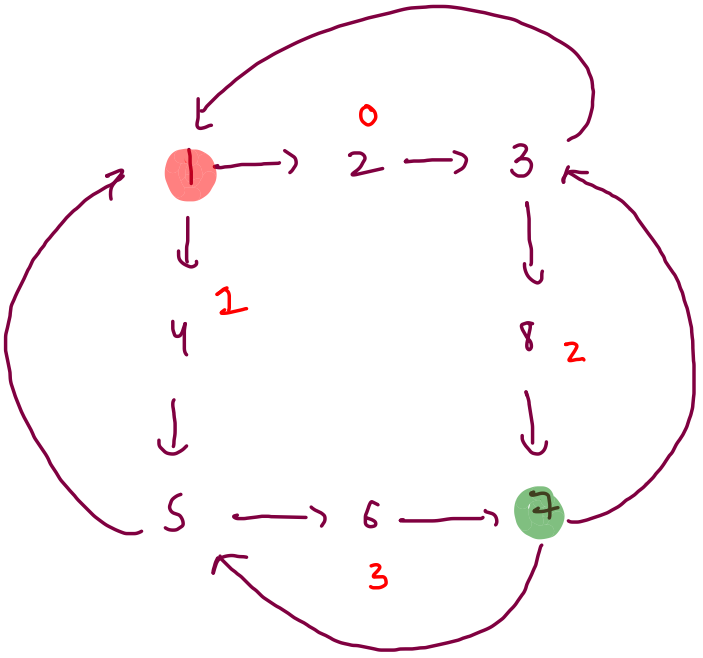
Bus routes

arr : [[1, 2, 3] , [1, 4, 5] , [3, 8, 7] , [5, 6, 7]]
0 1 2 3



1 -> 0, 1

2 -> 0

3 -> 0, 2

4 -> 1

5 -> 1, 3

6 -> 3

8 -> 2

7 -> 2, 3

src -> 1

dest -> 7

HM (Bus stand no. vs Bus)

1 → 0, 1

2 → 0

3 → 0, 2

4 → 1

5 → 1, 3

6 → 3

8 → 2

7 → 2, 3

vis (Bus stand) :

vis (Bus) :

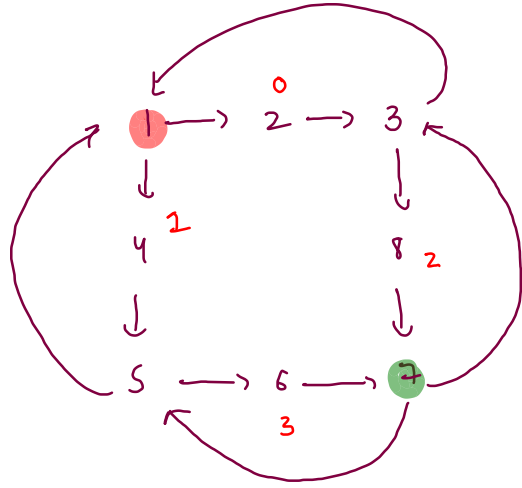
src = 1

dest = 7

dev = 2

c = 6

count [remove
mark
add nbr



```
routes = [[7,12],[4,5,15],[6],[15,19],
```

0 1 2 3

```java
for(int i=0; i < routes.length;i++) {
    int bus_no = i;
    for(int j = 0; j < routes[i].length;j++) {
        int bus_stop_no = routes[i][j];

        if(map.containsKey(bus_stop_no) == false) {
            ArrayList<Integer>list = new ArrayList<>();
            list.add(bus_no);
            map.put(bus_stop_no,list);
        }
        else {
            ArrayList<Integer>list = map.get(bus_stop_no);
            list.add(bus_no);
            map.put(bus_stop_no,list);
        }
    }
}
```

bus_no = 3

7 -> [0]

19 -> [3]

12 -> [0]

4 -> [1]

5 -> [1]

15 -> [1,3]

