7

5

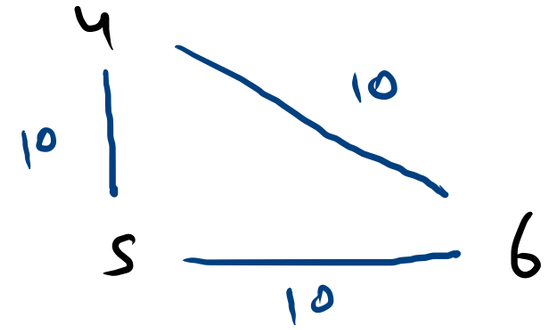
0 1 10

2 3 10

4 5 10

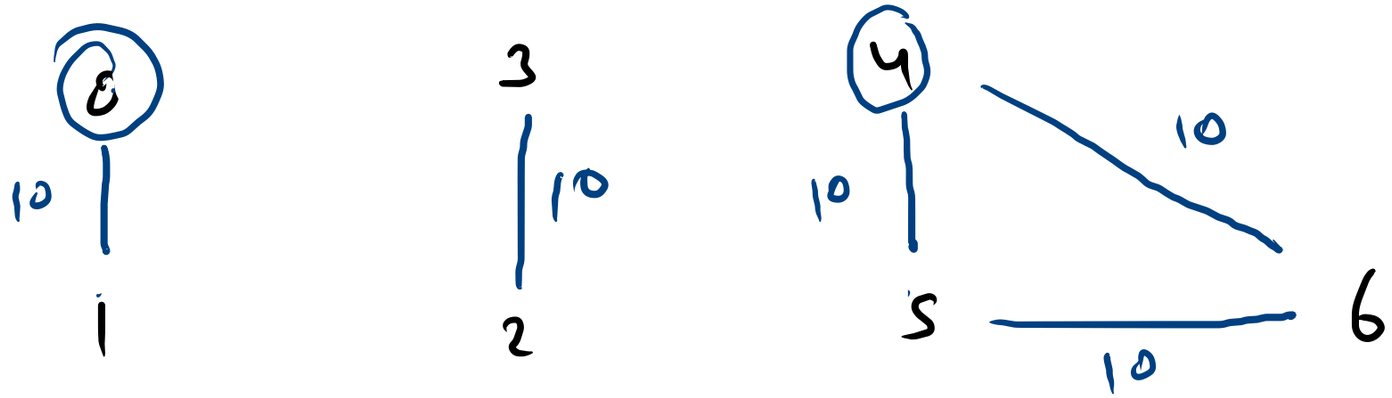
5 6 10

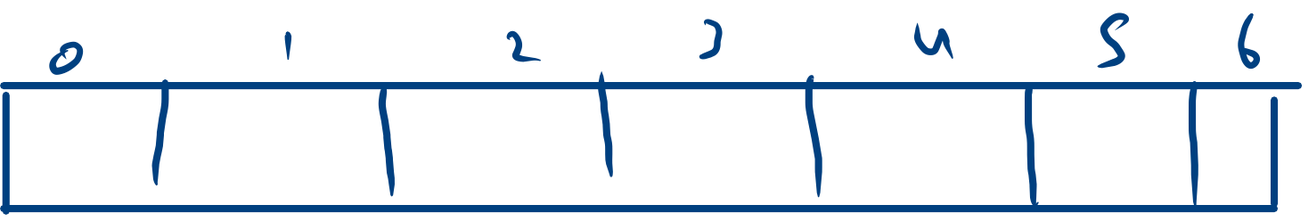
4 6 10

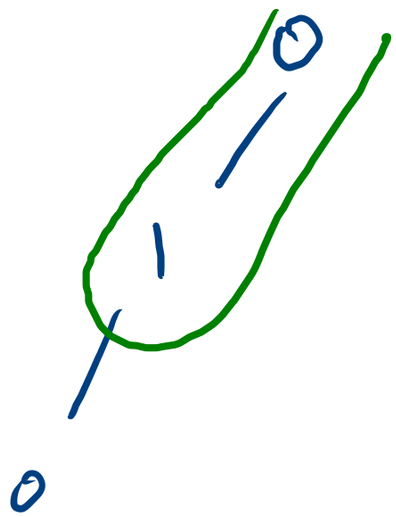


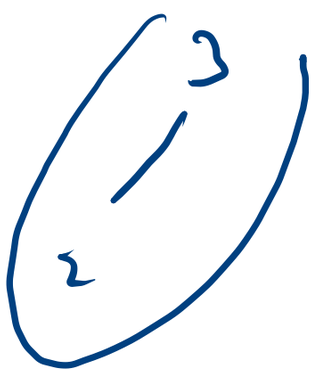
AL

[AL<Integer> [0, 1] , AL [3, 2] , AL [4, 5, 6]]

