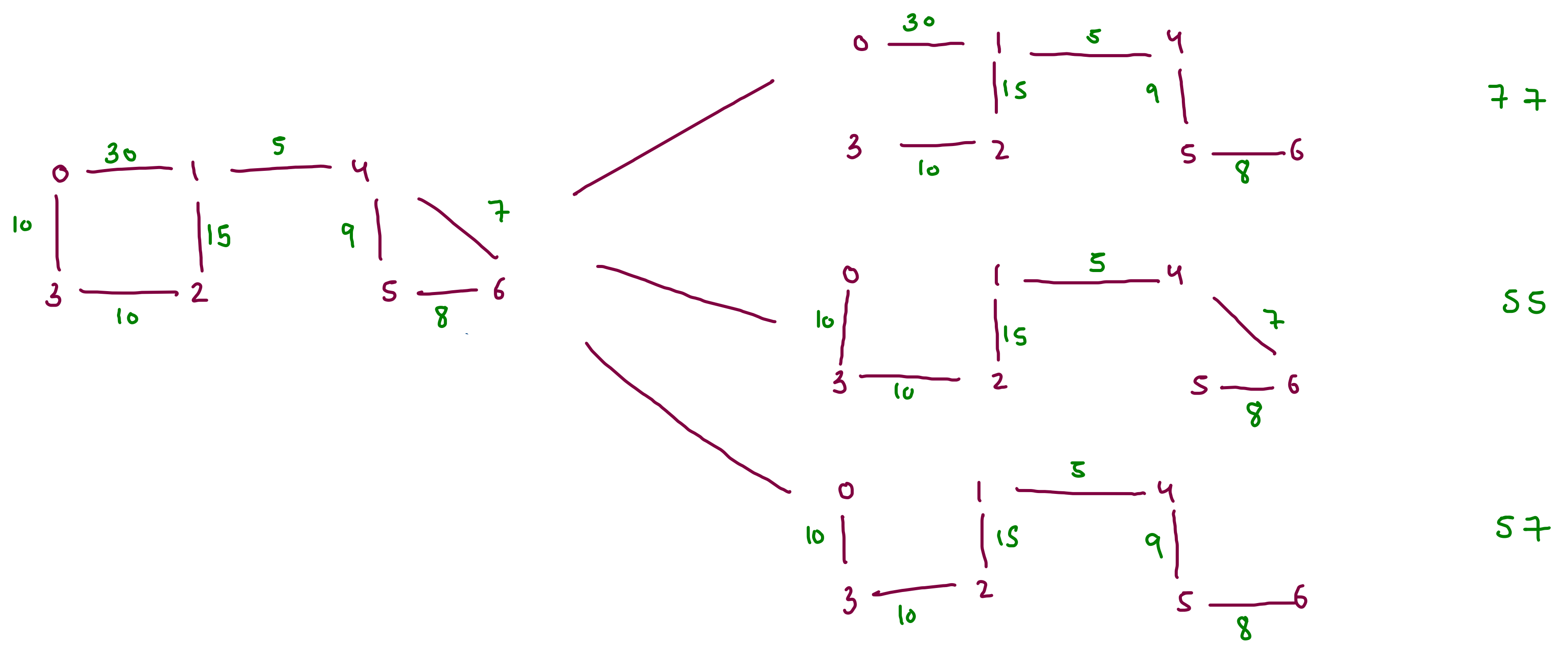
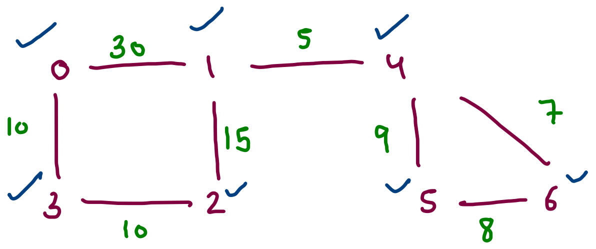
min spanning tree



