Interview Problems

Good Evening

Agenda

01. Possibbility of finishing courses
02. Topological sort
03 Minimum jumps to reach end
04. Buy sell stock (Infinite transaction)

5-6 problems

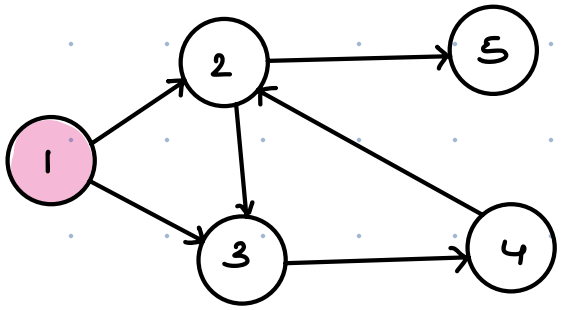
Possibility of finishing Courses

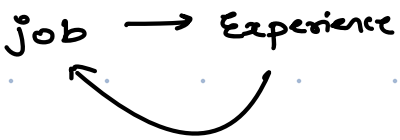
Given N courses with pre-requisites.

Check if it is possibble to finish all the courses

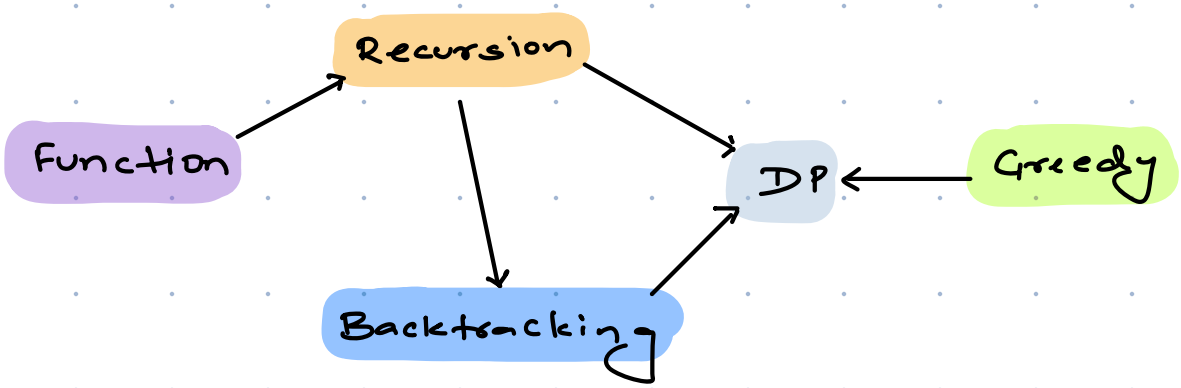
Course	pre-requisite for
1	2, 3
2	3, 5
3	4
4	2

{ 1 is prerequisite for 2 & 3 }





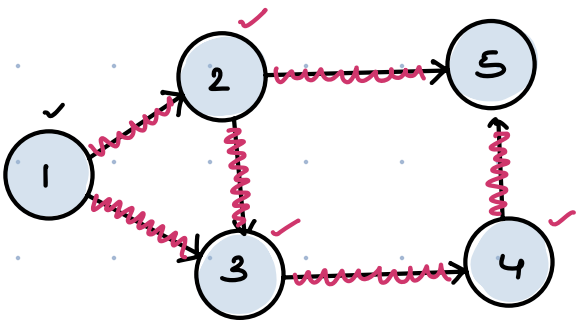
Note → In case of cycle, courses can't be finished



