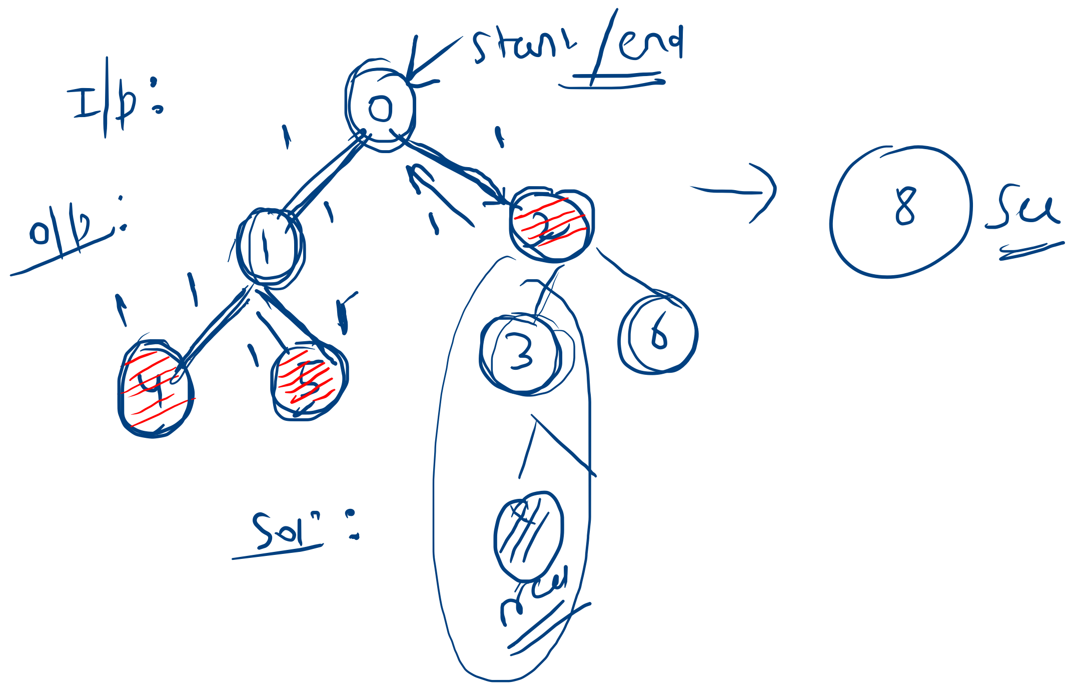
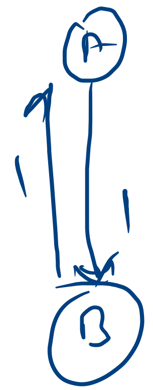
Minimum Time to Collect All Apples in a Tree

n = 7

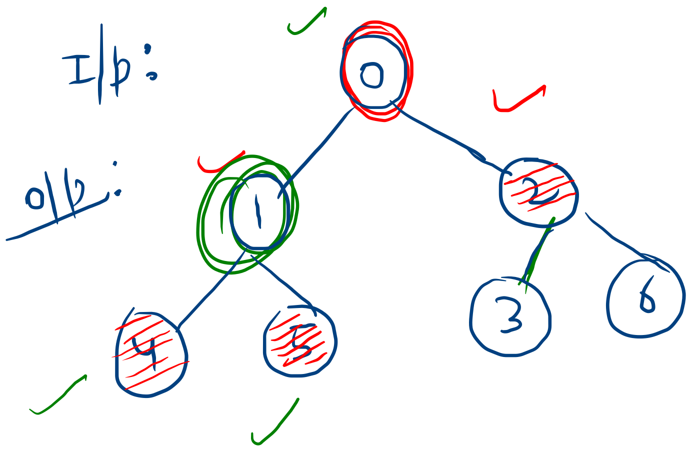




{0,1} ✓
{0,2} ✓
{1,4}
{1,5}
{2,3}
{2,6}

I/p:

o/p:



vector<vector<int>> adj;