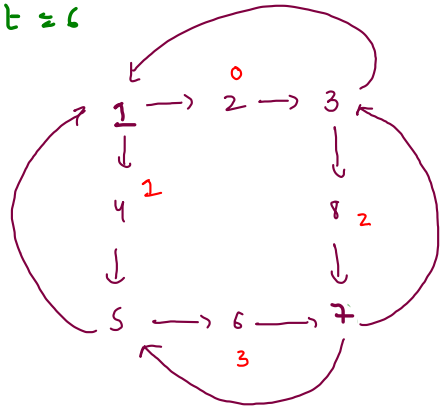
6 -> [2]

```
while(q.size() > 0) {
    int count = q.size();

    while(count-- > 0) {
        //remove
        int rem = q.remove();

        if(rem == tar) {
            return lev;
        }

        //mark
        if(bus_stop_vis.contains(rem) == true) {
            continue;
        }

        bus_stop_vis.add(rem);

        //add nbr
        for(int bus : map.get(rem)) {
            if(bus_vis.contains(bus) == false) {
                bus_vis.add(bus);
                for(int bus_stop : routes[bus]) {
                    if(bus_stop_vis.contains(bus_stop) == false) {
                        q.add(bus_stop);
                    }
                }
            }
        }
    }

    lev++;
}
```

