GRAPHS...

video-39

"let's make it easy too"

Hard

Leetcode -2642

If you have tried my "Graph Concepts & Qns" playlist, these Qns, will seem very easy. Do try it once :)

Facebook, Instagram → codestorywithMIK

(Twitter) → CSwithMIK

codestorywithMIK →

Design Graph with "Shortest Path Calculator"...

Company Tags :- will update soon...

2642. Design Graph With Shortest Path Calculator

Hint

Hard 243 25

Companies

There is a **directed weighted** graph that consists of `n` nodes numbered from `0` to `n - 1`. The edges of the graph are initially represented by the given array `edges` where `edges[i] = [from_i, to_i, edgeCost_i]` meaning that there is an edge from `from_i` to `to_i` with the cost `edgeCost_i`.

Implement the `Graph` class:

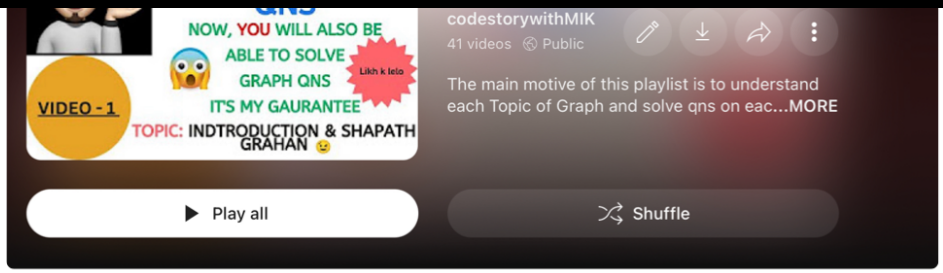
- `Graph(int n, int[][] edges)` initializes the object with `n` nodes and the given edges.
- `addEdge(int[] edge)` adds an edge to the list of edges where `edge = [from, to, edgeCost]`. It is guaranteed that there is no edge between the two nodes before adding this one.
- `int shortestPath(int node1, int node2)` returns the **minimum** cost of a path from `node1` to `node2`. If no path exists, return `-1`. The cost of a path is the sum of the costs of the edges in the path.

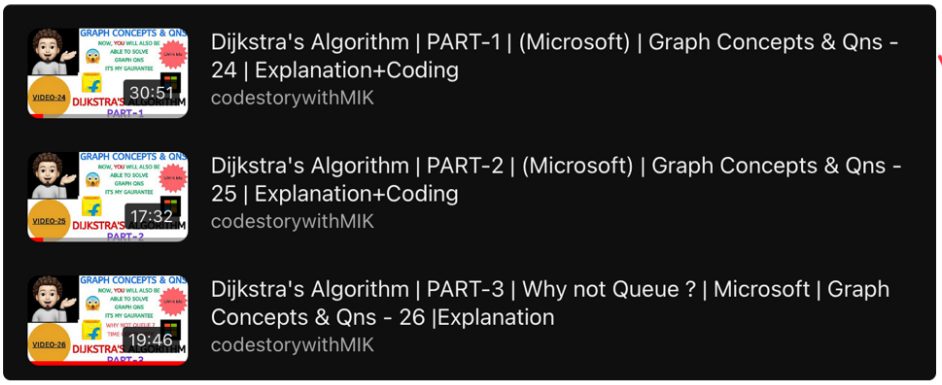
(Dijk.)

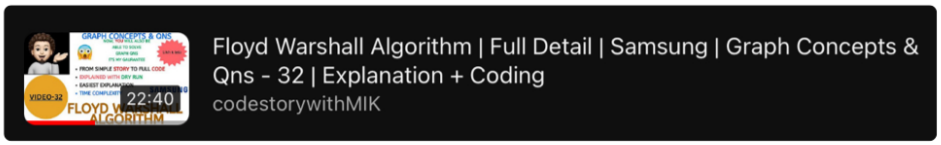
Pre-requisite :-

(•) Dijkstra's Algorithm.

(•) Floyd Warshal Algorithm.







Dijkstra's Algo

```
unordered_map <int, vector<pair<>>>adj;
Graph(int n, vector<vector<int>>& edges){
```

O(V+E).

u
v
cost

```
adj[u].push_back({v, cost});
}
```

M

void addEdge (vector<int> edge) {

O(1)

edge = {u, v, cost}

O(1).

adj[u].push_back({v, cost})

}

int shortestPath (int node1, int node2) {

Dijkstra Code

O(M * ElogV)

}

Floyd Warshall

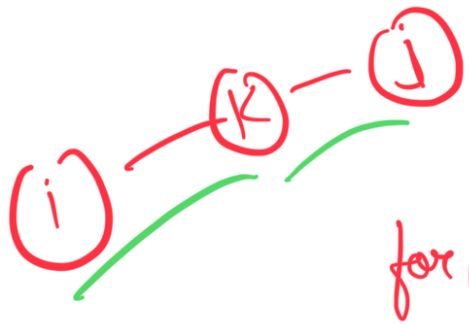
[V3][V4] O(1).

Graph (int n, vector<vector<int>>& edges) {

[n][n] → {10^9}

edges → u, v, cost

↳ [u][v] = cost ←




```
for(k=0; k<n; k++) {
    for(int i=0; i<n; i++) {
        for(int j=0; j<n; j++) {
            adj[i][j] = max(adj[i][j],
                adj[i][k] + adj[k][j]
        }
    }
}
```

Floyd Warshall

O(N^3)

```
void addEdge(vector<int> edge) {
    u, v, cost
    for(i=0; i<n)
        for(j=0; j<n) {
            adj[i][j] = min(adj[i][j],
                adj[i][u] +
                cost + adj[v][j]
        }
}
```

```
int ShortestPath(int node1, int node2) {
```

O(1)

```
    return adj[node1][node2] == 10^9 ? -1 : adjacency.
}
```