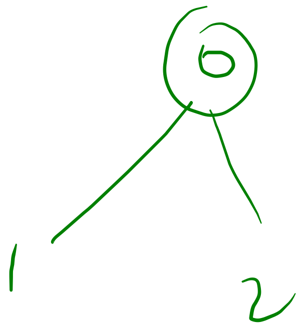
adj[][] =

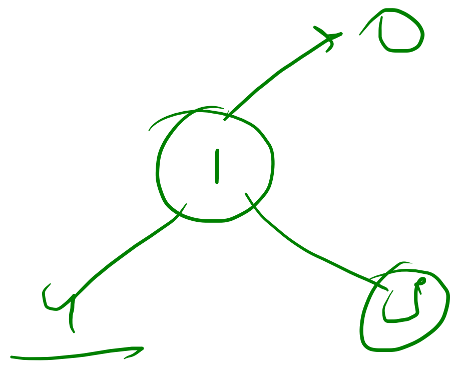
0	1	2	
1	0	4	5
2	0	3	6
3	2		
4	1		
5	1		
6	2		

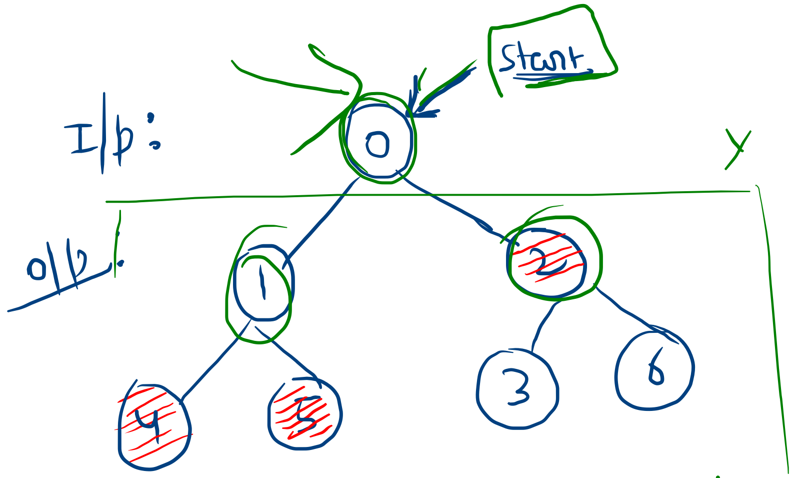
adj[][]

