GRAPHS...

video-35

"let's make it easy too"

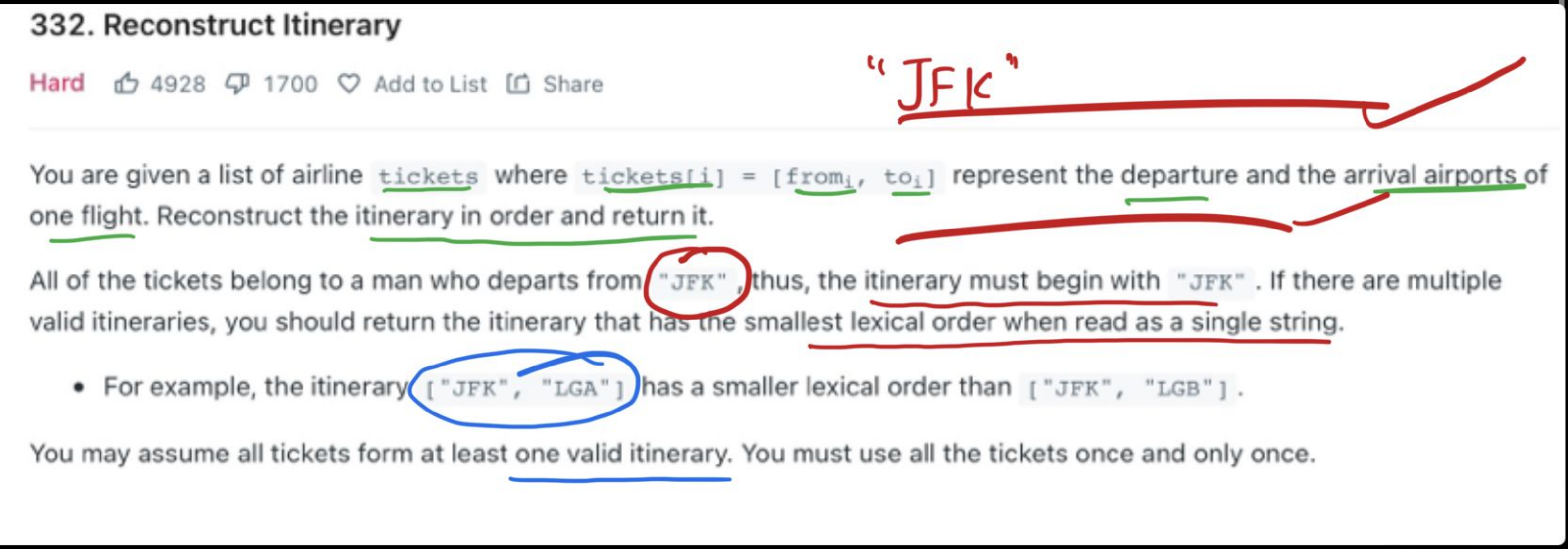
~~HARD~~

Leetcode - 332

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

Reconstruct Itinerary...

Company Page :-



332. Reconstruct Itinerary

Hard 4928 1700 Add to List Share

"JFK"

You are given a list of airline `tickets` where `tickets[i] = [from_i, to_i]` represent the departure and the arrival airports of one flight. Reconstruct the itinerary in order and return it.

All of the tickets belong to a man who departs from `"JFK"`, thus, the itinerary must begin with `"JFK"`. If there are multiple valid itineraries, you should return the itinerary that has the smallest lexical order when read as a single string.

- For example, the itinerary `["JFK", "LGA"]` has a smaller lexical order than `["JFK", "LGB"]`.

