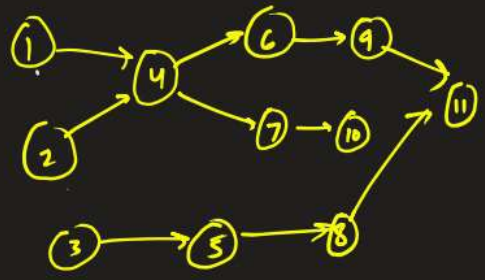
Trees & Graphs Lecture 7

Friday, 23 August 2024 6:03 AM

Topological Sorting



2, 1, 4, 6, 9, 7, 10, 3, 5, 8, 11

1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11.

① Using DFS:

dfs(3)

dfs(5)

dfs(8)

T = O(V+E)

S = O(V)

ans

11	9	6	10	7	4	1	2	8	5	3

ans reverse:-

3	5	8	2	1	4	7	10	6	9	11

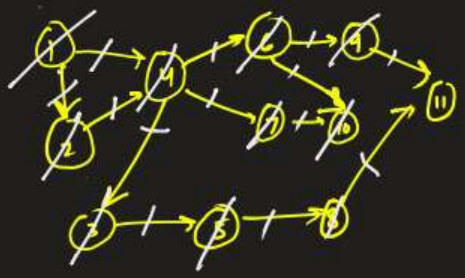
https://www.geeksforgeeks.org/problems/topological-sort/1

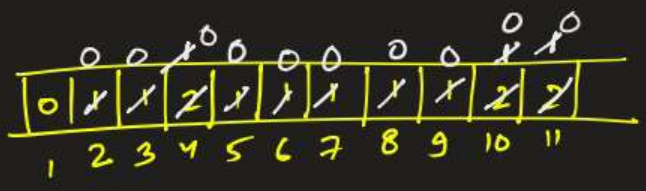
```cpp
// } Driver Code Ends
class Solution
{
    vector<bool> vis;
    vector<int> ans;

    void dfs(int s, vector<int> adj[]) {
        vis[s] = true;
        for(int v: adj[s])
            if(!vis[v])
                dfs(v, adj);
        ans.push_back(s);
    }

    public:
    //Function to return list containing vertices in Topological order.
    vector<int> topoSort(int V, vector<int> adj[]) {
        vis.assign(V, false);
        ans.clear();
        for(int i=0; i<V; i++)
            if(!vis[i])
                dfs(i, adj);
       reverse(ans.begin(), ans.end());
       return ans;
    }
};
```

② Kahn's Algorithm





$T = O(V+E)$

$S = O(V)$

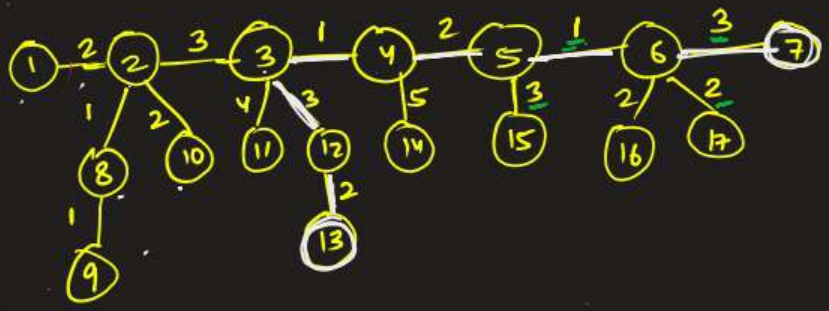
ans = 1 2 4 6 7 3 9 10 5 8 11

```python
from collections import deque

class Solution:

    #Function to return list containing vertices in Topological order.
    def topoSort(self, V, adj):
        id = [0]*V
        for u in adj:
            for v in u:
                id[v] += 1
        q = deque()
        for i in range(V):
            if id[i] == 0:
                q.appendleft(i)
        ans = []
        while q:
            ans.append(q[-1])
            for v in adj[q[-1]]:
                id[v] -= 1
                if id[v] == 0:
                    q.appendleft(v)
            q.pop()
        return ans
```

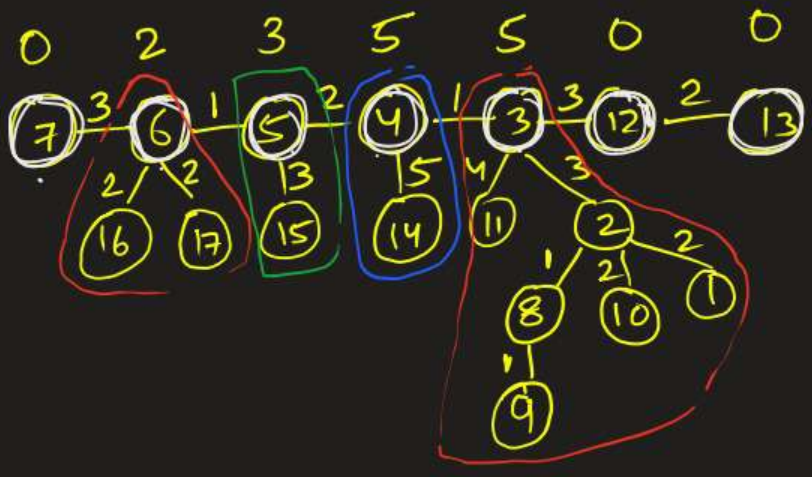
https://codeforces.com/contest/1004/problem/E



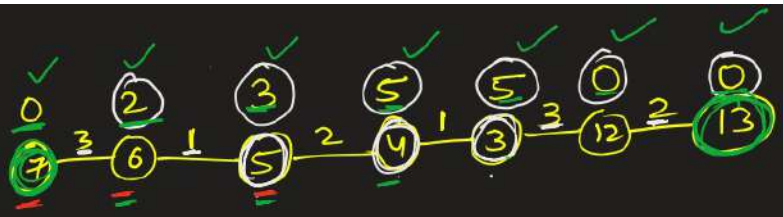
Diameter:- path on the tree with maximum distance

Step (1) Task: Find the diameter of the tree.