arr : [[1,2,3] , [1,4,5] , [3,8,7] , [5,6,7]]
0 1 2 3

src = 2

dest = 6



tar = ~~0~~ ~~1~~ ~~2~~ 3

1 -> 0,1

2 -> 0

3 -> 0,2

4 -> 1

5 -> 1,3

8 -> 2

7 -> 2,3

6 -> 3

HM bus stand no vs bus

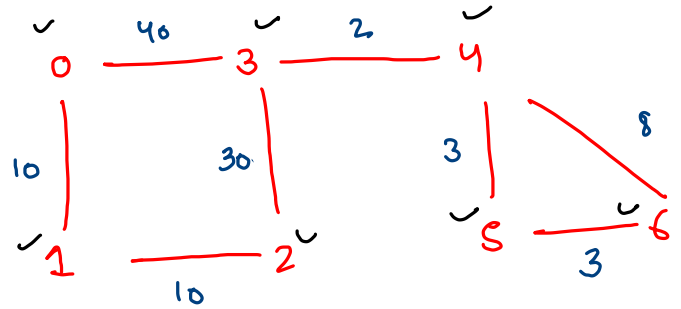
bus_vis : 0,1,2,3

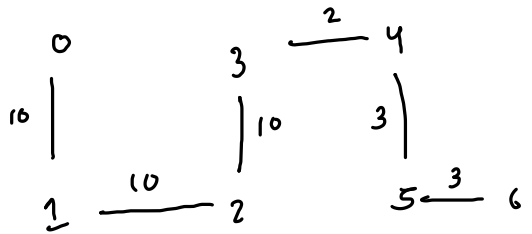
bus_stop_vis : 2,1,3,4,5,8,7

~~2~~	~~1~~	~~3~~	~~4~~	~~5~~	~~8~~	~~7~~	~~6~~	7

Prims

```
7
8
0 1 10
1 2 10
2 3 10
0 3 40
3 4 2
4 5 3
5 6 3
4 6 8
```



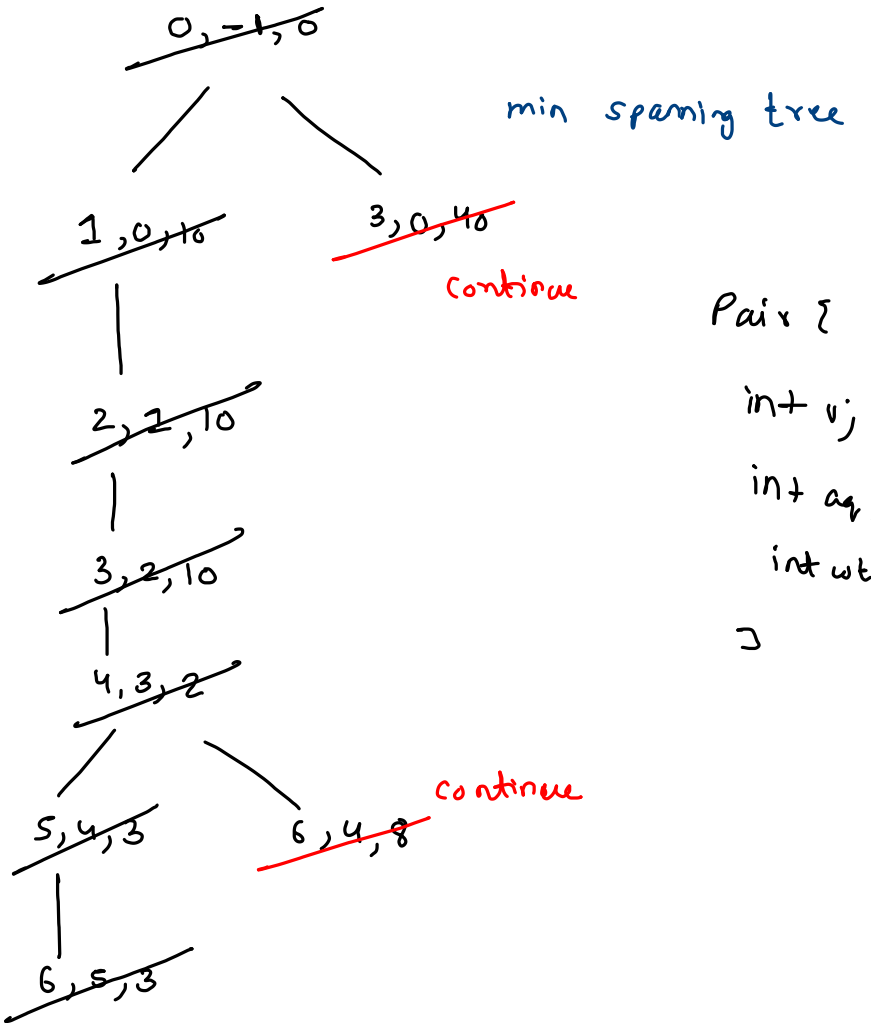


remove

mark ×

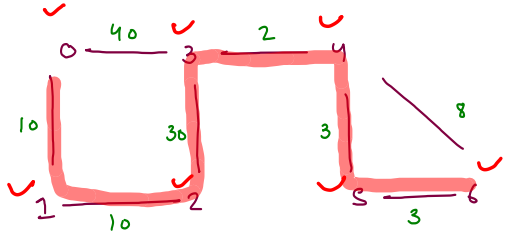
work

add

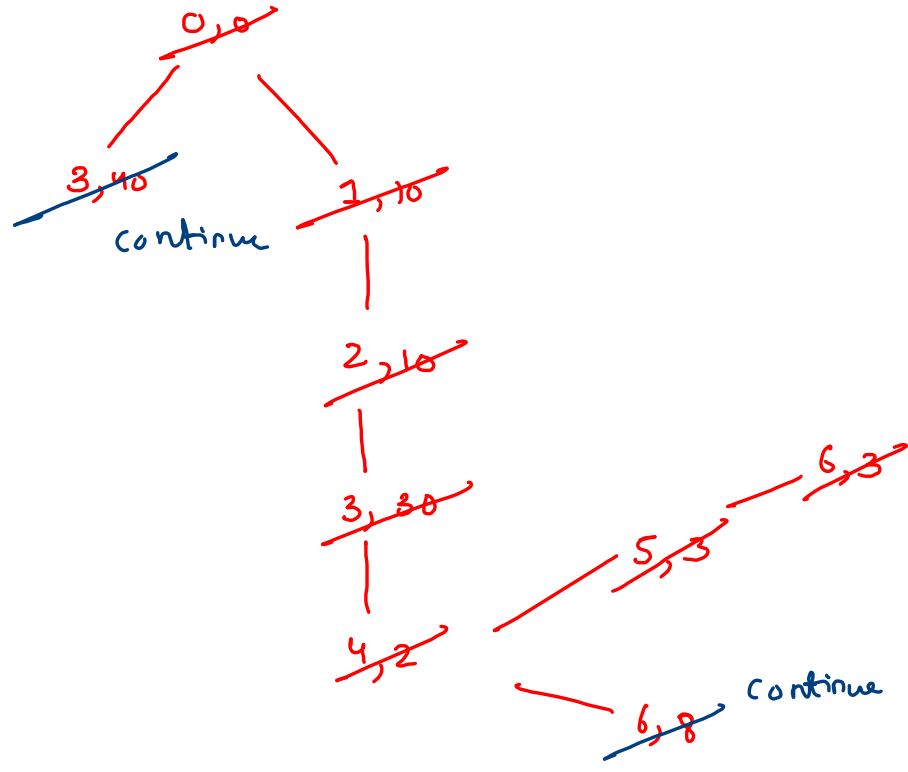



```
Pair {
    int v;
    int aq;
    int wt;
}
```

```java
public static int prims(ArrayList<ArrayList<Edge>>graph) {
    PriorityQueue<Edge>pq = new PriorityQueue<>();

    int cost = 0;
    pq.add(new Edge(0,0));
    boolean[]vis = new boolean[graph.size()];

    while(pq.size() > 0) {
        //remove
        Edge rem = pq.remove();

        //mark
        if(vis[rem.v] == true) {
            continue;
        }
        vis[rem.v] = true;

        //work
        cost += rem.wt;

        //add nbr
        for(Edge ne : graph.get(rem.v)) {
            int nbr = ne.v;
            int wt = ne.wt;
            if(vis[nbr] == false) {
                pq.add(new Edge(nbr,wt));
            }
        }
    }

    return cost
```



