Lecture :- Graphs - 3

Agenda

- Minimum spanning tree
- Dijkastra Algorithm

Minimum spanning tree

Given n islands and cost of construction of bridge b/w multiple pair of islands, find min cost of construction required such that it is possible to travel from one island to another island via bridges. If not possible, return -1.

ip n = 5

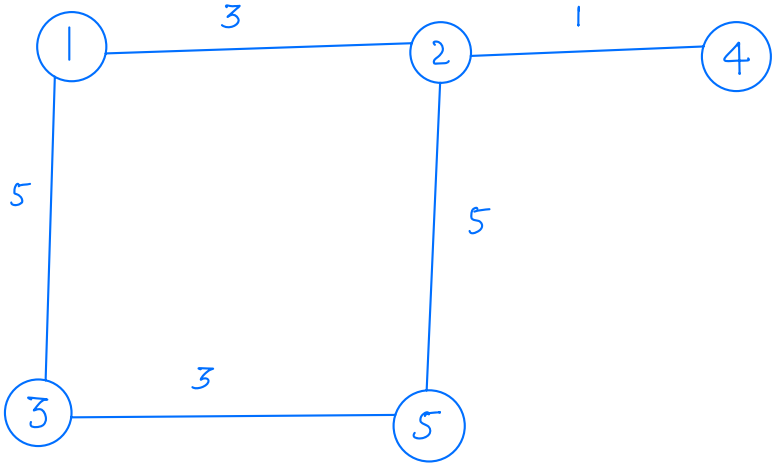
1 —3— 2

1 —5— 3

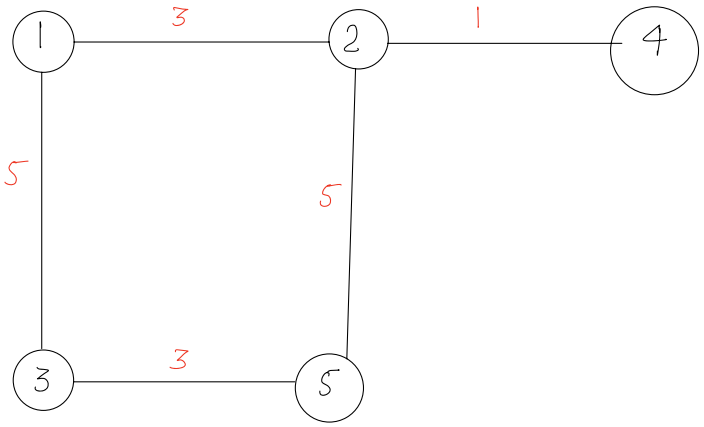
2 —1— 4

2 —5— 5

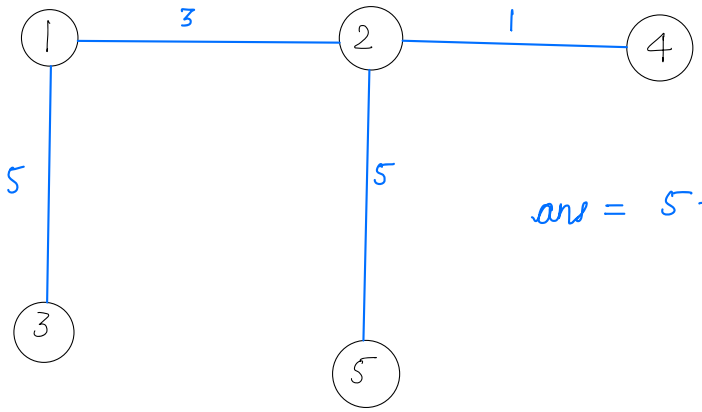
3 —3— 5



Min edges to connect n nodes ⇒ n-1 edges

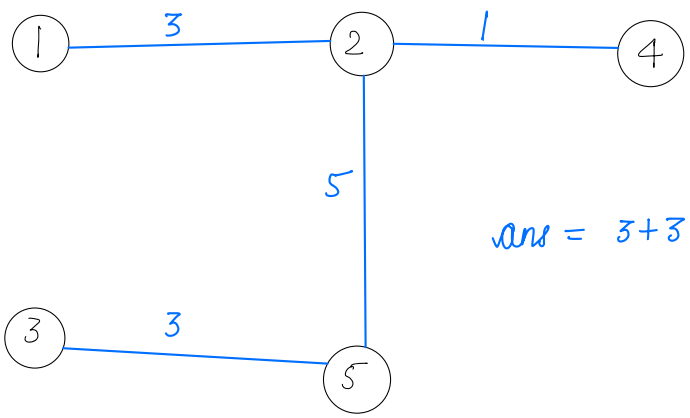


Case1



ans = 5+3+5+1 = 14

Case2



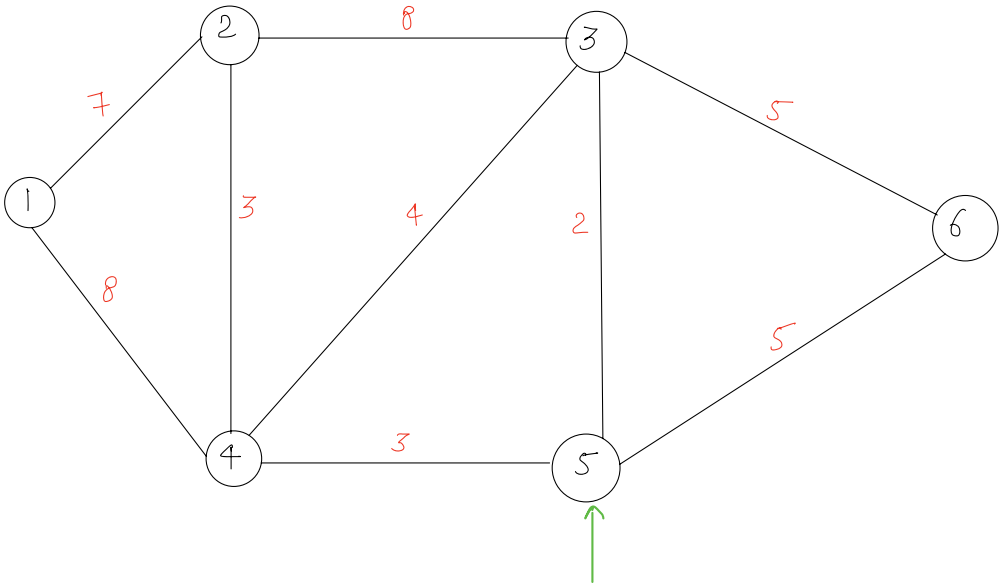
ans = 3+3+5+1 = 12.