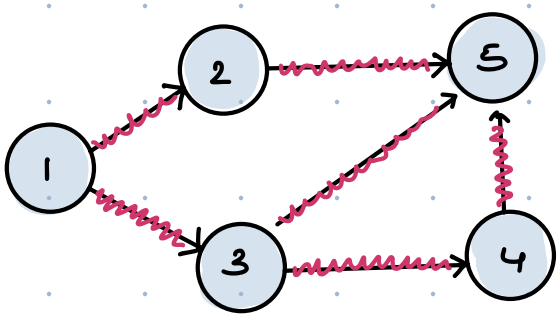
DSA curricullum

Function Greedy Recursion Backtracking DP

Function Recursion Backtracking Greedy DP



Ans - 1 2 3 4 5



Ans = 1 2 3 4 5 ✓

1 3 2 4 5 ✓

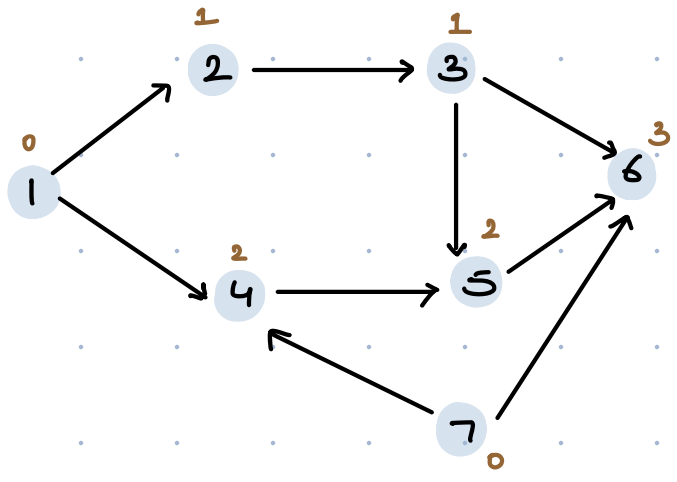
1 3 4 2 5 ✓

Topological Sort

Definition

Topological sort is a linear ordering of the vertices (nodes) in a directed acyclic graph (DAG) such that for every directed edge (u, v), vertex u comes before vertex v in the ordering. In other words, it arranges the nodes in such a way that if there is a directed edge from node A to node B, then node A comes before node B in the sorted order.

Find topological sort of below graph (Kahn's Algo)

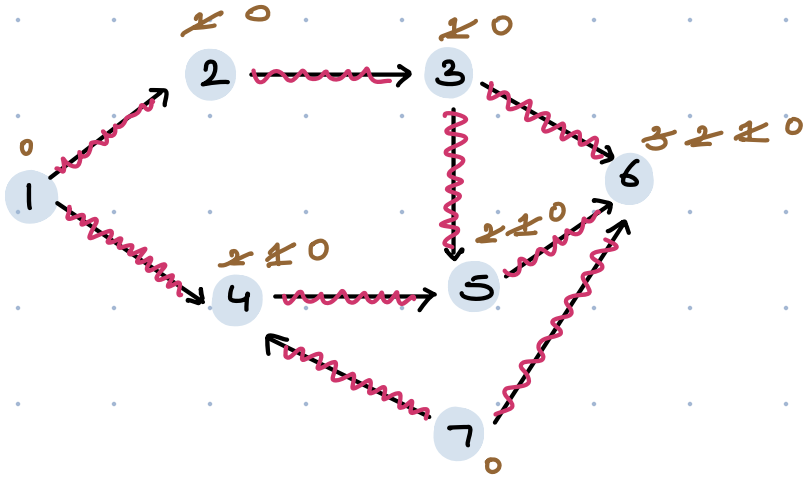


0 → { }
1 → {2, 4}
2 → {3}
3 → {5, 6}
4 → {5}
5 → {6}
6 → { }
7 → {4, 6}

01. Calculate indegree of every vertex

```
int[] indegree = new int[n+1];

for(i=0; i<=n; i++){
    for(int nbr : graph[i]){
        indegree[nbr]++;
    }
}
```

* Put all nodes with indegree 0 inside queue & start relaxing nbrs. If indegree[nbr] = 0, put nbr in queue.



