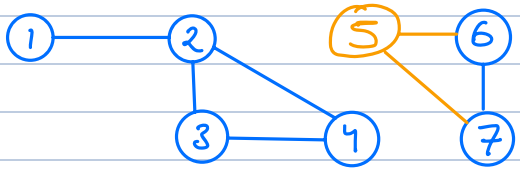
Graphs - 3

Topics To Cover:
1. DSU optimisation } Last Class
2. Application of DSU
3. BFT
4. Minimum Spanning Tree
5. Prim's Algo
6. Dijkstra's Algo.

Song: Call Me Maybe - Carly Rae Jepsen

Hi Everyone!!!

Breadth First Traversal ⟶ Level order Traversal



visited : T T T T T T T
0 1 2 3 4 5 6 7

Q : 1 2 3 4 5 6 7

Ans: 1 2 3 4 5 6 7

```
int [N+1] visited ;
for ( int i = 1 ; i ≤ N ; i++)
{
    if ( visited [i] == False )
        BFT ( graph, visited , i);
}

fn BFT ( graph , visited , i)
{
    Queue q ;
    q. insert (i);
```

```
visited [i] = True;
print (i);

while (q.isEmpty == False)
{
    temp = q.remove();
    for ( int nbr : graph[temp])
    {
        if ( visited [nbr] == False)
        {
            q.insert (nbr);
            visited [nbr] = True;
            print (nbr);
        }
    }
}
```

graph

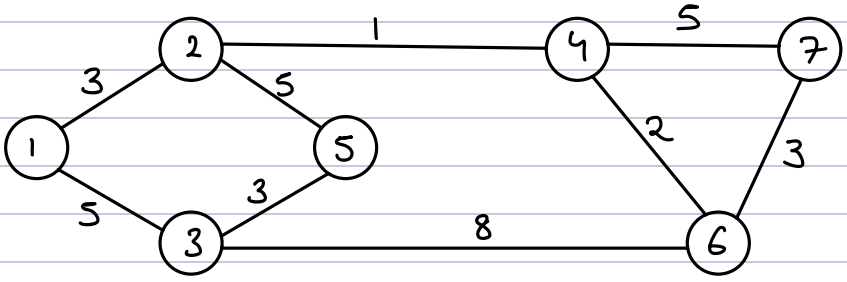
1 → {2, 3}

2 → {3, 4, 5}

3 → {4}

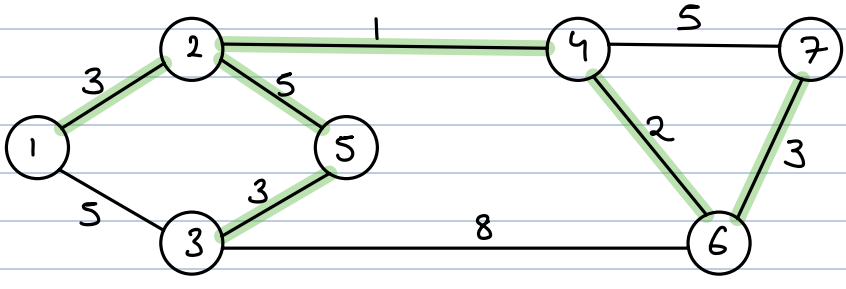
TC : $\theta(V+E)$

SC : $\theta(V)$



Q. N → No. of Flipkart centers & their possible connections with their cost to build

Find the min cost of constructing roads b/w centers such that it is possible to travel from any center to any other center.



Ans: 17

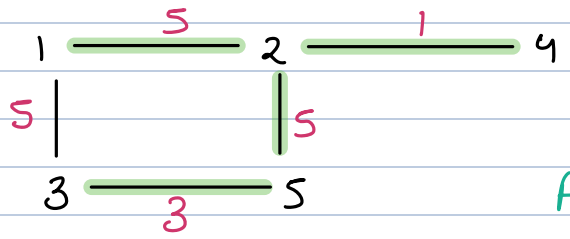
For 7 nodes → 7-1 = 6 edges

N nodes → N-1 edges

Sort of Tree
(Minimum Spanning Tree)

Minimum Spanning Tree :

N = 5



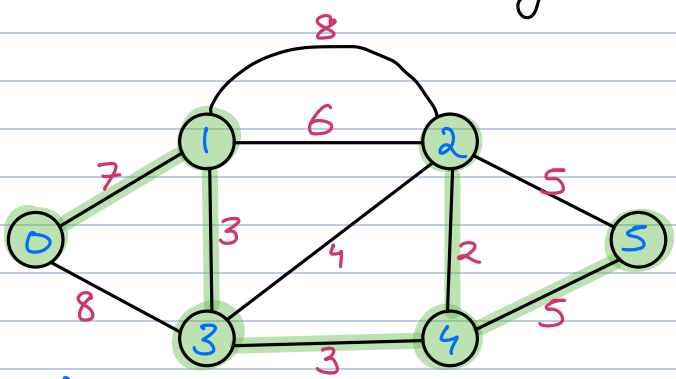
Ans : 14

Prim's Algo

* Start with any node as root, Keep adding othe nodes with minimum weight

greedy

Min Heap



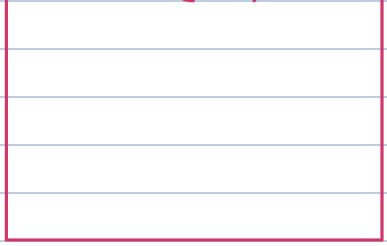
INPUT

	Pair (Vertex, wt)
0	(1,7), (3,8)
1	(0,7), (3,3), (2,6), (2,8)
2	(1,6), (1,8), (5,5), (4,2), (3,4)
3	(0,8), (1,3), (2,4), (4,3)
4	(3,3), (2,2), (5,5)
5	(2,5), (4,5)

