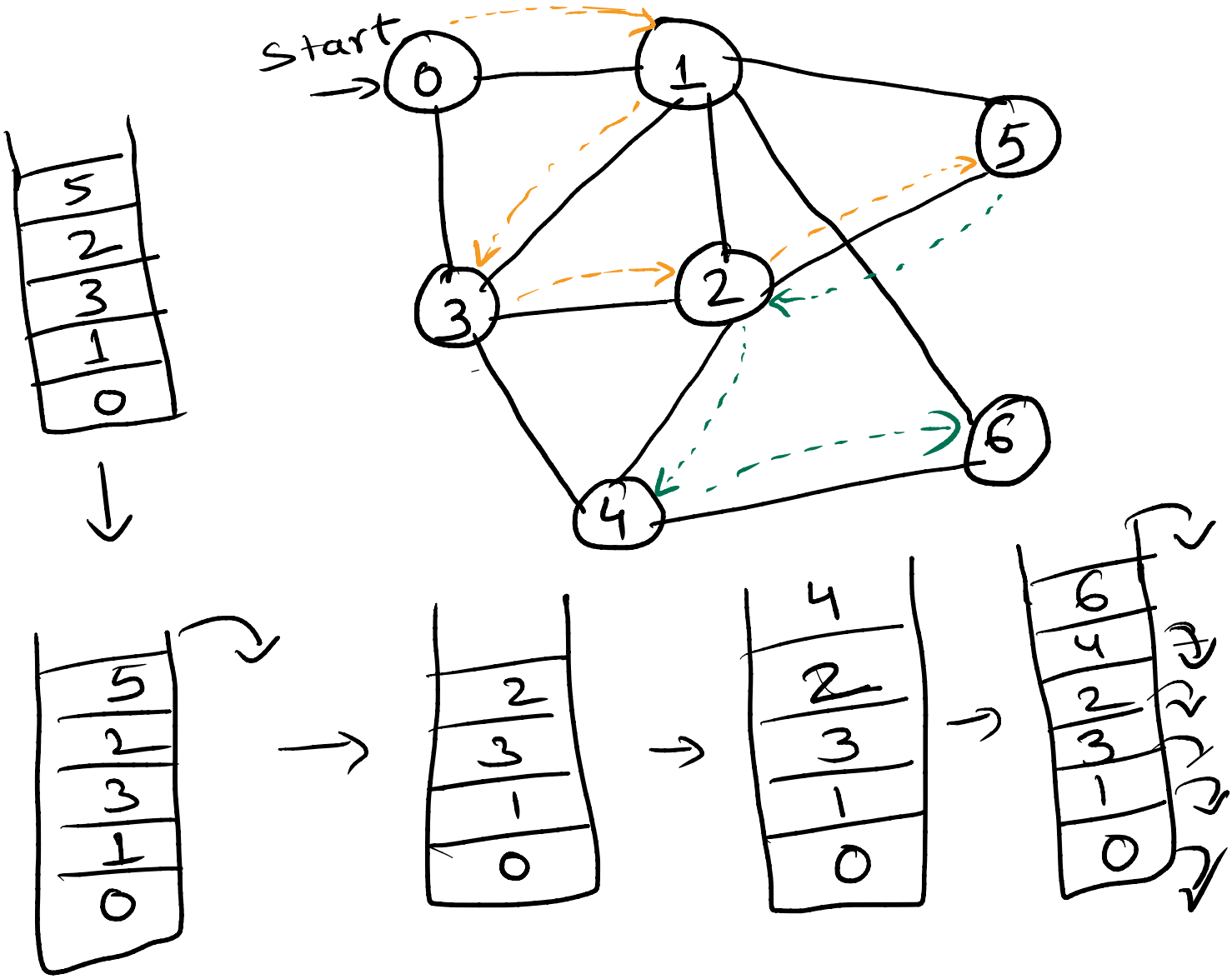
DFS (Depth first Search)

