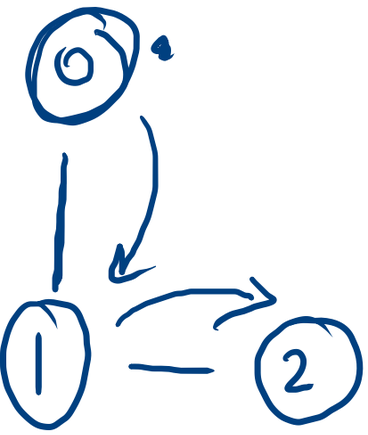
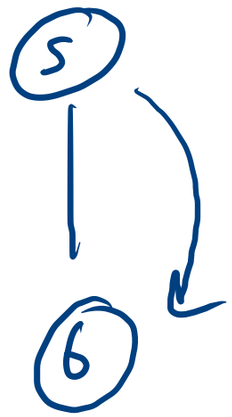
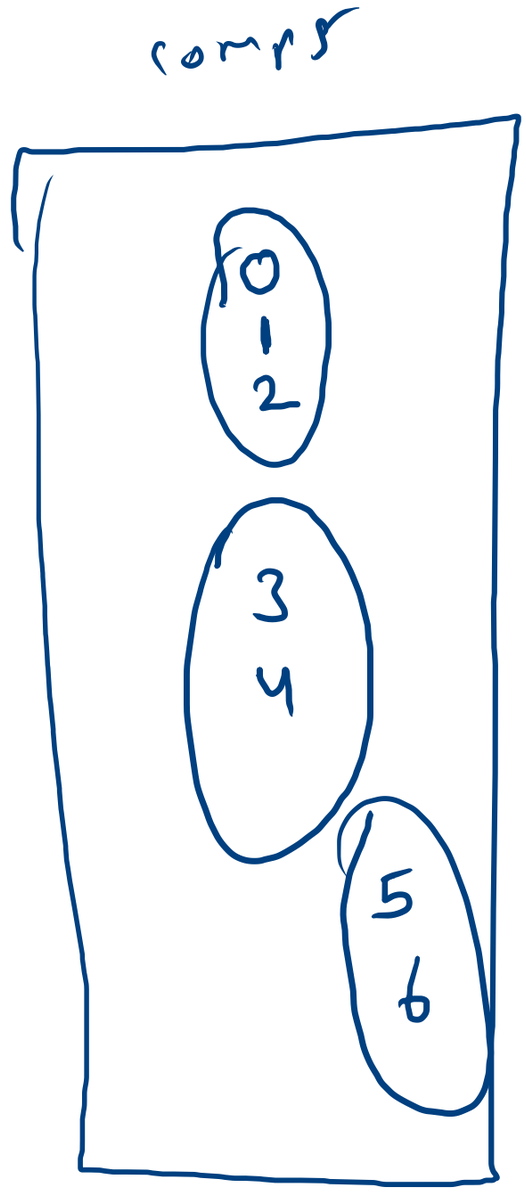