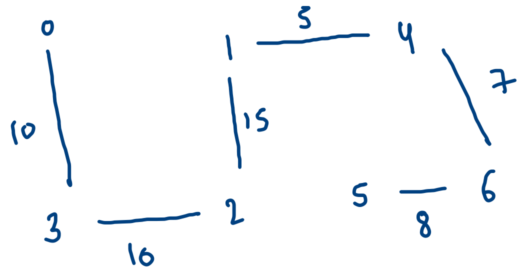


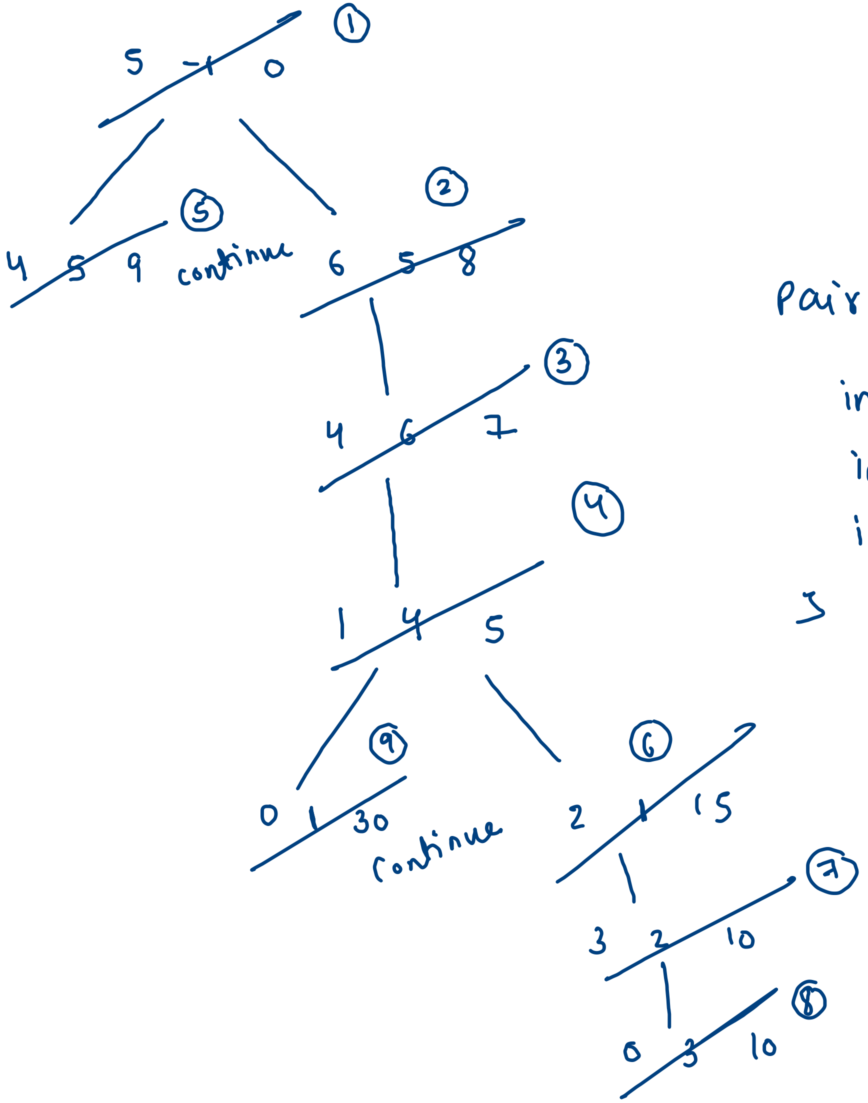
remove

mark*

work

add unvisited nbr



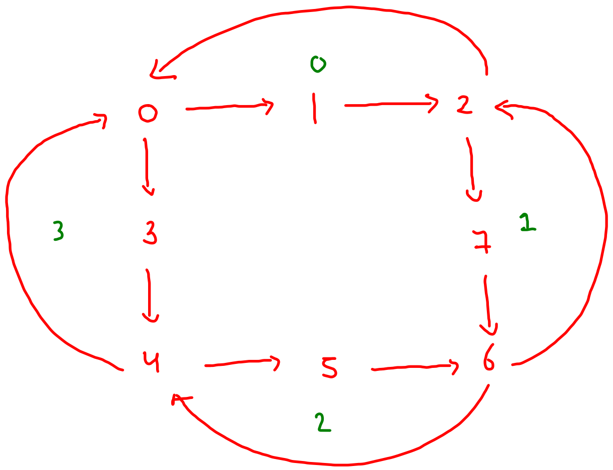


```
Pair {
    int v;
    int av;
    int wt;
}
```