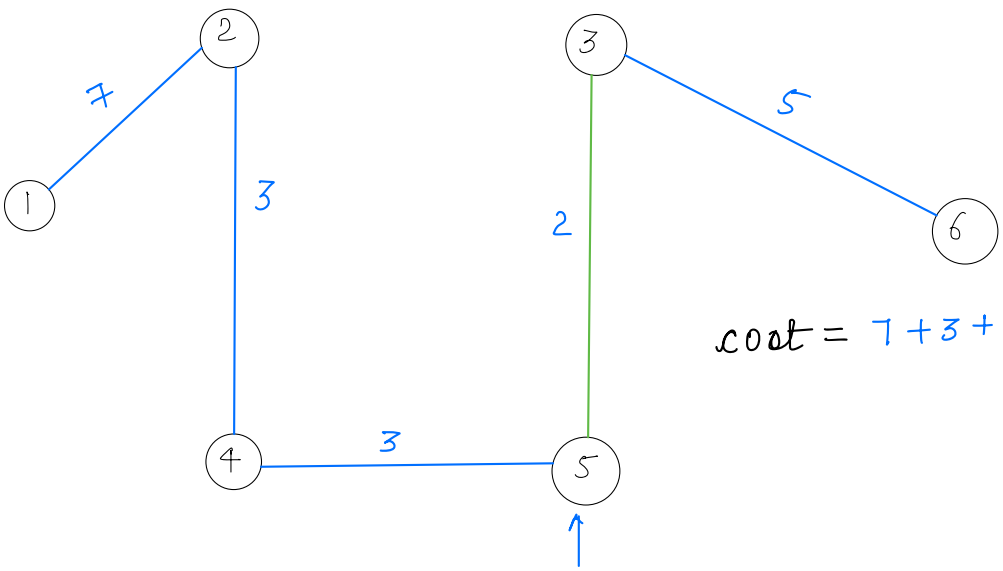
Minimum spanning tree

Tree generated from a connected graph such that all nodes are connected and sum of weights of all selected edges is minimum.

- Kruskal
- Prims

Prims Algorithm





cost = 7 + 3 + 3 + 2 + 5 ⇒ 20

Min heap

~~(4,3)~~ ~~(3,2)~~ ~~(6,5)~~ (2,8) ~~(4,4)~~ ~~(6,5)~~ (1,8) ~~(2,3)~~ ~~(1,7)~~

visited[] =

0	1	2	3	4	5	6
	~~f~~ t	~~f~~ t	~~f~~ t	~~f~~ t	~~f~~ t	~~f~~ t

Pseudocode

```
int prims(n, edges[][]) {
    1. Create a graph.
    PriorityQueue<Pair> pq;
    vis[n+1];
    2. Insert any random node in
       your heap
    cost = 0;
    while(!pq.isEmpty()) {
        Pair curr = pq.poll();
        v = curr.v
        wt = curr.wt;
        if(vis[v]) {
            continue;
        }
        cost += wt;
        vis[v] = true;
        List<Pair> children = graph[v];
        for(Pair child: children) {
            if(!vis[child.v]) {
                pq.add(child);
            }
        }
    }
    return cost;
}
```

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

TC: E log E

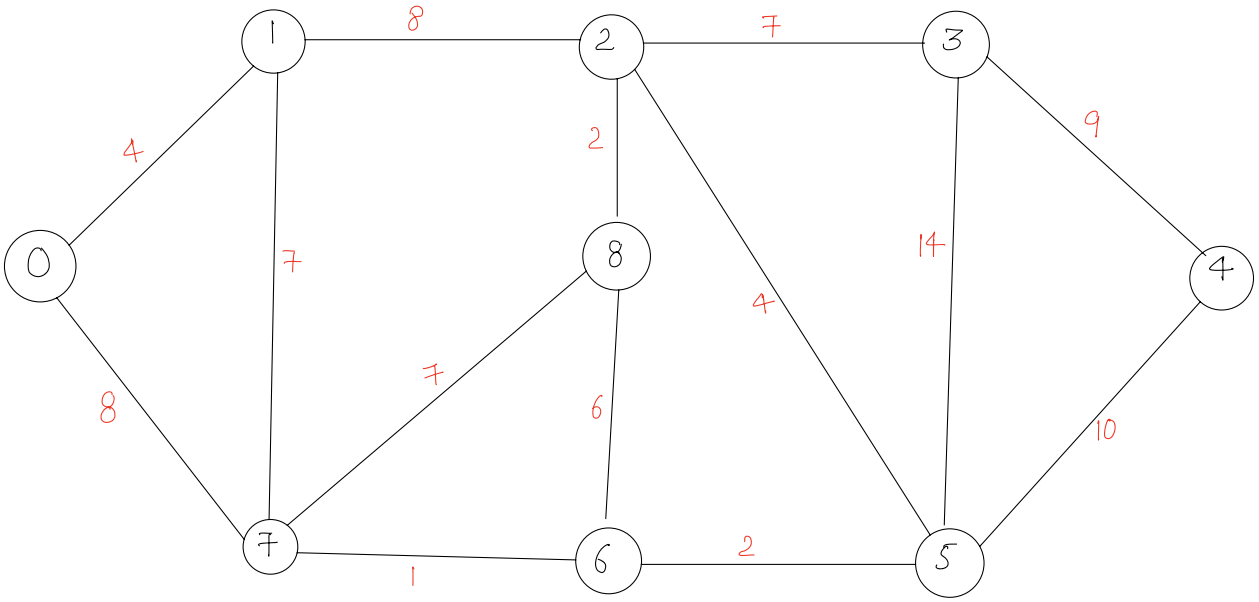
SC: O(e+n)