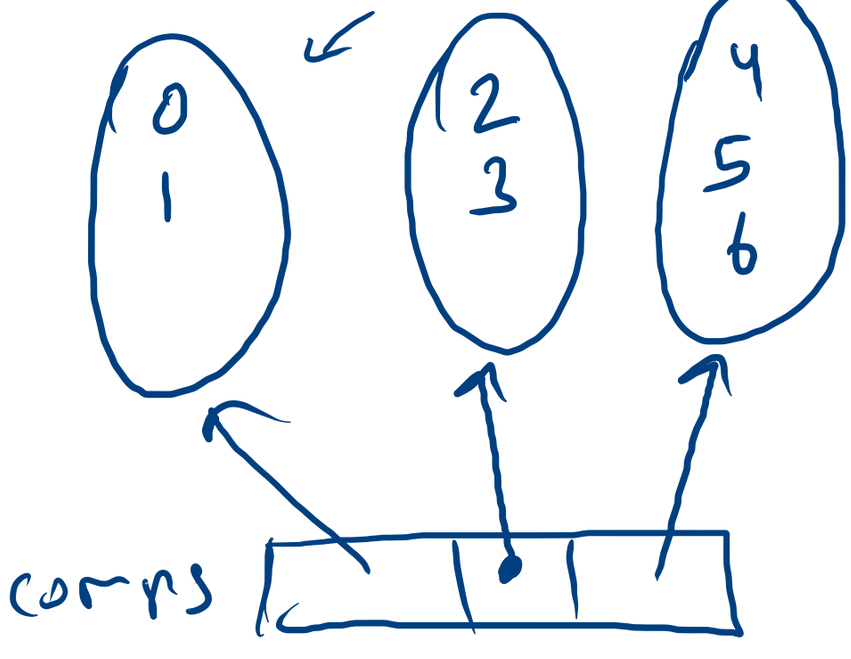


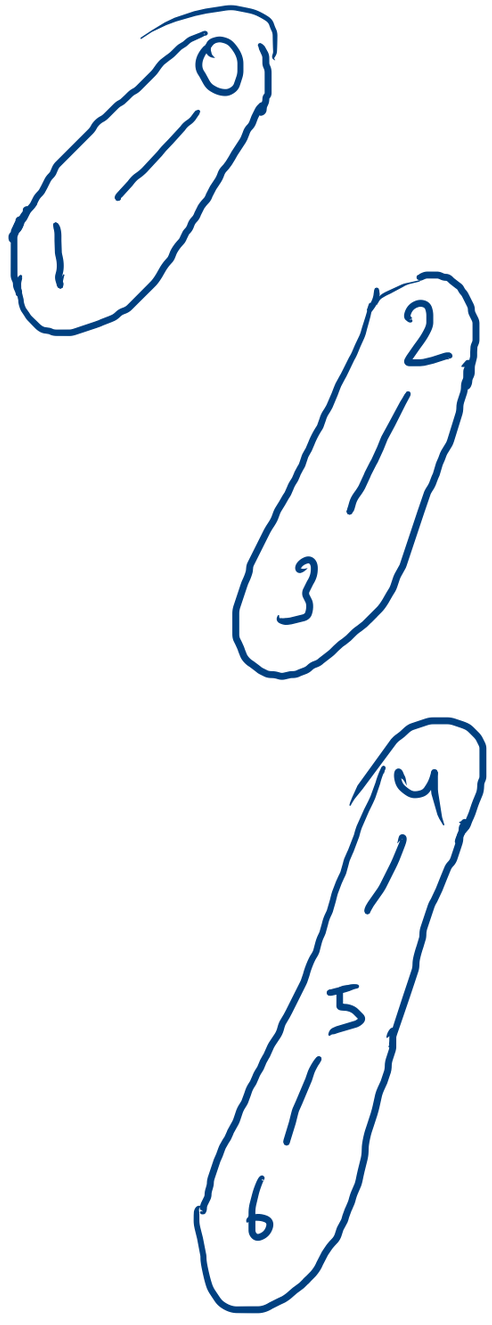


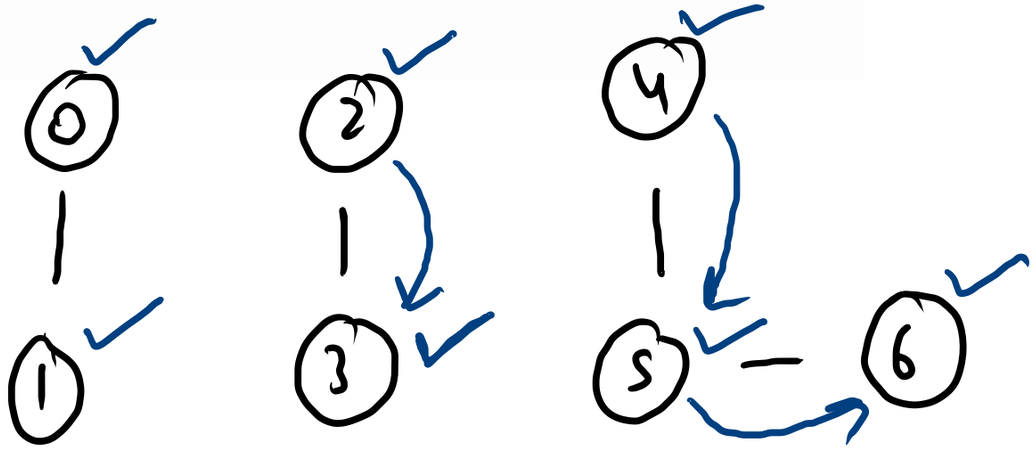
comp

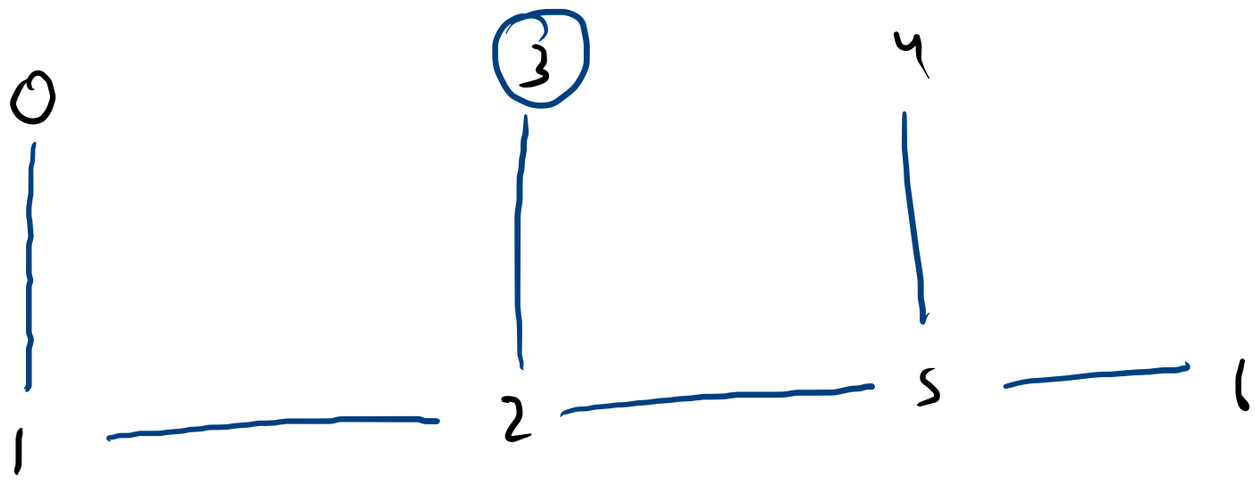
```
boolean visited[] = new boolean[vtces];

    for(int i=0;i<vtces;i++){
        if(visited[i] == false){
            ArrayList<Integer> comp = new Arrayl
            gcc(graph, i, visited, comp);
            comps.add(comp);
        }
    }

    System.out.println(comps);
}

public static void gcc( ArrayList<Edge>[] graph,

    visited[src] = true;
    comp.add(src);
    for(Edge e: graph[src]){
        if(visited[e.nbr] == false){
            gcc(graph, e.nbr, visited, comp);
        }
    }
}
```

