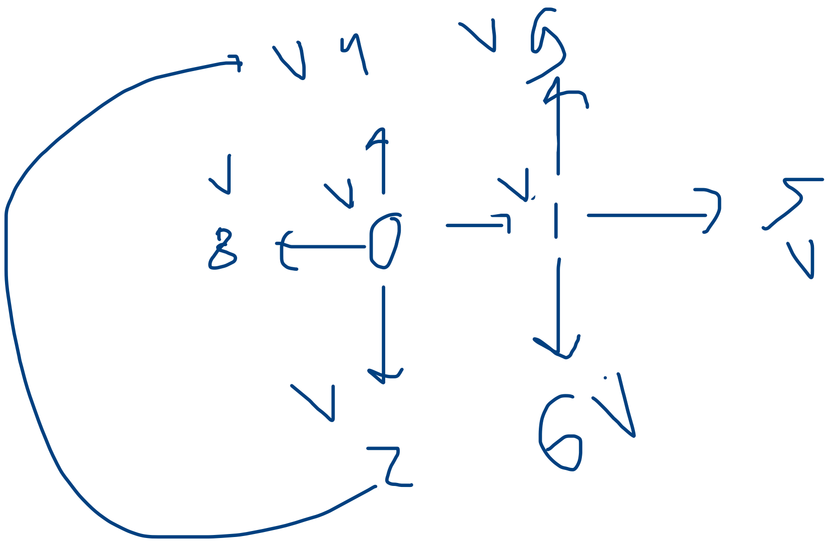
```
0 1
0 2
0 4
0 8
1 5
1 6
1 9
2 4
3 7
3 8
5 8
6 7
6 9
```



~~0~~ ~~1~~ ~~2~~ $\boxed{4}$ 8

5 9 6 $\boxed{7}$

~~0~~ ~~1~~ ~~2~~ ~~4~~ 8 9

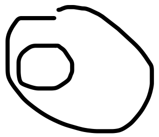
5 6

0

|

BFS vs DFS

BFS



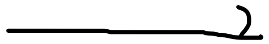
dest close
to src

Time: O(V + E)

Req More
space

FIFO

DFS



dest far to
src

Time: O(V + E)

Req Less
space

LIFO

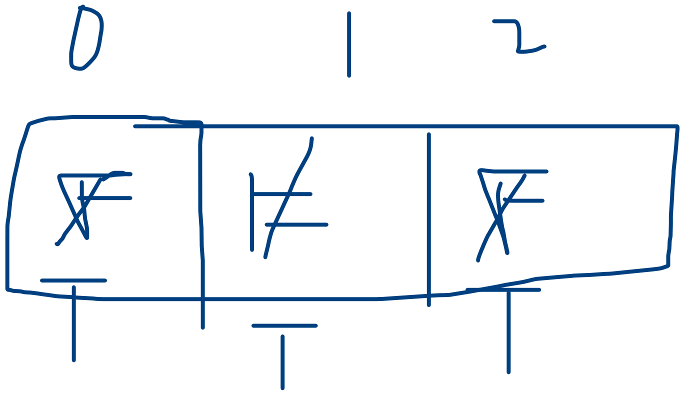
Number of Provinces / num of comp

DFS

$$\begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 2 & 0 & 0 & 1 \end{array}$$

$$\boxed{num = \cancel{0}\ 1+1}$$

$\downarrow$ 0 $\quad$ $\downarrow$ 1 $\quad$ $\downarrow$ 2



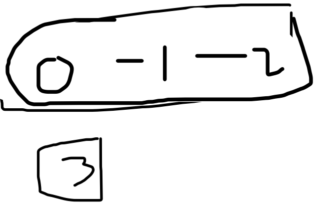
DFS $\longrightarrow$ call

mark all cities which is directly plus undirectly connected with 0 as true

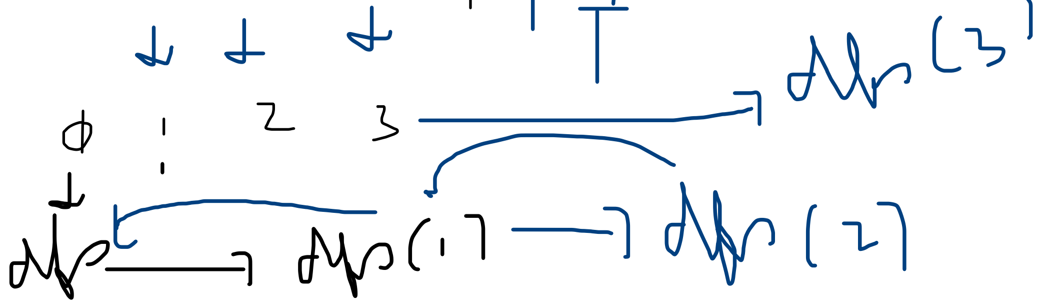
$n_c = \cancel{0}\ \cancel{1}\ 2$

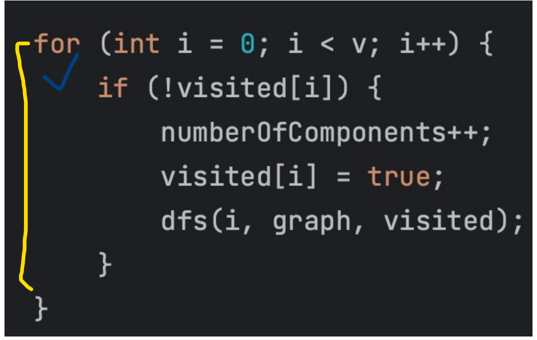
DFS

$$\begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 2 & 0 & 1 & 1 & 0 \\ 3 & 0 & 0 & 0 & 1 \end{array}$$

