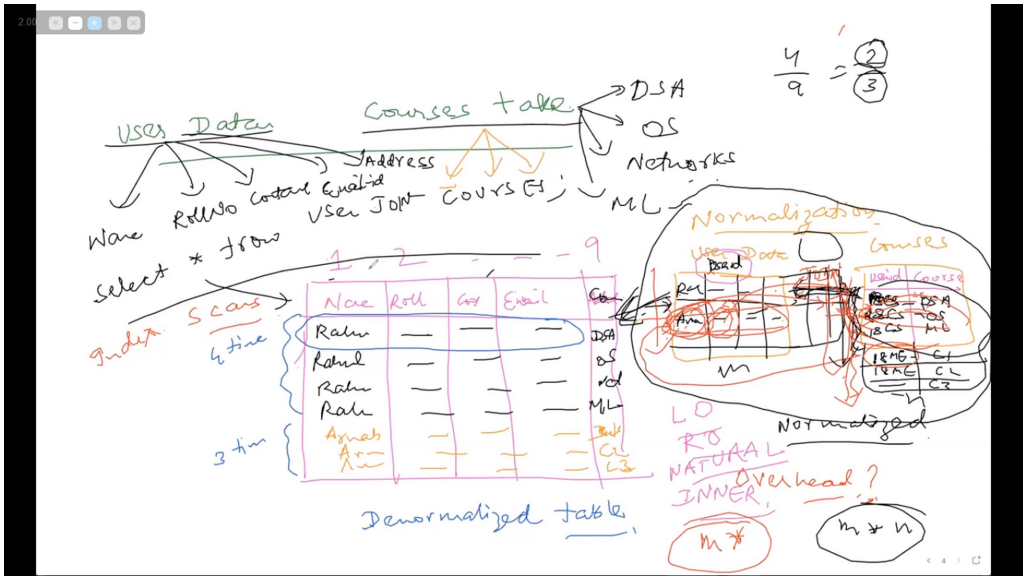
LEC 3:

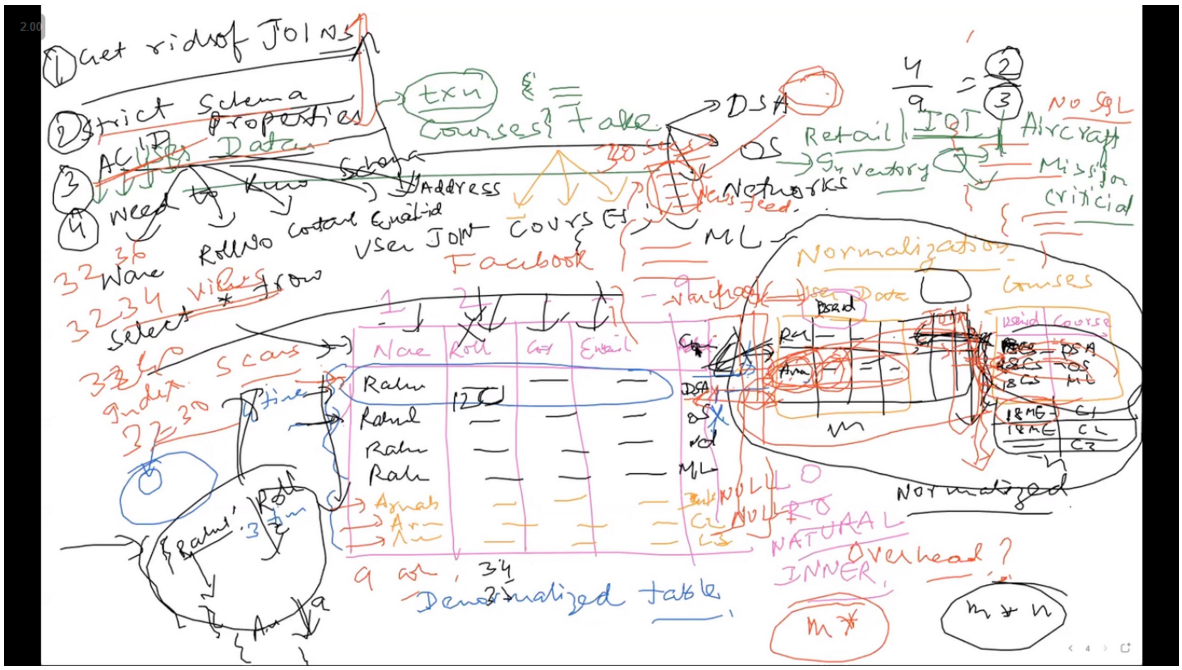
Brief discussion on Load balancers, data sharding and its reasons:

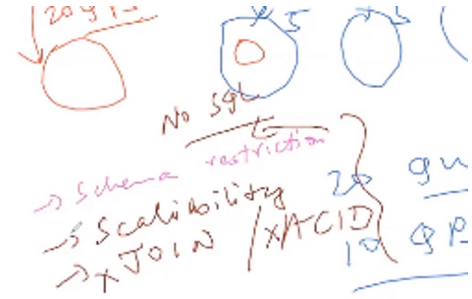
- Fault tolerance -> replication of data to avoid data loss if one server gets destroyed
- Parallel searching : since indexes improve searching time
- Storage - by dividing data into blocks/tablets and storing them into different nodes, we are also increasing the cached memory for each node as RAM is 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











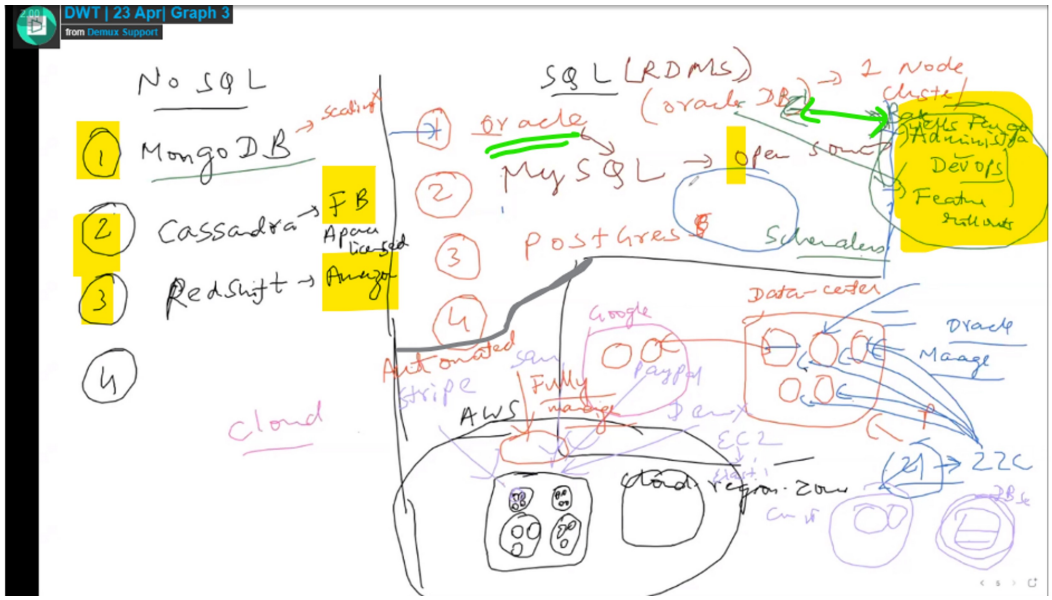
WHY need of JSON arised (when SQL was there)? Basically different btween SQL and NoSQL ::

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

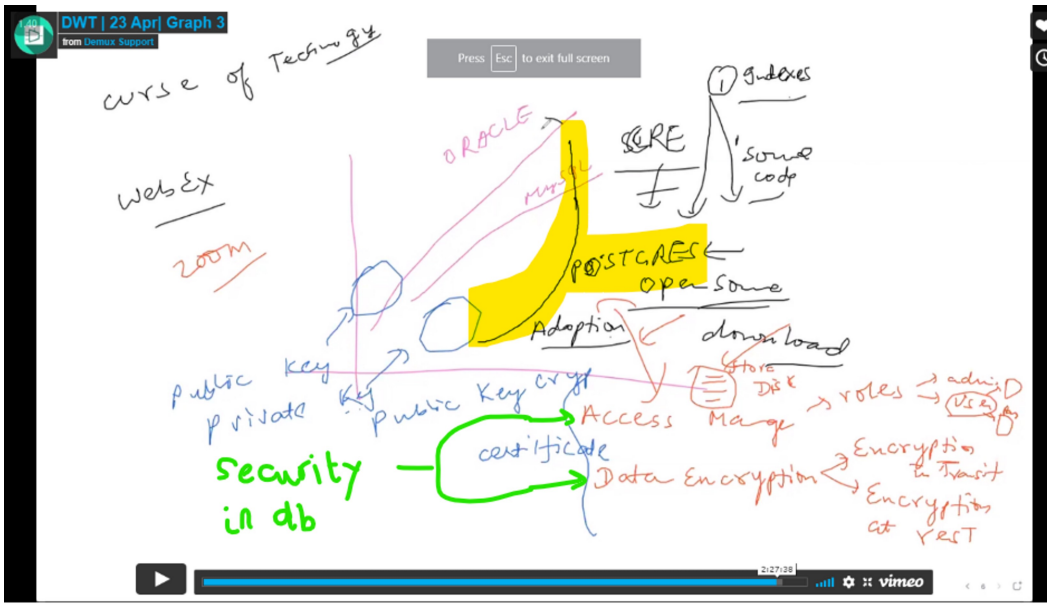
b. banking Reason: for example in banking if during fund transfer money got out of your account and the receiving end wale ka server crashes at that point then you cannot afford to loose your money ,here ACID properties of SQL save the day
c. up to date data access of inventory items
d. Concurrent data 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 DISTRIBUITED SQL database .



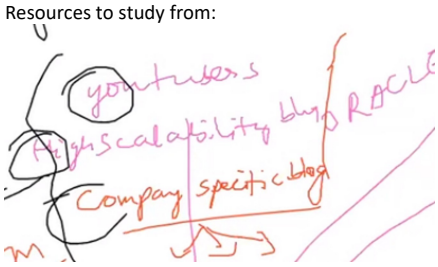
- Postgres is the most growing SQL database at present(see the adoption chart)



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

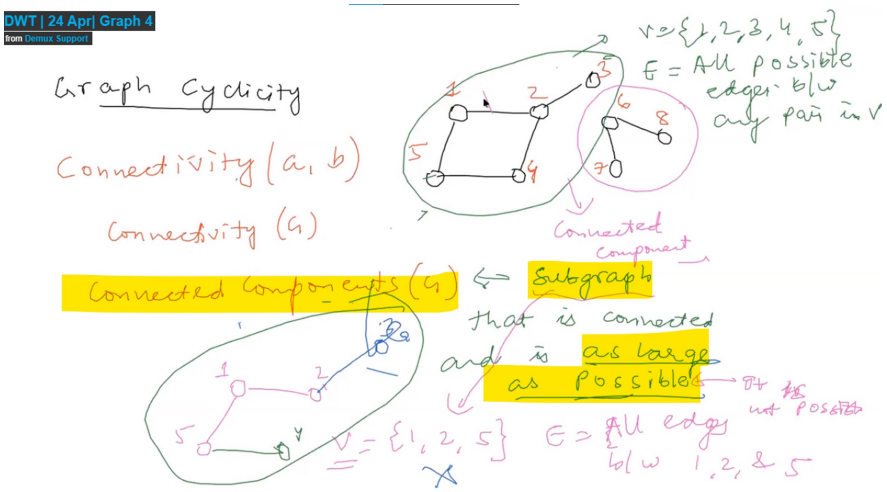
- Yugabyte is also doing that only : its mission is to have one single database for any type of use: like you come say these are the features you want and you get it (like already defined in the model)

Resources to study from:

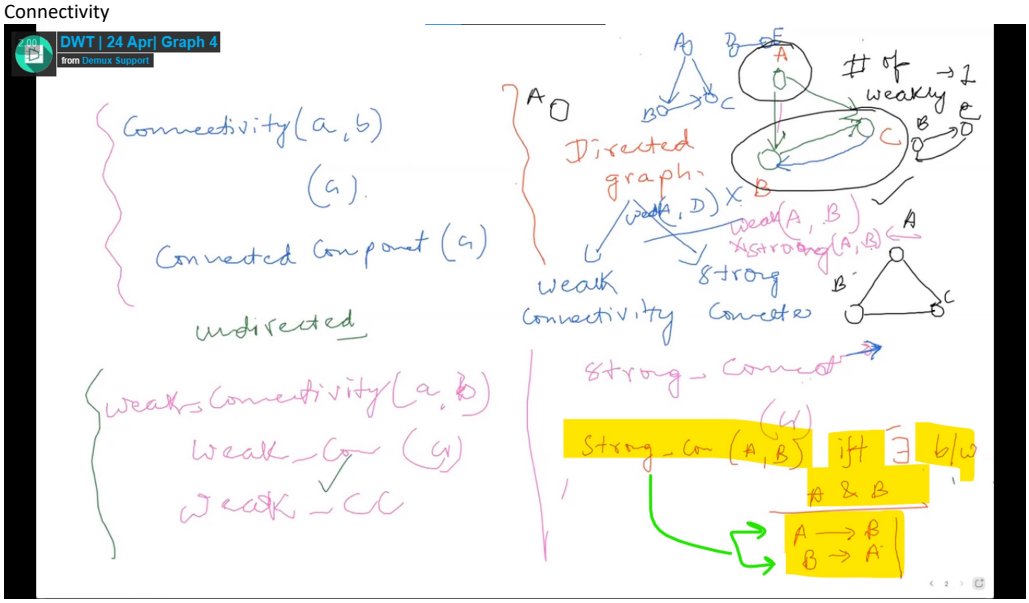


==========
====

LEC 4:



- Cyclicity
- Connectivity



-

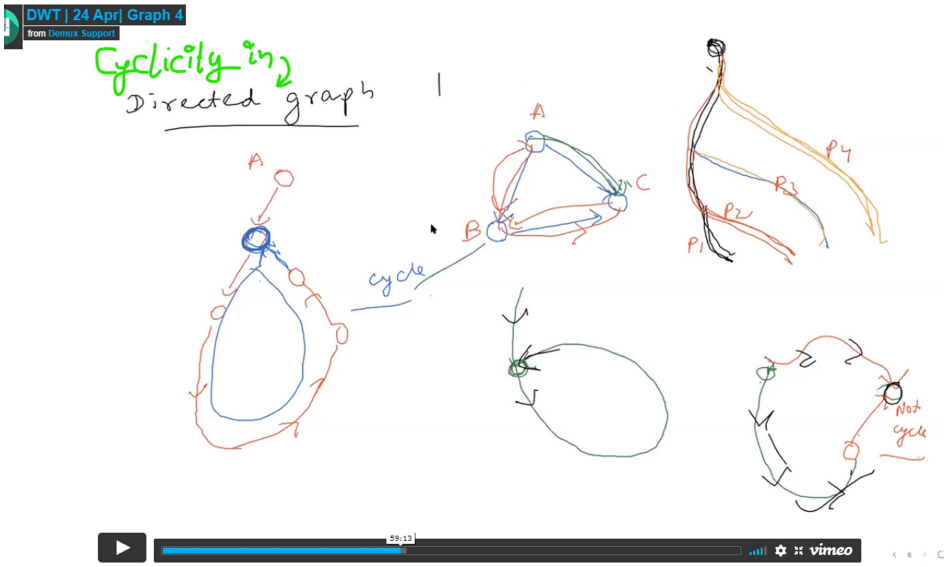
Cyclicity in undirected & directed graph:

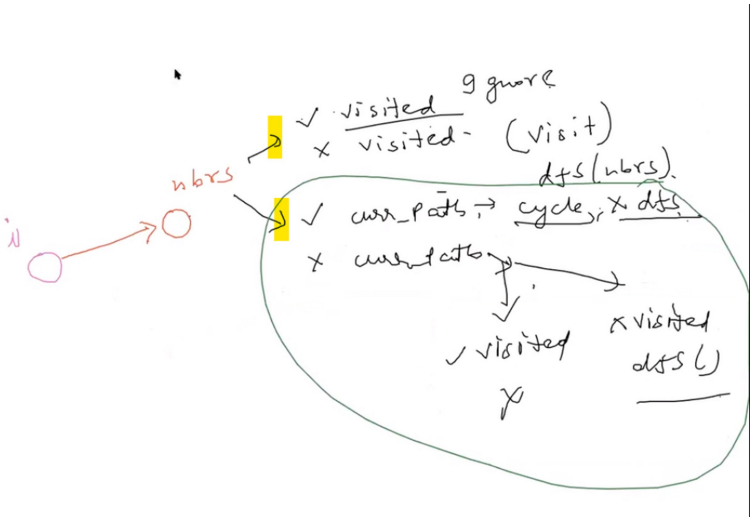
- Undirected : if we visit a start node twice from 2 different nodes then we have a cycle
- Directed: if while visiting a path we revisit a already visited node then that

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

Q)

```
// Leetcode 207. Course Schedule ::
```





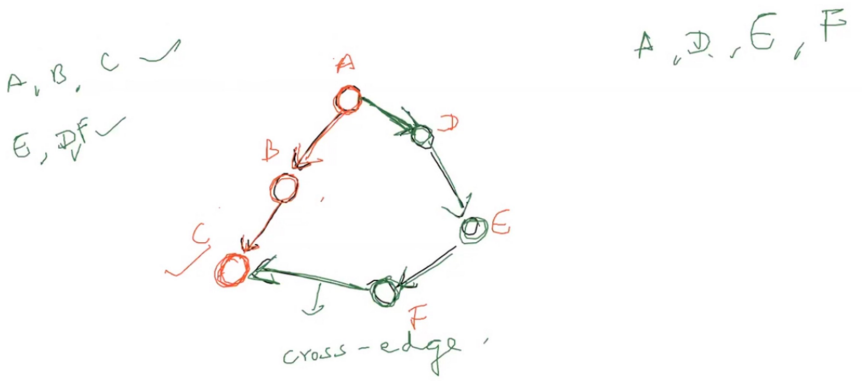
Graphs
Demux

Audio recording started: 17:43 09 February 2022

Graphs
Demux

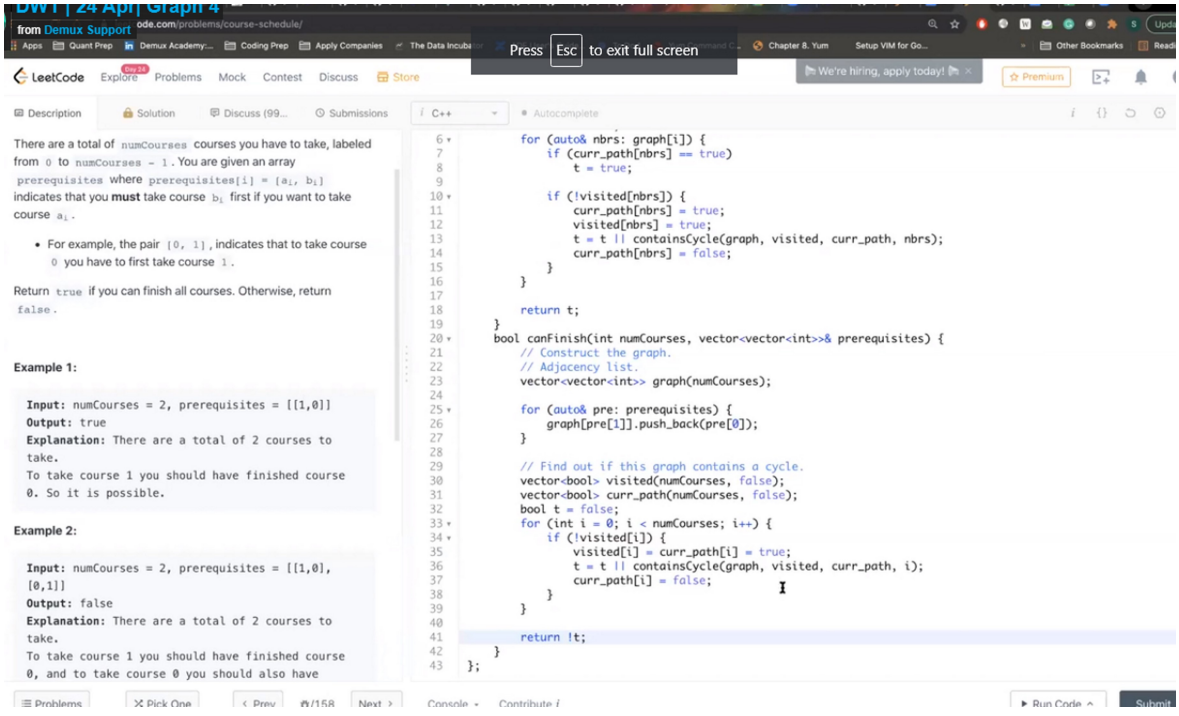
Audio recording started: 17:46 09 February 2022

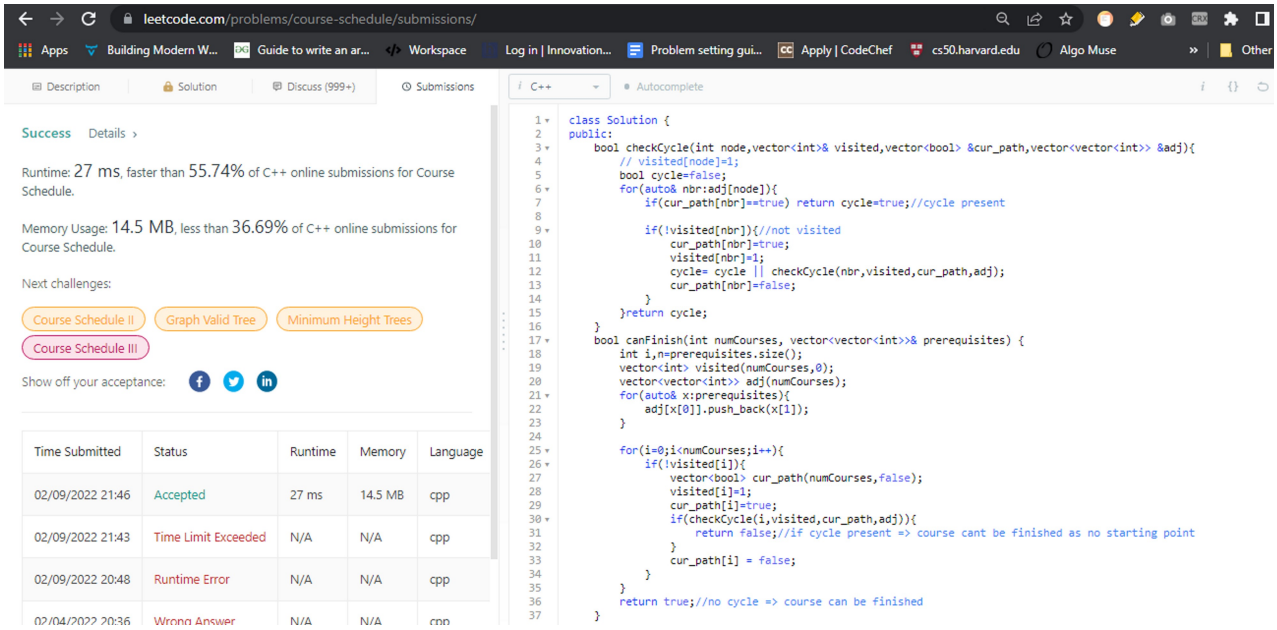
Audio recording started: 17:43 09 February 2022



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

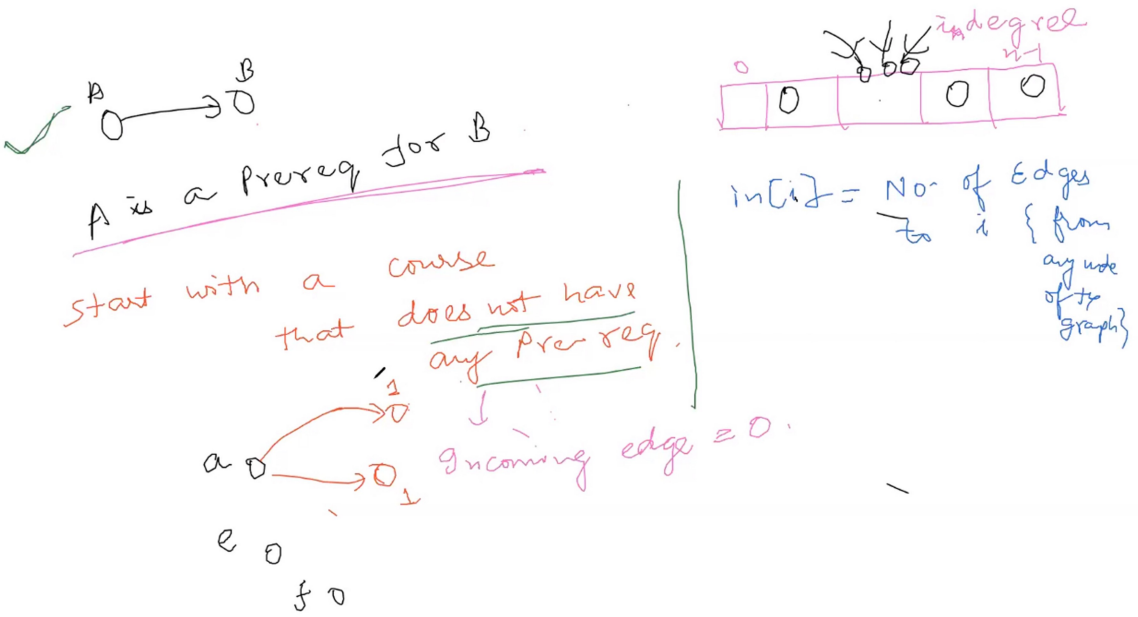
```cpp
class Solution {
public:
    bool containsCycle(vector<vector<int>>& graph, vector<bool>& visited, vector<bool>& curr_path, int i) {

        bool t = false;
        for (auto& nbrs: graph[i]) {
            if (curr_path[nbrs] == true)
                t = true;

            if (!visited[nbrs]) {
                curr_path[nbrs] = true;
                visited[nbrs] = true;
                t = t || containsCycle(graph, visited, curr_path, nbrs);
                curr_path[nbrs] = false;
            }
        }

        return t;
    }
    bool canFinish(int numCourses, vector<vector<int>>& prerequisites) {
        // Construct the graph.
        // Adjacency list.
        vector<vector<int>> graph(numCourses);

        for (auto& pre: prerequisites) {
            graph[pre[1]].push_back(pre[0]);
        }

        // Find out if this graph contains a cycle.
        vector<bool> visited(numCourses, false);
        vector<bool> curr_path(numCourses, false);
        bool t = false;
        for (int i = 0; i < numCourses; i++) {
            if (!visited[i]) {
                visited[i] = curr_path[i] = true;
                t = t || containsCycle(graph, visited, curr_path, i);
```



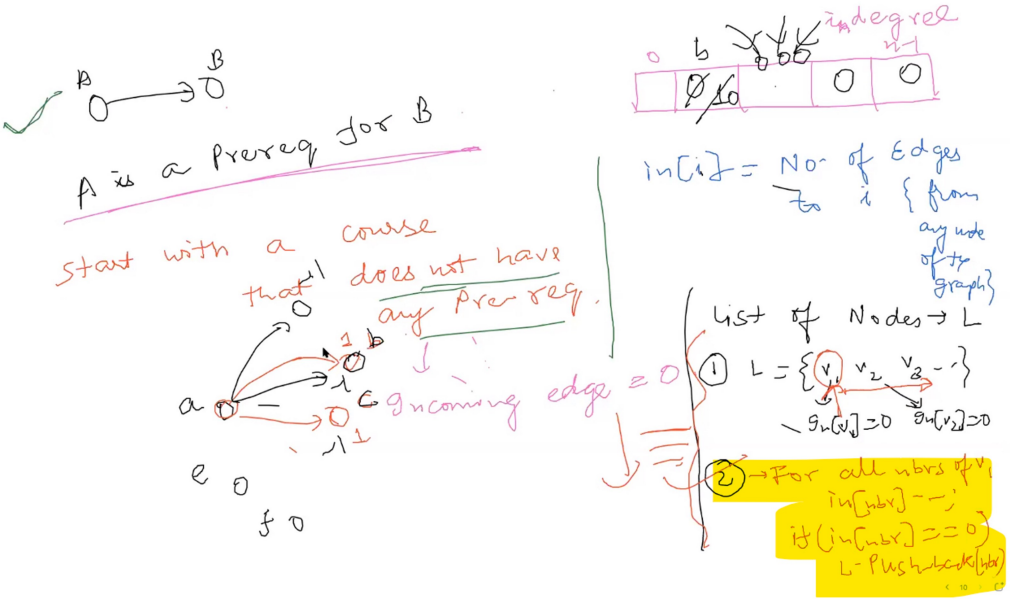


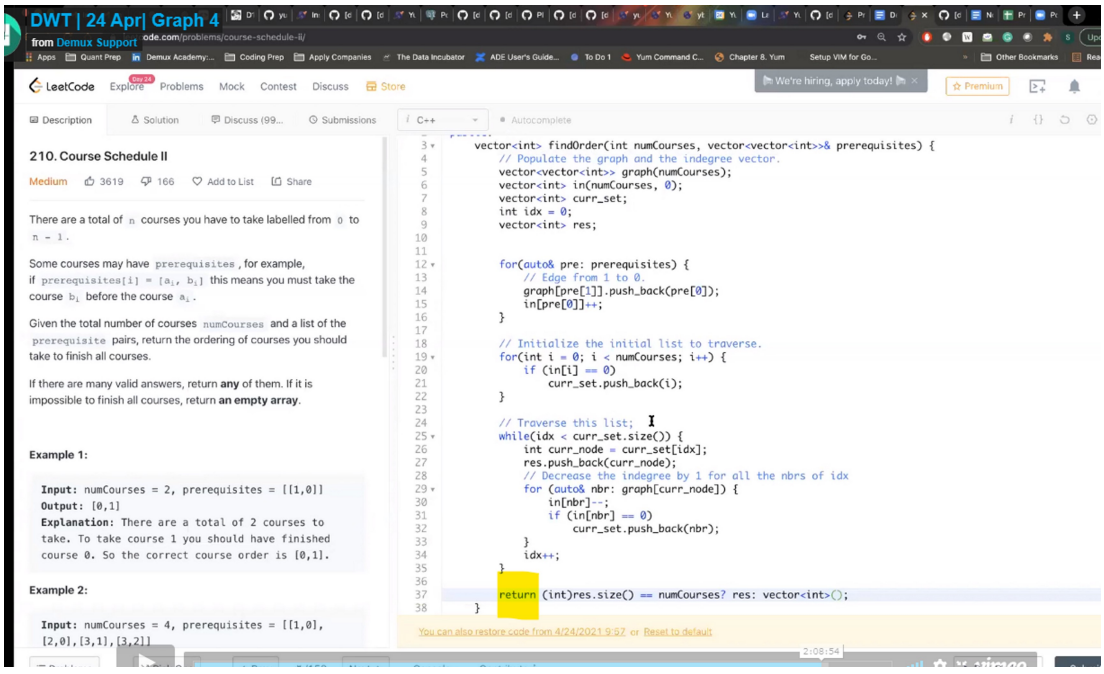
==========

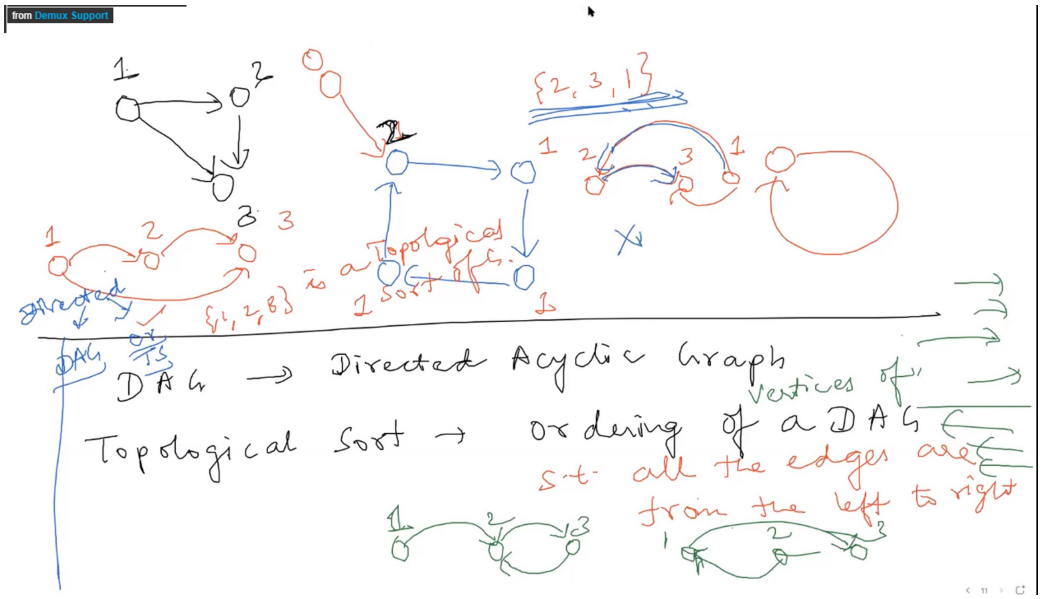
Course schedule ii:



Visit in increasing order of indegree


A is a Prereq for B
Start with a course that does not have any Prereq.
indegree
in[i] = No. of Edges to i {from any node of the graph}
incoming edge = 0
List of Nodes → L
For all nbrs of v
if(in[nbr] == 0)
L - Push_back(nbr)


DWT | 24 Apr| Graph 4
from Demux Support
LeetCode
Explore
Problems
Mock
Contest
Discuss
Store
Description
Solution
Discuss (99...
Submissions
C++
Autocomplete
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 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 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]]
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;
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>();
}
You can also restore code from 4/24/2021 9:57 or Reset to default
2:08:54

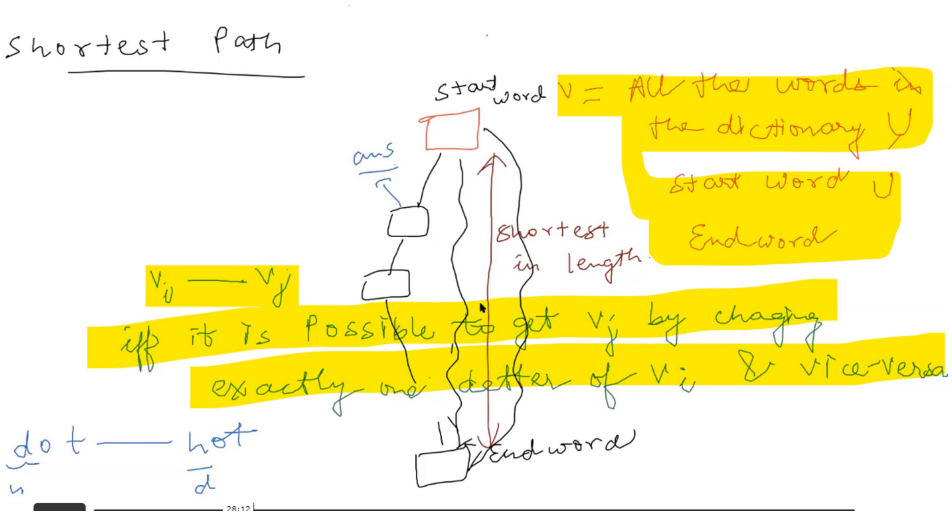


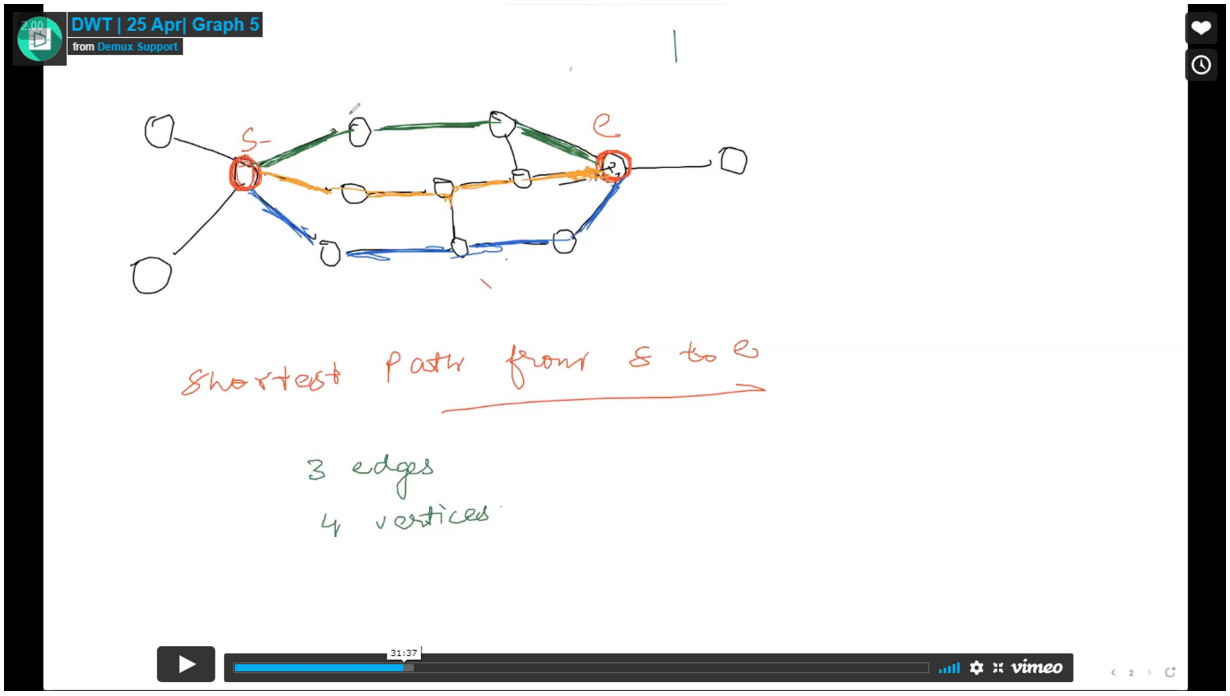
- Topological sort:: ordering of a DAG such that all the edges are in one direction (eg all edges are from either left to right/ all edges are from right to left)
- Eg in above image for

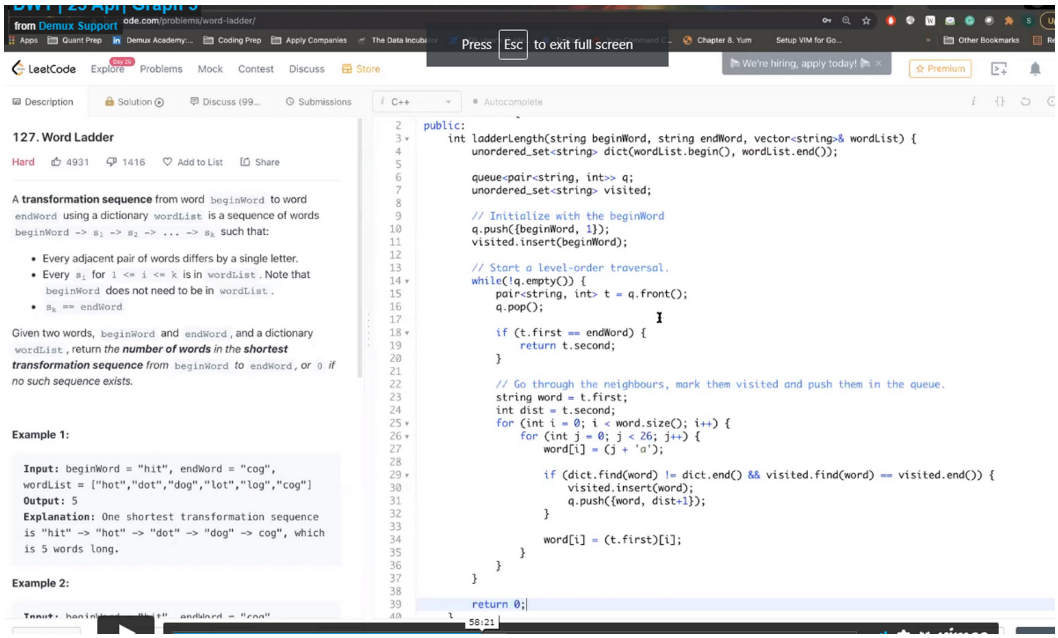
==========

Lec 5:

Q) Leetcode :Word Ladder







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 W... | 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 <= 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:

```
class Solution {
public:
    //1. model of graph : nodes as wordlist elements & beginWord  (undirected edges bw elements at distance 1)
    //2. find shortest distance between beginWord & endWord nodes : use level order traversal for it(bfs)
    int ladderLength(string beginWord, string endWord, vector<string>& wordList) {
        int ans=0,i,j,level=0;
        unordered_set<string> visited,dict;
        queue<string> curLevel;

        for(auto& word:wordList){
            dict.insert(word);
        }
        curLevel.push(beginWord); //start with the beginWord
        visited.insert(beginWord);//mark it as visited

        while(!curLevel.empty()){
            int len=curLevel.size();
            level++;// increment level
            while(len>0){
                len--;
                string cur,temp;
                cur=temp=curLevel.front();
                curLevel.pop();

                for(i=0;i<cur.size();i++){
                    for(j=0;j<26;j++){
                        cur[i] = 'a'+j;// change the i'th character in current word to all possible letters 'a' to 'z'
                        // if(cur==endWord) return level+1;
                        if(dict.find(cur)!=dict.end() && visited.find(cur)==visited.end()){
                            curLevel.push(cur);
                            visited.insert(cur);//mark visited
                            if(cur==endWord) return level+1;
                        }
                    }cur=temp;// change cur to same initial word in the queue
                }
            }
        }
        return 0;
```

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 adviced 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 W... | Guide to write an ar... | Workspace | Log in | Inno

Description | Solution | Discuss (999+) | Submissions

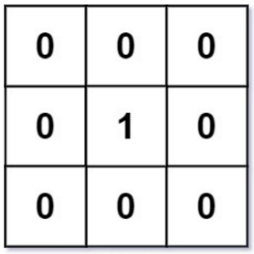
542. 01 Matrix

Medium 4088 198 Add to List Share

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

The distance between two adjacent cells is `1`.

Example 1:

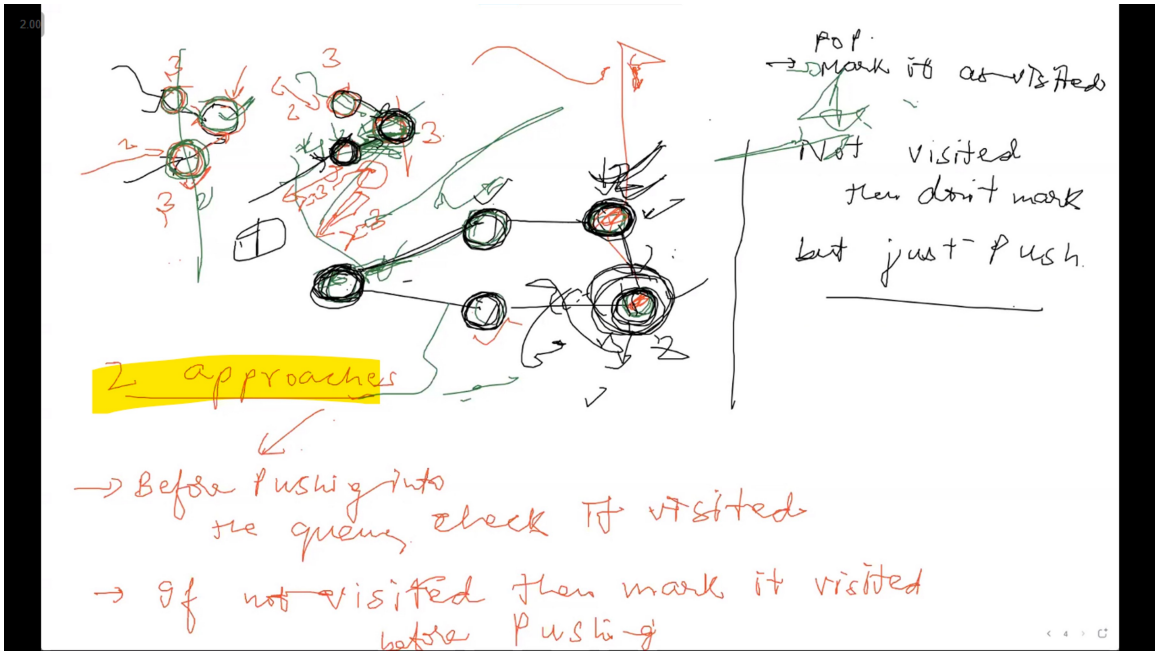


0	0	0
0	1	0
0	0	0

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

Example 2:

Problems | Pick One | Prev 542/2173



My soln::

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

Apps | Building Modern W... | 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

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

C++ | Autocomplete

```cpp
class Solution {
public:
    int dx[4]={-1,1,0,0};
    int dy[4]={0,0,-1,1};
    bool inRange(int i,int j,int n,int m){// kina like a inRange() function with modified aim to find distance
        // int r=dist.size(),c=dist[0].size(),x=1e5;
        if(i<0 || j<0 || i>=n || j>=m) return false;
        return true;
    }

    vector<vector<int>> updateMatrix(vector<vector<int>>& mat) {
        int n=mat.size(),m=mat[0].size(),i,j,distance=0;
        queue<pair<int,int>> curset;// current set to be iterated upon
        vector<vector<int>> visited(n,vector<int>(m,-1));
        for(i=0;i<n;i++){
            for(j=0;j<m;j++){
                if(mat[i][j]==0){
                    visited[i][j]=distance;//initial distance =0 for all 0's in th matrix
                    curset.push({i,j});
                }
            }
        }

        while(!curset.empty()){
            int len=curset.size();
            distance++;//increment the level for level order traversal
            while(len>0){
                len--;
                i=curset.front().first,j=curset.front().second;
                curset.pop();
                for(int x=0;x<4;x++){
                    int a=i+dx[x],b=j+dy[x];
                    if(inRange(a,b,n,m) && visited[a][b]==-1){
                        //inrange and unvisited
                        visited[a][b]=distance;
                        curset.push({a,b});
                    }
                }
```

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

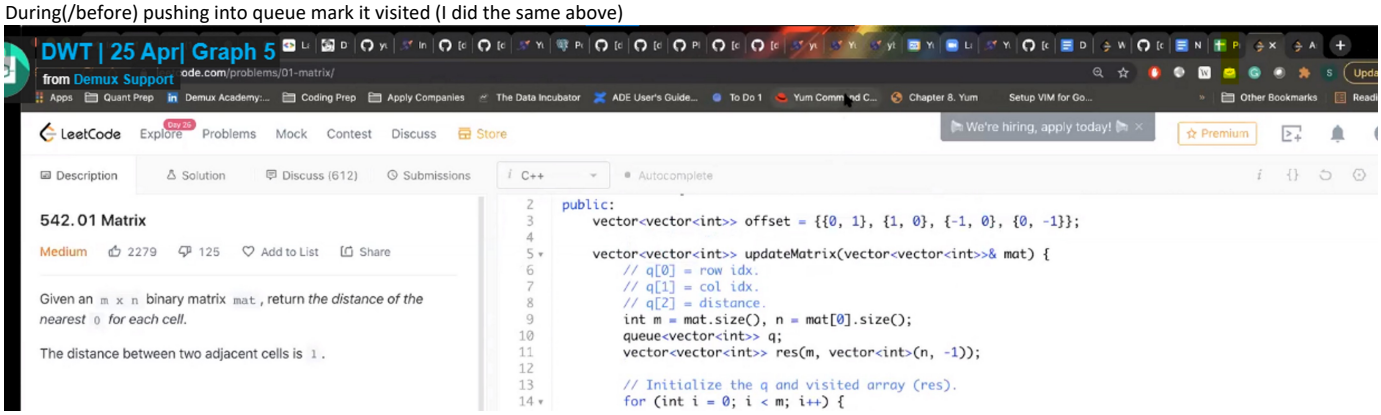
Minimum Operations to Remove Adjacent Ones in Matrix