1 7 2 4 3 5 6 ◯

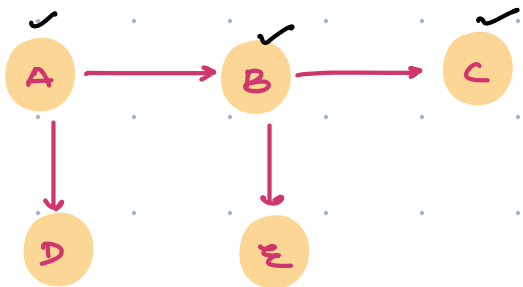
Ans = 1 7 2 4 3 5 6

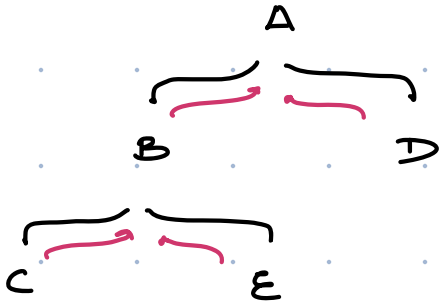
```
for (i=0; i<N; i++){
    if (indegree[i]==0) q.add(i);
}

while (q.size()>0){
    int rem = q.remove();
    print(rem);
    for (int nbr : graph[rem]){
        indegree[nbr]--;
        if (indegree[nbr]==0) q.add(nbr);
    }
}
```

TC : O(V+E)

SC : O(V)

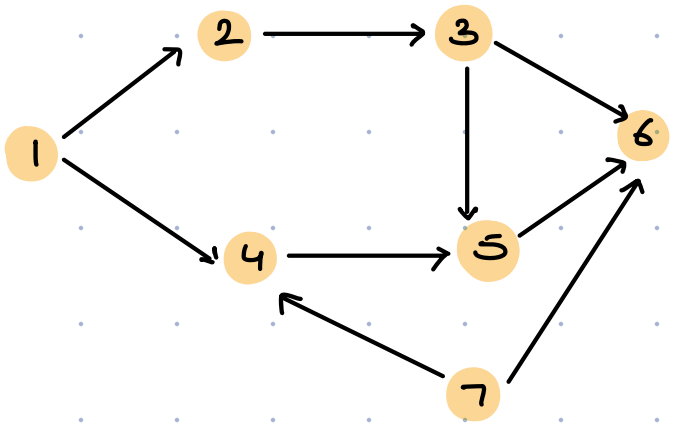
* Topological Sort (Right to left)





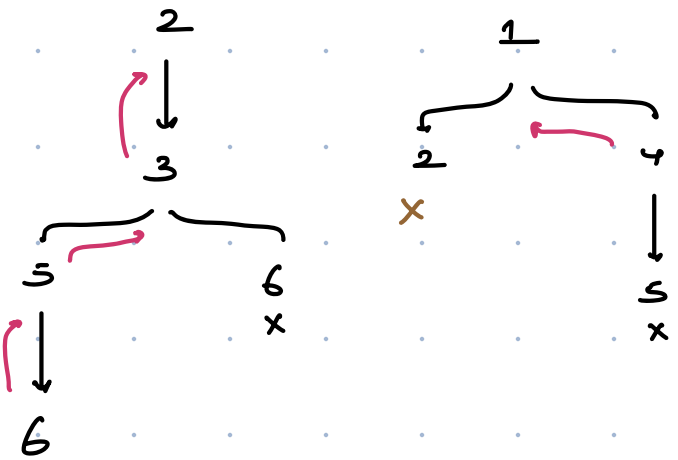
C E B D A

reverse = A D B E C

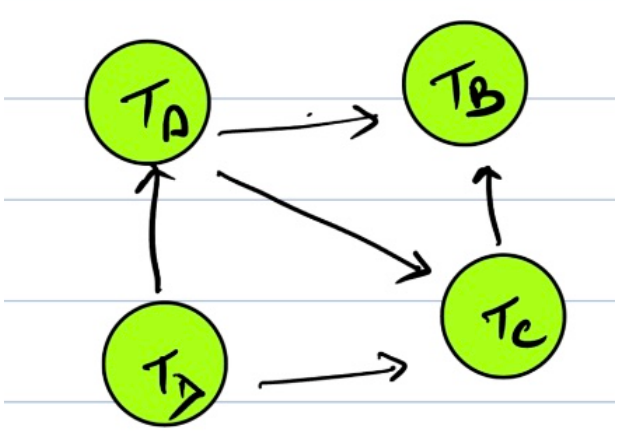


6 5 3 2 4 1 7

Rev = 7 1 4 2 3 5 6



```
Stack <I> st = new Stack <>();
int [] vis = new int [n];

for (i=0; i<N; i++){
    if (vis[i] == false) dfs (graph, i, vis, st)
}

void dfs (graph, src , vis, st ){
    vis [src] = 1 ;
    for ( int nbr : graph [src]){
        if (vis [nbr] == false) dfs (graph, nbr, vis, st):
    }
    st.push(src)
}

while (st.size() > 0){
    print (st.pop()):
}
```