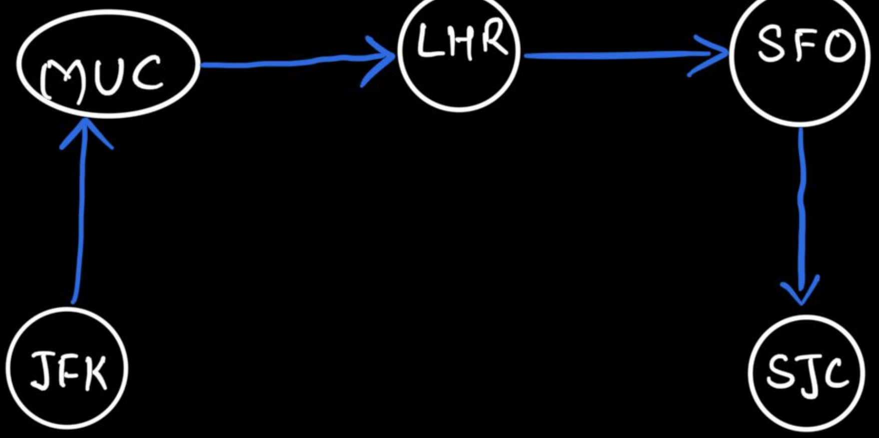
You may assume all tickets form at least one valid itinerary. You must use all the tickets once and only once.