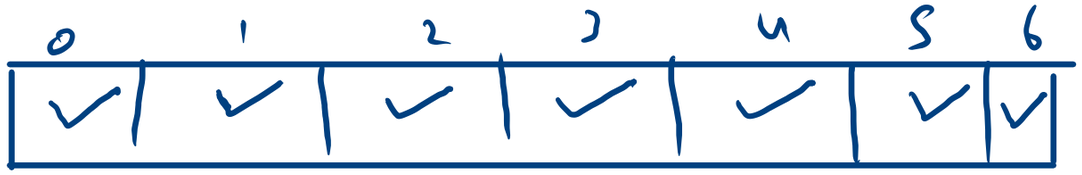

0
10
1
3
10
2
4
10
10
5
6
10


0
1
2
3
4
5
6

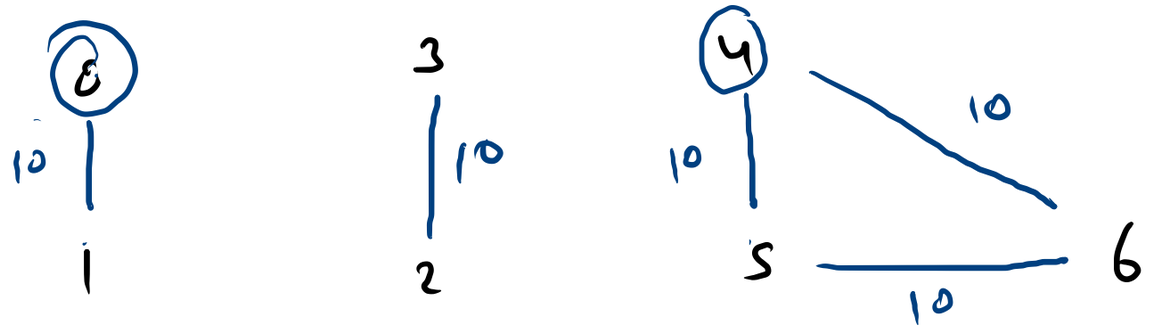

0
1
0

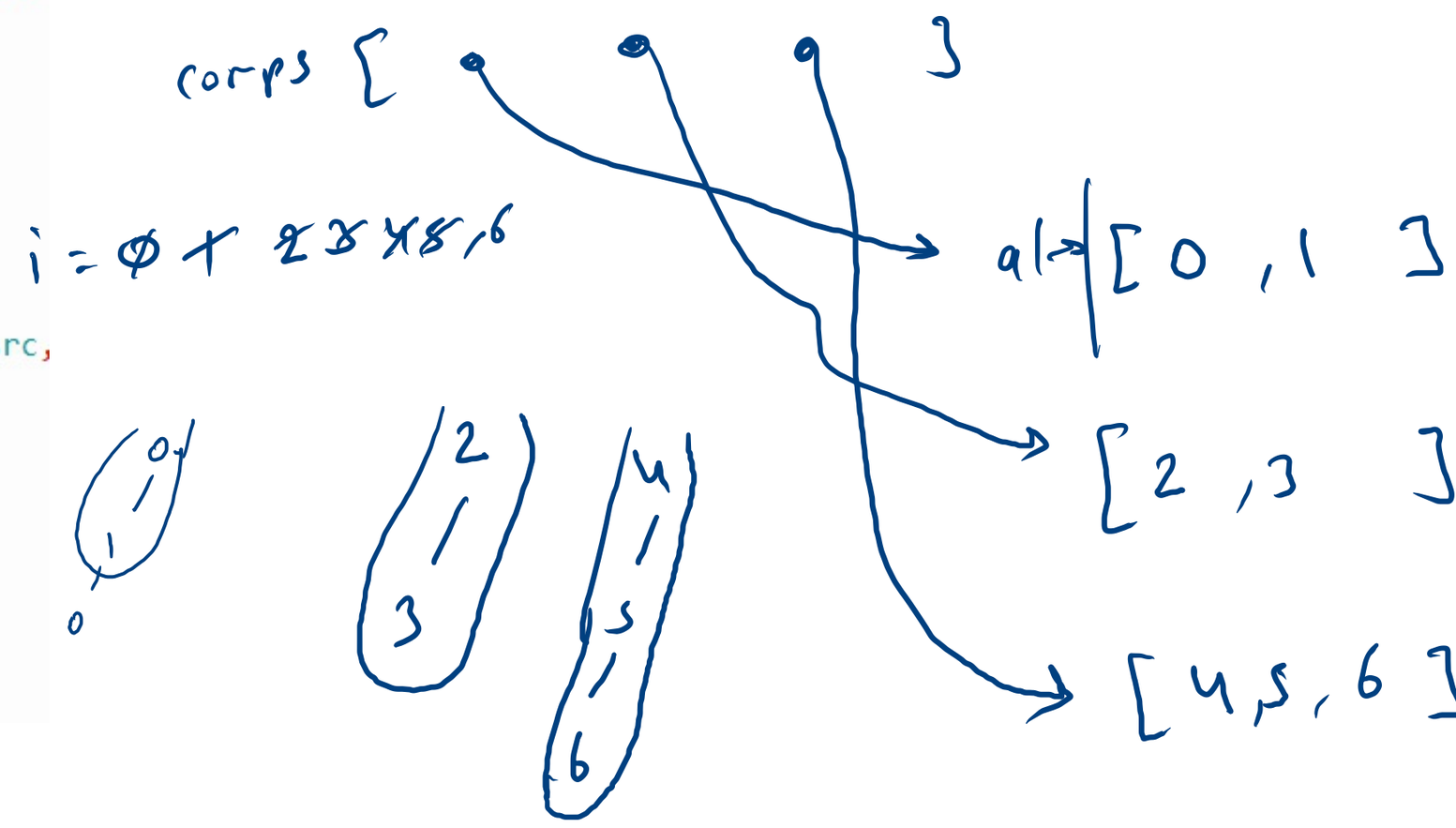

3
2


4
5
6
1
4



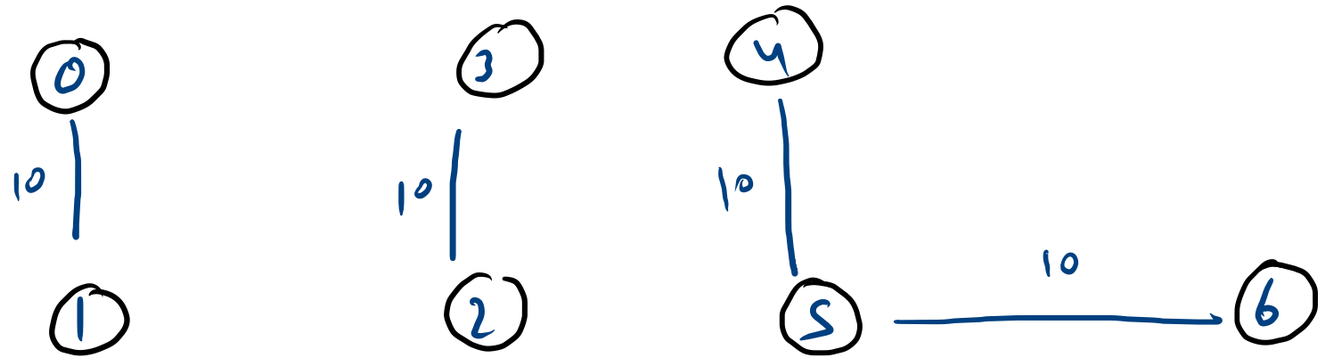

```java
boolean visited[] = new boolean[vtces];
for(int i=0;i<vtces;i++){
    if(visited[i] == false){
        ArrayList<Integer> al = new ArrayList<>();
        dfs(graph, i, al, visited);
        comps.add(al);
    }
}

System.out.println(comps);


ublic static void dfs(ArrayList<Edge>[] graph, int src,
    visited[src] = true;
    al.add(src);
    for(Edge edge: graph[src]){
        if(visited[edge.nbr] == false){
            dfs(graph, edge.nbr, al, visited);
        }
    }
}
```