0 1 2 3 4

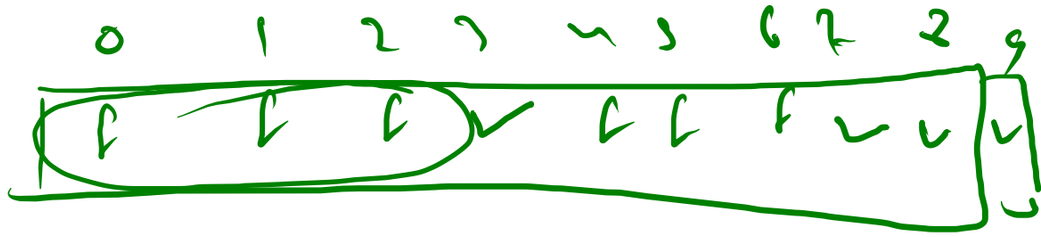


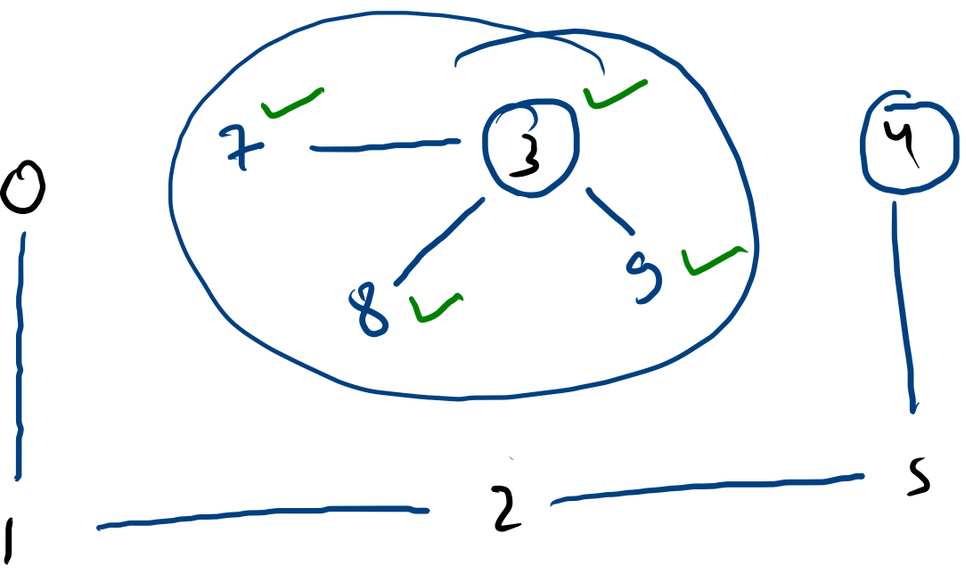




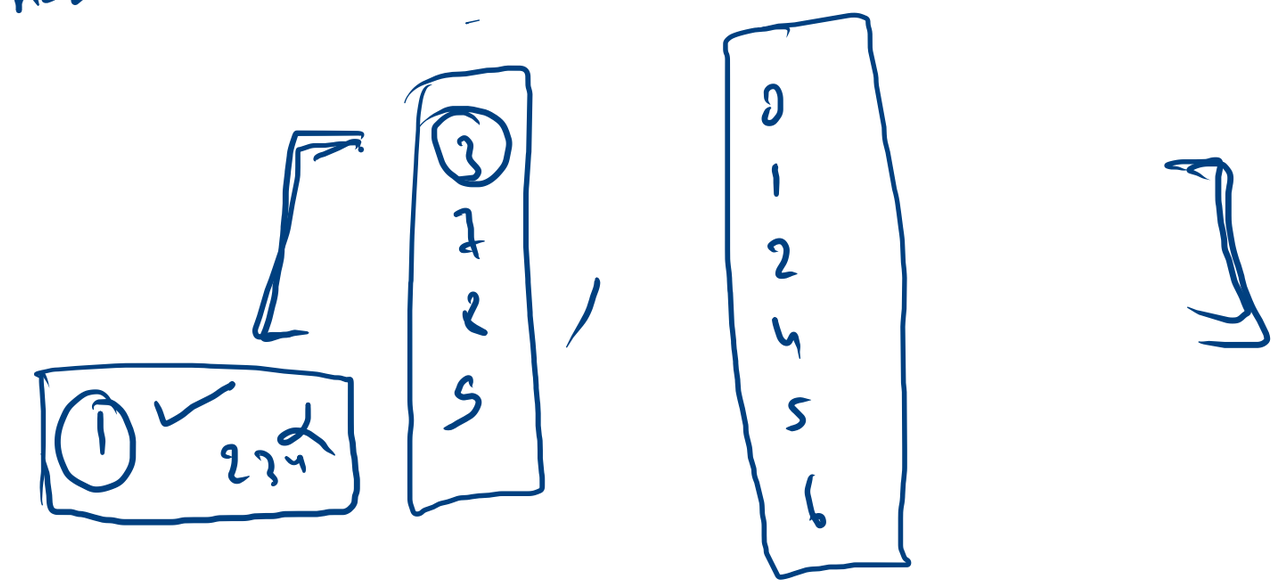


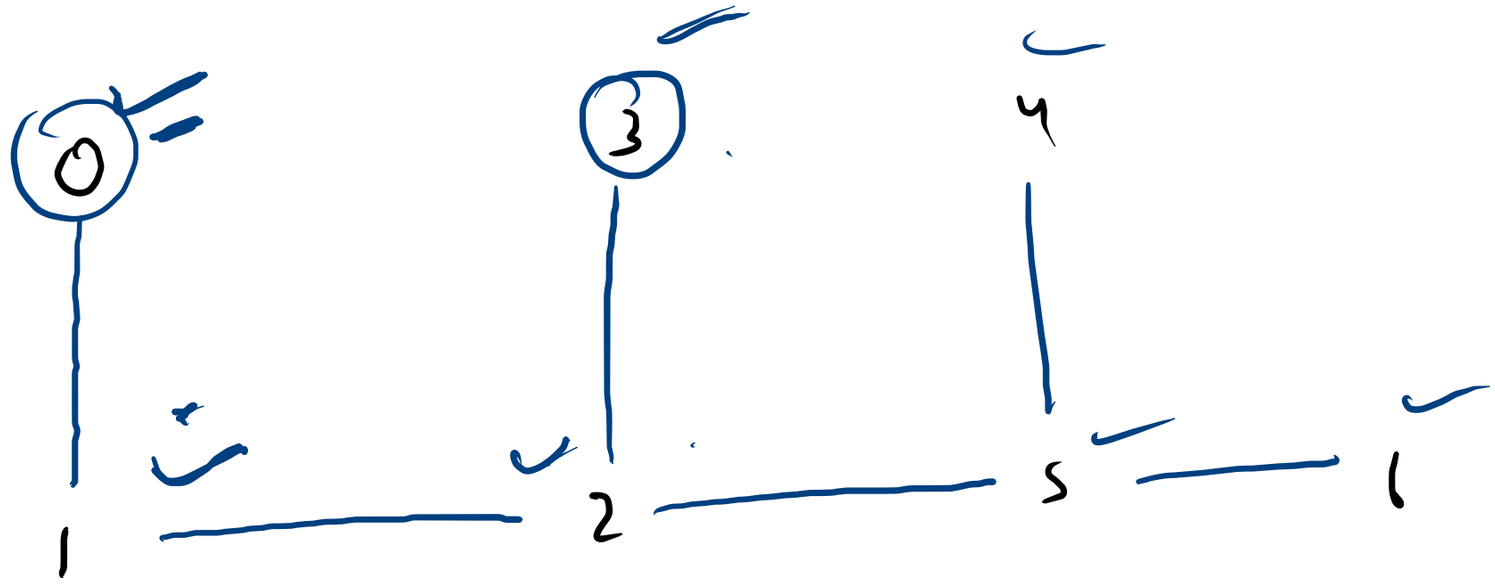
connected

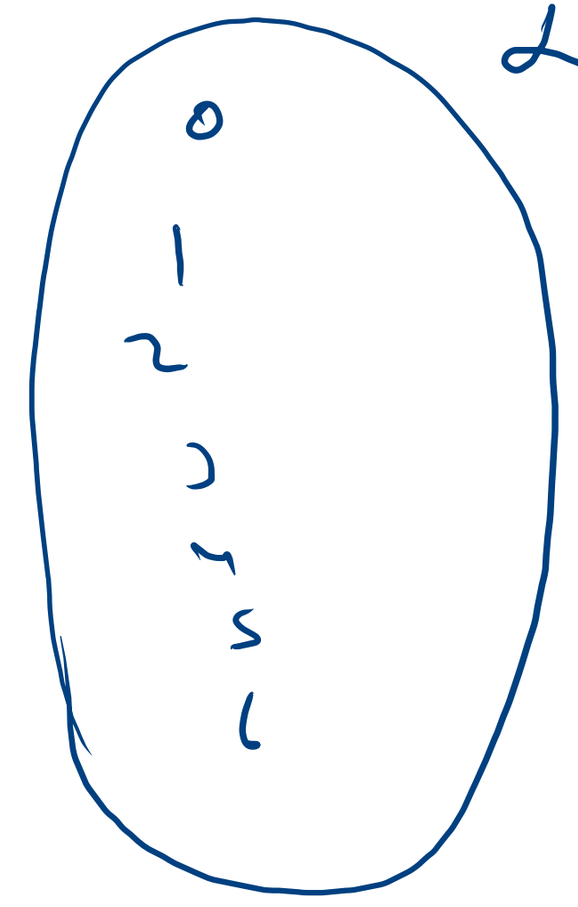




not connected





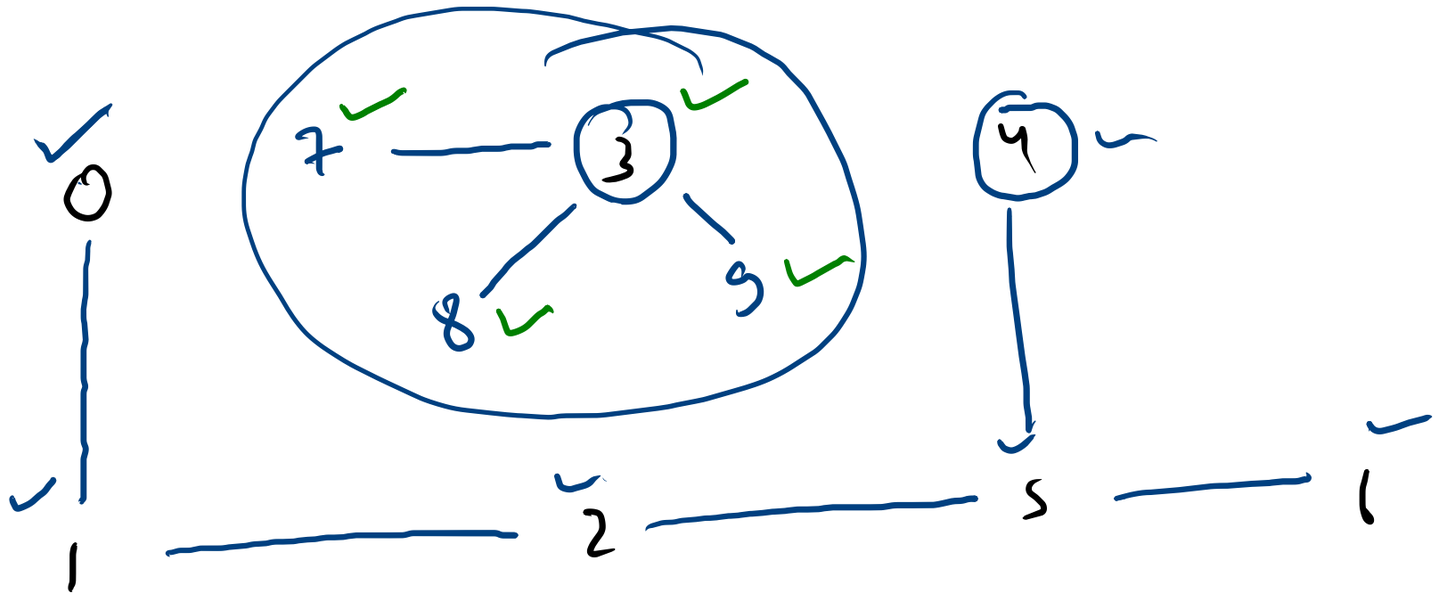


