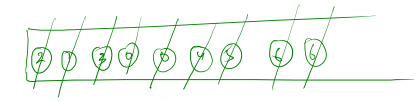
Graph - 1 : Introduction, Graph, Representations, DFS

Graph - 2 : DFS applications, Hamiltonian*

Graph - 3 : BFS, BFS applications

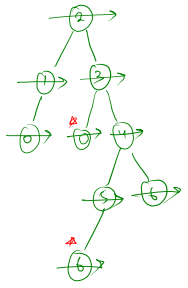
- ✓ Breadth First Traversal
- ✓ Is Graph Cyclic
- ✓ Is Graph Bipartite
- ✓ Spread Of Infection
- Shortest Path In Weights
- Minimum Wire Required To Connect All Pcs
- ✓ Order Of Compilation
- Iterative Depth First Traversal

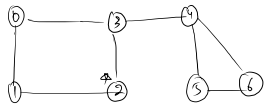
→ A.D.

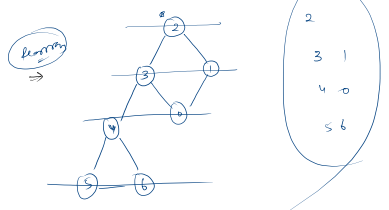


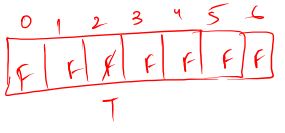
BFS introduction :- Breadth first Search

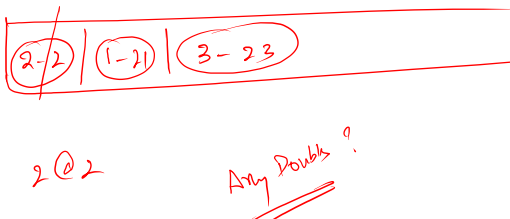
Level by level

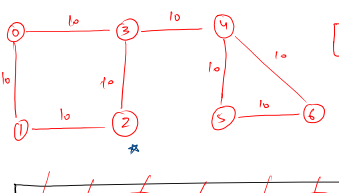


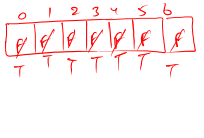


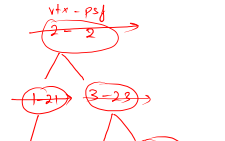


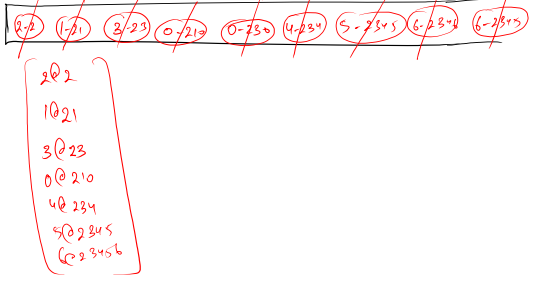


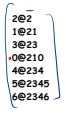








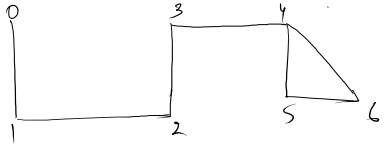


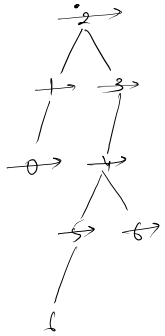


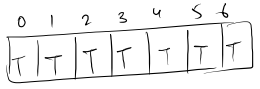
2@2

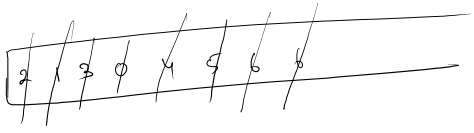
Any Doubts ?