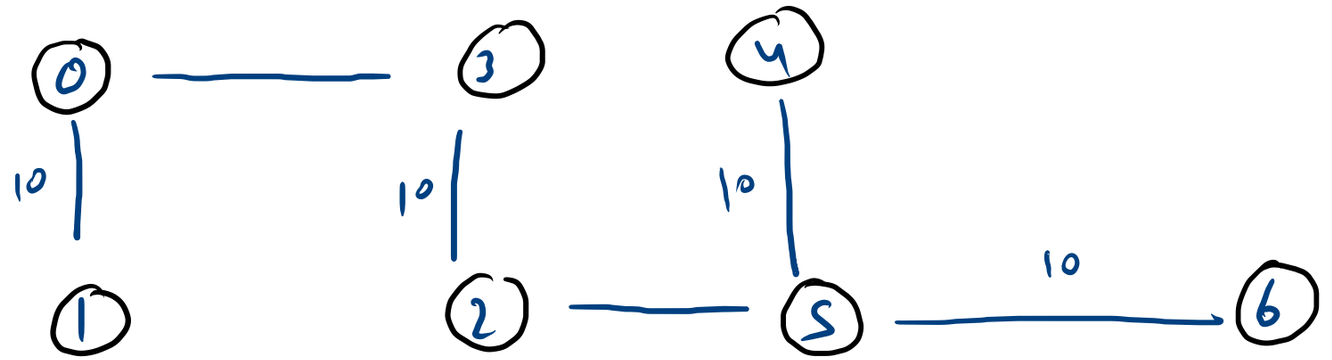


Is Graph Connected

11.V

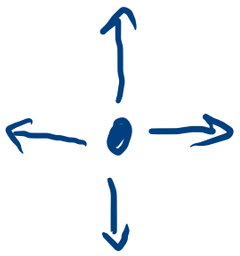


false

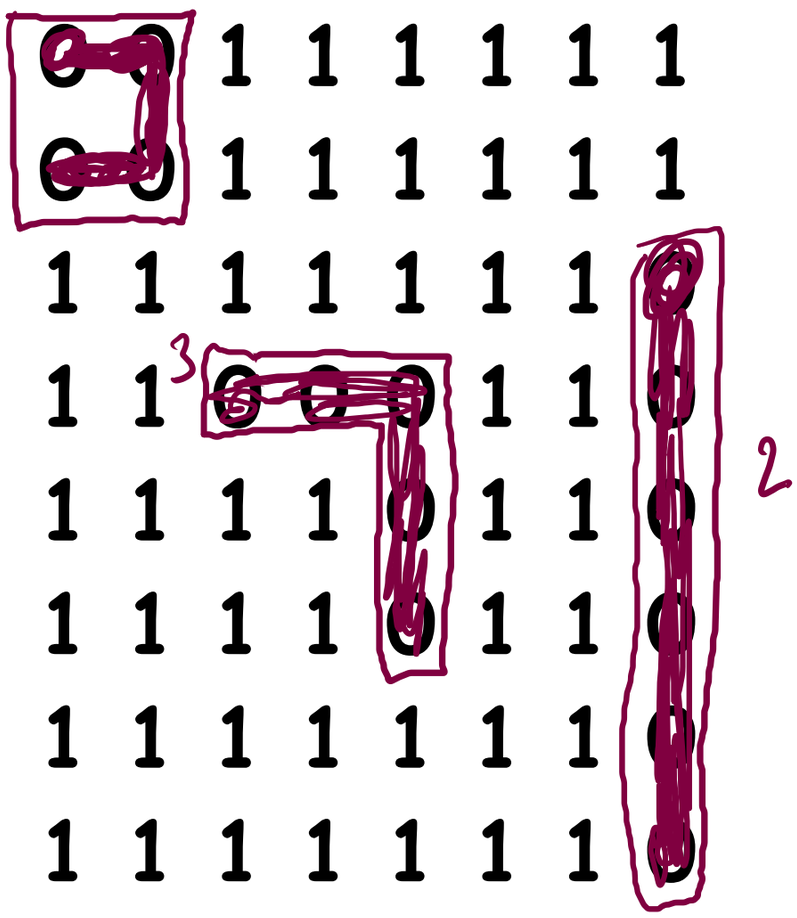


true

8x8



1

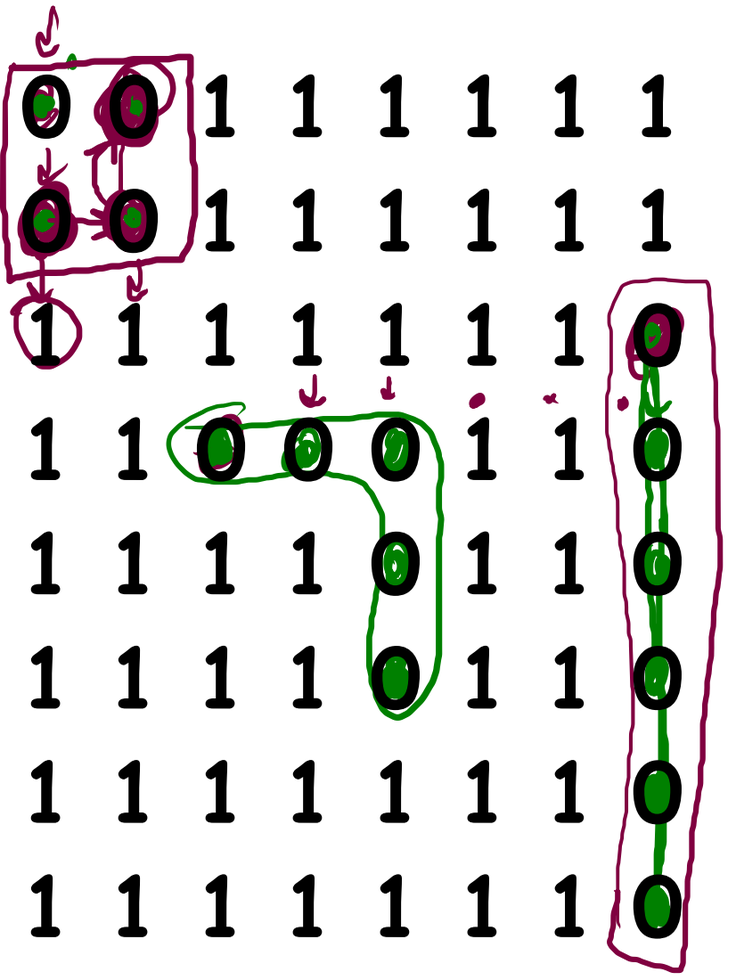


0 → land

1 → water

count Islands

count 0 1 2 3



↓ ↑ → ←

```
int count=0;
boolean visited[][] = new boolean[m][n];
for(int i=0;i<m;i++){
    for(int j=0;j<n;j++){
        if(arr[i][j] == 0 && visited[i][j] == fal
            markCC(arr, i, j, visited);
            count++;
        }
    }
}
System.out.println(count);
```

```
ublic static void markCC(int arr[][], int i, int j,
    if(i<0 || j<0 || i>=arr.length || j>=arr[0].leng

    if(arr[i][j] == 1||visited[i][j] == true)return;

    visited[i][j] = true;
    markCC(arr, i+1,j, visited);
    markCC(arr, i-1,j, visited);
    markCC(arr, i,j+1, visited);
    markCC(arr, i,j-1, visited);
```

no: frien

7 [0 ... 6]