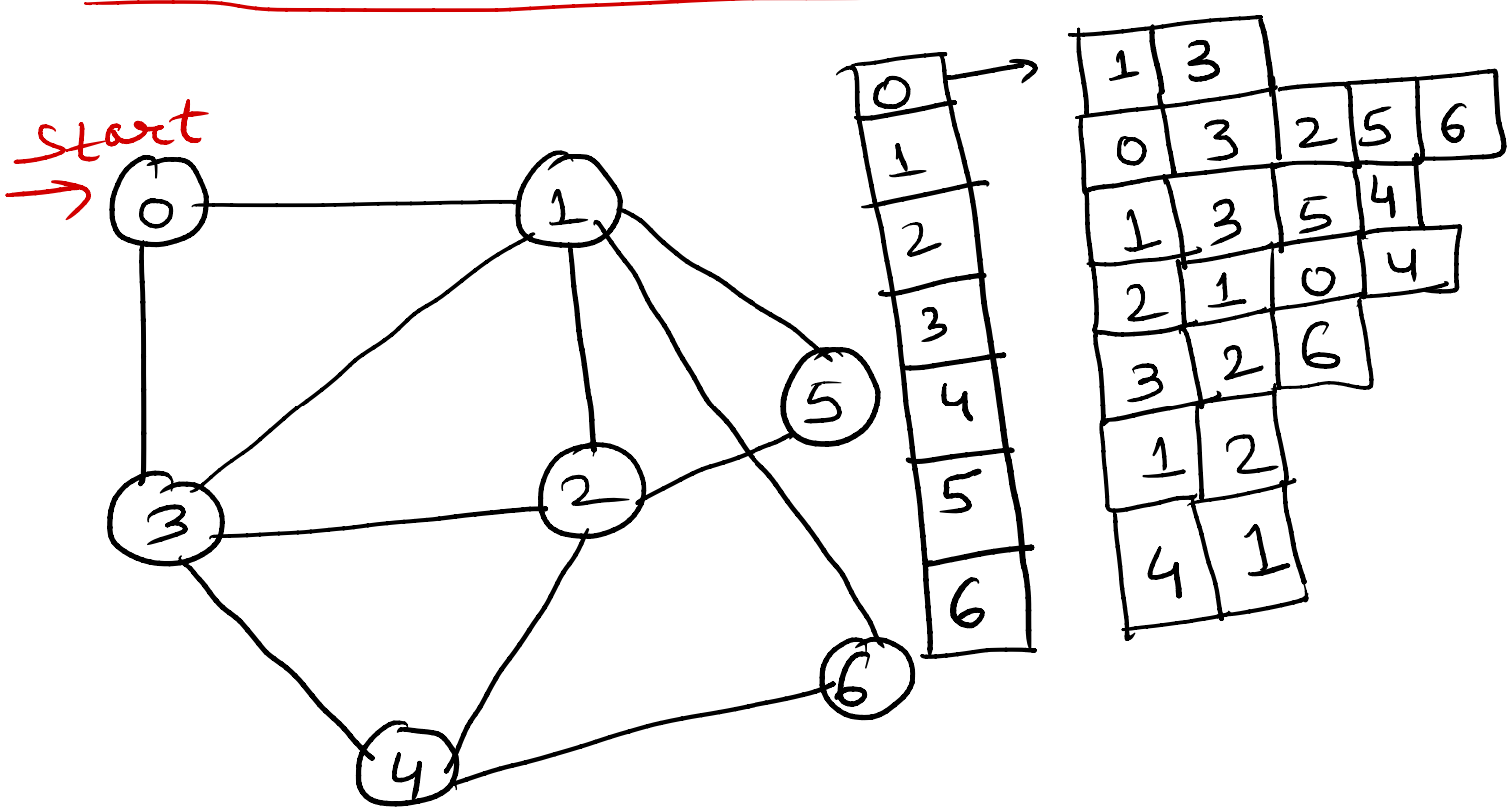
→ stack is used



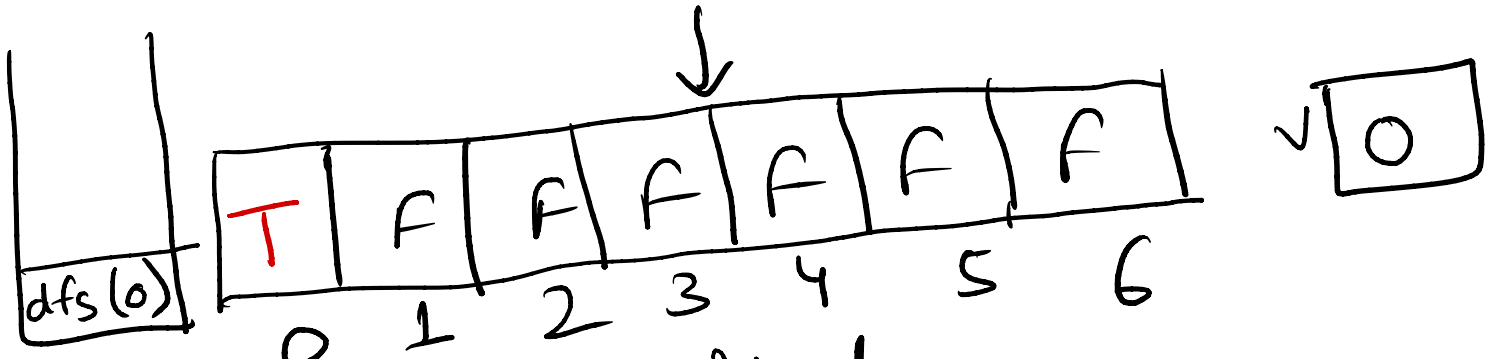
DFS : 0, 1, 3, 2, 5, 4, 6

Traversal

DFS (Depth first Search)