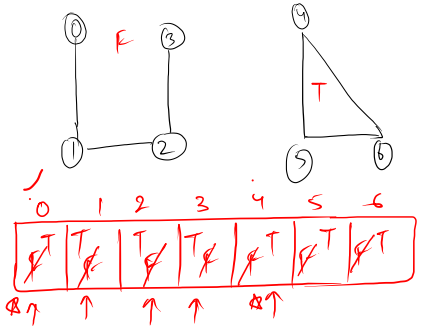
```
public static class BFSPair {
  int vtx;
  String psf;
  BFSPair(int vtx, String psf) {
    this.vtx = vtx;
    this.psf = psf;
  }
}
public static void BFS(ArrayList<Edge>[] graph, int vtx) {
  Queue<BFSPair> queue = new ArrayDeque<>();
  queue.add(new BFSPair(vtx, vtx + ""));
  boolean[] vis = new boolean[graph.length];

  while (queue.size() > 0) {
    BFSPair pair = queue.remove();

    if (!vis[pair.vtx]) {
      vis[pair.vtx] = true;

      System.out.println(pair.vtx + "@" + pair.psf);

      for (Edge e : graph[pair.vtx]) {
        if (!vis[e.nbr]) {
          queue.add(new BFSPair(e.nbr, pair.psf + e.nbr));
        }
      }
    }
  }
}
```

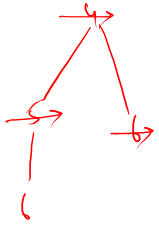




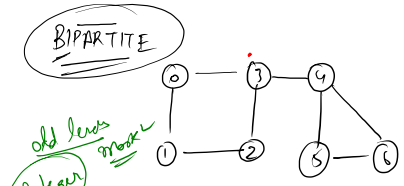




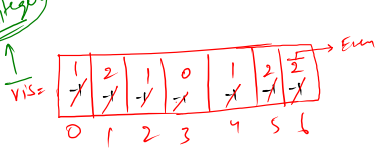


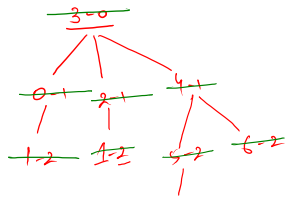



```java
public static boolean isCyclic(ArrayList<Edge>[] graph){
    boolean[] vis = new boolean[graph.length];

    for(int vtx = 0 ; vtx < graph.length ; vtx++){
        if(!vis[vtx]){
            boolean res = isCyclicHelper(graph, vtx,vis); //  comp
            if(res){
                return true;
            }
        }
    }
    return false;
}
public static boolean isCyclicHelper(ArrayList<Edge>[] graph,int vtx,boolean[] vis){
    Queue<Integer> queue = new ArrayDeque<>();
    queue.add(vtx);

    while(queue.size() > 0){
        int tvtx = queue.remove();
        if(vis[tvtx]){
            return true;
        }else{
            vis[tvtx] = true;
            for(Edge e : graph[tvtx]){
                if(vis[e.nbr] == false){
                    queue.add(e.nbr);
                }
            }
        }
    }

    return false;
}
```

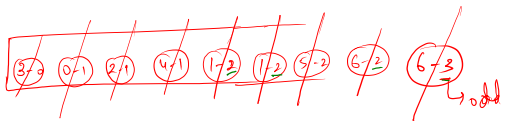


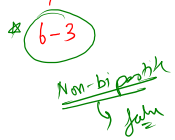
① Graph → divide → 2 Parts
② Intersection = ϕ
③ v_1 —— v_2 & $v_1 \in$ Part 1
then v_2 must ∈ Part 2