1

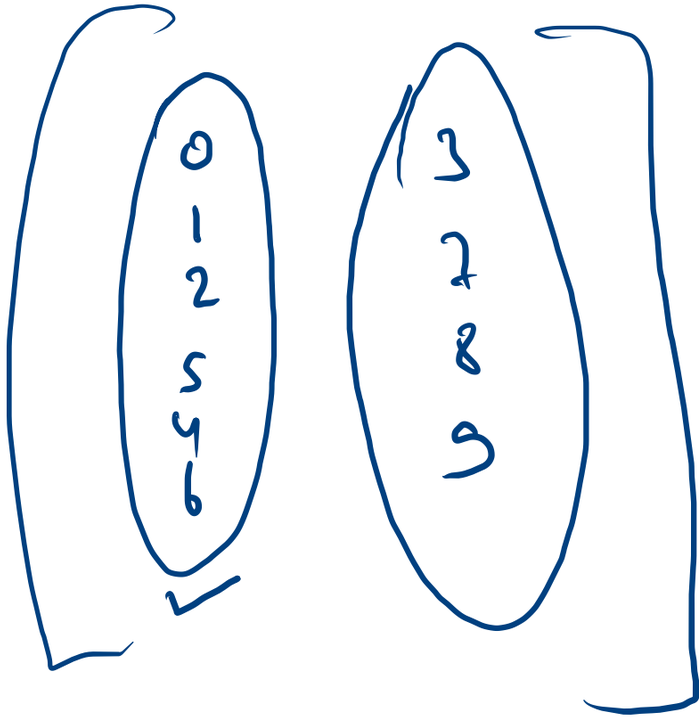
0 (✓)

0 (✓)

recursion



O(V)



O(V)

0, 1, 2, 5, 4, 6

	0	1	
0	0	0	1
	0	0	1
	1	1	1

$\{0,0\}, \{0,1\}$

land

row

col

8 8

0	0	1	1	1	1	1	1
0	0	1	1	1	1	1	1
1	1	1	1	1	1	1	0
1	1	0	0	0	1	1	0
1	1	1	1	0	1	1	0
1	1	1	1	0	1	1	0
1	1	1	1	1	1	1	0
1	1	1	1	1	1	1	0