5

0 1

2 3

4 5

5 6

C1 C2 C3

0 3 4

1 2 5 — 6

→ [2, 2, 3]

1st , 2nd

C1 != C2

C2

C1 C3

C2 C3

— , —

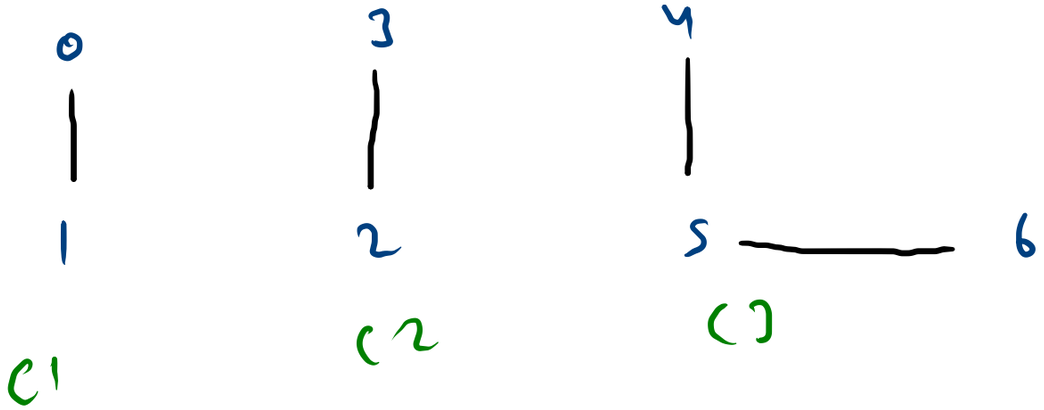
0 1 d

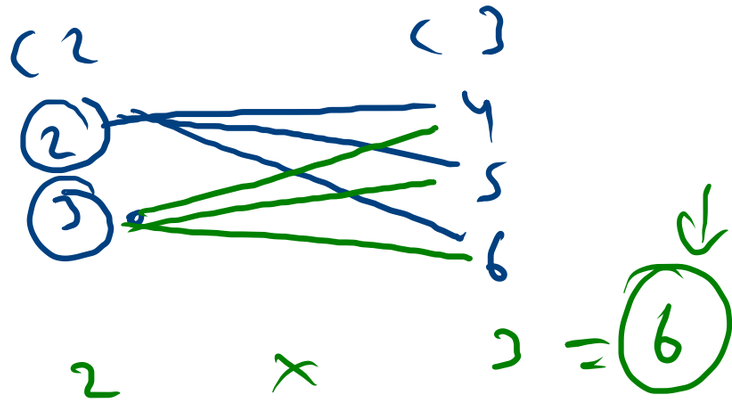
0 2,3 4,5,6

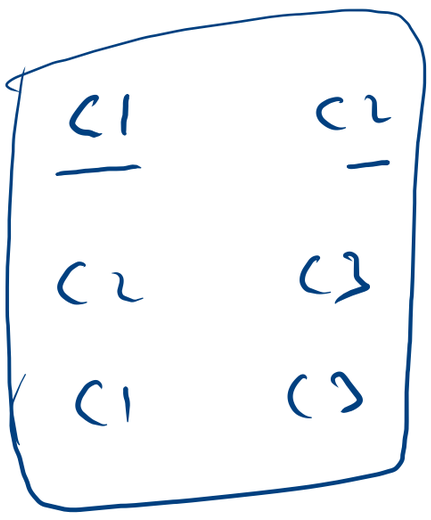
1 2,3,4,5,6

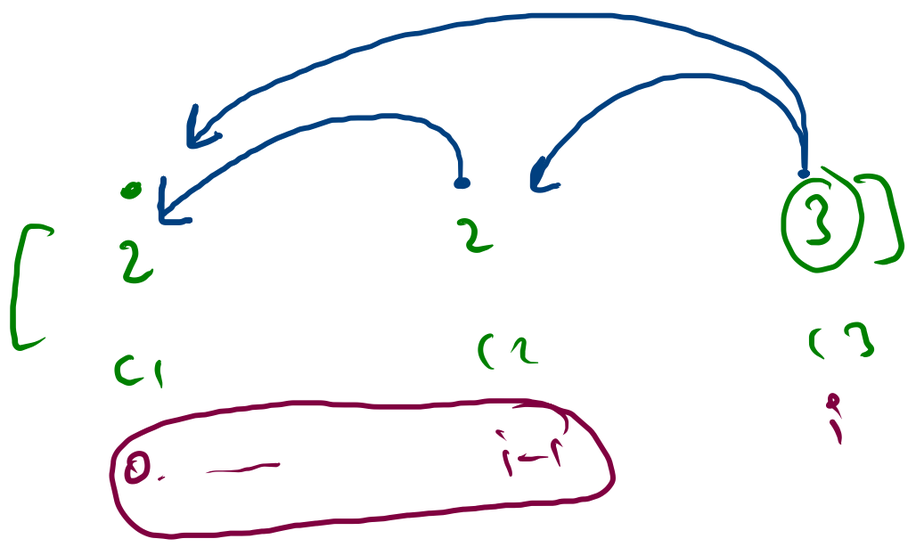
4 5,6 d

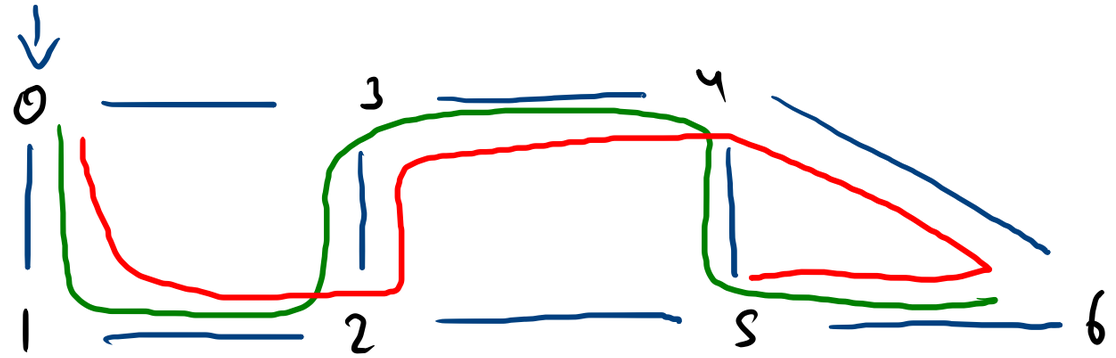
0	2	1	2	2	4
0	3	1	3	2	5
0	4	1	4	2	6
0	5	1	5	3	4
0	6	1	6	3	5
				3	6


