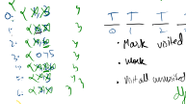
Depth first Search

0 1 5 3 4 2 6



- Mark visited
- Work
- Visit all unvisited neighbours

graph.get(4)

- Mark visited
- Work
- Visit all unvisited neighbours

dfs(node){
visited[node] = true
System(node)
for(int nbr: graph.get(node)){
if(!visited[nbr]) dfs(nbr)
}
}

DFS

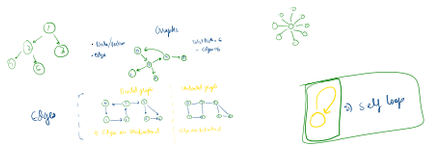
graph.get(4)

```
ArrayList<ArrayList<Integer>> graph = new

//initialize internal arraylist
for(int i = 0;i<n;i++){
    graph.add(new ArrayList<Integer>());
}

//connect the vertices according to the g
for(List<Integer> edge: edges){
    int u = edge.get(0);
    int v = edge.get(1);

    graph.get(u).add(v);
    graph.get(v).add(u);
}
```

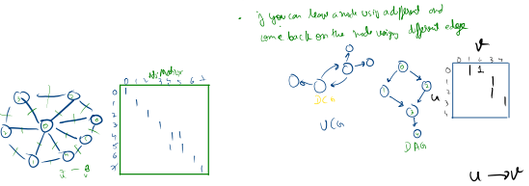
0 1
0 3
5 3
3 4
4 2
4 6
2 6

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

1 5
u v



- If you can leave a node using a different and come back on the node using different edge



u → v → u.v

u → v → uv vu

Adj List

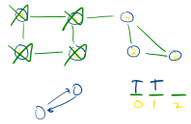
1. Array of ArrayList Of 7

[null
null
null
null
null
null
null]

```
ArrayList<Integer>[] arr = new ArrayList<>[7];

for(int i = 0;i<V;i++){
    arr[i] = new ArrayList<>();
}
```

2. Initialize all Internal ArrayLists

[{}
{}
{}
{}
{}
{}
{}]



0 1 3 5

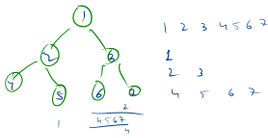
- Remove node for Queue
- Mark
- WORK
- Add/Visit all the unvisited NBR

I I _ I _ I _
0 1 2 3 4 5 6

4

→ DFS Back
→ Queue PQ
1D 2D
Mix Veg

~~Hard~~



1 2 3 4 5 6 7

1
2 3
4 5 6 7

```
for(int nbr: graph[cur]){
    if(vis[nbr] == false) q.add(nbr);
}
```

pop(3) 1. {2,45}