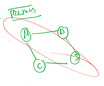
Graph — Time Sol & Space complexity

E.g.

Graph → Map → Adj List

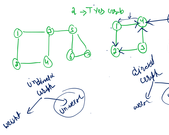
Facebook — Friends



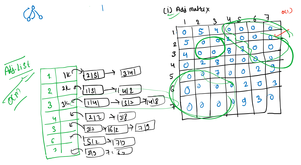
Time → Visited

Graph → Time ×

2 → Two Graph

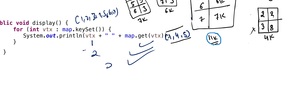


(1) Adj matrix



Graph

Time



2+2 (V+E)/2

Bipartite

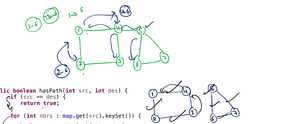
```
public void display() {
    for (int vtx : map.keySet()) {
        System.out.println(vtx + " " + map.get(vtx));
    }
}
```

```
    public void removevertex(int v1) {// v1=3
        for (int nbrs : map.get(v1).keySet()) {
            map.get(nbrs).remove(v1);// (1-->3)
        }
    }
```

Disconnected graph

Connected

```
public boolean hasPath(int src, int des) {
    if (src == des) {
        return true;
    }
    for (int nbrs : map.get(src).keySet()) {
        boolean ans = hasPath(nbrs, des);
        if (ans) {
            return ans;
        }
    }
    return false;
}
```



g.AddEdge(1, 4, 6);
g.AddEdge(1, 2, 10);
g.AddEdge(2, 3, 7);
g.AddEdge(3, 4, 5);
g.AddEdge(4, 5, 1);
g.AddEdge(5, 6, 4);
g.AddEdge(7, 5, 2);
g.AddEdge(6, 7, 3);

```
public boolean hasPath(int src, int des, HashSet<Integer> visited) {
    if (src == des) {
        return true;
    }
    visited.add(src);
    for (int nbrs : map.get(src).keySet()) {
        if (!visited.contains(nbrs)) {
            boolean ans = hasPath(nbrs, des, visited);
            if (ans) {
                return ans;
            }
        }
    }
    return false;
}
```

```
public void PrintPathJoint src, int des, HashSet<Integer> visited, String ans) {
    if (src == des) {
        System.out.println(ans+des);
        return;
    }
    visited.add(src);
    for (int nbrs : map.get(src).keySet()) {
        if (!visited.contains(nbrs)) {
            PrintPath(nbrs, des, visited, ans + src);
        }
    }
    visited.remove(src);
}
```

breadth first search

BFS % of FS

depth first search

1. remove
2. if visited continue
3. visited add
4. Self work
5. Add nbrs (unvisited)

5 ताकत इसे चाहिए
Queue इसे चाहिए

BFS 10:50 - 10:40 to 10

```
public boolean BFS(int src, int des) {
    Queue<Integer> q = new LinkedList();
    HashSet<Integer> visited = new HashSet();
    q.add(src);
    while (!q.isEmpty()) {
        // 1. remove
        int rv = q.remove();
        // 2. if visited continue
        if (visited.contains(rv)) {
            continue;
        }
        // 3. visited
        visited.add(rv);
        // 4. Self work
        if (rv == des) {
            return true;
        }
        // 5. Add nbrs (unvisited
        for (int nbrs : map.get(rv).keySet()) {
            if (!visited.contains(nbrs)) {
                q.add(nbrs);
            }
        }
    }
    return false;
}
```

BFS

1. remove
2. visited
3. Self work
4. Add nbrs