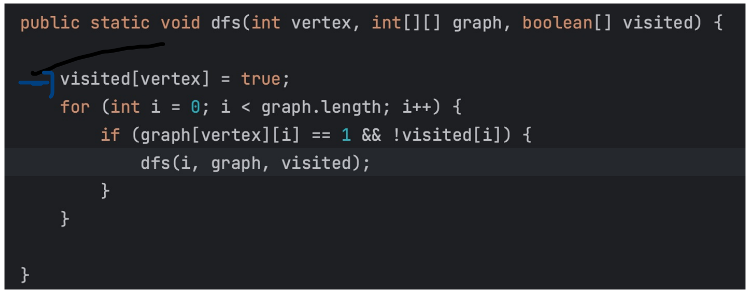


0	1	2	3
~~F~~ T	~~F~~ T	~~F~~ T	F





```java
for (int i = 0; i < v; i++) {
    if (!visited[i]) {
        numberOfComponents++;
        visited[i] = true;
        dfs(i, graph, visited);
    }
}
```

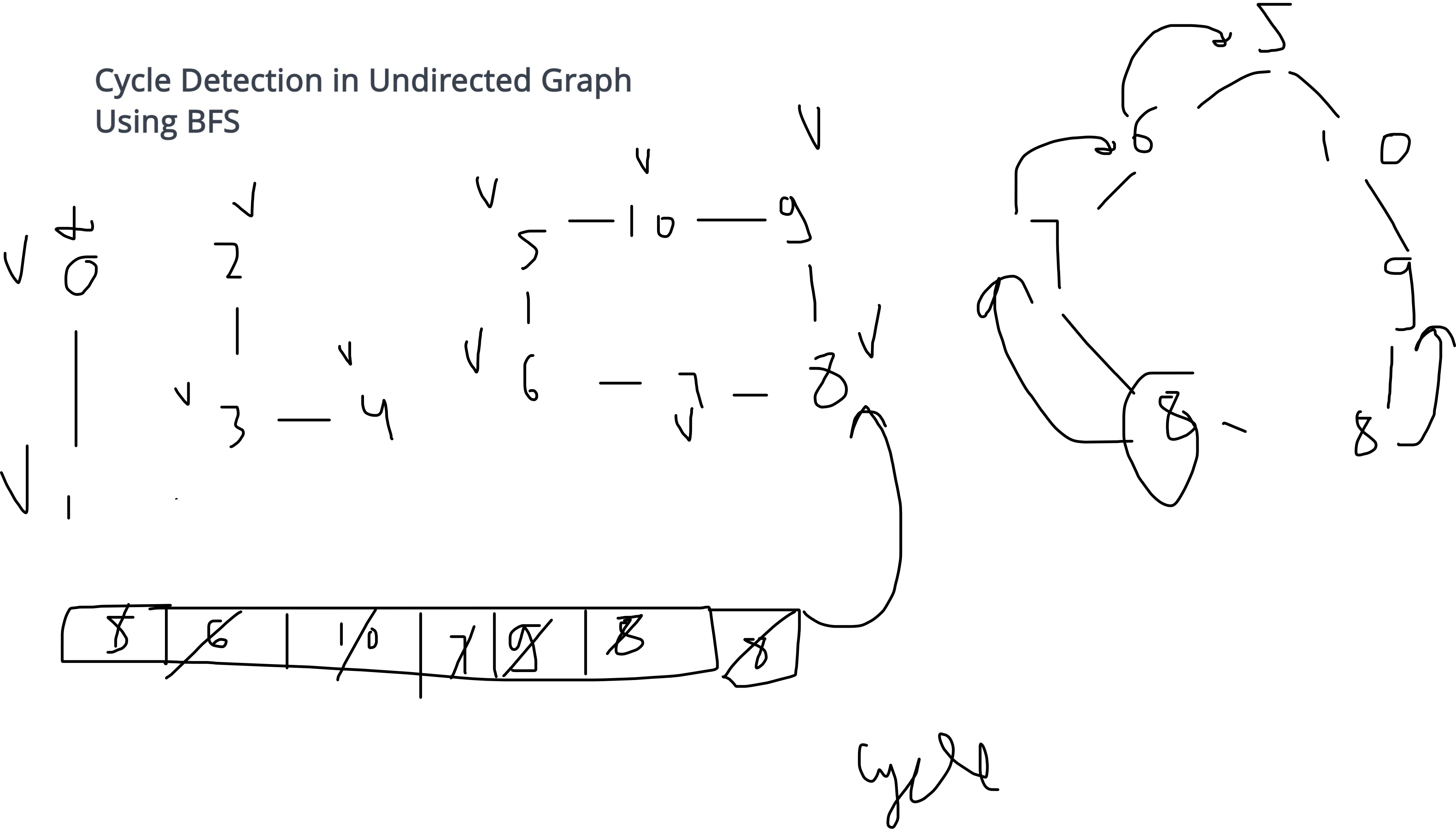


```java
public static void dfs(int vertex, int[][] graph, boolean[] visited) {

    visited[vertex] = true;
    for (int i = 0; i < graph.length; i++) {
        if (graph[vertex][i] == 1 && !visited[i]) {
            dfs(i, graph, visited);
        }
    }

}
```

Cycle Detection in Undirected Graph Using BFS



0
1
2
3
4
5
10
9
6
7
8
5
6
10
7
9
8
8
cycle