n = 7

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



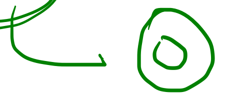


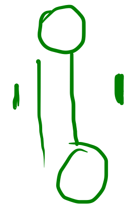


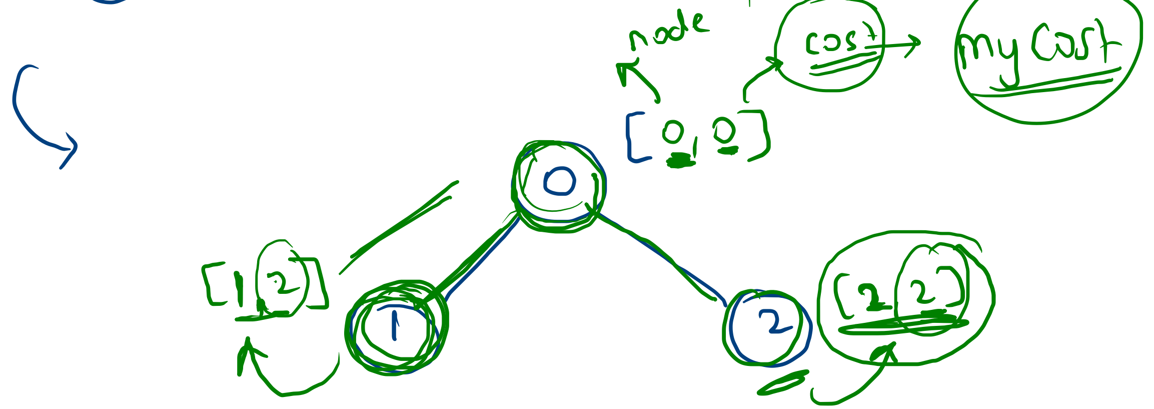
→ now we will make fun^n

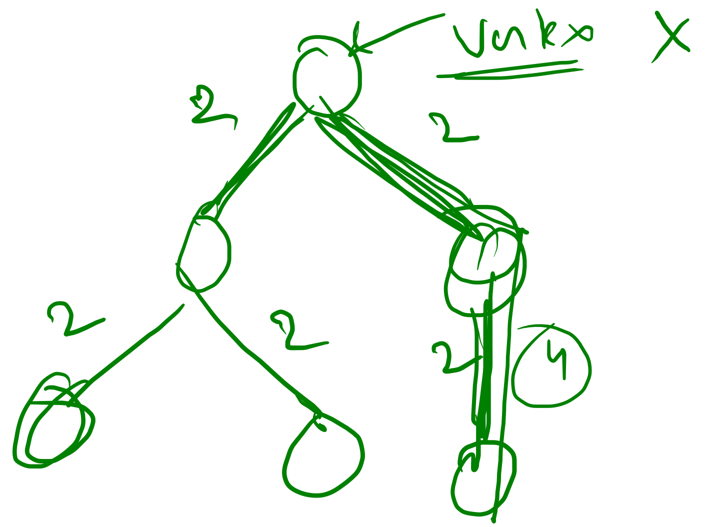
[∴ start from 0 vertex
end at 0 vertex]

→ dfs(node, my cost)





