Graph 3

Good Evening

Today's content

01. Minimum spanning Tree
    - → Kruskal Algo
    - → Prim's Algo
02. Another BFS
03. Djikstra Algorithm

01. Given N islands & cost of construction of a bridge b/w multiple pair of islands. Find minimum cost of construction required such that it is possible to travel from one island to any other island via bridges.

If not possible, return -1;

eg: N = 7 E = 9

1 —3— 2

1 —5— 3

2 —1— 4

2 —5— 5

3 —3— 5

4 —2— 6

3 —8— 6

4 —5— 7

6 —3— 7

01. Graph must be connected.

DFS BFS DSU

02 Min cost → Minimum edges/ bridges

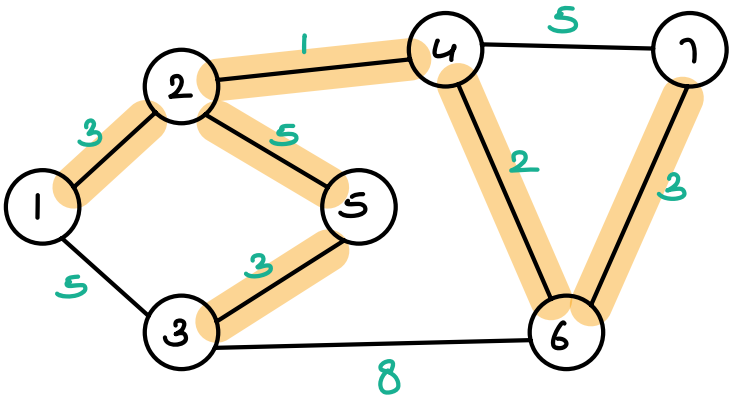
N islands → N-1 edges

⇓

Tree

$\sum$ wt of selected edges is minimum

Minimum spanning Tree



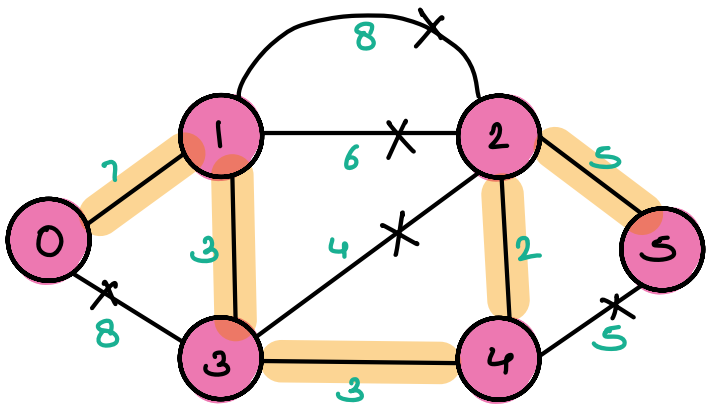
Ans = 17

* Kruskal Algo

→ Select the edge with minimum weight

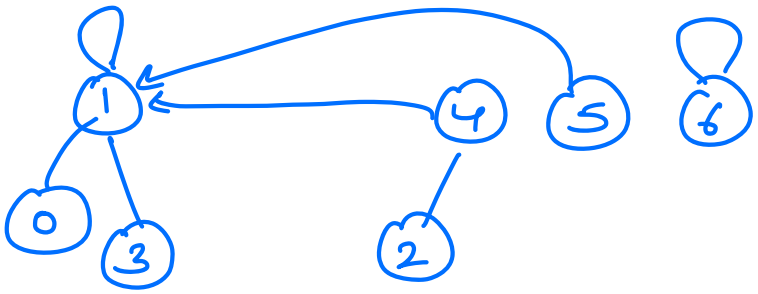
→ Should not form a cycle

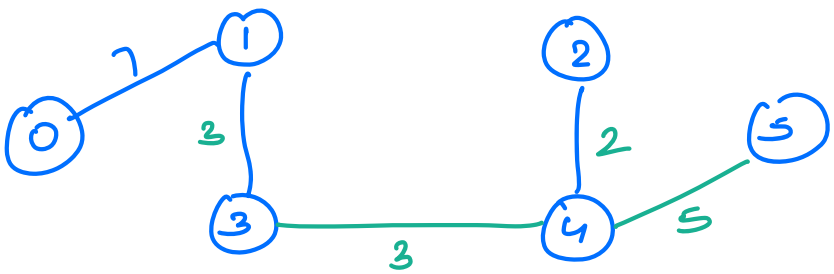
→ Until unless, entire graph is connected