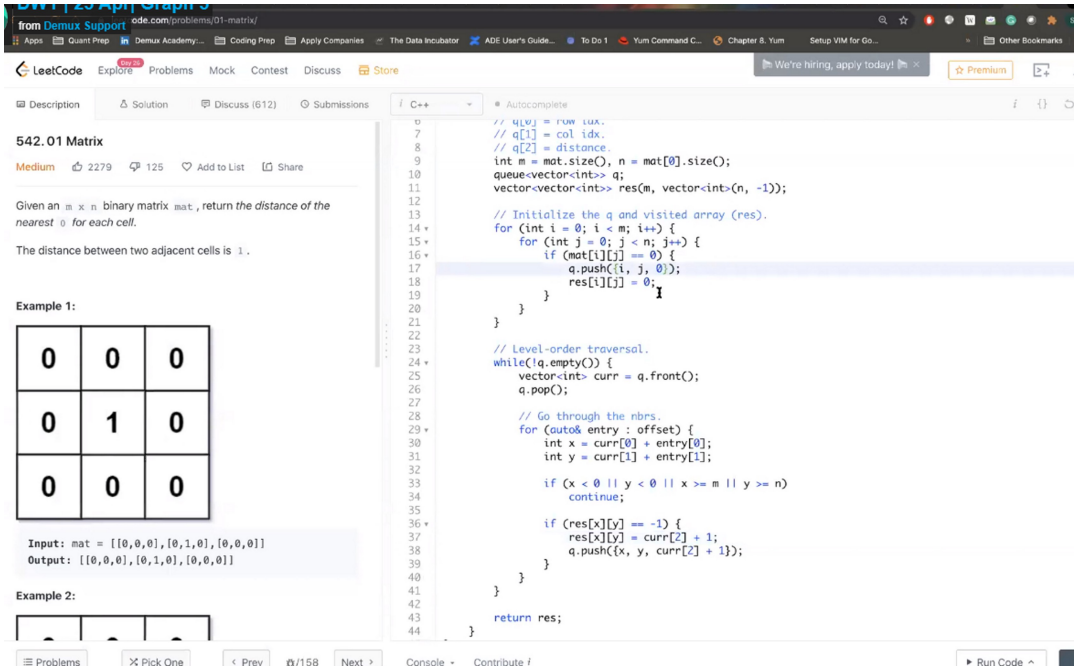
Show off your acceptance:

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

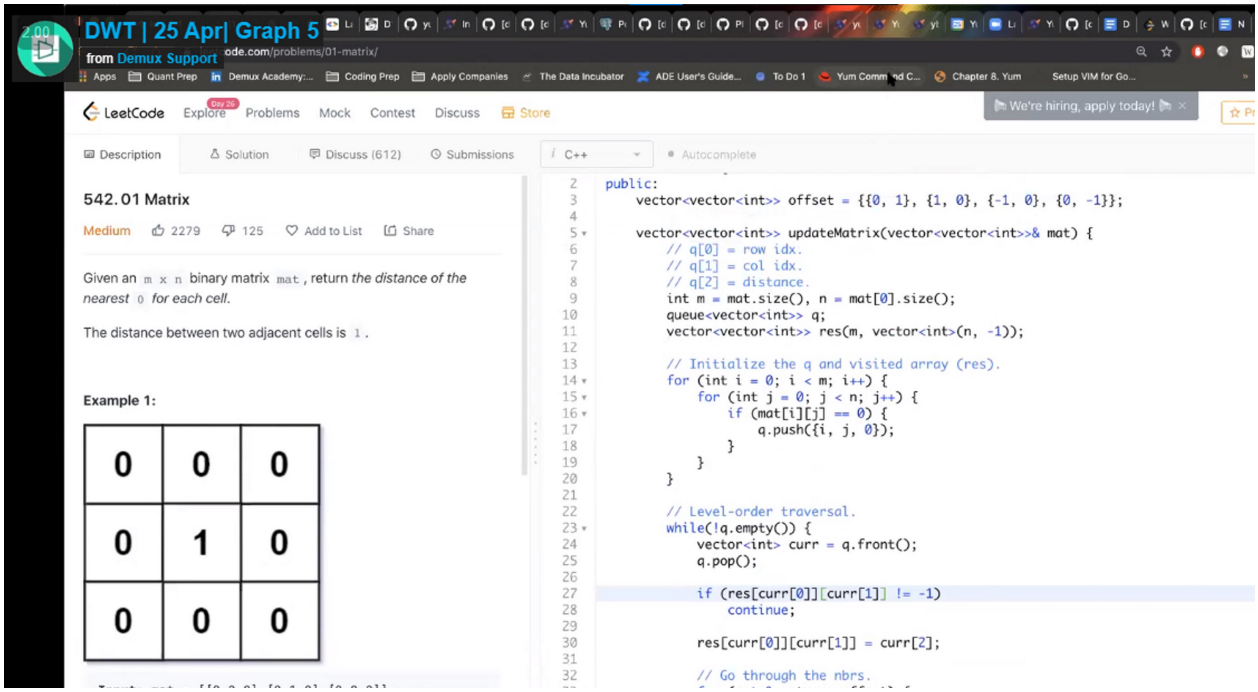
```
    }

    vector<vector<int>> updateMatrix(vector<vector<int>>& mat) {
        int n=mat.size(),m=mat[0].size(),i,j,distance=0;
        queue<pair<int,int>> curset;// current set to be iterated upon
        vector<vector<int>> visited(n,vector<int>(m,-1));
        for(i=0;i<n;i++){
            for(j=0;j<m;j++){
                if(mat[i][j]==0){
                    visited[i][j]=distance;//initial distance =0 for all 0's in th matrix
                    curset.push({i,j});
                }
            }
        }

        while(!curset.empty()){
            int len=curset.size();
            distance++;//increment the level for level order traversal
            while(len>0){
                len--;
                i=curset.front().first,j=curset.front().second;
                curset.pop();
                for(int x=0;x<4;x++){
                    int a=i+dx[x],b=j+dy[x];
                    if(inRange(a,b,n,m) && visited[a][b]==-1){
                        //inrange and unvisited
                        visited[a][b]=distance;
                        curset.push({a,b});
                    }
                }
            }
        }
        return visited;

    }
};
```

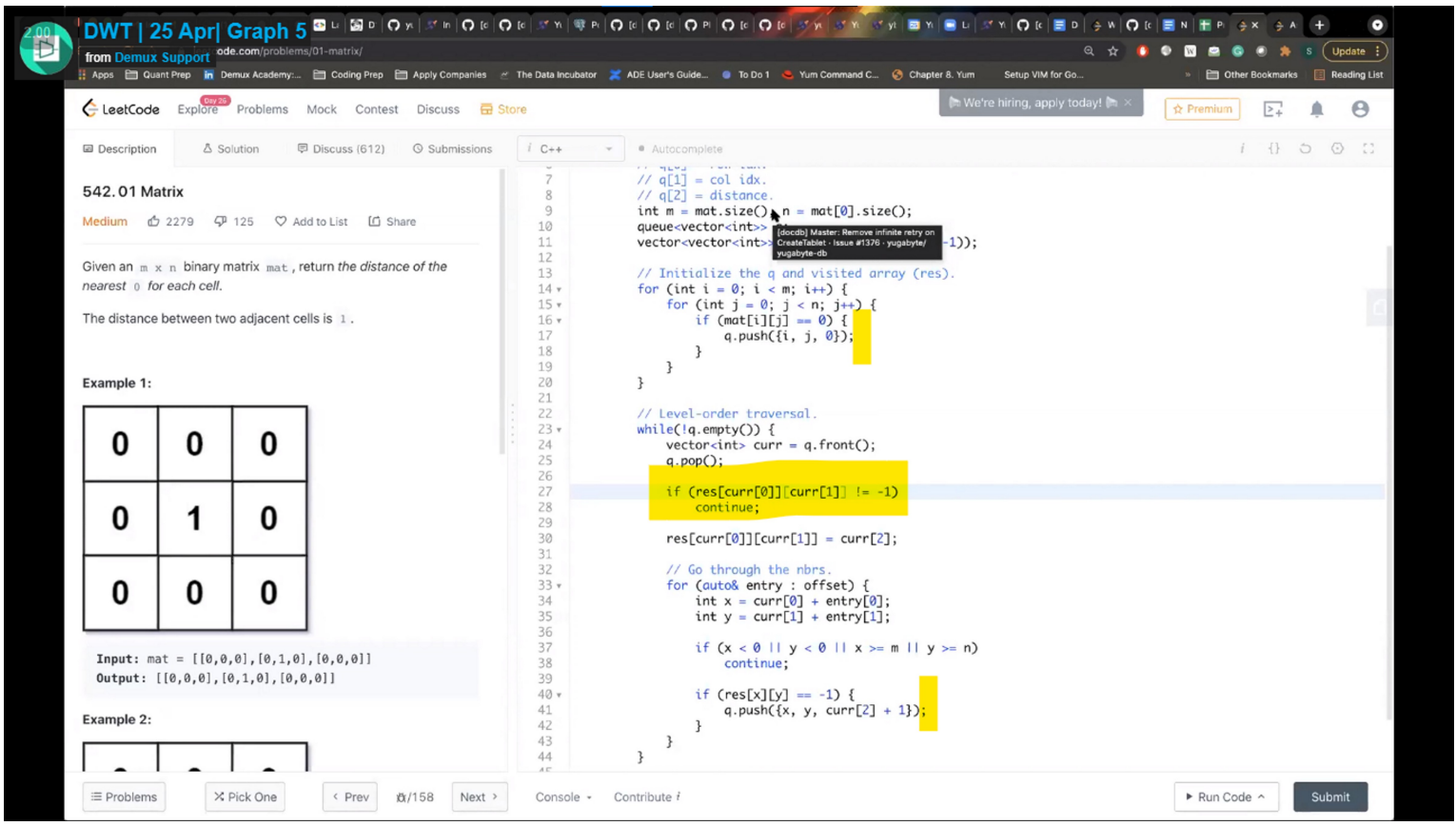
Sir's soln: Approach 1:
During(/before) pushing into queue mark it visited (I did the same above)