815. Bus Routes

[[0, 1, 2], [2, 7, 6], [4, 5, 6], [0, 3, 4]]

0 1 2 3



bus stand → buses

src = 1

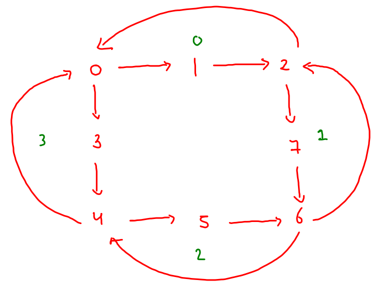
0 →	0, 3
1 →	0
2 →	0, 1
3 →	3
4 →	2, 3
5 →	2
6 →	1, 2
7 →	1

0, 3

dest = 6

[[0, 1, 2], [2, 7, 6], [4, 5, 6], [0, 3, 4]]

0 1 2 3



vis (bus stand) : 1, 0, 2

bus stand → buses vis (buses) : 0, 3

queue
↓
Pair
(bus stand, dev)

0 → 0, 3

1 → 0

2 → 0, 1

3 → 3

4 → 2, 3

5 → 2

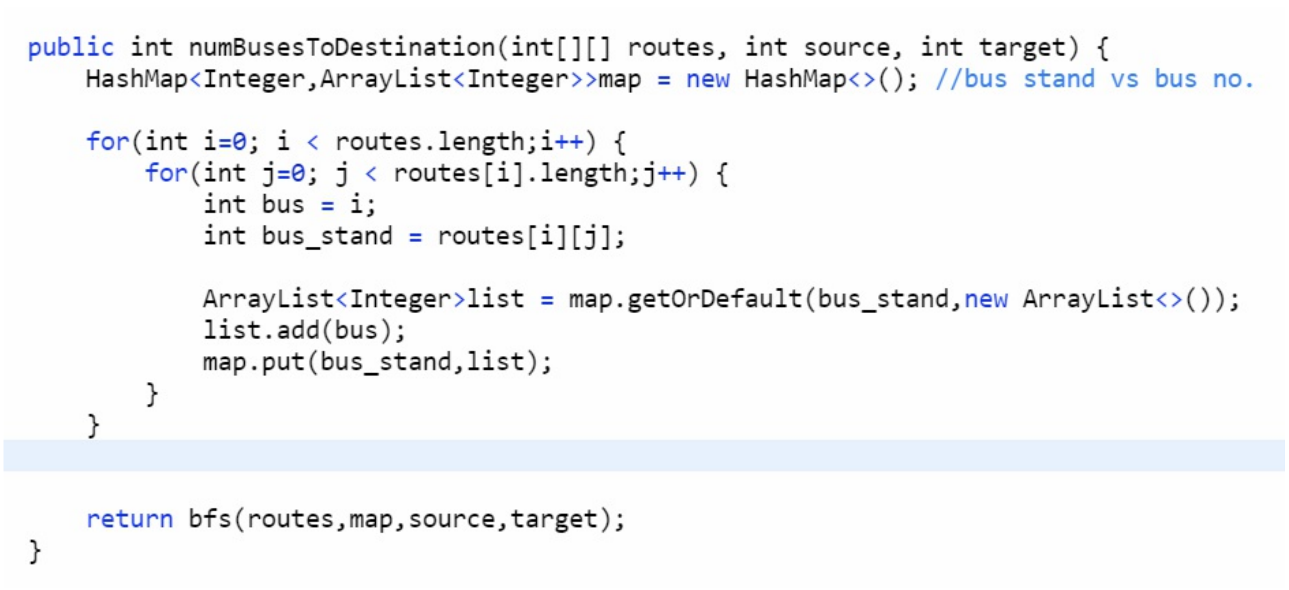
6 → 1, 2

7 → 1

| ~~1,0~~ | ~~0,1~~ | ~~2,1~~ | 3,2 | 4,2 | 7,2 | 6,2 |

src = 1

dest = 7



```java
public int numBusesToDestination(int[][] routes, int source, int target) {
    HashMap<Integer,ArrayList<Integer>>map = new HashMap<>(); //bus stand vs bus no.

    for(int i=0; i < routes.length;i++) {
        for(int j=0; j < routes[i].length;j++) {
            int bus = i;
            int bus_stand = routes[i][j];

            ArrayList<Integer>list = map.getOrDefault(bus_stand,new ArrayList<>());
            list.add(bus);
            map.put(bus_stand,list);
        }
    }

    return bfs(routes,map,source,target);
}
```