⇒ T_D T_A T_C T_B

Problem 2 Minimum Jumps to Reach End

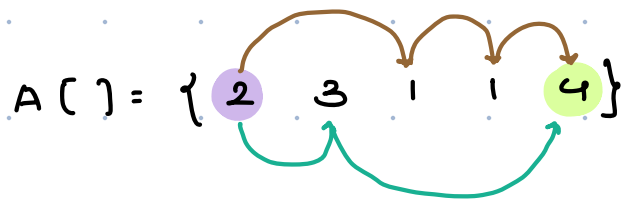
Description:

You are given a 0-indexed array of integers arr of length n. You are initially positioned at arr[0].

Each element arr[i] represents the maximum length of a forward jump from index i. In other words, if you are at arr[i], you can jump to any arr[i + j] where:

- 0 <= j <= arr[i]
- i + j < n

Return the minimum number of jumps to reach arr[n - 1]. The test cases are generated such that you can reach arr[n - 1].



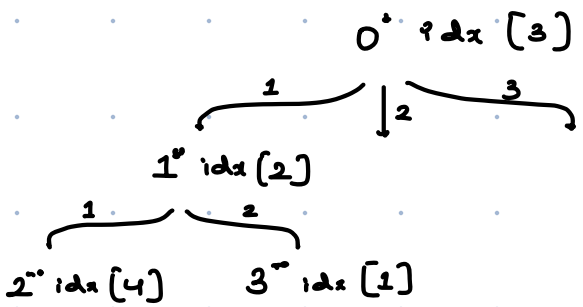
jumps = 3

jumps = 2

A[ ] = { 0 1 4 } jump = ∞ (not possible)

A[ ] = { 3 2 4 } jump = 1

A[ ] = { 3 2 4 1 2 3 0 0 1 }
0 1 2 3 4 5 6 7 8




```
int minjump ( A[], int i )        // i → 0
    if ( i == A.length - 1) return 0;
    if ( i ≥ A.length) return ∞;
    mini = ∞
    for ( jump = 1 ; jump ≤ A[i] ; jump++){
        int rec = minjump ( A , i + jump)
        mini = Math.min ( mini, rec);
    }
    return mini == ∞ ? mini : mini + 1 ;
}
```

TC: O(max of A)^n

SC: O(n)

A[] = { 3 2 4 1 2 3 0 0 1 }
idx: 0 1 2 3 4 5 6 7 8

dp[] = { 3 3 2 3 2 1 ∞ ∞ 0 }

dp[i] = min jumps to reach last idx

```
dp[n-1] = 0
for (i=n-2 ; i≥0 ; i--){
    int mini = ∞
    for (j=1 ; j ≤ A[i] && i+j<n ; j++){
        mini = Math.min (mini, dp[i+j]);
    }
    dp[i] = mini == ∞ ? ∞ : mini+1;
}
return dp[0];
```

TC: O(n * max)
SC: O(n)