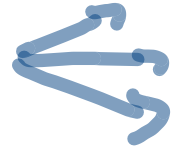
Cost = 0 + 10 + 10 + 30 + 2 + 3 + 3 —



zero one matrix

x_0 y_0 x_1 y_1

$dist = |x_0 - x_1| + |y_0 - y_1|$

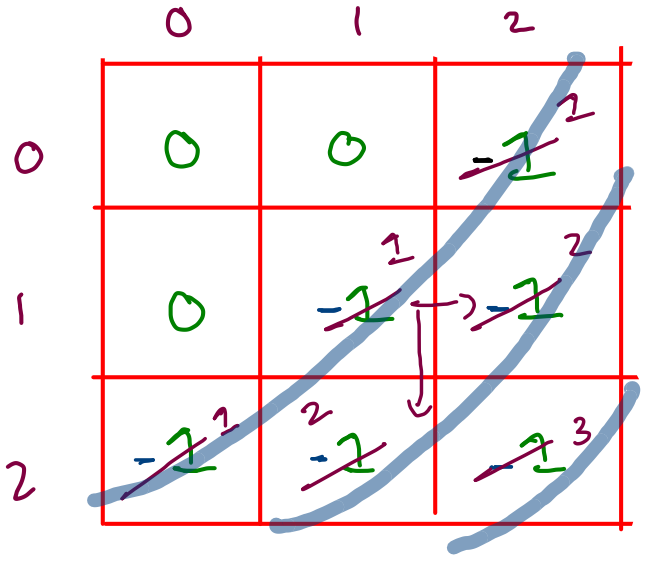


0	0	0
0	1	0
1	1	1

ans

0	0	0
0	1	0
1	2	1

why we should start 0?



remove

add nbr → mark & work

0's

↓

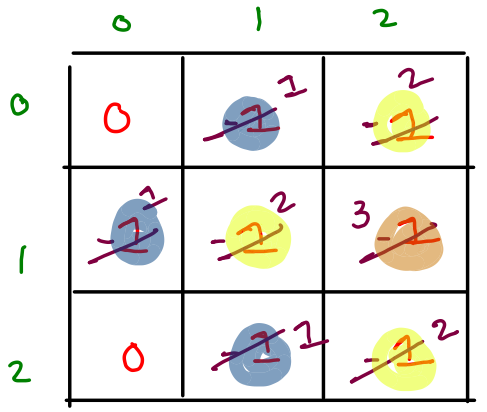
1's 1 away

↓

1's 2 away

0,0	0,1	1,0	1,1	0,2	2,0	1,2	2,1	2,2

```
while(q.size() > 0) {

    //remove
    Pair rem = q.remove();
    int ri = rem.i;
    int rj = rem.j;

    //add nbr
    for(int i=0; i < 4;i++) {
        int ni = ri + dir[i][0];
        int nj = rj + dir[i][1];

        if(ni >= 0 && ni < mat.length && nj >= 0 && nj < mat[0].length && mat[ni][nj] == -1) {
            mat[ni][nj]  = mat[ri][rj] + 1;
            q.add(new Pair(ni,nj));
        }
    }
}

return mat;
```



0 -> water

1 -> land

	0	1	2
0	0 1	1 -x- 0	0 1
1	1 -x- 0	2 -x- 0	1 -x- 0
2	0 1	1 -x- 0	0 1

1	1	1
1	2	1
1	1	1

0,0	0,2	2,0	2,2	0,1	1,0	2,1	1,2	1,1

(i) find nearest 1 for every 0.

(ii) find max of these dist

max = 2

(iii) q.add() -> 1's