map → bus stand vs bus

0 → 0, 3

1 → 0

2 → 0, 1

7 → 1

6 → 1, 2

4 → 2, 3

5 → 2

3 → 3

[[0, 1, 2], [2, 7, 6], [4, 5, 6], [0, 3, 4]]

0 1 2 3

```java
while(q.size() > 0) {
    Pair rem = q.remove();

    if(rem.bus_stand == dest) {
        return rem.lev;
    }

    for(int bus : map.get(rem.bus_stand)) {
        if(vis_bus.contains(bus) == false) {
            vis_bus.add(bus);
            for(int bus_stand : routes[bus]) {
                if(vis_bus_stand.contains(bus_stand) == false) {
                    q.add(new Pair(bus_stand,rem.lev + 1));
                    vis_bus_stand.add(bus_stand);
                }
            }
        }
    }
}
```

src = 3

dest = 7

$$\begin{array}{cccc} [\,[0,1,2], & [2,7,6], & [4,5,6], & [0,3,4]\,] \\ 0 & 1 & 2 & 3 \end{array}$$

vis_bus_stand : 3, 0, 4, 1, 2, 5, 6

vis_bus : 3, 0, 2, 1

map → bus stand vs bus

0 → 0, 3

1 → 0

2 → 0, 1

7 → 1

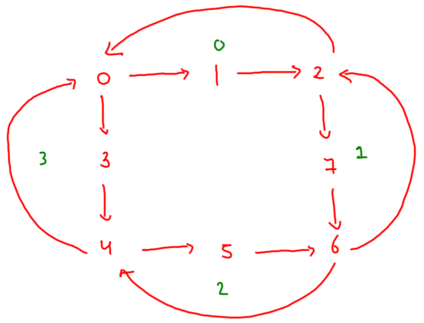
6 → 1, 2

4 → 2, 3

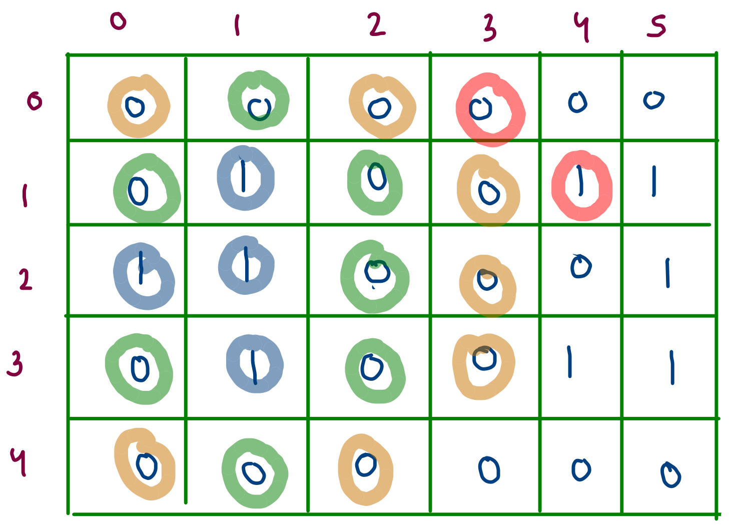
5 → 2

3 → 3

$\cancel{3,0}$	$\cancel{0,1}$	$\cancel{4,1}$	$\cancel{1,2}$	$\cancel{2,2}$	$\cancel{5,2}$	$\cancel{6,2}$	$\cancel{7,3}$