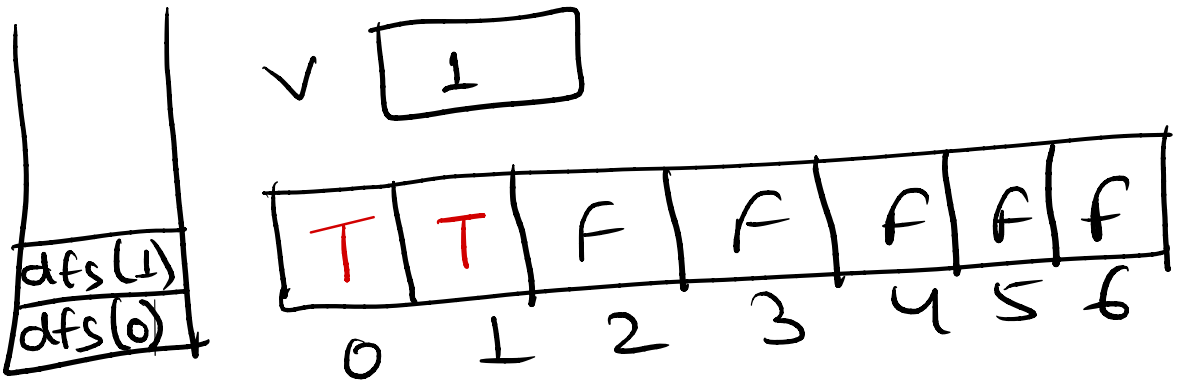
Visited



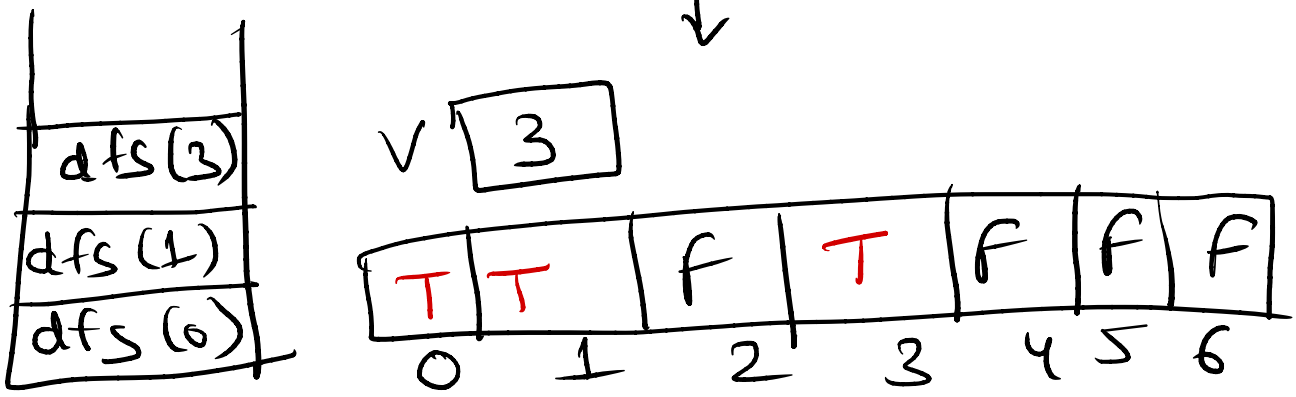
Visited

Adj u of '0' = {1, 3}

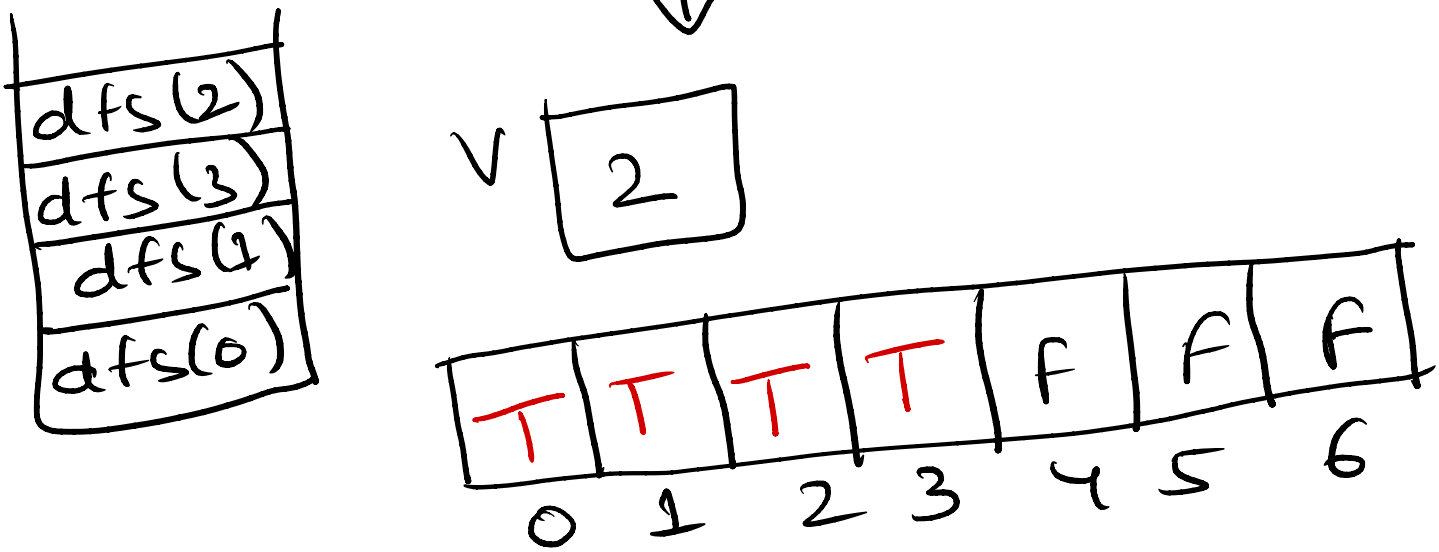
↓



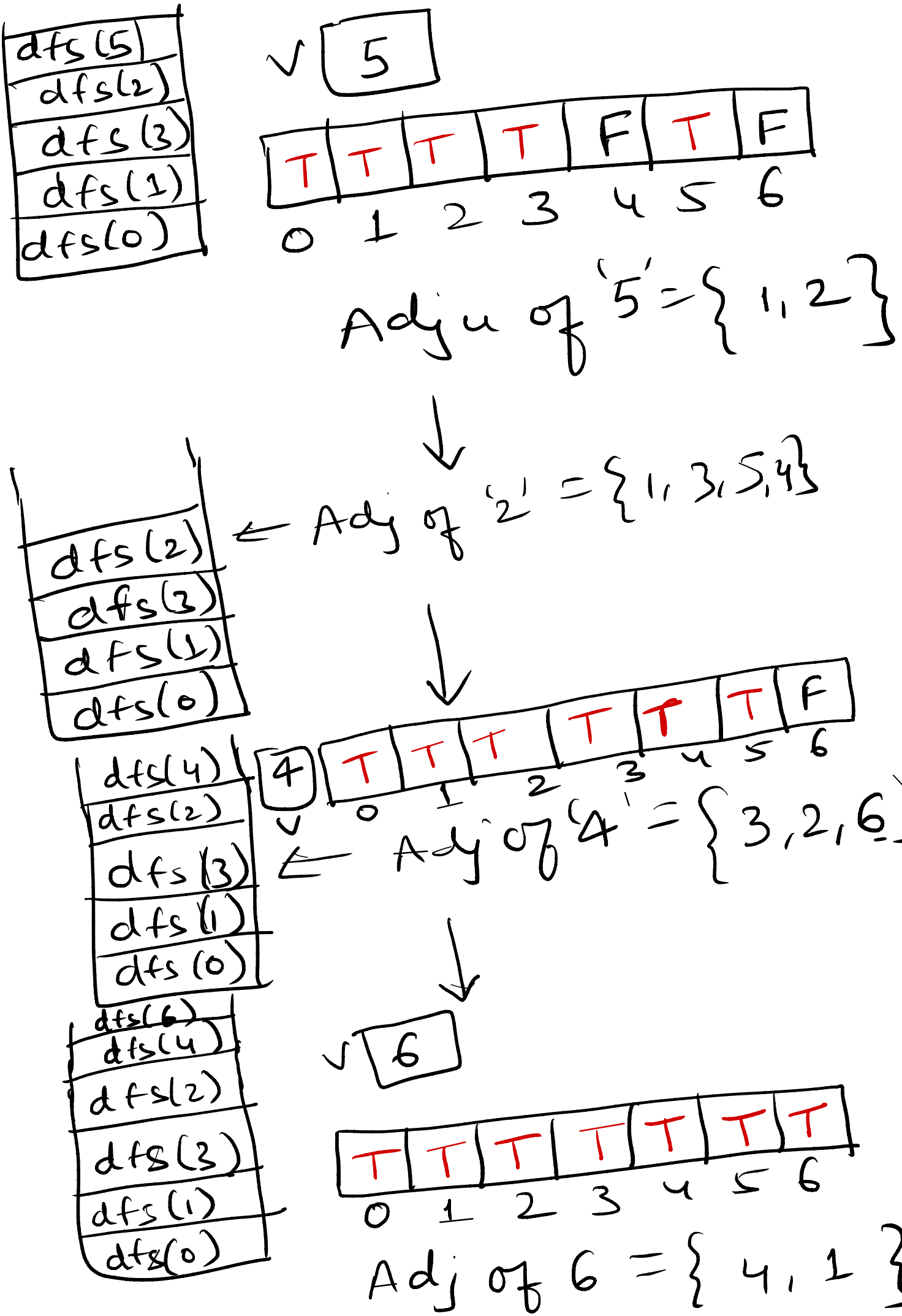
Adj u of '1' = { 0, 3, 2, 5, 6 }