Approach 2:
This approach is used in djikstra's algorithm

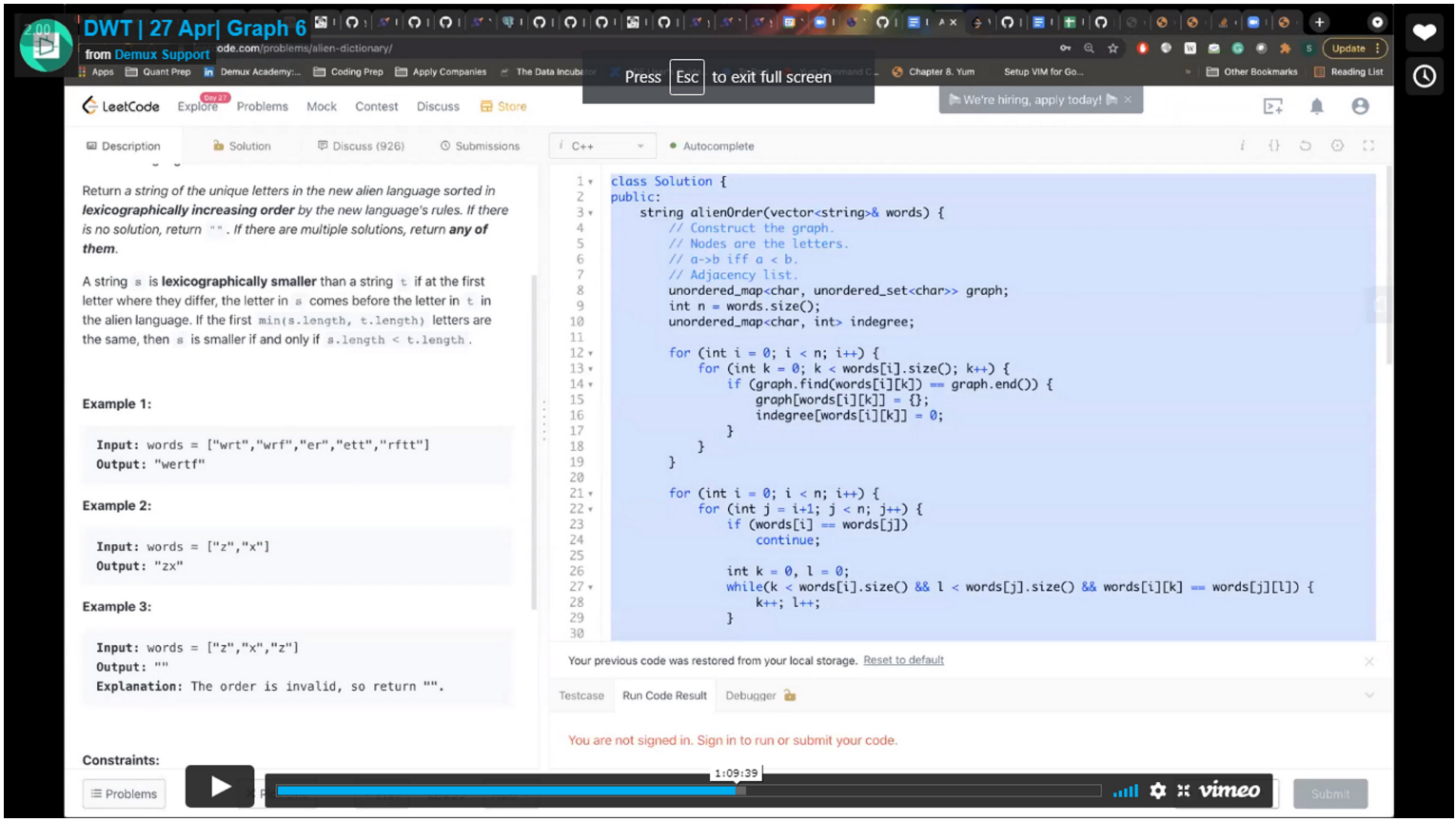


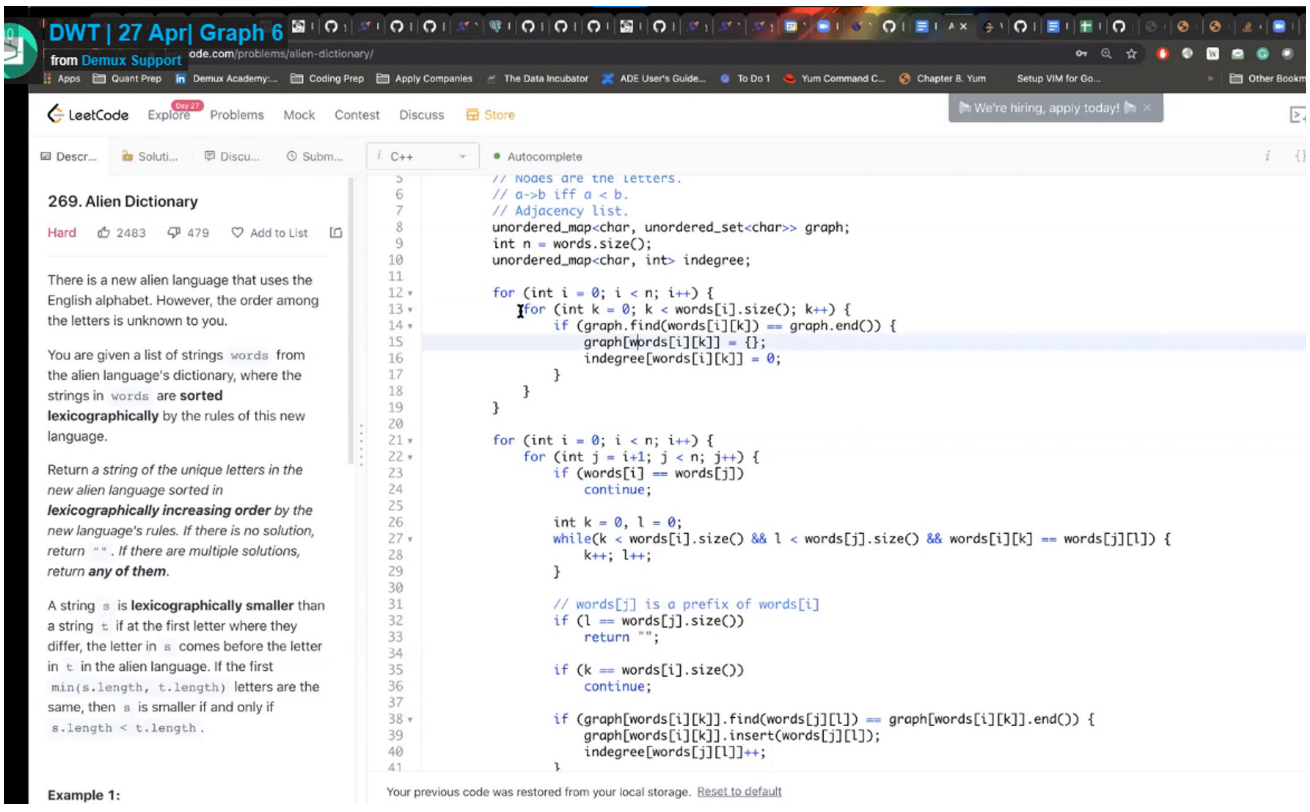


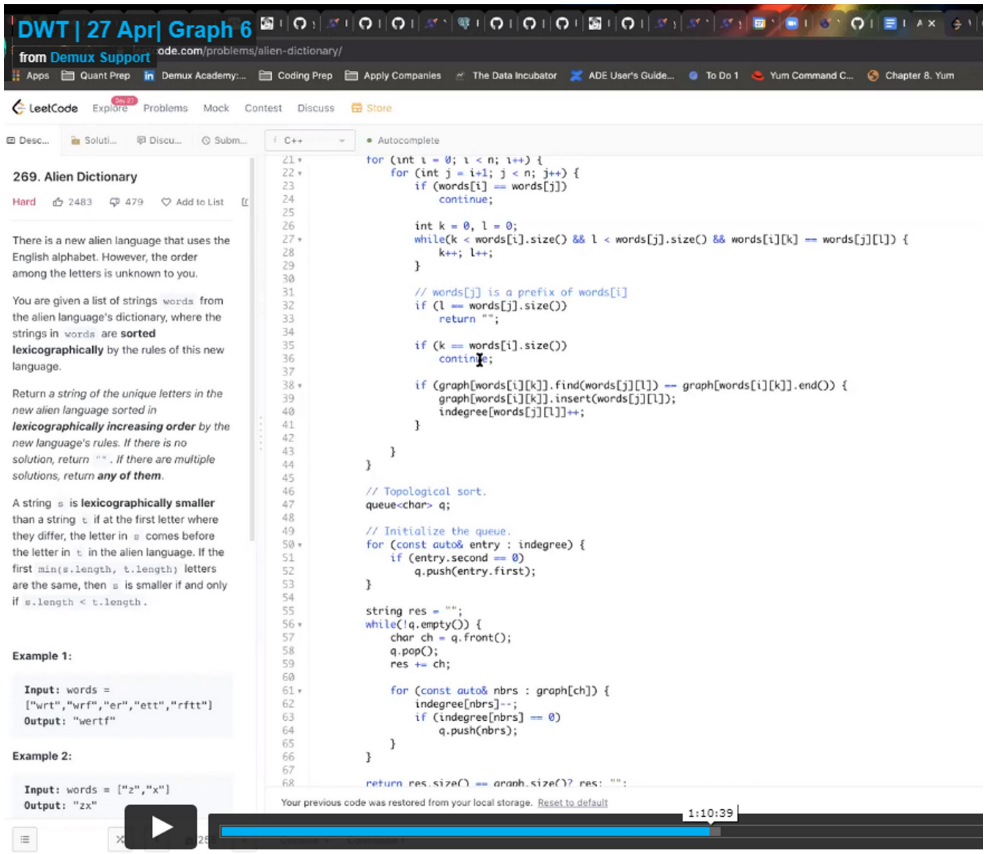
==========

Lec 6:

Q) leetcode 269. alien dictionary


DWT | 27 Apr| Graph 6
from Demux Support
Press Esc to exit full screen
LeetCode
Explore
Problems
Mock
Contest
Discuss
Store
Description
Solution
Discuss (926)
Submissions
C++
Autocomplete
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:
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++;
}
Your previous code was restored from your local storage. Reset to default
Testcase
Run Code Result
Debugger
You are not signed in. Sign in to run or submit your code.
Problems
1:09:39
vimeo
Submit

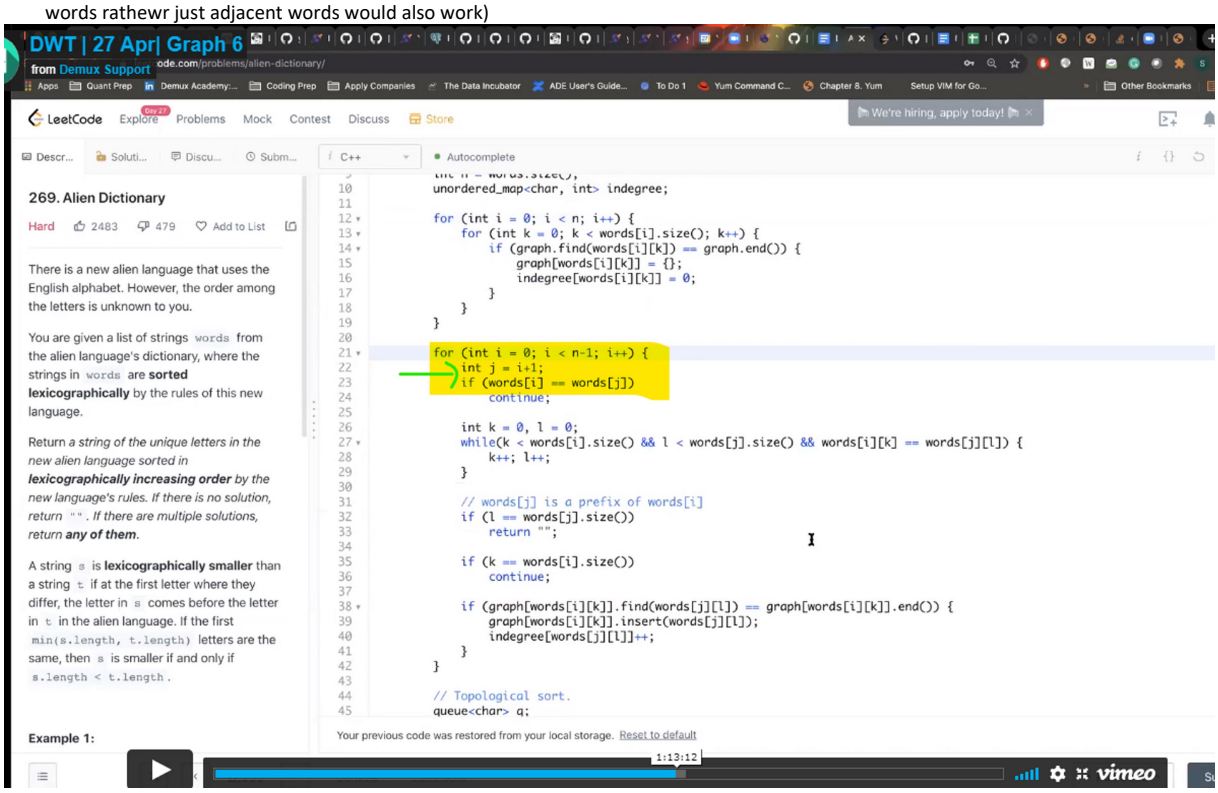

DWT | 27 Apr| Graph 6
from Demux Support
LeetCode
Explore
Problems
Mock
Contest
Discuss
Store
269. Alien Dictionary
Hard
2483
479
Add to List
There is a new alien language that uses the English alphabet. However, the order among the letters is unknown to you.
You are given a list of strings words from 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:
// 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++;
}
// words[j] is a prefix of words[i]
if (l == words[j].size())
return "";
if (k == 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

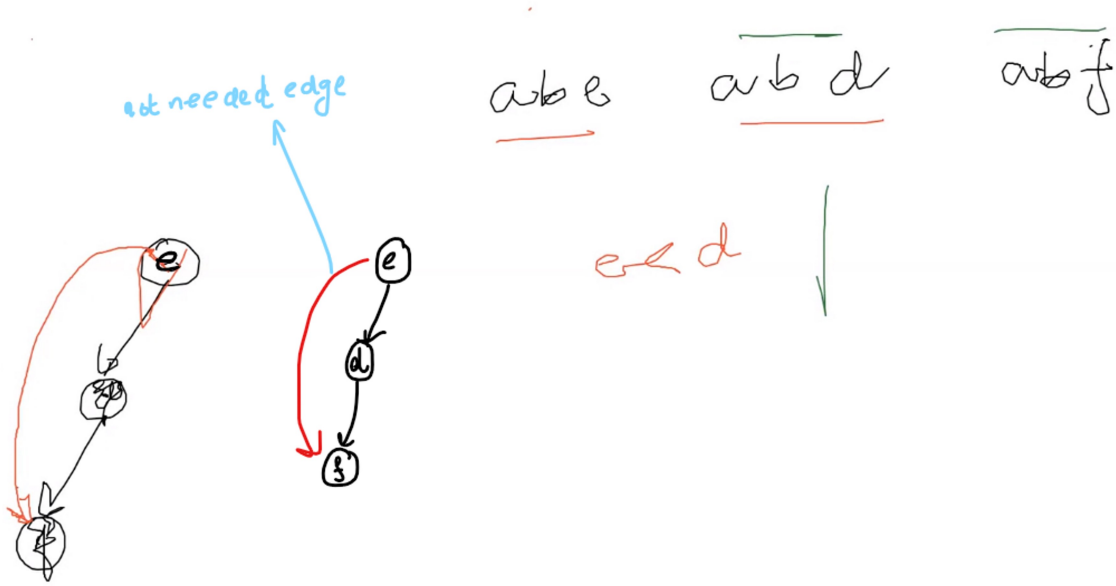


-

Optimized code for above:

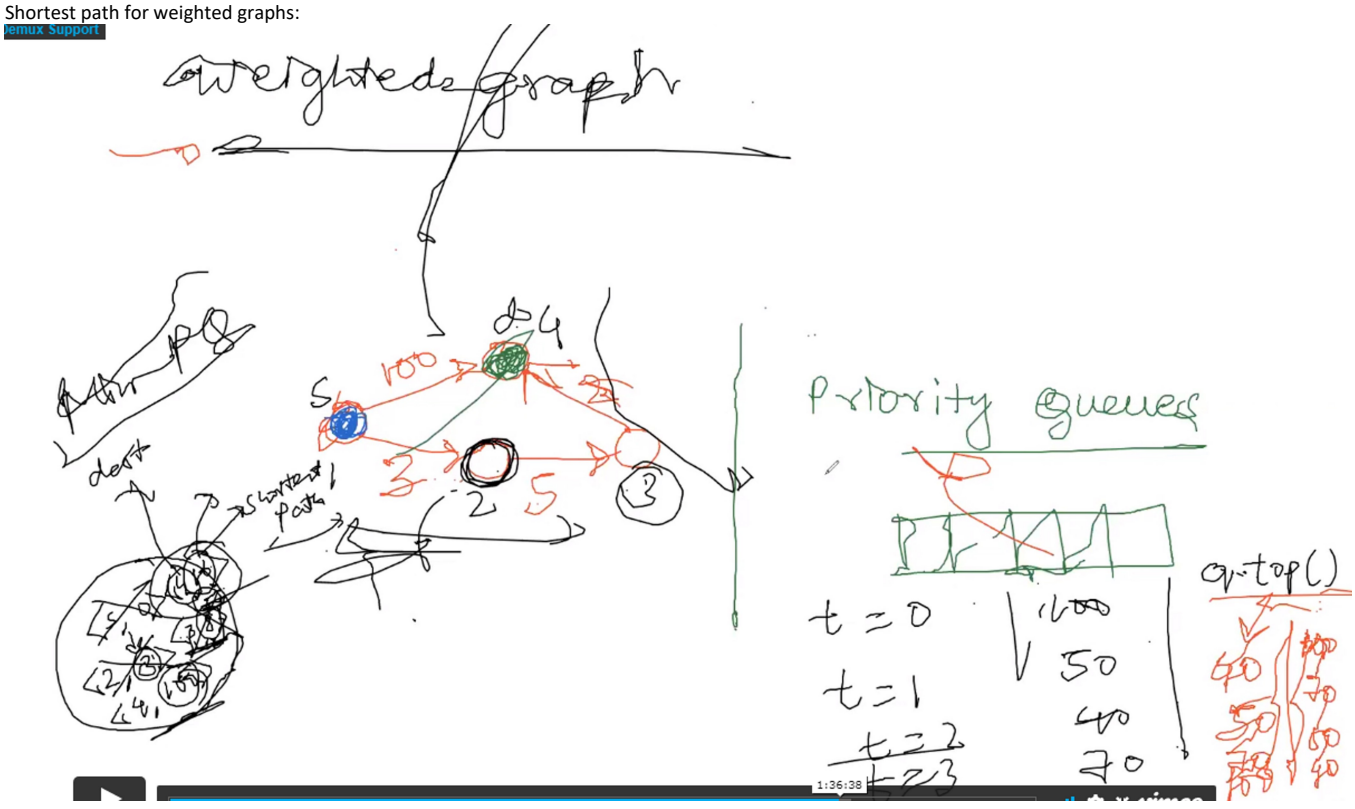
- Here we capture only the relations between adjacent words only(as the relationship is transistive so those wilol be auitomatically captured in the graph itself, so we don't need to populate the relations between all the words rathewr just adjacent words would also work)





=----------

Shortest path for weighted graphs:



Q)

Sir ka solnm:

DWT | 27 Apr| Graph 6

from Demux Support

ode.com/problems/network-delay-time/

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 | Reading List

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

Descrip... | Solution | Discus... | Submis... | C++ | Autocomplete

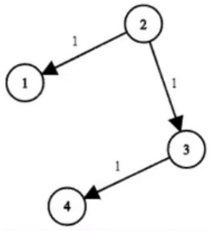
743. Network Delay Time

Medium | 2517 | 246 | Add to List | Sh

You are given a network of `n` nodes, labeled from `1` to `n`. You are also given `times`, a list of travel times as directed edges `times[i] = (ui, vi, wi)`, where `ui` is the source node, `vi` is the target node, and `wi` 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>>, /* Type of priority */ 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++;
        // Shortest path from k to t.second.