A[] = { 2 3 0 1 4 }
idx: 0 1 2 3 4

dp[] = { 2 1 ∞ 1 0 }

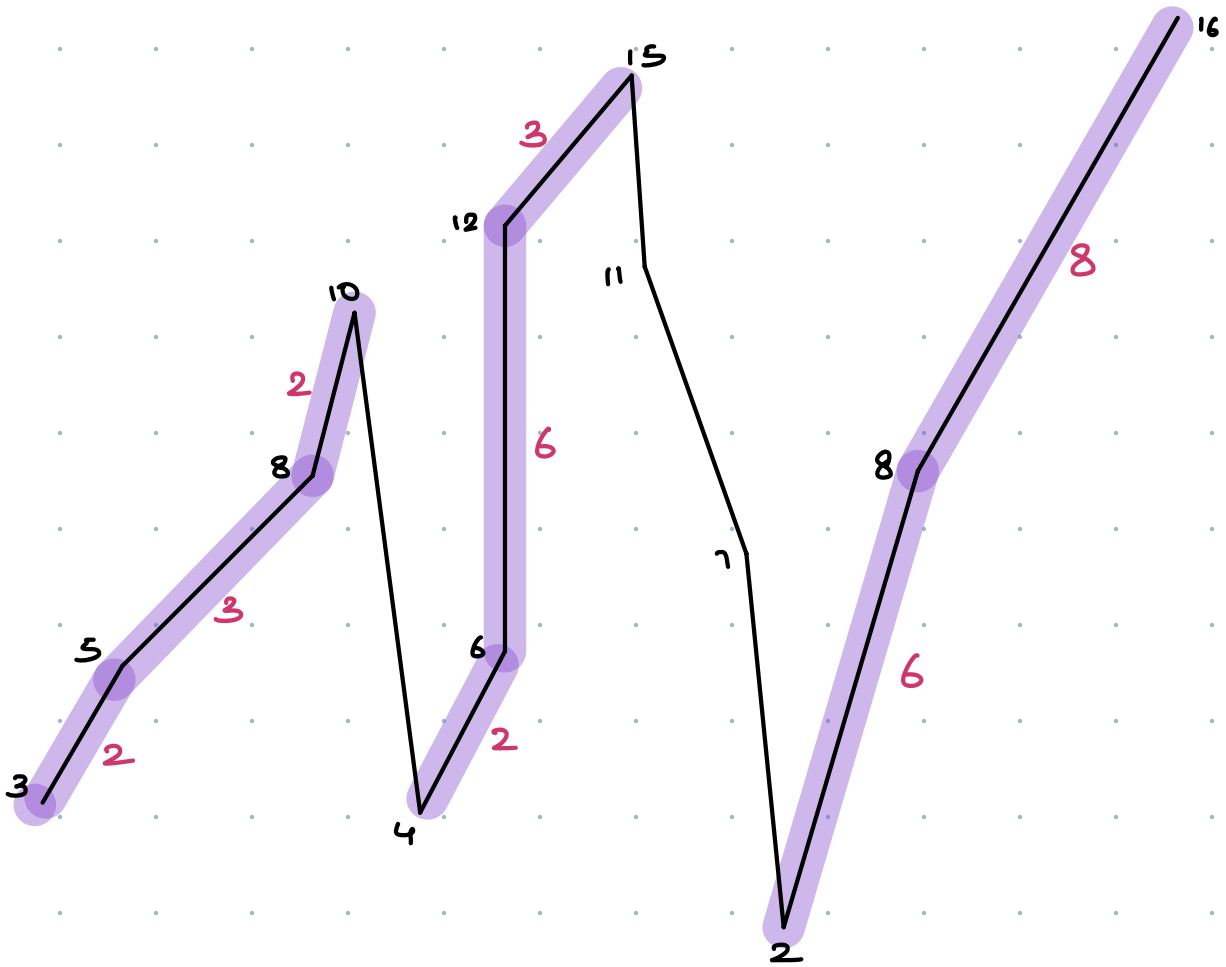
Problem 3 Maximum Profit from Stock Prices

FIGQLUT

Problem Statement

Given an array A where the i-th element represents the price of a stock on day i, the objective is to find the maximum profit. We're allowed to complete as many transactions as desired, but engaging in multiple transactions simultaneously is not allowed.

A[] = { 3 5 8 10 4 6 12 15 11 7 2 8 16 }




```
profit = 0
for (i=0 ; i<n-1 ; i++){
    if (A[i+1] > A[i]){
        profit = A[i+1] - A[i];
    }
}
return profit;
```

TC : O(n)

SC : O(1)

Prediction

HashMap → subarr with sum = 0, sum = B,
Longest consecutive sequence
Minimum window substring

DP / Trees

Graph → No. of islands, rotten oranges,
Nearest Distance from 1s, Possibbility of
finishing course

prims & Dijkstra

Sorting or Searching

↓

Custom comparator

→ A+ magical no
→ Search in rotated sorted Arr
→ Aggressive cows / Painter partition /
Koko eating banana / Books Allocation

LL / Stack

↓

Reversing / middle of LL / cycle in LL

Finding min/max on left or right
Killer fish
Asteroid collision
Maximum in window → Deque