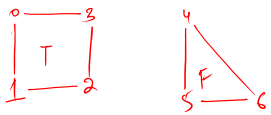


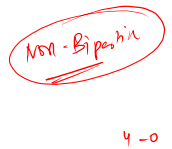


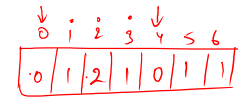
Even	odd
3	0
	2
1	4
5	
6	6

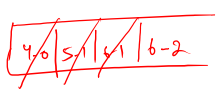


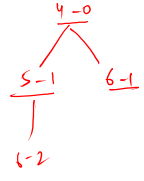


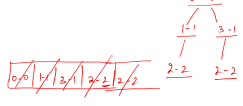













```java
public static class isBipartitePair{
    int vtx,level;
    isBipartitePair(int vtx,int level){
        this.vtx = vtx;
        this.level = level;
    }
}
public static boolean isBipartiteHelper(ArrayList<Edge>[] graph,int vtx,int[] visited){
    Queue<isBipartitePair> queue = new ArrayDeque<>();
    queue.add(new isBipartitePair(vtx, 0));

    while(queue.size() > 0){
        isBipartitePair pair = queue.remove();

        if(visited[pair.vtx] == -1){
            visited[pair.vtx] = pair.level;

            for(Edge e : graph[pair.vtx]){
                if(visited[e.nbr] == -1){
                    queue.add(new isBipartitePair(e.nbr, pair.level+1));
                }
            }
        }else{
            if(pair.level != visited[pair.vtx]){
                return false;
            }
        }
    }
    return true;
}
public static boolean isBipartite(ArrayList<Edge>[] graph){
    int visited[] = new int[graph.length];
    Arrays.fill(visited, -1);
    for(int vtx = 0 ; vtx < graph.length ; vtx++){
        if(visited[vtx] == -1){
            boolean res = isBipartiteHelper(graph,vtx,visited);
            if(!res){
                return false;
            }
        }
    }
    return true;
}
```

7 → vertices
8 → Edges

0 1 10
1 2 10
2 3 10
0 3 10
3 4 10
4 5 10
5 6 10
4 6 10

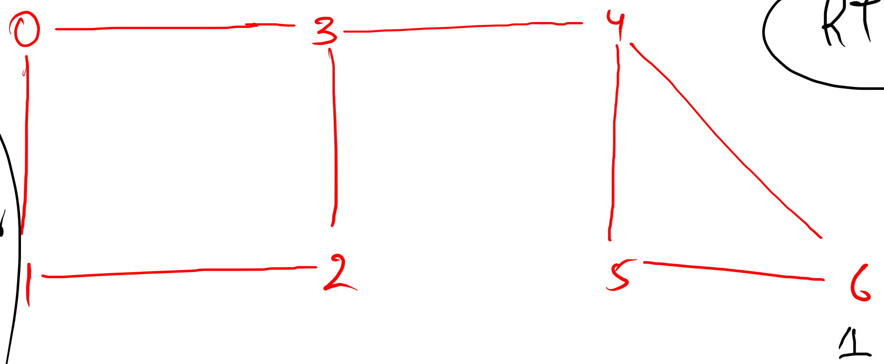
6
3

3 BFS

H.W.