        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;
}
};
```

1/158 | Console | Contribute | 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 <= k <= n <= 100`
- `1 <= times.length <= 6000`
- `times[i].length == 3`
- `1 <= ui, vi <= n`
- `ui != vi`
- `0 <= wi <= 100`

My soln:

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

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 C
Network Delay Time.

Next challenges:

The Time When the Network Becomes Idle

Second Minimum Time to Reach Destination

Show off your acceptance:

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

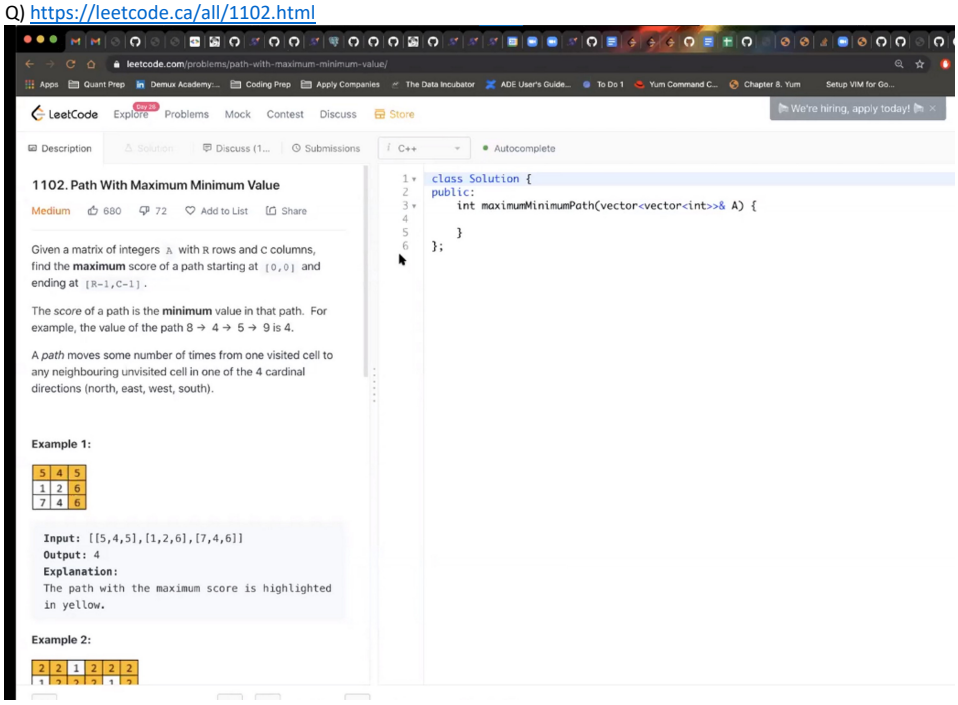
```cpp
class Solution {
public:
    int networkDelayTime(vector<vector<int>>& times, int n, int k) {
        // Level order traversal : bfs -> but since its weighted we maintain a *priority queue*
        // maintain runninng max for time & visited set also
        // at last check if visited set size == n, if not then return -1 else return running max
        unordered_set<int> visited;//we could use array of size n also for this(more effective)
        // Weighted BFS(weighted level order traversal) => priority queue
        priority_queue<pair<int,int>, vector<pair<int,int>>, greater<pair<int,int>>> pq; // <time,node> :sort by time
        int i,j,time=0,tmax=0;

        vector<vector<pair<int,int>>> adj(n);// ui :<vi,weight>
        for(const auto& x:times){
            adj[x[0]-1].push_back({x[1]-1,x[2]});//buidling graph (0 : n-1)
        }
        k=k-1;//making 0 based nodes
        pq.push({time,k});//pushing initial node k into the pq
        // Level order traversal
        while(!pq.empty()){
            int t,node;
            t = pq.top().first, node = pq.top().second;
            pq.pop();
            visited.insert(node);
            tmax = max(tmax,t);
            if(visited.size()==n) return tmax;//this has to be inside the loop
            //Push all neighbours
            for(const auto& nbr:adj[node]){
                if(visited.find(nbr.first)==visited.end()){//unvisited nbr
                    pq.push({ t + nbr.second , nbr.first});//{t+wt,nxt_node}
                }
            }
        }
        if(visited.size()==n) return tmax;
        return -1;
    }
```

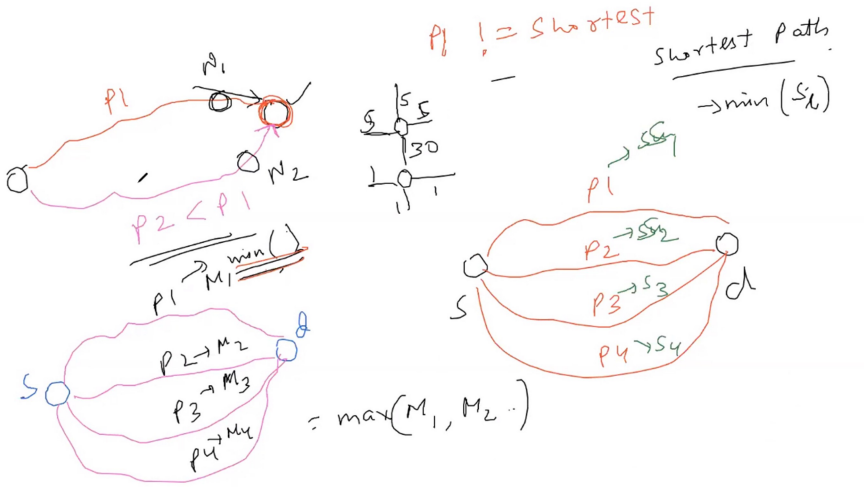
743/2174 Console Contribute i Run Code Sub

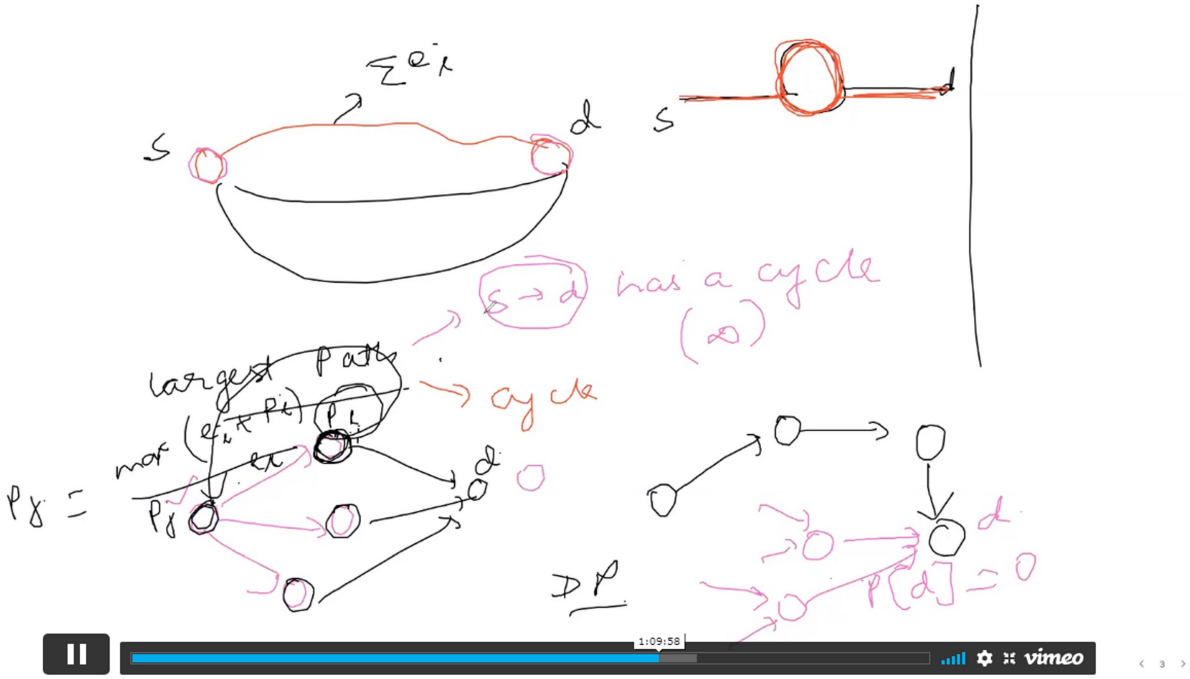
==========
==========

LEC - 7

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



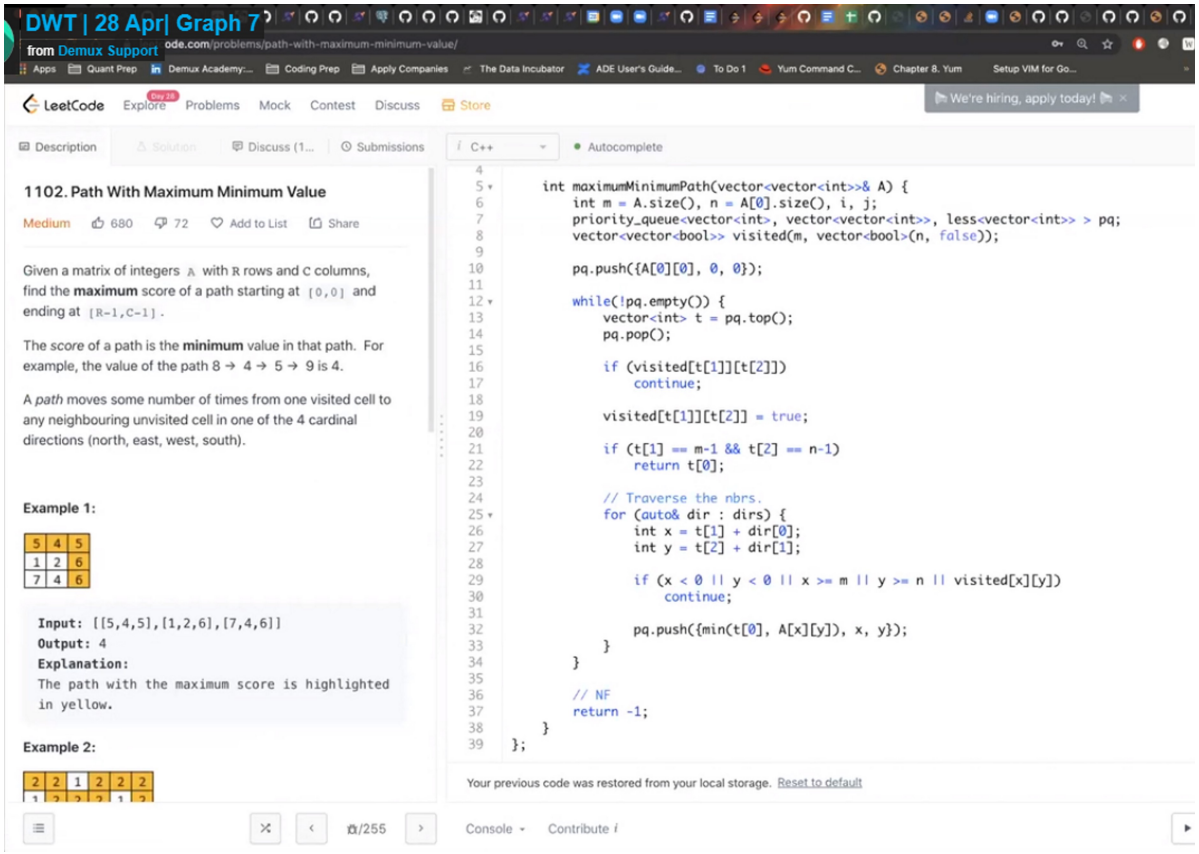


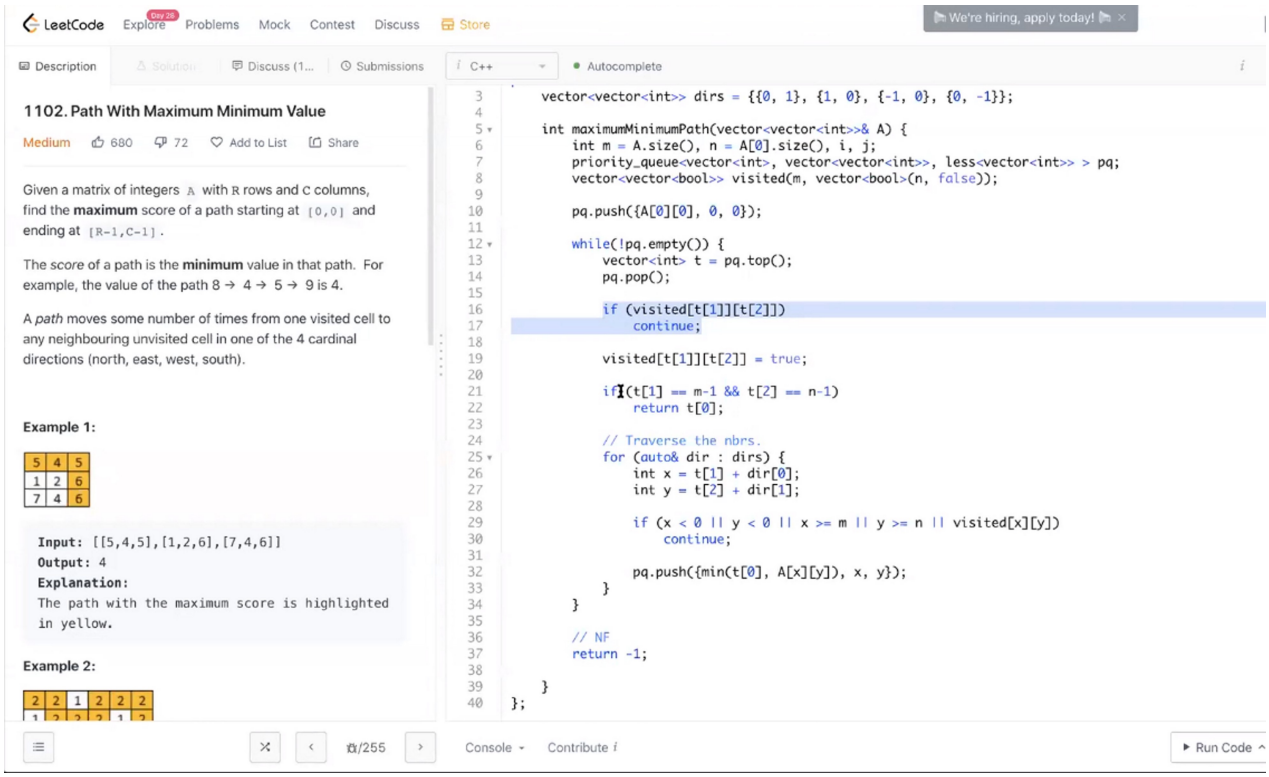


Soln 1 : using pq ::

C++ Autocomplete

```cpp
class Solution {
public:
    vector<vector<int>> dirs = {{0, 1}, {1, 0}, {-1, 0}, {0, -1}};

    int maximumMinimumPath(vector<vector<int>>& A) {
        int m = A.size(), n = A[0].size(), i, j;
        priority_queue<vector<int>, vector<vector<int>>, less<vector<int>> > pq;
        vector<vector<bool>> visited(m, vector<bool>(n, false));

        pq.push({A[0][0], 0, 0});