↑ heap ↑ visited | graph

Break: 7:51 - 8:01

Dijkstra's Algorithm

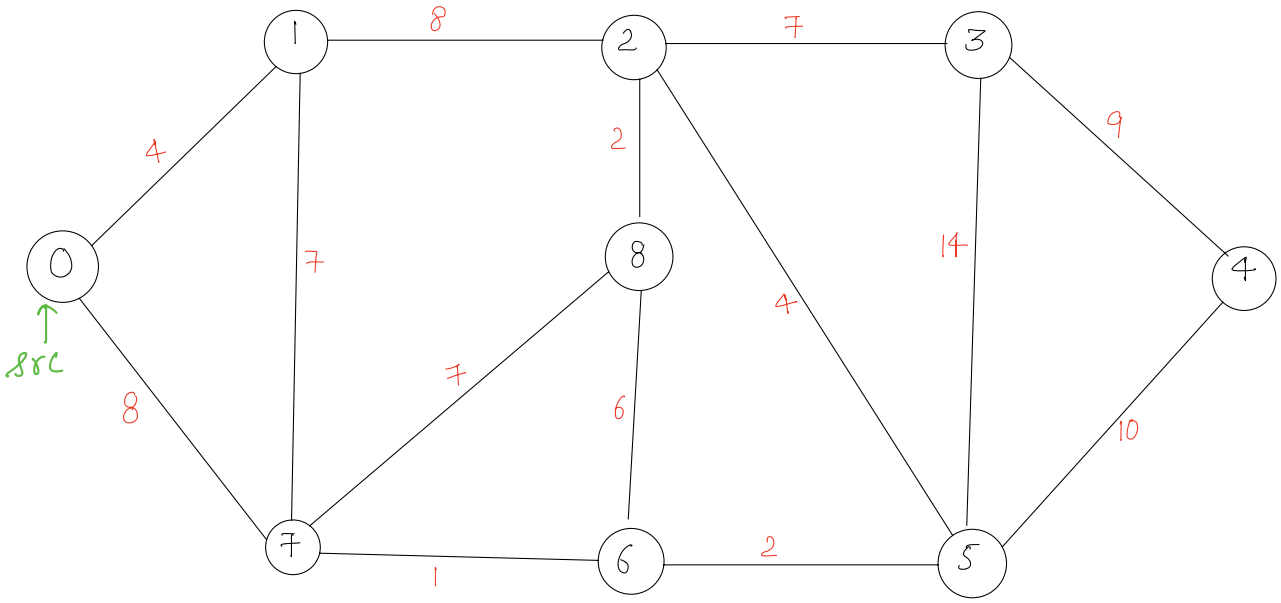
[Min distance from any src to all vertices]