ans = 0 + 2 + 3 + 3 + 5 + 7

01. Sort the edges on the basis of their weight in the ascending order





$\Sigma = 7+3+3+2+5 = 20$

2 — 2 — 4 ✓

1 — 3 — 3 ✓

3 — 3 — 4 ✓

2 — 4 — 3 ✗

4 — 5 — 5 ✓

2 — 5 — 5 ✗

1 — 6 — 2 ✗

0 — 7 — 1 ✓

0 — 8 — 3 ✗

1 — 8 — 2 ✗

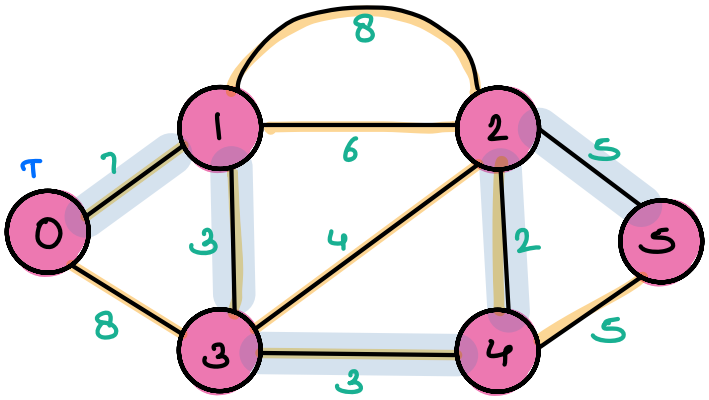
* Select edge one by one if (u, v) doesn't belong to the same set, add it to your ans.

```
if (union(u,v) == true){
    ans += wt(u,v)
}
```

TC: O(ElogE + E)

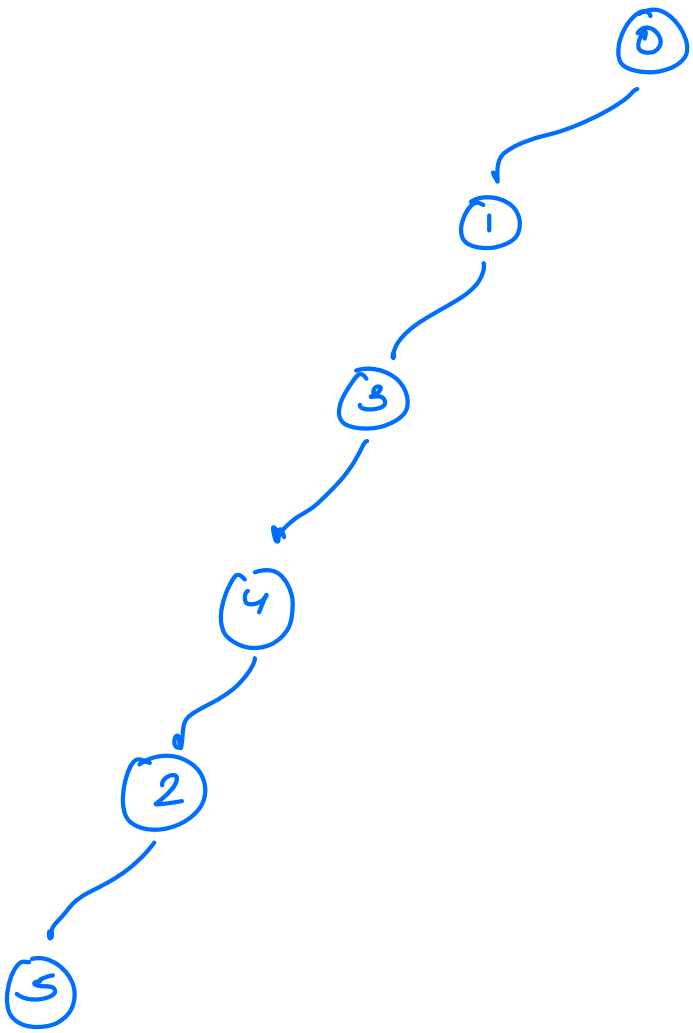
SC: O(N + E)