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

            if (visited[t[1]][t[2]])
```





Rewatch this uestion lec 7 :: rewatch it::

==========

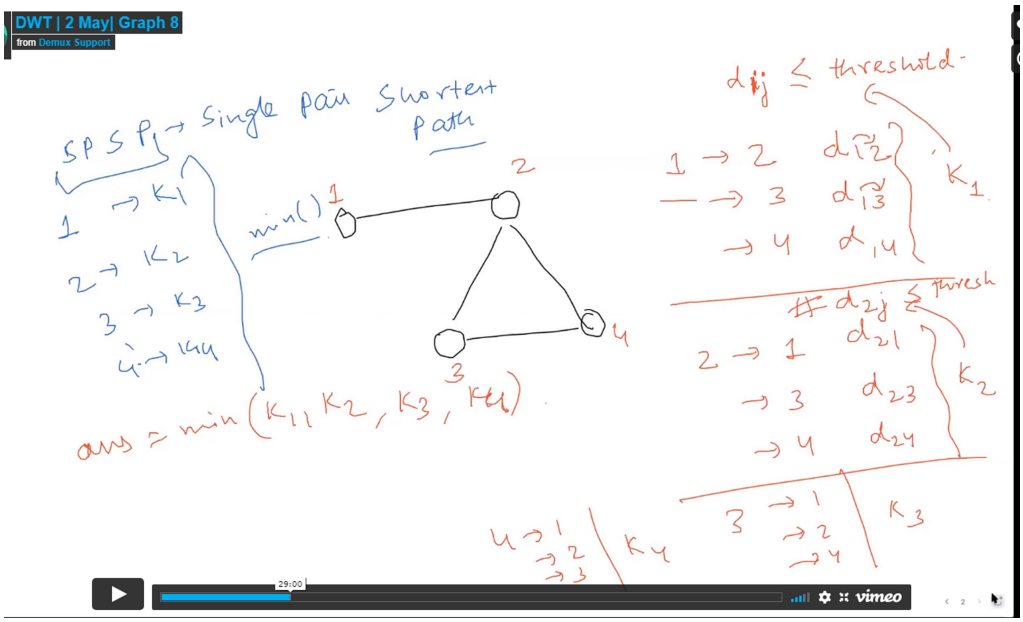
Lec 8 ::

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

From <https://leetcode.com/problems/find-the-city-with-the-smallest-number-of-neighbors-at-a-threshold-distance/>

- https://cp-algorithms.com/graph/all-pair-shortest-path-floyd-warshall.html

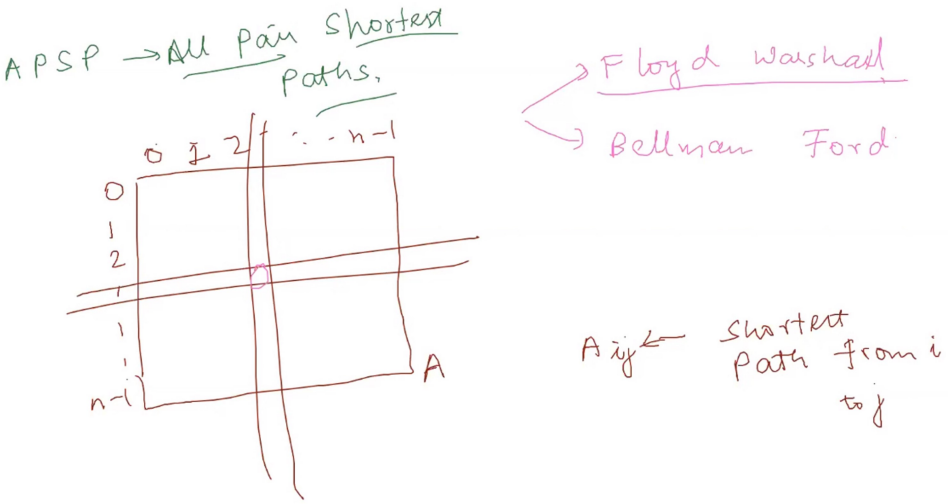


Brute force :

- single pair shortest path

Efficient:

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



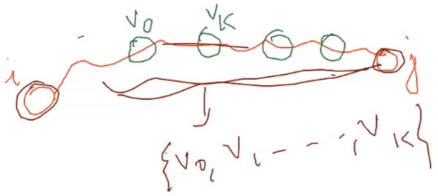
Floyd Warshall

0, 1, 2, ... n-1

$[V_0, V_1, V_2]$

A_{ijk} ← Shortest path from i to j that contains only vertices from $[V_0 \ldots V_k]$ ⓑ

$> V_k$

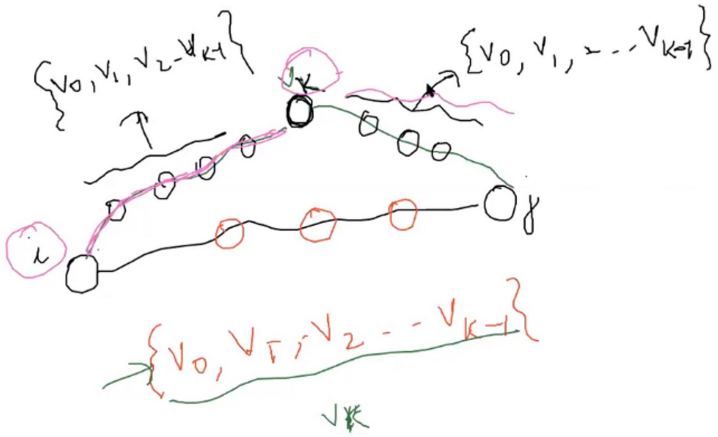


WT | 2 May| Graph 8

om Demux Support

Floyd Warshall

$$A_{ijK} = \min\left(A_{ijK-1}\ ,\ A_{iKK-1} + A_{KjK-1}\right)$$

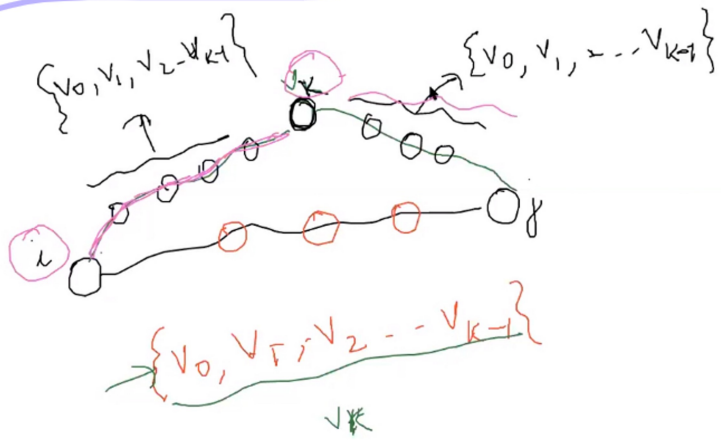


WT | 2 May| Graph 8

pm Demux Support

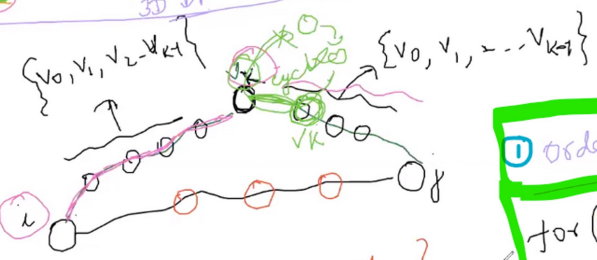
Floyd Warshall

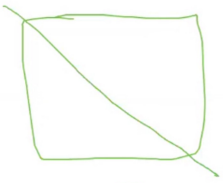
$$A_{ijk} = \min\left(A_{ijk-1},\ A_{ik(k-1)} + A_{kj(k-1)}\right)$$



$$A_{ijk} = \min\left(A_{ijk-1},\ A_{ik(k-1)} + A_{kj(k-1)}\right)$$

3D DP





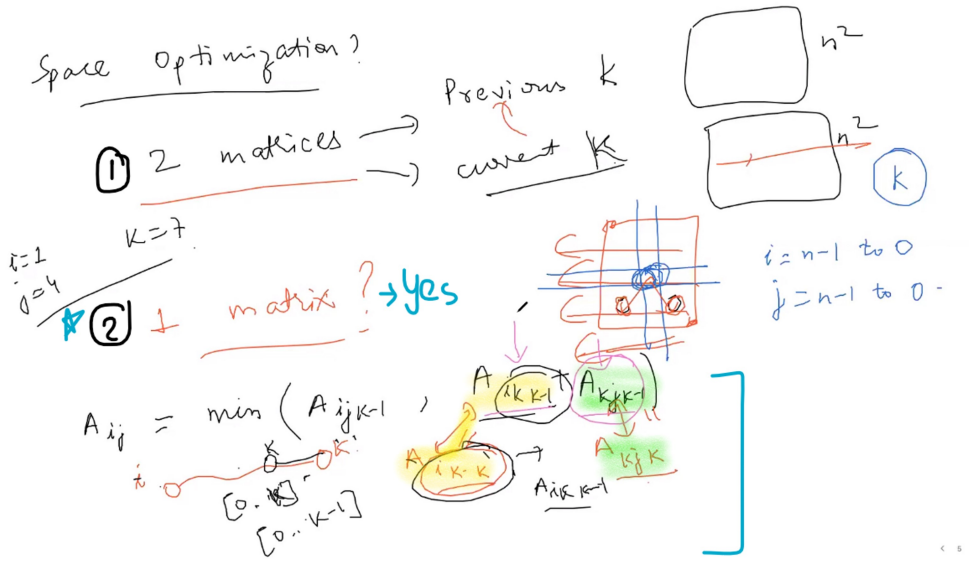
1. Order of iteration

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

2. Base Case

$A(i,i) = 0$: $i \to i$, $k \to k-1$

∞ : $i \to j$ if w_{ij}

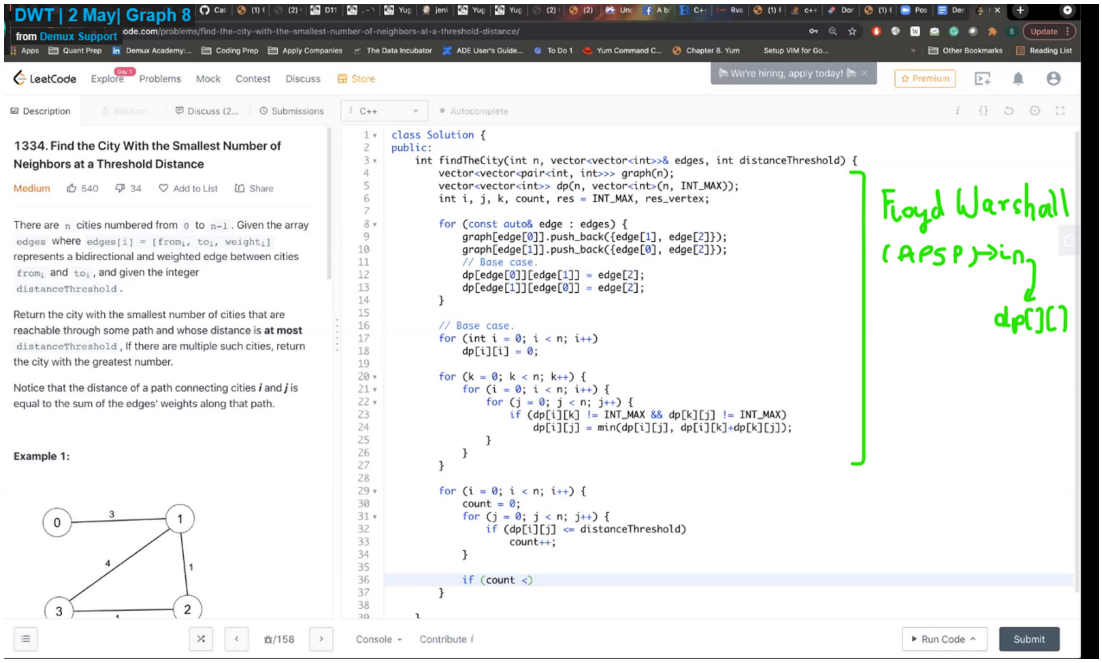


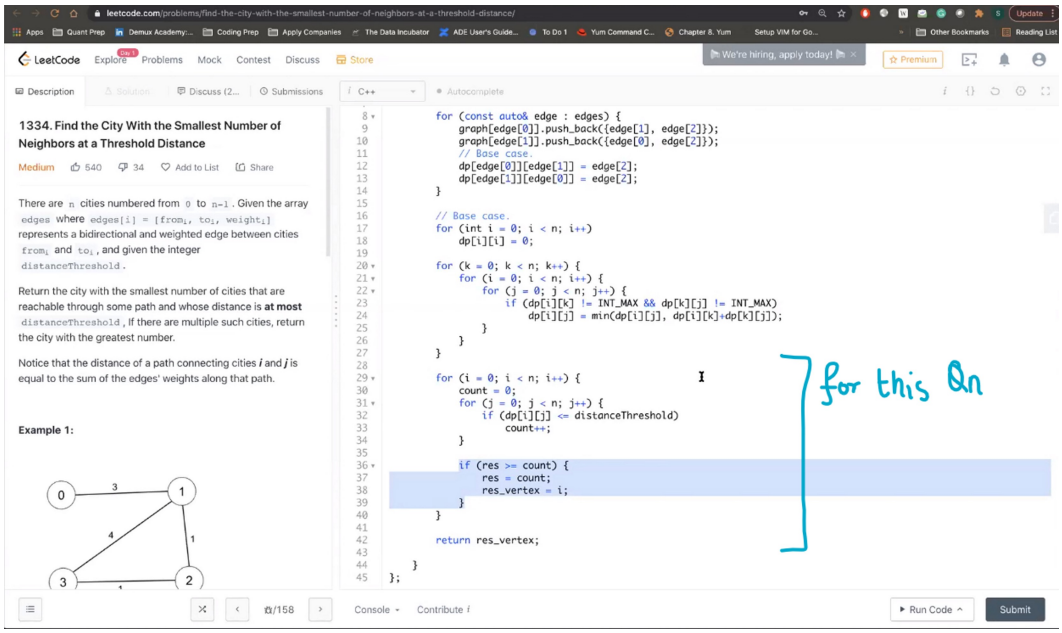
NOTE:

A(i)(j)(k) = minimum metric(eg: path sum) to reach form node I to node j ; with intermediate nodes belonging to the set: {v0,v1,…,vk}

Things we need to consider:

1. Order of iteration (/order of filling the table):