0
3
4
1
2
5
6


2
3
4
5
6
2 x 3 = 6


c1
c2
c2
c3
c1
c3

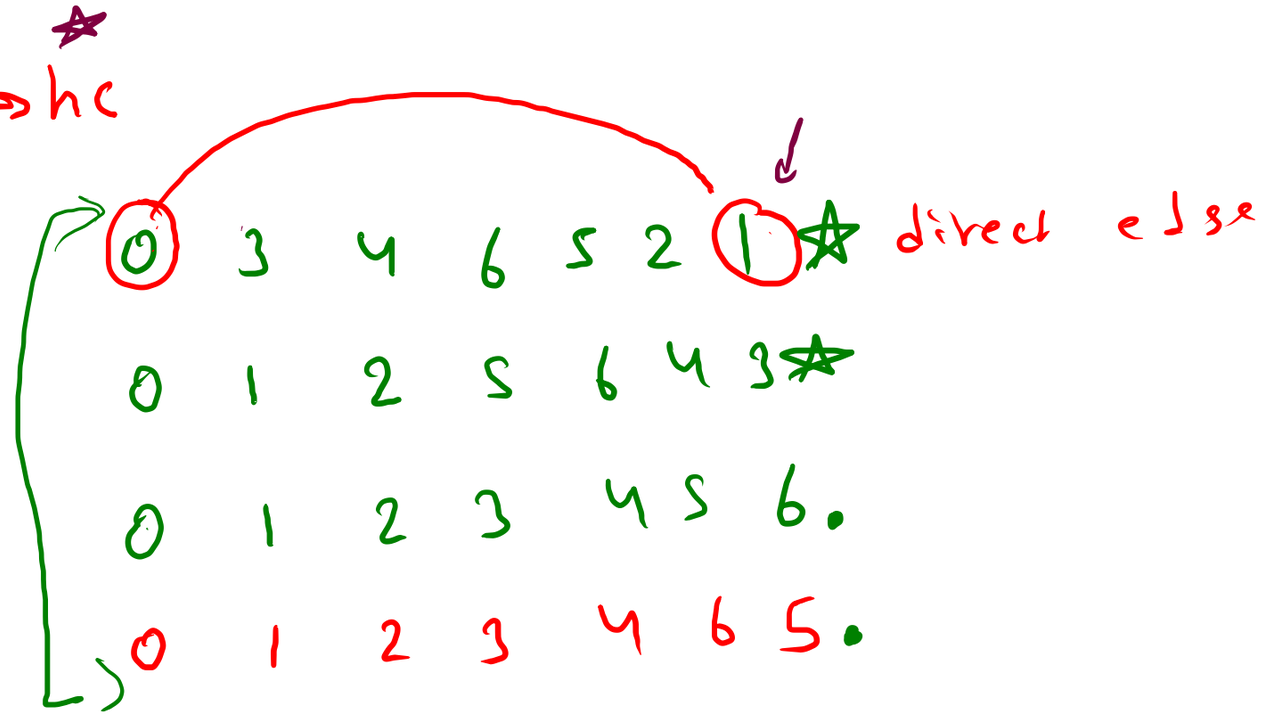

2
2
3
c1
c2
c3
0
i-1
i



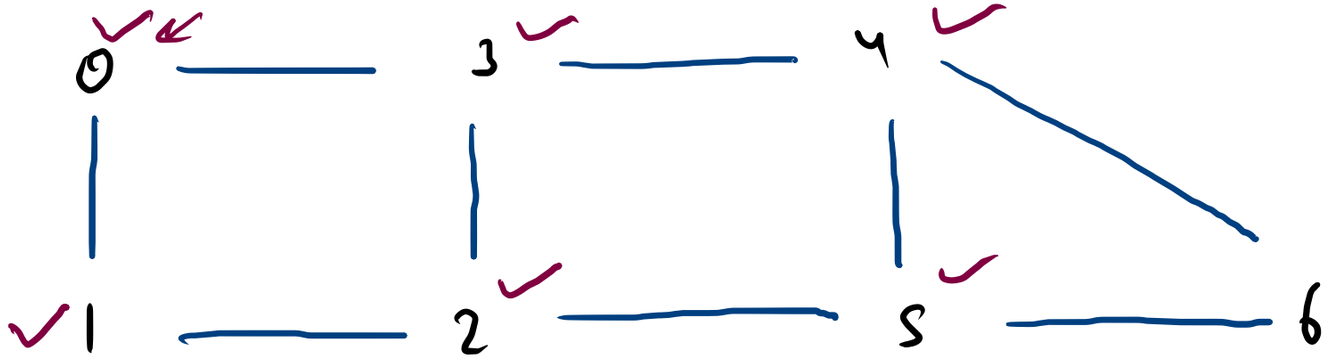
graph src visited path, (cst) , osrc

src → 0

hp → hc



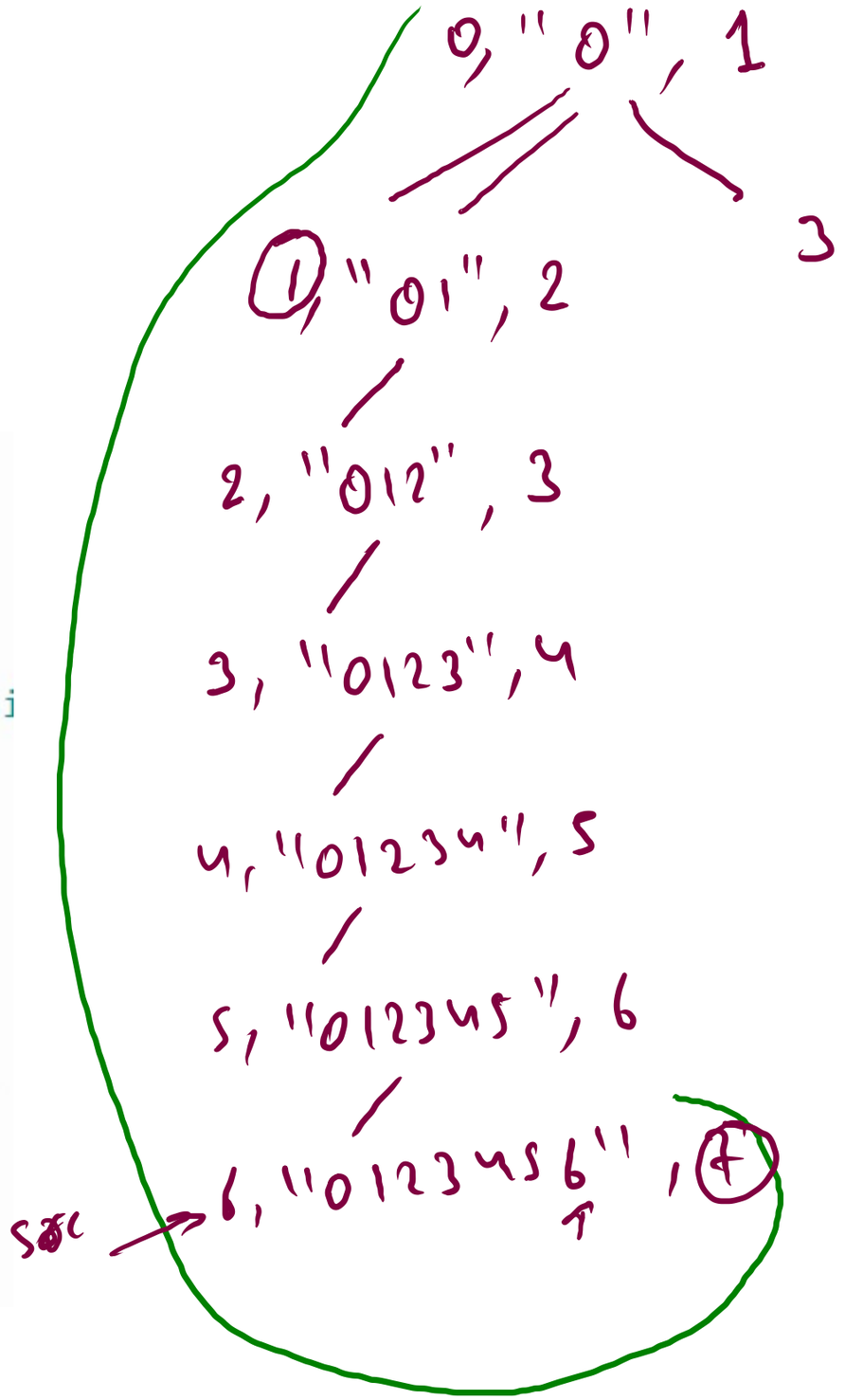
src → dest



src = 0

```
    boolean visited[] = new boolean[vtces];
    hasPath(graph, src, visited, ""+src, 1);