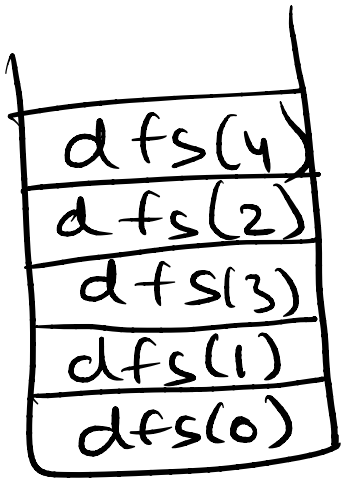


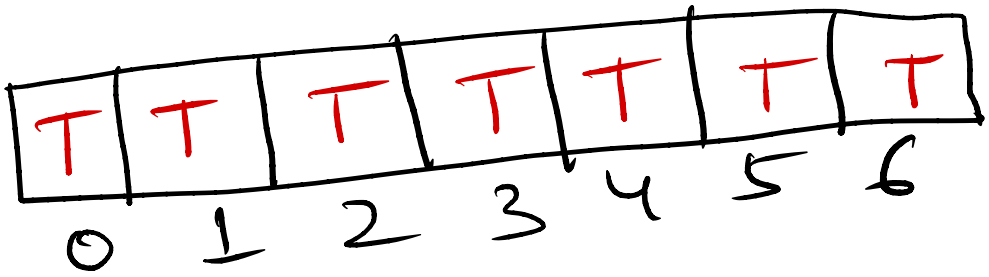
Adj u of '3' = { 2, 1, 0, 4 }



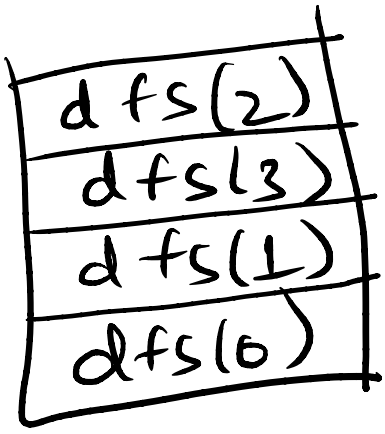
Adj u of '2' = { 1, 3, 5, 4 }


dfs(5)
dfs(2)
dfs(3)
dfs(1)
dfs(0)
5
T T T T F T F
0 1 2 3 4 5 6
Adj of '5' = { 1, 2 }
Adj of '2' = { 1, 3, 5, 4 }
dfs(2)
dfs(3)
dfs(1)
dfs(0)
T T T T T T F
0 1 2 3 4 5 6
dfs(4)
dfs(2)
dfs(3)
dfs(1)
dfs(0)
4
Adj of '4' = { 3, 2, 6 }
dfs(6)
dfs(4)
dfs(2)
dfs(3)
dfs(1)
dfs(0)
6
T T T T T T T
0 1 2 3 4 5 6
Adj of 6 = { 4, 1 }