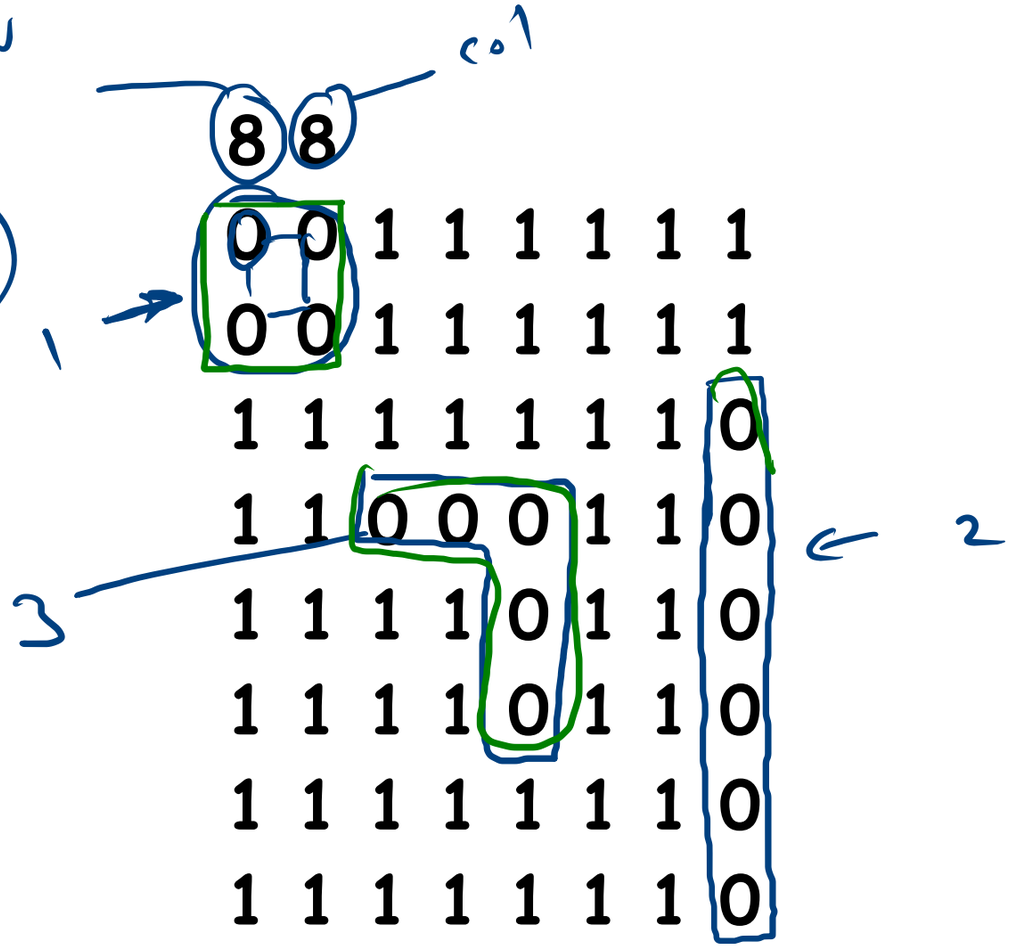
1

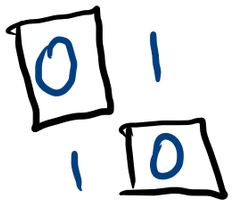
2

3

0 — land

1 → water





$\boxed{0}$ 1

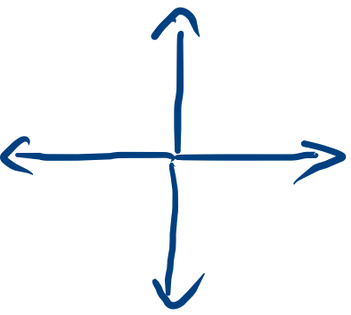
1 $\boxed{0}$

2

$\boxed{3}$

$\boxed{2}$

1 and 2



8 8

0 0 1 1 1 1 1 1

0 0 1 1 1 1 1 1

1 1 1 1 1 1 1 0

1 1 0 0 0 1 1 0

1 1 1 1 0 1 1 0

1 1 1 1 0 1 1 0

1 1 1 1 1 1 1 0

1 1 1 1 1 1 1 0

```java
int count = 0;
for(int i=0;i<m;i++){
    for(int j=0;j<n;j++){
        if(arr[i][j] == 0 && !visited[i][j]){