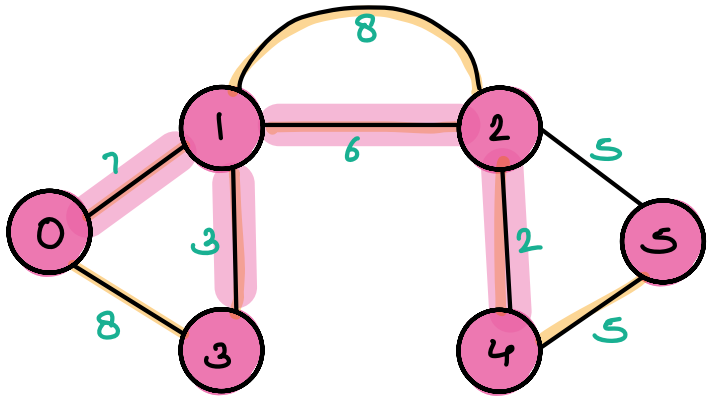
* Prim's Algo



→ Start with any node as root & iterate on its neighbor having minimum weighted edge. Will also be needing a vis arr

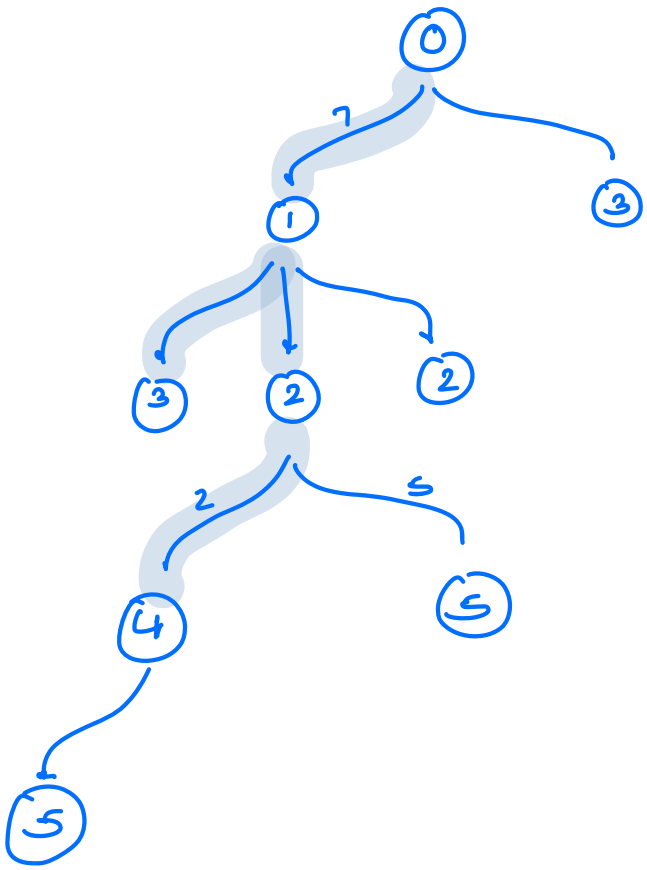
vis[] = 0, 1, 3, 4, 2, 5





vis =

T	T	T	T	T	T
0	1	2	3	4	5



01. Select any node & mark it as true in visited arr

02. Insert all the connected edges of this root

02. while (heap.size() > 0)

01. remove the edge with min wt from heap

if (vis[v] == true) continue;

else

vis[v] = true

ans += wt;

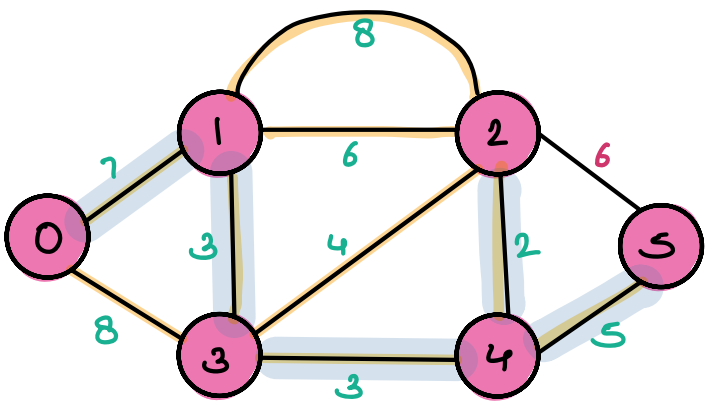
for (int nbr : graph[v]) {

if (vis[nbr] == false) {

heap.insert(wt, nbr);