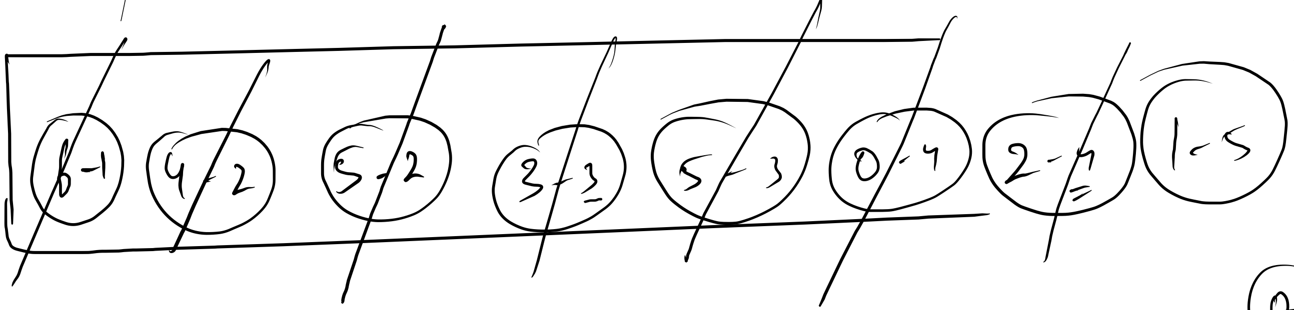
Pair
→ Pid
→ TOI

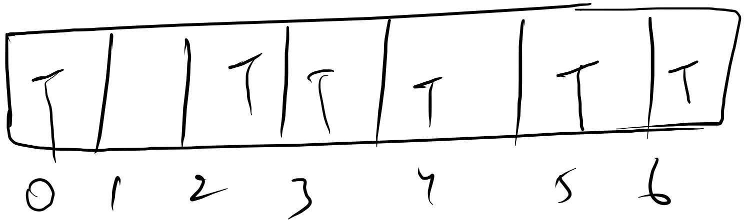
process odd nbr



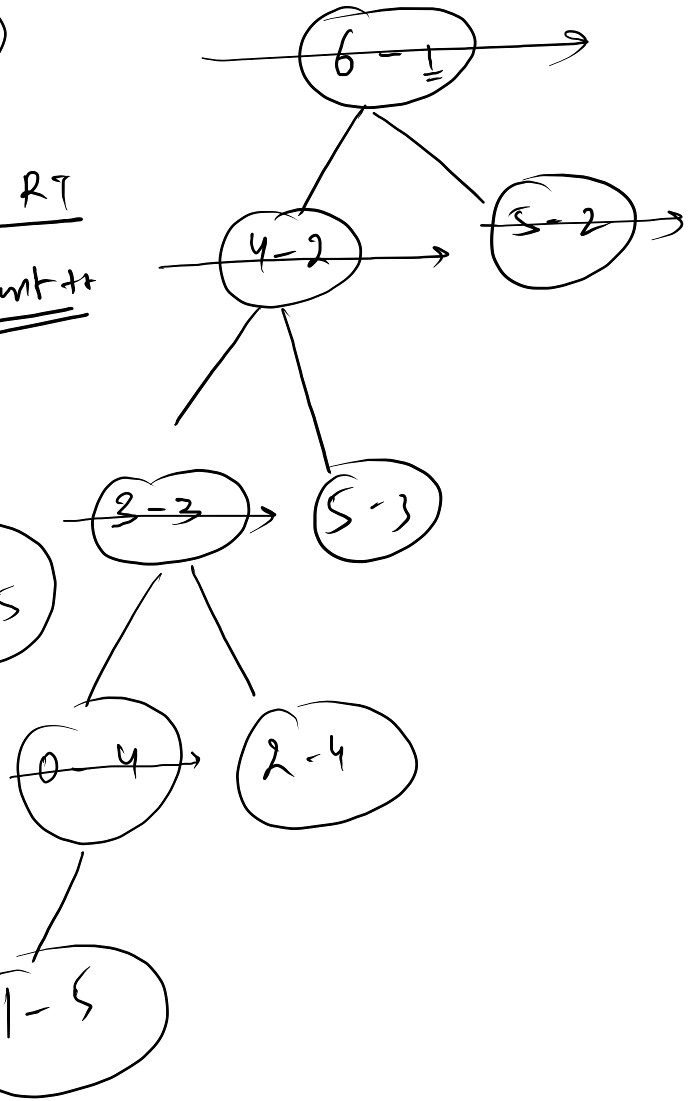
RT = 3 sec

TOI <= RT
→ Count++





Count = 0 1 2 3 4



Introduction To Graphs And Its Representation											
Has Path?	Easy	10	Auth	0	Public	Sol	0				
Print All Paths	Easy	10	Auth	0	Public	Sol	1				
Multisolver - Smallest, Longest, Ceil, Floor, Kthlargest Path	Easy	10	Auth	0	Public	Sol	2				
Get Connected Components Of A Graph	Easy	10	Auth	0	Public	Sol	3				
Is Graph Connected	Easy	10	Auth	0	Public	Sol	4				
Number Of Islands	Easy	10	Auth	0	Public	Sol	5				
Perfect Friends	Easy	10	Auth	0	Public	Sol	6				
Hamiltonian Path And Cycle	Easy	10	Auth	0	Public	Sol	7				
Knights Tour	Easy	10	Auth	0	Public	Sol	8				
Breadth First Traversal	Easy	10	Auth	0	Public	Sol	9				
Is Graph Cyclic	Easy	10	Auth	0	Public	Sol	10				
Is Graph Bipartite	Easy	10	Auth	0	Public	Sol	11				
Spread Of Infection	Easy	10	Auth	0	Public	Sol	12				
Shortest Path In Weights	Easy	10	Auth	0	Public	Sol	13				
Minimum Wire Required To Connect All Pcs	Easy	10	Auth	0	Public	Sol	14				
Order Of Compilation	Easy	10	Auth	0	Public	Sol	15				
Iterative Depth First Traversal	Easy	10	Auth	0	Public	Sol	16				
		10	Auth	0	Public	Sol	17