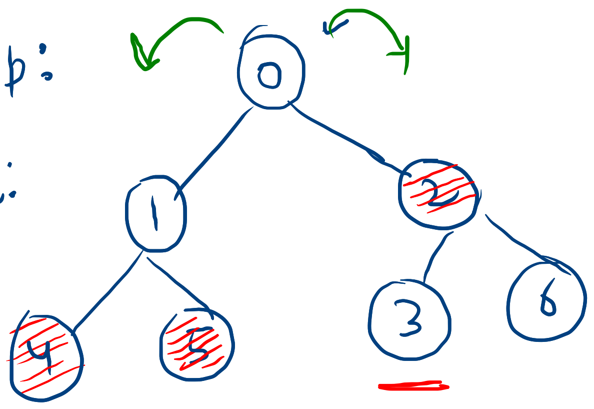


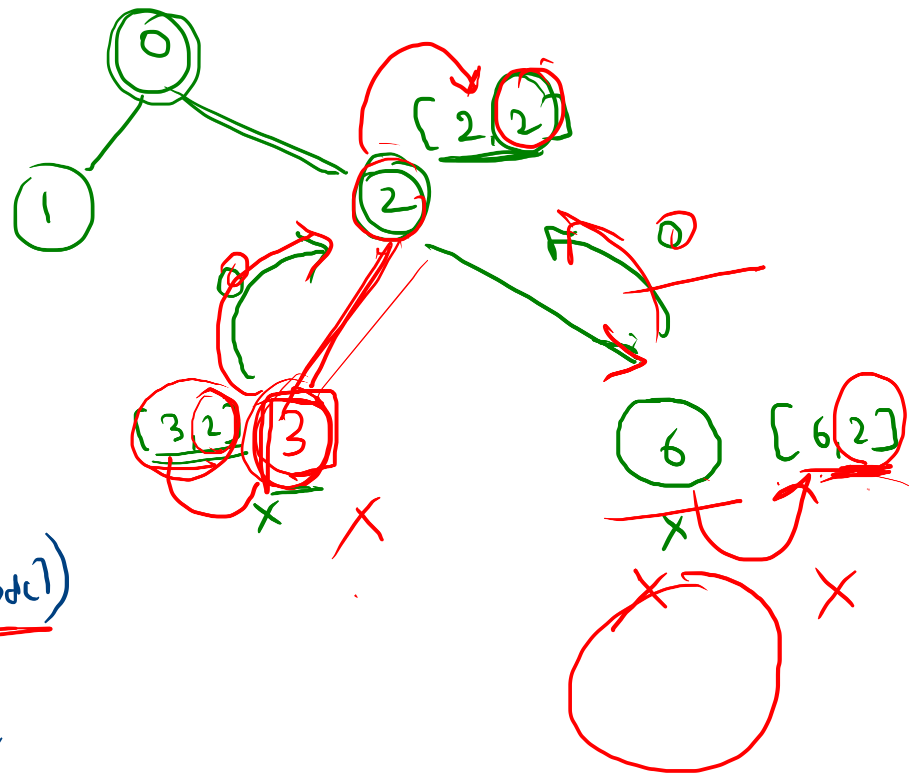


visited[ ]

I/p:

o/p:





0 → 1, 2

1

$\boxed{2 \to 0, 3, 6}$

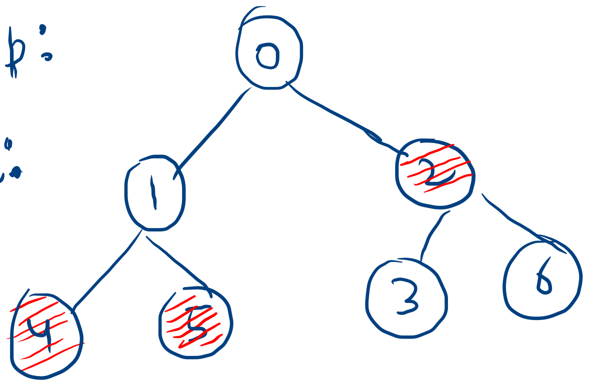
$\boxed{6 \to 2}$

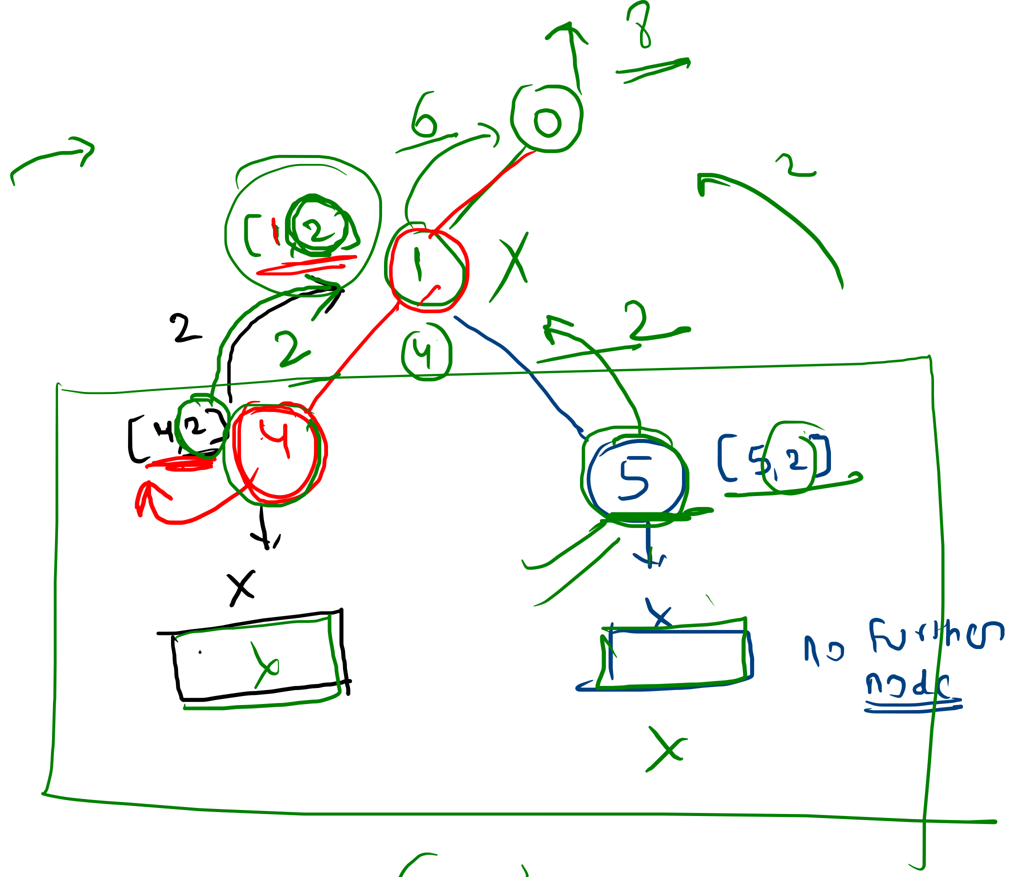
if(visited[node])