   a. Form the recursive relation its 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,…. & so on
2. Base Case
   a. dp(ij) = -1 ; if no edge between nodes i & j
   b. dp(ij) = Wij ; if there is a edge between node I & j and weight of the edge is Wij
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 wont affect our ans, therefore using only 1 matrix will suffice. Another way to think about it is : the updated value id 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 form I to k with intermediate nodes as v0,v1,..,vk-1 is same as going form I to k with intermediate nodes as v0,v1,..,vk ; (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)

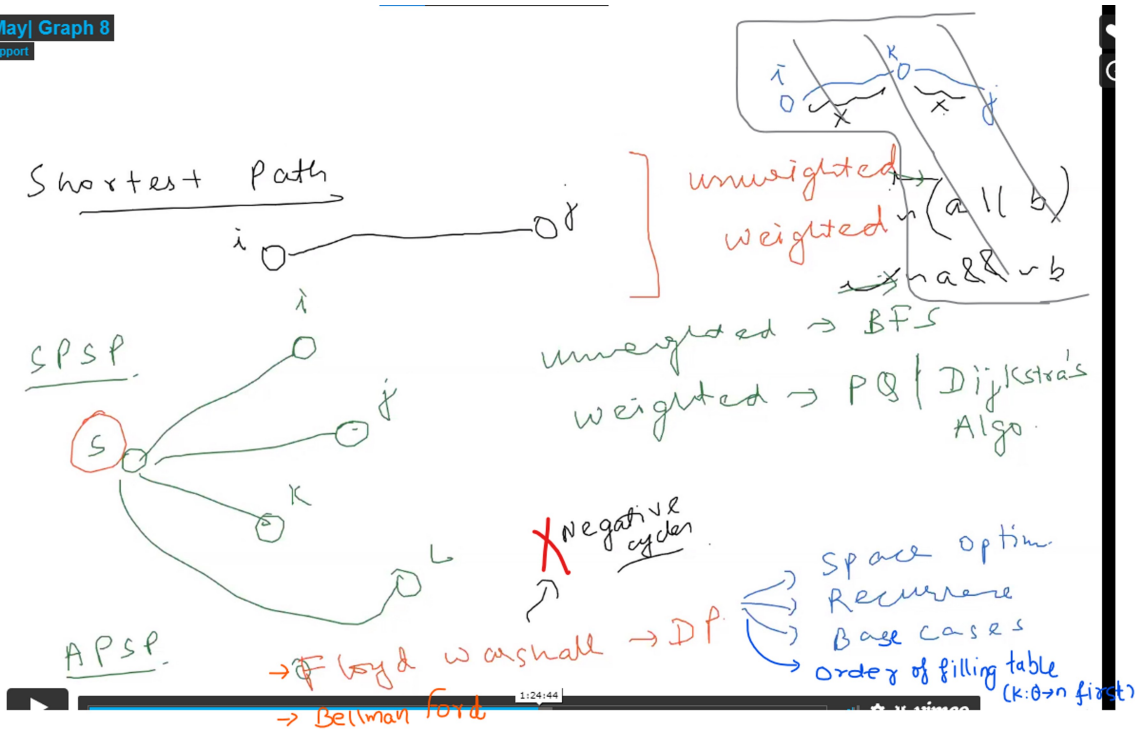




Overview om shortest path:

/T | 2 May| Graph 8

Demux Support

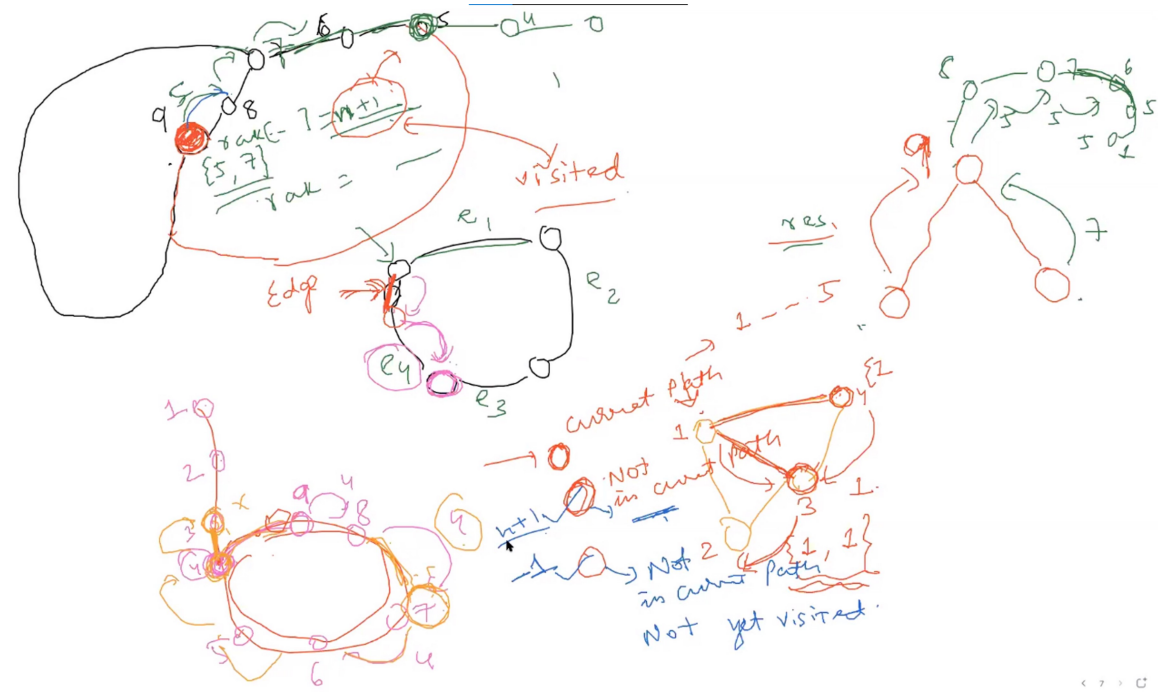


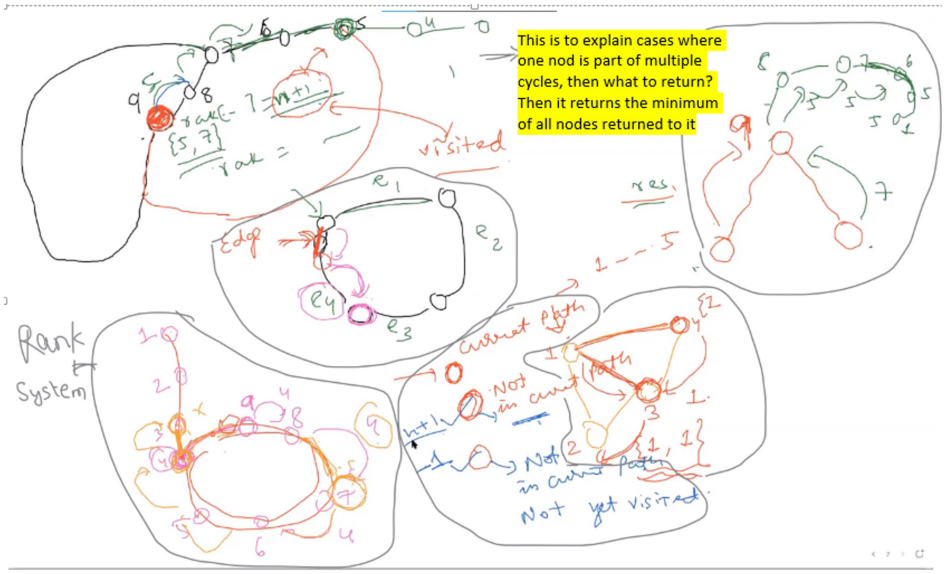
==========

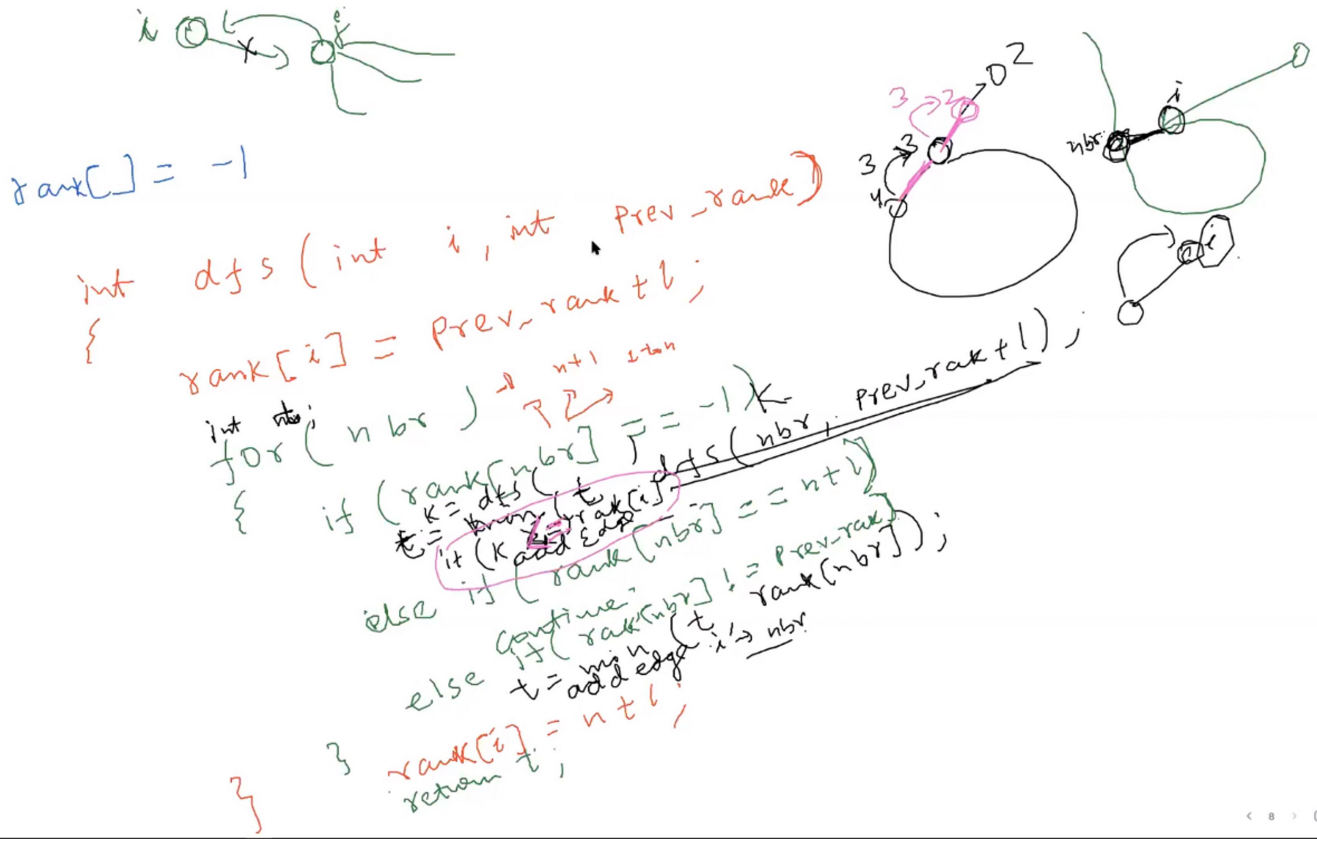
Leetcode 1192. Critical Connections in a Network

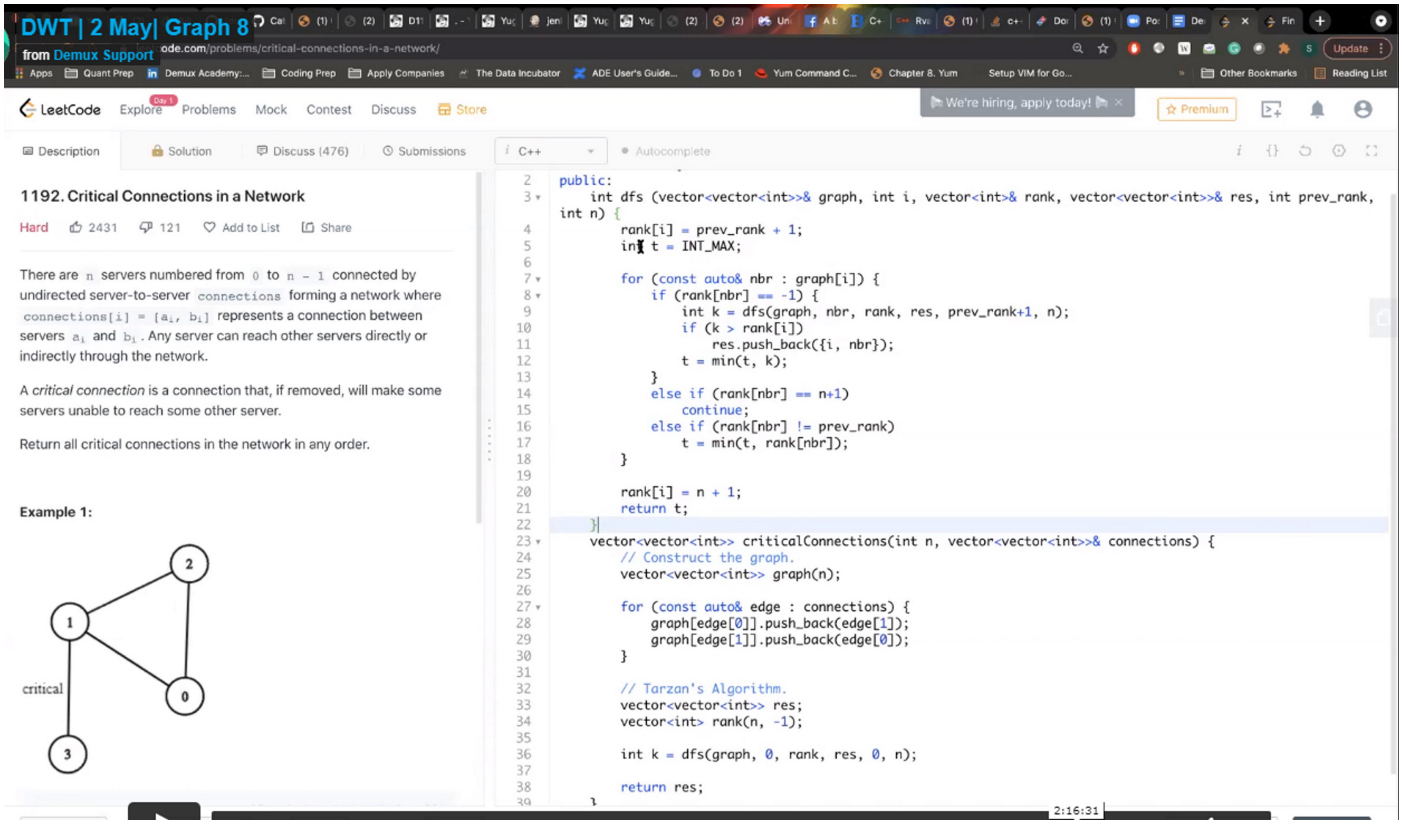
From <https://leetcode.com/problems/critical-connections-in-a-network/>

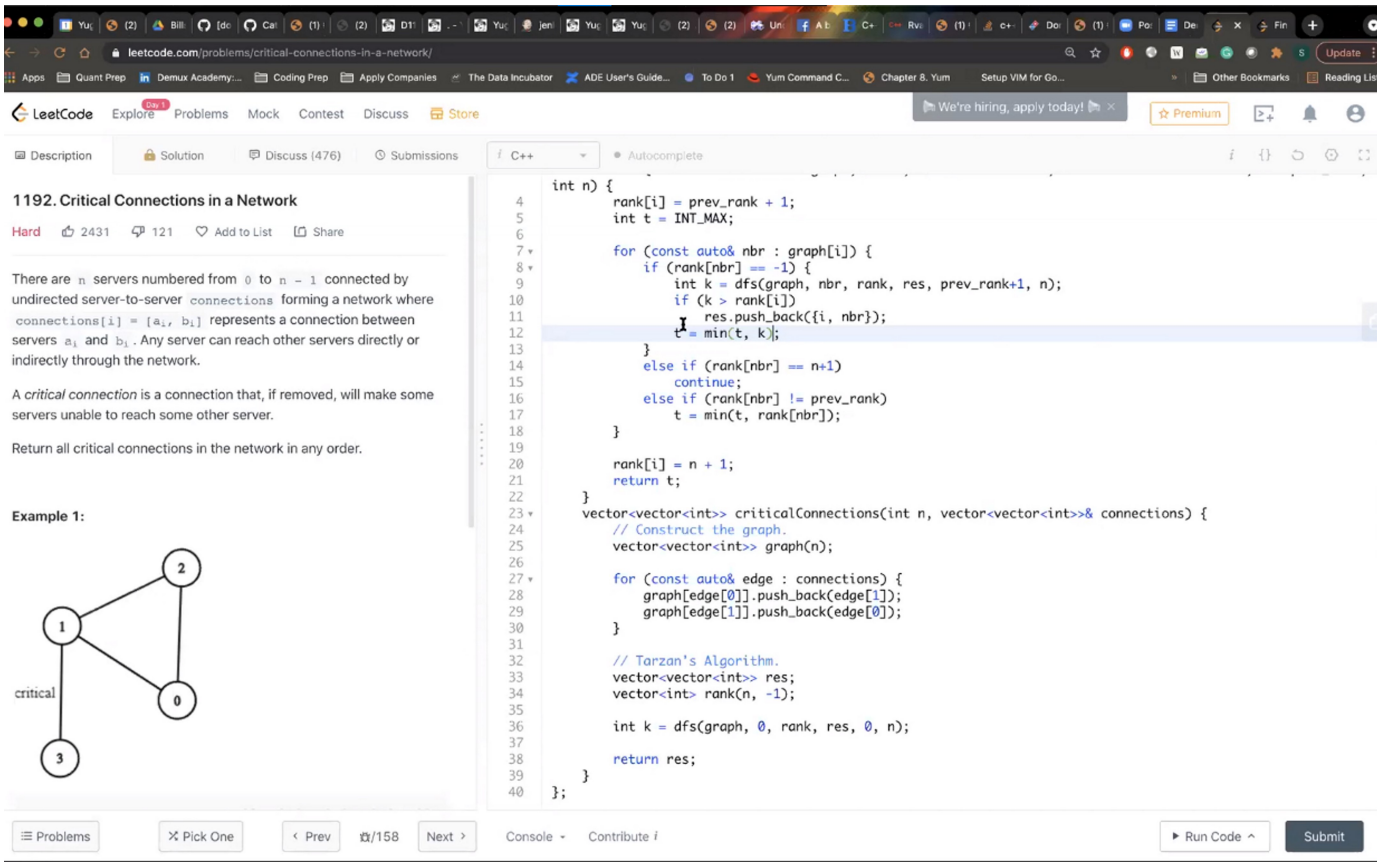
Here we have to tell bridges - any edge which is not a part of any cycle




This is to explain cases where one nod is part of multiple cycles, then what to return? Then it returns the minimum of all nodes returned to it
visited
res
edge
Rank
System
current path
Not in current path
Not yet visited


rank[_] = -1
int dfs (int i, int prev_rank)
{
rank[i] = prev_rank + 1;
for (nbr)
if (rank[nbr] != -1)
k = dfs(nbr, prev_rank+1);
if (k add edge)
else if (rank[nbr] == n+1)
else continue;
if (rank[nbr] != prev_rank)
rank(nbr));
t = min(t, rank[nbr])
rank[i] = n+1;
return t;
}
}
nbr





- 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/