numTickets = 4

Example :- { ["MUC", "LHR"], ["JFK", "MUC"], ["SFO", "SJC"], ["LHR", "SFO"] }

LHR →	SFO , xyz
MUC →	LHR
JFK →	MUC
SFO →	SJC

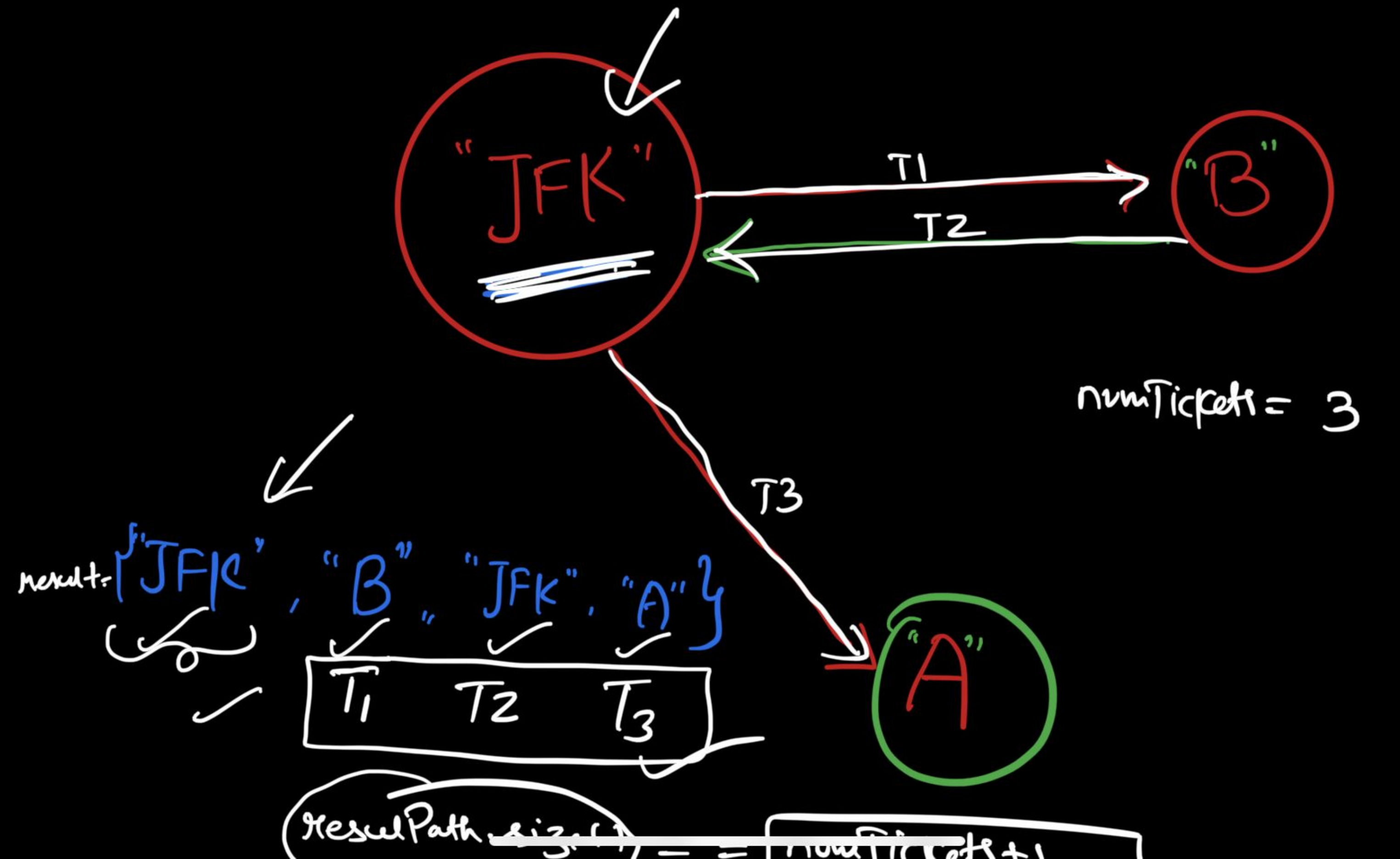
} adj



Output :- ["JFK", "MUC", "LHR", "SFO", "SJC"]

(•) Lexical order में जाना
if you have multiple options.

(•) But if that doesnot give me the answer, then explore other options.



"JFK"
"B"
"A"
T1
T2
T3
numTickets = 3
result = {"JFK", "B", "JFK", "A"}
T1 T2 T3
resultPath.size() == numTickets+1

resultPath.size() == numTickets+1

```
⇒ if (result.size() == numTicket+1) {
          result = Path;
          return True;
  }
```

Story to Code....

① adj

```
unordered_map < string , vector<string>> adj ;
```

② adj को Populate.

③ sort (adj में जो neigbors → lexically).

③ sort (adj में सभी neigbors → lexically).

④ DFS ("JFK", path) ; //vector<string>

//result = path.

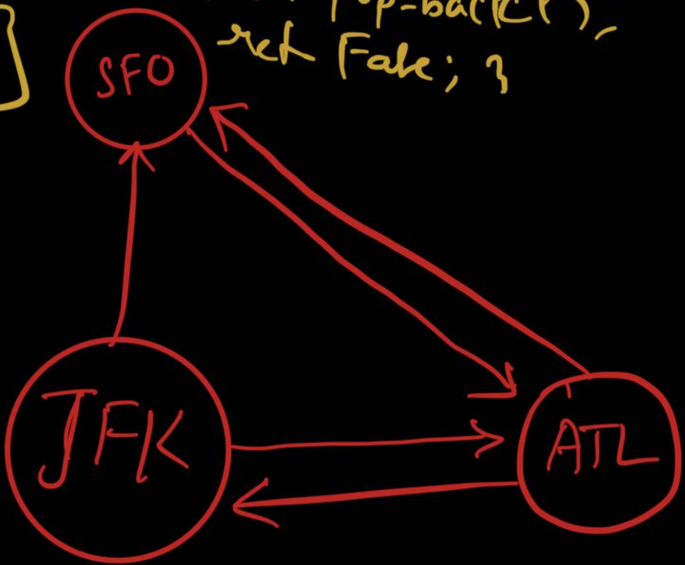
```
bool DFS (string fromcity, vector<str>& Path) {

    Path.push_back (fromCity);

    if ( path.size() == numTickets+1) {
        result = Path;
        return True;
    }

    vector<string> & neighbors = adj[fromCity];

    for (int i=0; i< neigbrs.size(); i++) {
        string toCity = neighbor[i].
for (int i=0;  i< neighbors.size() ; i++) {
    String toCity = neighbor[i];
    erase   → neighbors.erase( ith index से erase);
    explore → if (DFS(to City, Path)) {
                  return True;
              }
    Putback → neighb.insert (ith index);
}
Path.pop_back();
ret False; }
```



$(V+E)^2$

$(V+V)^2$

$O=(V)^2$

JFK → ~~ATL~~, SFO

ATL → ~~JFK~~, SFO

SFO → ATL