- From any node, find the farthest node (a)
- From a, find the farthest node (b)
- [a-b] is a diameter



at most k shops

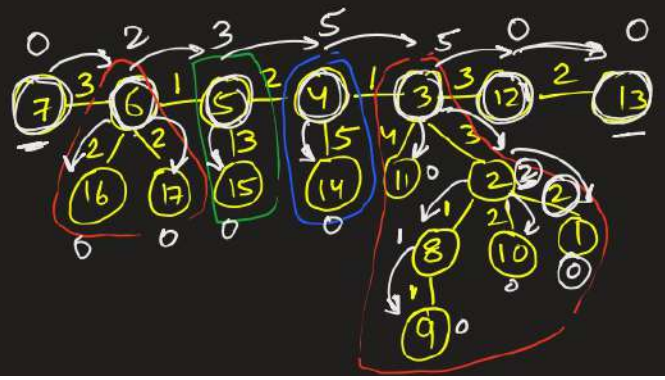


$k = 3$

$\max(3, 5, 5, 3+0, 3+2+0, 1+2, 3+1+0) = 5$

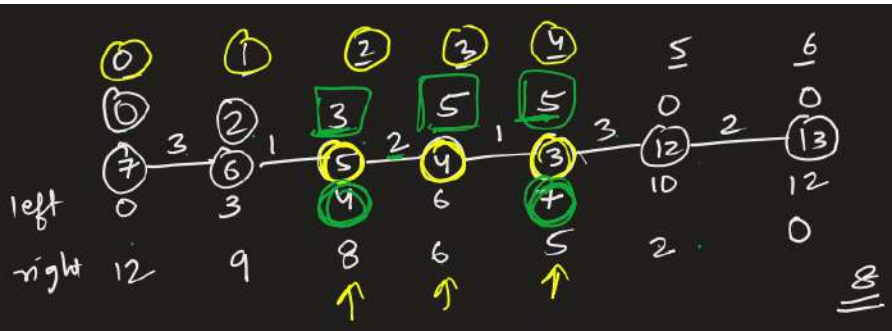
5 4 3 12 13 6 7

Step 2: Find the diameter path & max. distance of the path nodes to any of its subtree children.



↑ rnd = max distance of node to its children

Step 3



8

→ max(left(5), right(3), max(md))

left[6] = (left[7] + a[6-7], md[6])

S: 0 → ds - r + 1

k max ice cream stalls

r = min(k, size(dp))

k = 3, r = 3

```cpp
umap<int, int> a[100001];
bool vis[100001];
int n,k,u,v,d;
pii dfs(int s) {
    vis[s] = true;
    pii dis;
    int md = 0, node = s;
    for(auto p: a[s]) {
        if(!vis[p.first]) {
            dis = dfs(p.first);
            if(dis.second+p.second > md) {
                md = dis.second+p.second;
                node = dis.first;
            }
        }
    }
    return mp(node, md);
}
pii dia_nodes() {
    FOR(i,0,n+1) vis[i] = false;
    pii p1 = dfs(1);
    FOR(i,0,n+1) vis[i] = false;
    pii p2 = dfs(p1.first);
    return mp(p1.first, p2.first);
}
bool dfs_path(int s, int d, vi &dp, vi &md) {
    vis[s] = true;
    if(s == d) {
        dp.pb(d);
        return true;
    }
    bool idp = false;
    for(auto e: a[s]){
        if(!vis[e.first]) {
            if(dfs_path(e.first, d, dp, md)) {
                dp.pb(s);
                idp = true;
            }
            else
                md[s] = max(md[s], e.second+md[e.first]);
        }
    }
    return idp;
}
```

```cpp
void solve() {
    cin >> n >> k;
    FOR(i, 0, n-1) {
        cin >> u >> v >> d;
        a[u][v] = d;
        a[v][u] = d;
    }
    pii dn = dia_nodes();
    // cout << dn.first << " " << dn.second << endl;
    FOR(i, 0, n+1) vis[i] = false;
    vi dp, md(n+1, 0);
    dfs_path(dn.first, dn.second, dp, md);
    // for(auto i: dp) cout << i << " " << md[i] << endl;
    int ds = dp.size(), mmd = *max_element(md.begin(), md.end());
    vi left(ds, 0), right(ds, 0);
    FOR(i, 1, ds)
        left[i] = max(md[dp[i]], left[i-1]+a[dp[i]][dp[i-1]]);
    RFOR(i, ds-2, 0)
        right[i] = max(md[dp[i]], right[i+1]+a[dp[i]][dp[i+1]]);
    int r = min(k, ds), ans = INT_MAX;
    FOR(i, 0, ds-r+1)
        ans = min(ans, max(mmd, max(left[i], right[i+r-1])));
    cout << ans;
}
```