            gcc(i, j, arr, visited);
            count++;
        }
    }
}
```

count = 0 to 3

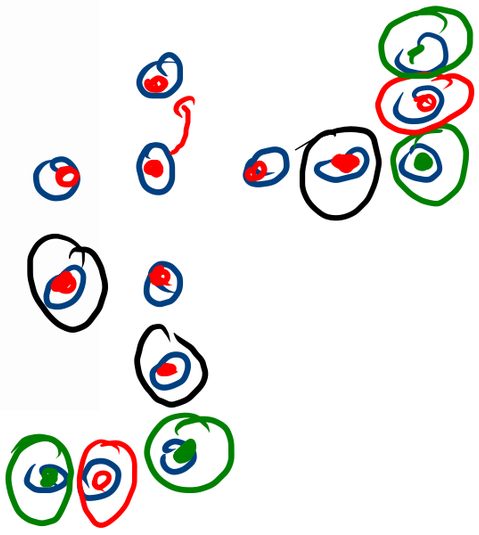
hxr

O(hxm)

```java
public static void gcc(int i, int j, int
    if(i<0 || j< 0||i>=arr.length ||
    j>= arr[0].length || arr[i][j] == 1
    || visited[i][j])return;

    visited[i][j] = true;

    gcc(i, j+1, arr, visited);
    gcc(i, j-1, arr, visited);
    gcc(i+1, j, arr, visited);
    gcc(i-1, j, arr, visited);
```

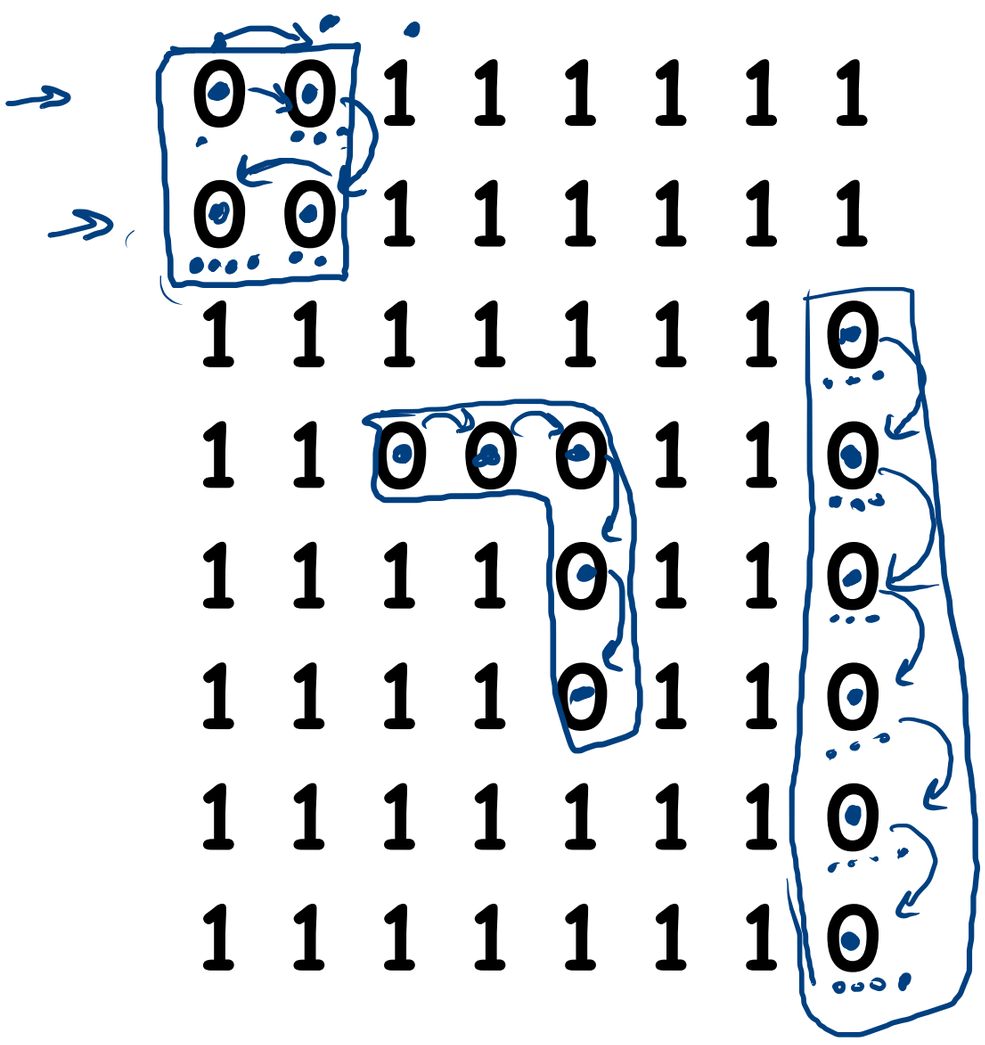
O(nxm)