Shortest Bridge - LeetCode



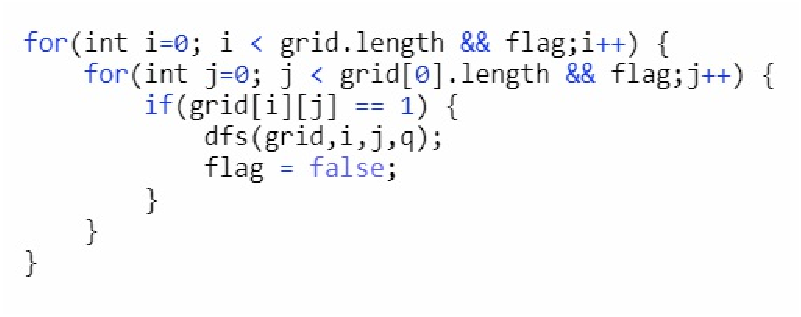
$$
\begin{array}{c|cccccc}
 & 0 & 1 & 2 & 3 & 4 & 5 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 1 & 1 \\
2 & 1 & 1 & 0 & 0 & 0 & 1 \\
3 & 0 & 1 & 0 & 0 & 1 & 1 \\
4 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}
$$

blue : 0th lev

green : 1st lev

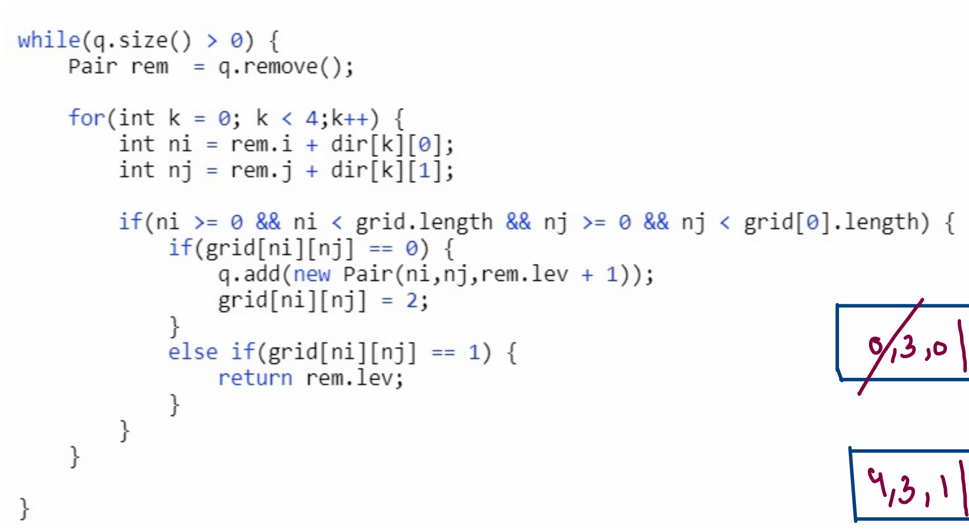
orange : 2nd lev

red : 3rd lev



```java
for(int i=0; i < grid.length && flag;i++) {
    for(int j=0; j < grid[0].length && flag;j++) {
        if(grid[i][j] == 1) {
            dfs(grid,i,j,q);
            flag = false;
        }
    }
}
```

	0	1	2	3
0	0	0	~~0~~ 2	~~1~~ 2
1	1	1	~~0~~ 2	~~1~~ 2
2	1	0	~~0~~ 2	~~1~~ 2
3	1	~~0~~ 2	~~1~~ 2	~~1~~ 2
4	0	~~0~~ 2	~~1~~ 2	~~0~~ 2



```java
while(q.size() > 0) {
    Pair rem  = q.remove();

    for(int k = 0; k < 4;k++) {
        int ni = rem.i + dir[k][0];
        int nj = rem.j + dir[k][1];

        if(ni >= 0 && ni < grid.length && nj >= 0 && nj < grid[0].length) {
            if(grid[ni][nj] == 0) {
                q.add(new Pair(ni,nj,rem.lev + 1));
                grid[ni][nj] = 2;
            }
            else if(grid[ni][nj] == 1) {
                return rem.lev;
            }
        }
    }

}
```

~~0,3,0~~	~~1,3,0~~	~~2,3,0~~	~~3,3,0~~	~~3,2,0~~	~~4,2,0~~	~~0,2,1~~	~~1,2,1~~	2,2,1

4,3,1	3,1,1	4,1,1	0,1,2