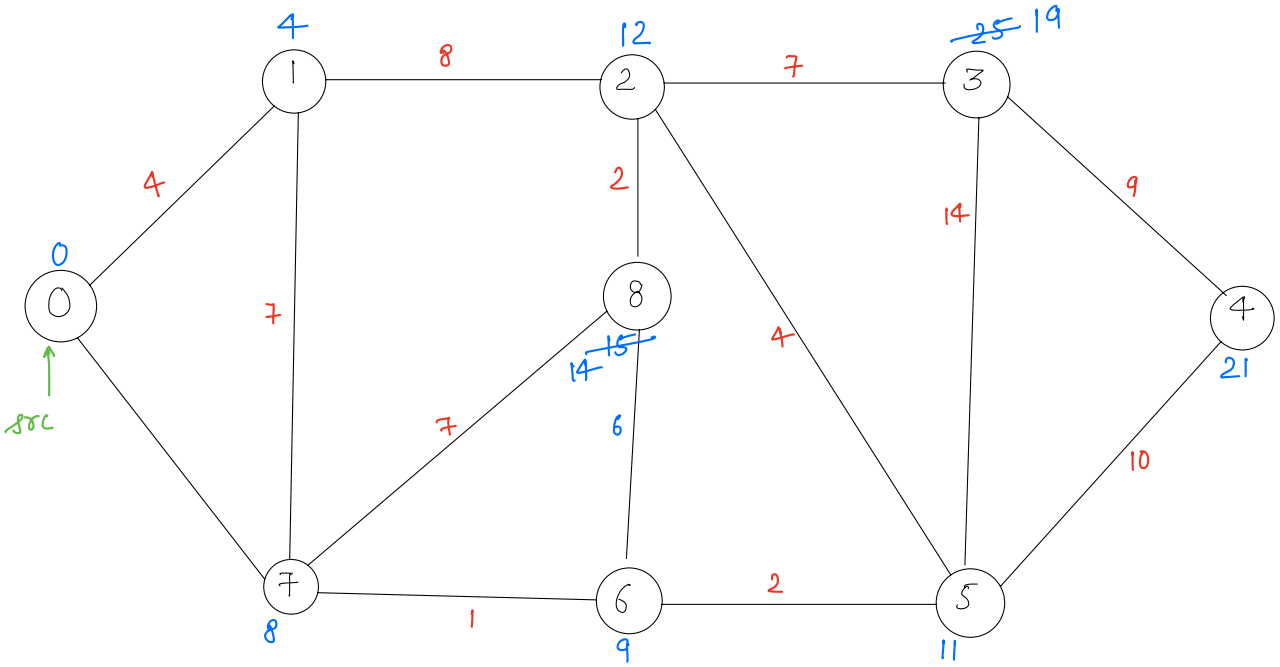


output

src	dest	distance
0	1	4
0	2	12
0	3	19
0	4	21
0	5	11
0	6	9
0	7	8
0	8	14

Example





dist[] ⇒

0	1	2	3	4	5	6	7	8
0	∞ 4	∞ 12	~~∞ 25~~ 19	∞ 21	∞ 11	∞ 9	∞ 8	~~∞ 15~~ 14

visited ⇒

0	1	2	3	4	5	6	7	8
t	t	t	t	t	t	t	t	t

min heap ⇒

~~(1, 4)~~ ~~(7, 8)~~ ~~(2, 12)~~ ~~(6, 9)~~ ~~(8, 15)~~ ~~(5, 11)~~ ~~(4, 21)~~ (3, 25) ~~(8, 14)~~ ~~(3, 19)~~

Pseudocode

```
int[] dijkastra( List<Pair> graph[], int src, int n) {
    dis[n+1];
    for(i= 0; i<=n; i++) {
        if (i != src) {
            dis[i] = ∞;
        }
    }
    vis[n+1]; vis[src]=true;
    priority Queue<Pair> pq;
    for( Pair child: graph[src]) {
        pq.add (child);
    }
    while( !pq.isEmpty()) {
        Pair p = pq.poll();
        if (vis[p.v]) {
            continue;
        }
        vis[p.v] = true;
        List<Pair> children = graph[p.v];
        for(Pair child:children) {
            int currDist = dis[p.v] + child.wt;
            if ( currDist < dis[child.v]) {
                dis[child.v] = currDis;
                pq.add ( new Pair (child.v, currDis));
            }
        }
    return dis;
}
```

TC:
SC: O(V+E)