BFS



dfs

hxm



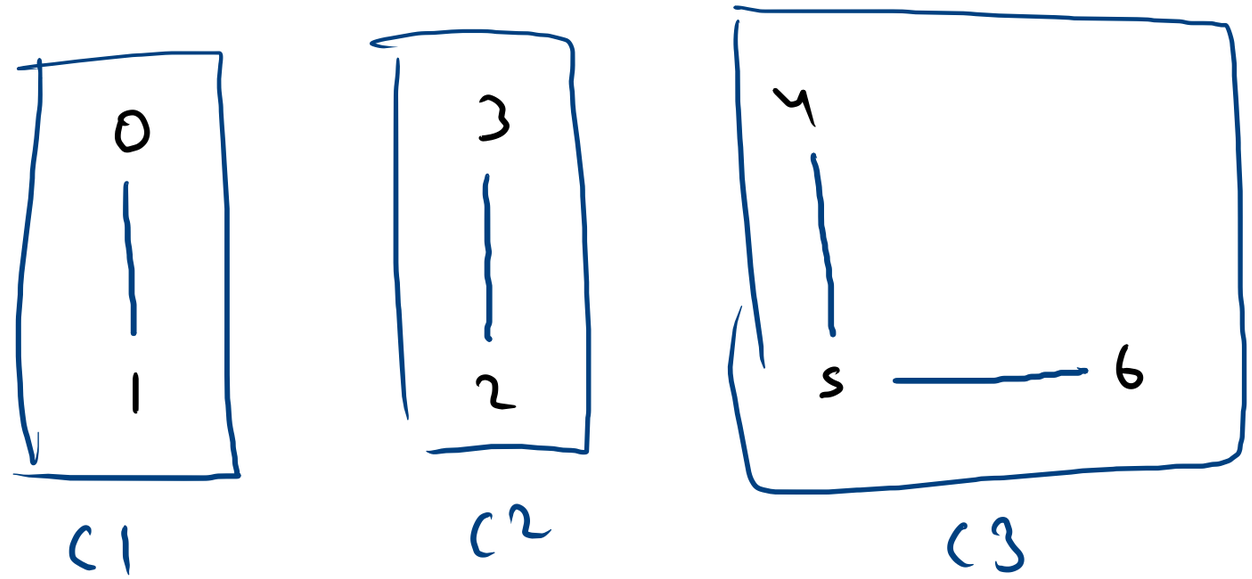
0 ... 6

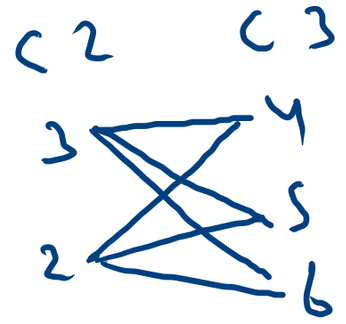
7

5

0 1
2 3
4 5
5 6
4 6

graph





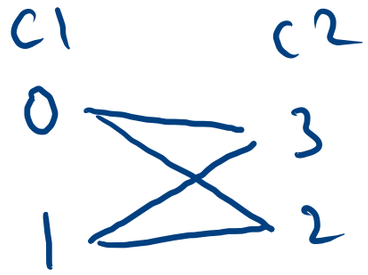
3 + 3

? ?