return 0;

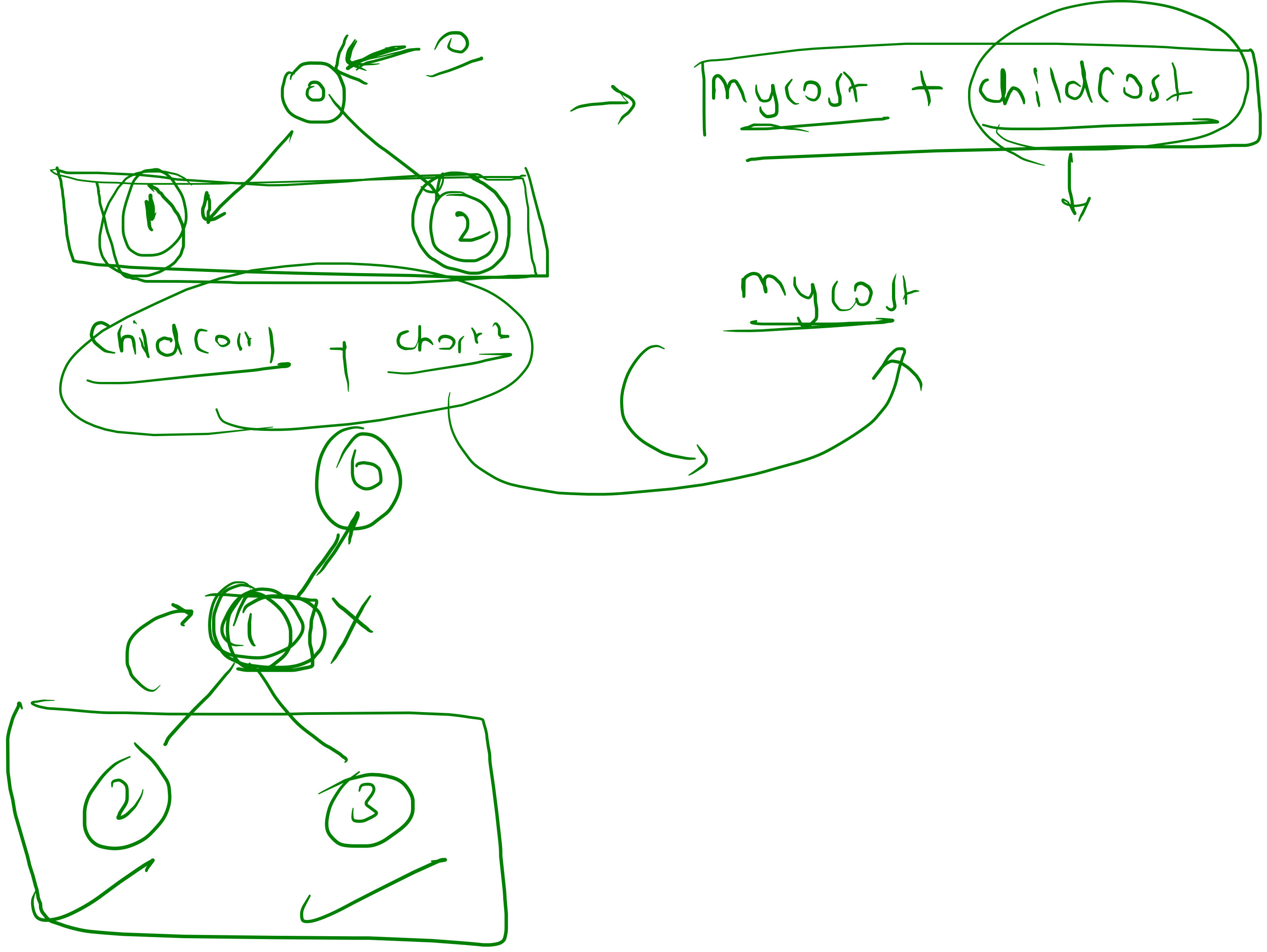
I/p:

O/p:



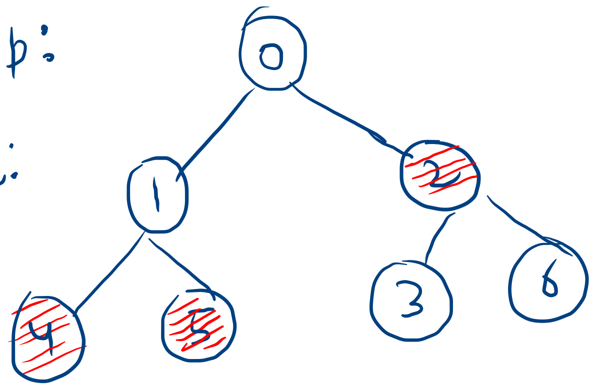


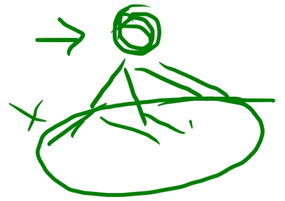
Cost


0
0
1
2
mycost + childcost
childcost1 + chost2
mycost
6
1
2
3

I/p:

O/p:

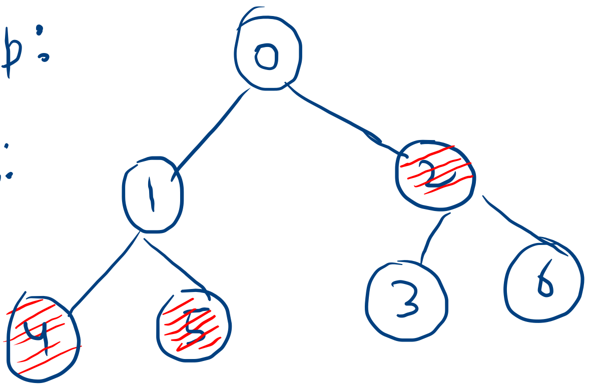




```
int dfs( int node, int mycost, vector<bool > hasApple)
{
    if( visited[node] ) return 0;
    visited[node] = True;

    int childCost = 0;
    for( auto child : adj[node] )
    {
        childCost += dfs( child, 2, hasApple );

    }

    if( childcost == 0  And  !hasApple[node] )
        return 0;

}
```

I/p:

o/p:



```
int dfs( int node, int mycost, vector<bool> hasApple)
{
    if( visited[node] ) return 0;
    visited[node] = True;

    int childCost = 0;
    for( auto child : adj[node] )
    {
        childCost += dfs( child, 2, hasApple );
    }

    if( childcost == 0 And !hasApple [node] )
        return 0;

    return myCost + childcost;
}
```