}

}



vis

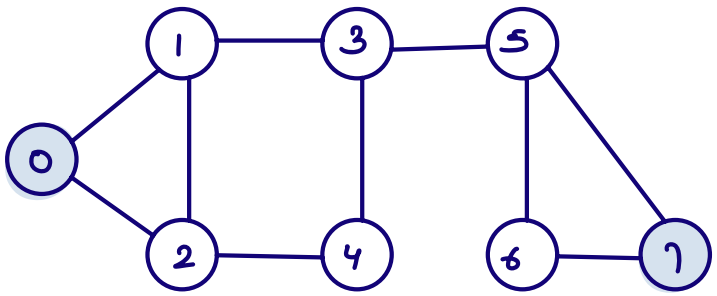
T	T	T	T	T	T
0	1	2	3	4	5

wt, nbr

heap

10:06 → 10:16 pm

Q Find the min no. of edges to reach v starting from u in undirected simple graph



u = 0

v = 7

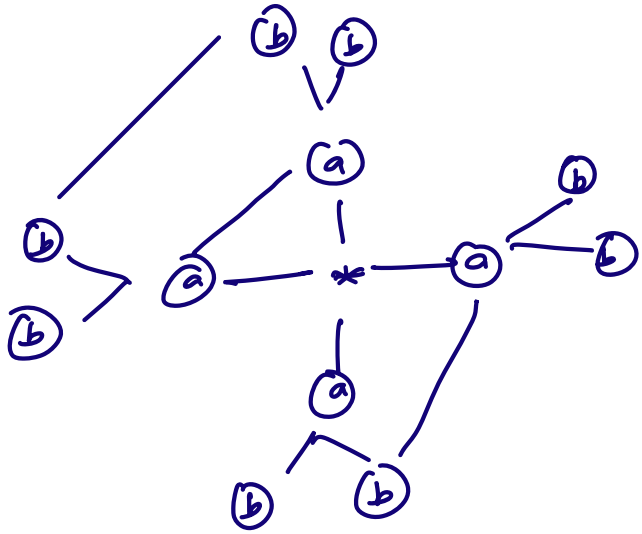
Ans = 4

* Solution → BFS from source

0	1	2	3	4	5	6	7

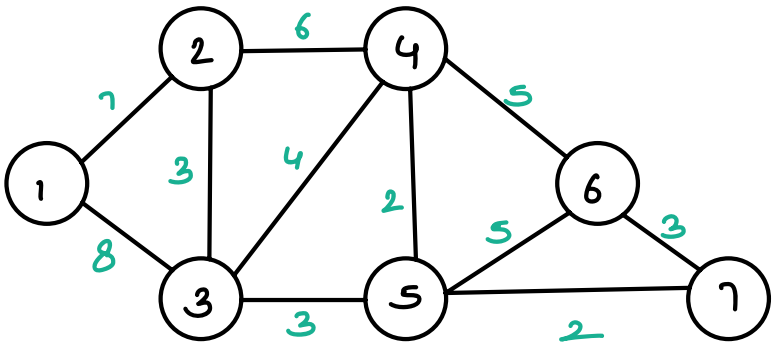
0 1 2 3 4

* Level order linewise



* Djikstra's Algorithm

There are N cities in a country, you are living in city-1. Find minimum distance to reach every other city from city 1.

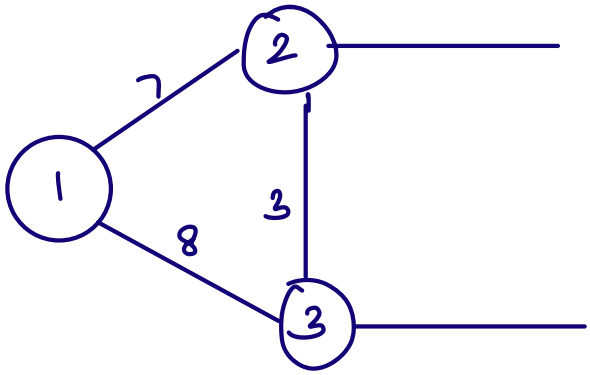


d[ ] =

x	0	7	8	12	11	16	13
0	1	2	3	4	5	6	7

d[i] = minimum distance to reach i^th city from src city.