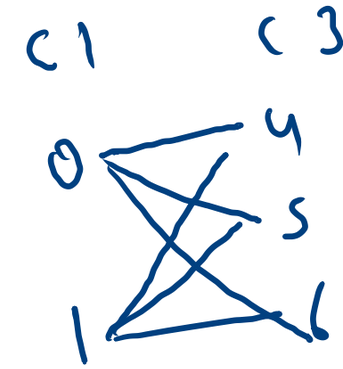
10 C1 C2, C3 *

6 C2 C3 *

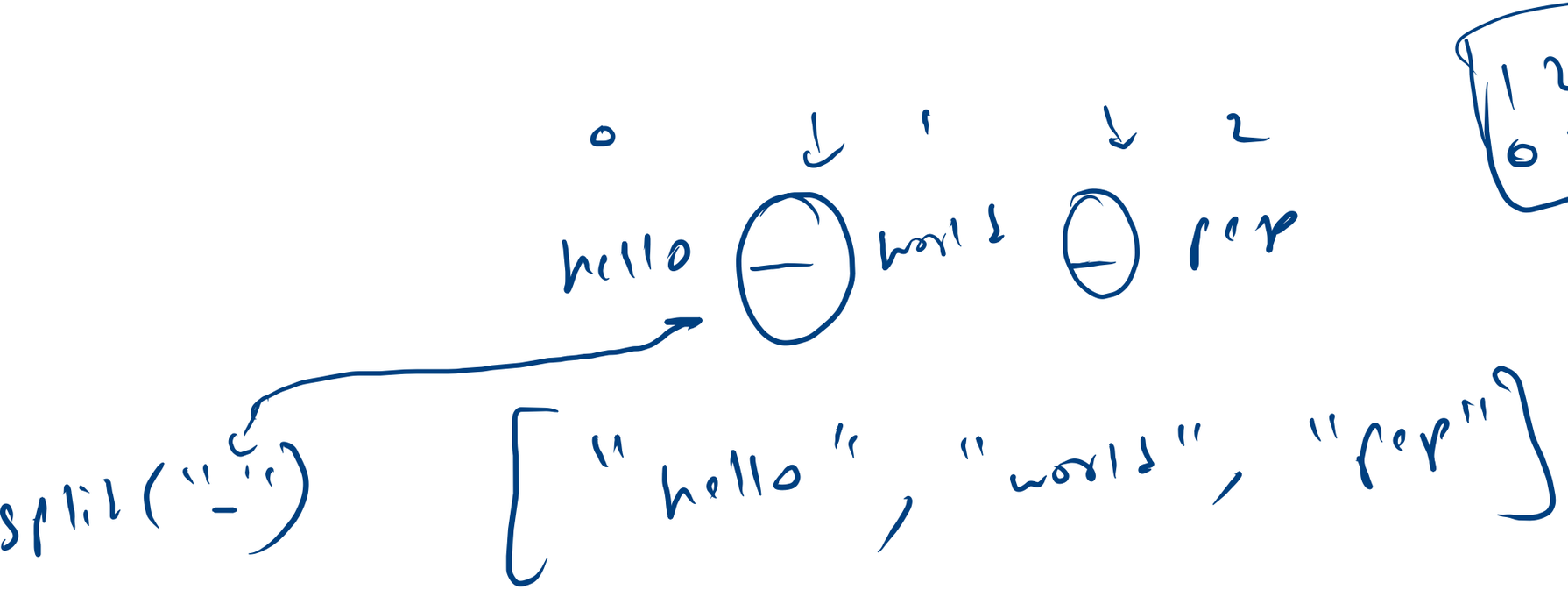
* 16

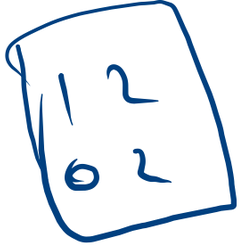


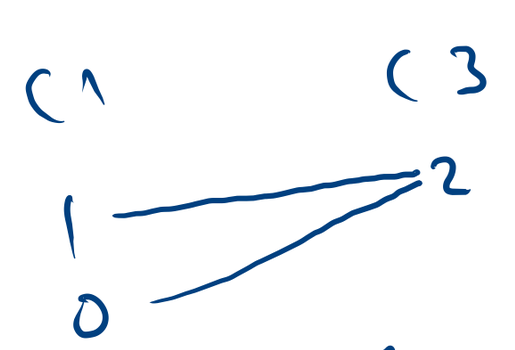
0 3
0 2
1 3
1 2

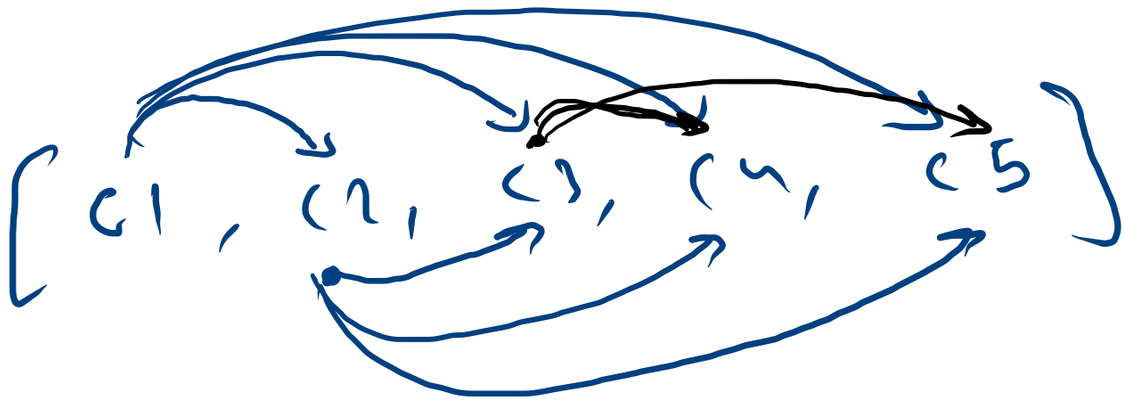


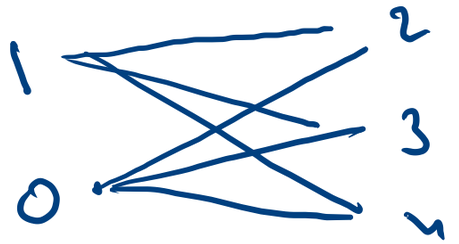
4 +3+3 10

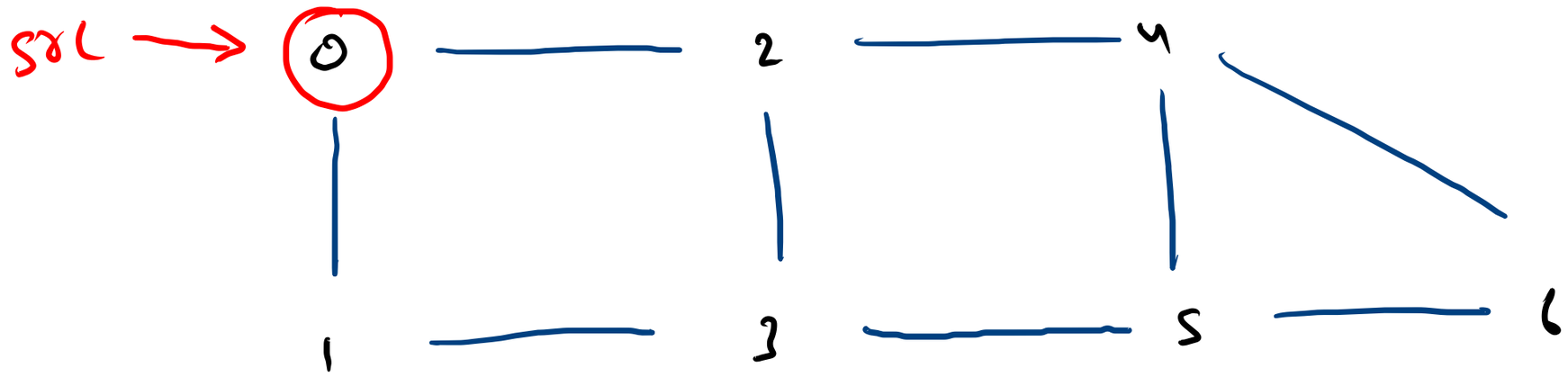

split("-")
"hello", "world", "pop"




C1
C3
C1 x C3




C1 x C2



cycle

all visited once

0 1 3 2 4 5 6. path

0 1 3 2 4 6 5. path

0 2 4 6 5 3 1 ✱ cycle

0 1 3 5 6 4 2 ✱ cycle