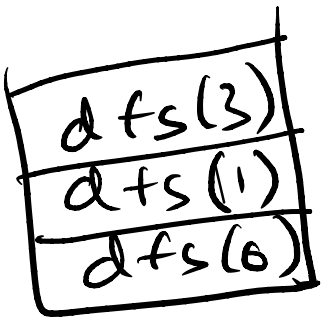




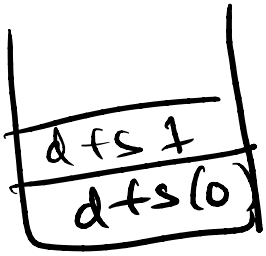
Adj of 4 = { 3, 2, 6 }



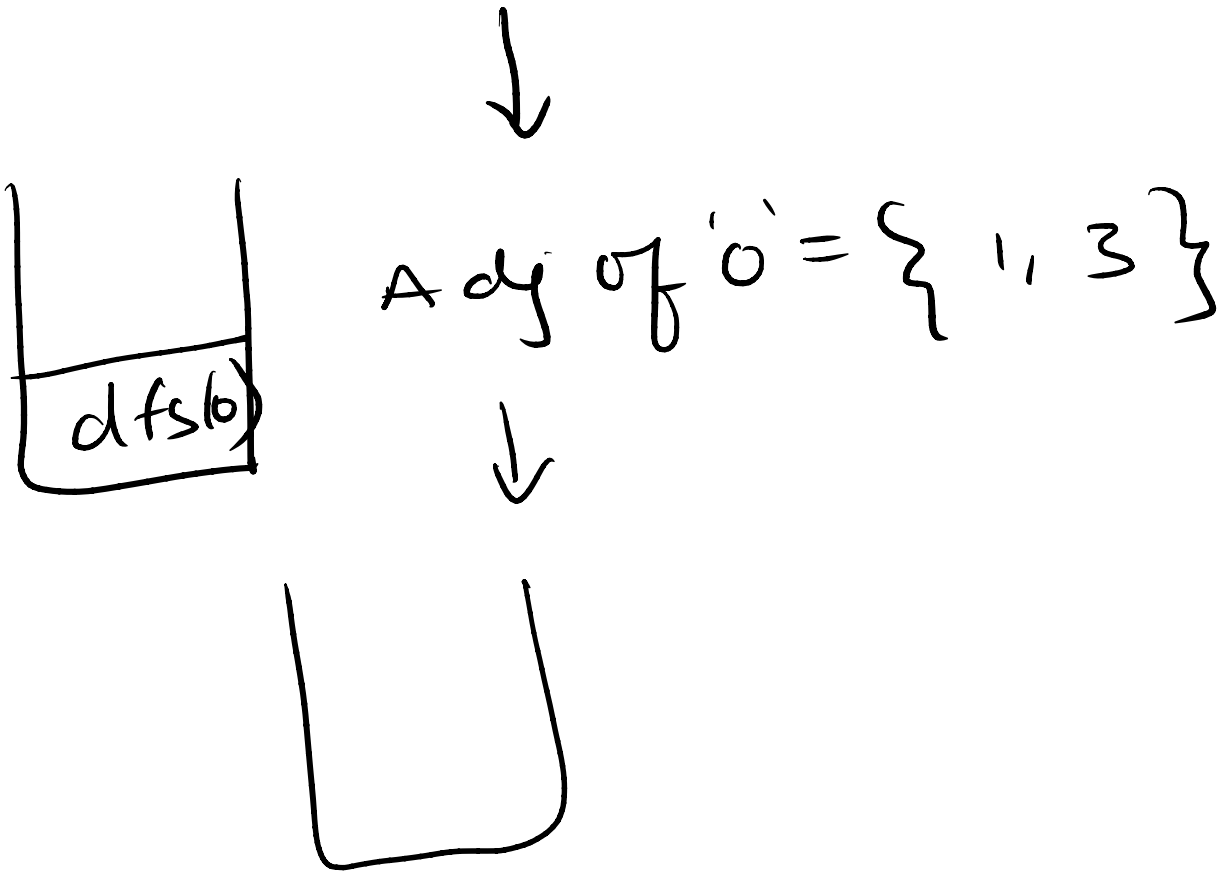
Adj of '2' = {1, 3, 5, 4}



Adj. of '3' = {2, 1, 0, 4}



Adj of '1' = {0, 3, 2, 5, 6}



DFS → 0, 1, 3, 2, 5, 4, 6

One of the possible traversal

Pseudocode for DFS

```
Initialize visited Array

DFS(v)
{
    visited[v] = true;          ---- O(1)
    print(v);                   } C
    for each 'u' adjacent to 'v'
    {
        if (visited[u] = false)
            DFS(u);
    }
}
```

N_v

Space Complexity:

visited: V
stack: V
Adj List: V+E
O(V+E)

Time Complexity:

$(C + N_i) + (C + N_i) + (C + N_i) + \cdots + (C + N_i)$

$= cV + \sum N_i$ for all vertices

$= cV + E$

$= O(V+E)$

BFS	DFS
→ Uses Queue	→ Uses Stack
→ Used for finding single source shortest path	→ may traverse more edges to reach a destination vertex
→ Not suitable for games & puzzles	→ More suitable
→	