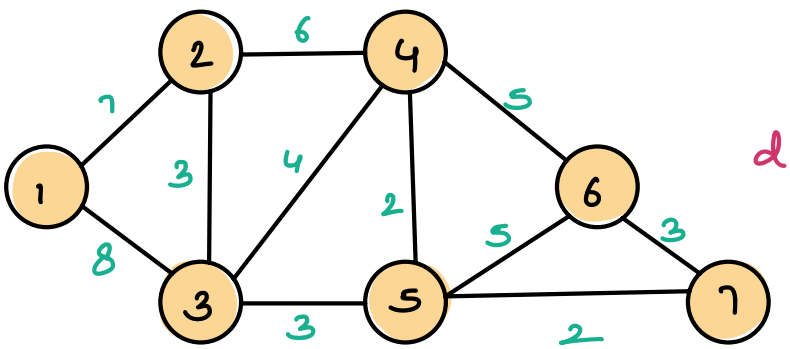
d[src] = 0;



d[ ] =

	x	0	~~∞~~ 7	∞	∞	∞	∞	∞
	0	1	2	3	4	5	6	7

minimum wt so far starting from src



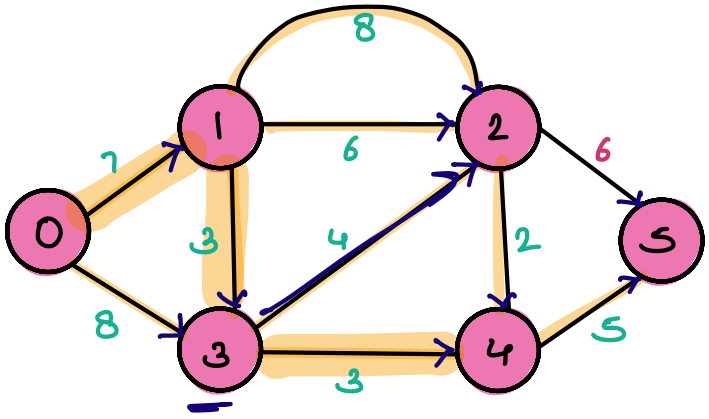
dist =

	x	0	~~∞~~ 7	~~∞~~ 8	~~∞~~ 12	~~∞~~ 11	~~∞~~ 16	~~∞~~ 13
	0	1	2	3	4	5	6	7

```
dist [N+1] ;      // initialise it with ∞

dist [src] = 0;

Minheap < pair > heap;

heap.insert (src)

while (heap.size() > 0) {

    Pair rp = heap.extractmin();

    if ( dis[rp.v] != Integer.MAX_VALUE) {

        continue;

    }

    else

        dis[rp.v] = wsf;      // wsf = rp.wsf

        for ( int nbr : graph[rp.v]) {

            if (dis[nbr] == ∞) {

                heap.insert (wsf + edgewt , nbr);

            }

        }

    }

return dis[];
```

TC: O(E log E)

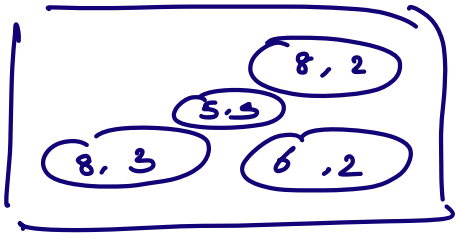
SC: O(E)



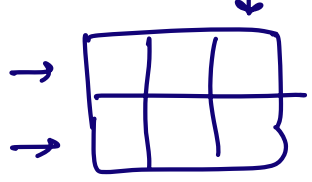
4, 2

0

7, 1



heap



```
public int compare ( int [] a, int [] b)
{
    if ( a[2] < b[2] ) return -1;
    else if ( a[2] > b[2] ) return 1;
    else return 0;
}
```