}

public static void hasPath(ArrayList<Edge> graph[], int src, boolean[]visi
    if( ??? ){
        System.out.println(path);
        return;
    }
    visited[src] = true;
    for(Edge edge: graph[src]){
        if(visited[edge.nbr] == false){
            hasPath(graph, edge.nbr, des, visited, path+edge.nbr, csf+1);
        }
    }
    visited[src] = false;
}
```

graph.len = 7 ?



7

8

0 1 10

1 2 10

2 3 10

0 3 10

3 4 10

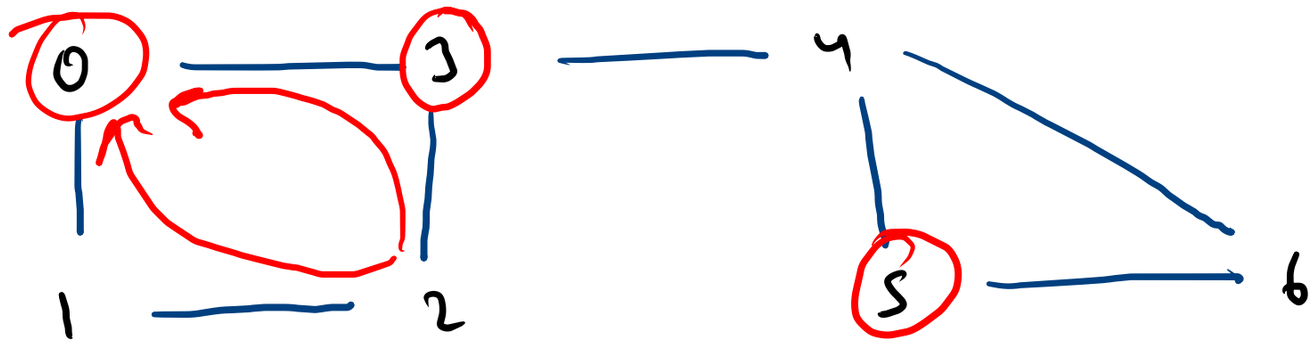
4 5 10

5 6 10

4 6 10

2

src = 2



3 @ 2 3

0 @ 2 10

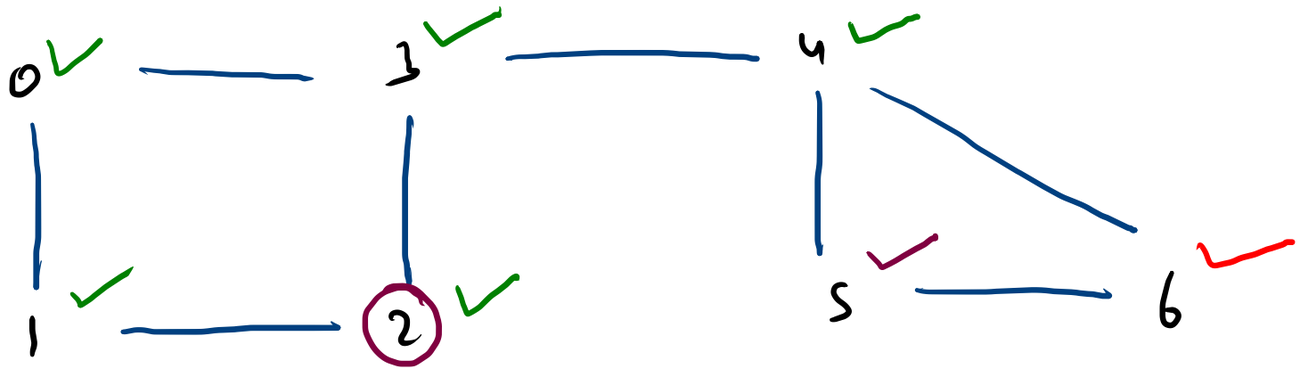
1 @ 2 1

4 @ 2 3 4

5 @ 2 3 4 5

6 @ 2 3 4 6

2 @ 2



y visit more child

already visited

2 @ 2

1 @ 21

0 @ 210

4 @ 234

5 @ 235

6 @ 2346