Ans = 0+7+3+3+2+5 = 20

Temp →

(Vertex, wt)



Min Heap. (wt)

Break 10:31 – 10:38

Song : It will Rain
– Bruno Mars.

```
PriorityQueue <pair> heap;
visited [N]
visited [0] = True;
heap.insert (pair(0,0));
Ans = 0;
while (heap.isEmpty == False)
{
    pair temp = heap.poll / heap.pop;
    if (visited [temp.v] == True) Continue;
    visited [temp.v] = True;
    Ans = Ans + temp.wt; ---> Ans array for Dijkstra's
    for (pair nbr : graph [temp.v])
    {
        if (visited [nbr.v] == False)
        {
            heap.push (pair (nbr.v, nbr.wt))
        }
    }
}
return Ans;
```

```
class pair {
    int v;
    int wt;
}
```

Dijkstra's Algo

(nbr.wt + temp.wt)

TC: $\Theta(E \log E)$ → $\Theta(V + E \log E)$

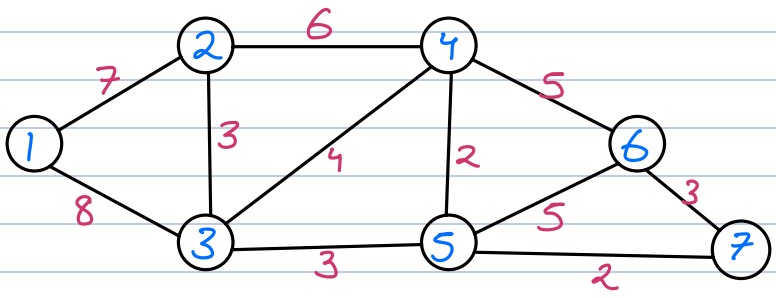
SC: $\Theta(E)$

Dijkstra's Algo

Q. There are N cities and you live in city-1. Find min dist to reach every city from city-1

Ans:

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



INPUT

Pair (Vertex, wt)

0	(1,7), (3,8)
1	(0,7), (3,3), (2,6), (2,8)
2	(1,6), (1,8), (5,5), (4,2), (3,4)
3	(0,8), (1,3), (2,4), (4,3)
4	(3,3), (2,2), (5,5)
5	(2,5), (4,5)

Ans: 0 7 12 8 11 16

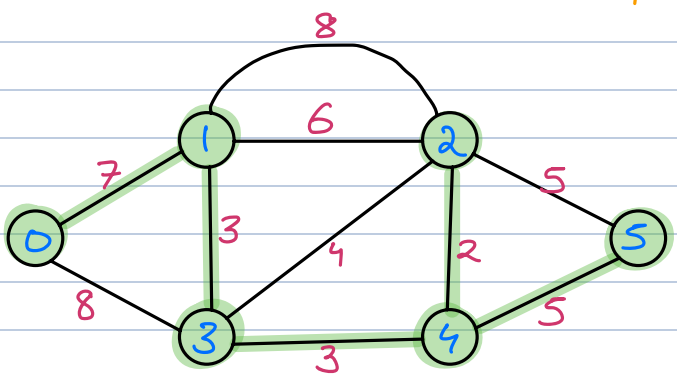
0 1 2 3 4 5

* if Ans[i] = ∞ ∴ not visited else visited

Temp

(Vertex, wt)

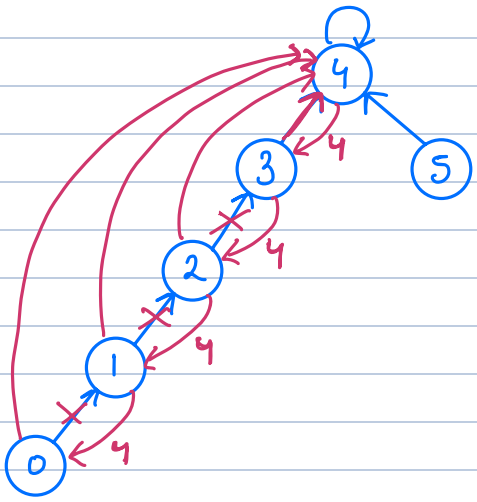
Min Heap. (wt)



Last class Notes:

Optimise DSU
- → Union By Rank → TODO → $O(\log_2 V)$
- → Path Compression → $O(1)$ (Amortized)

Path Compression



	basic approach	Path Compression
root(0)	5	5
root(1)	4	2
root(2)	3	2
root(3)	2	2
root(4)	1	1

```
int root (int x)
{
    if (x == par[x]) return x;
    r = root(par[x]);
    par[x] = r;
    return r;
}
```

TC: $\Theta(1)$ Amortized.

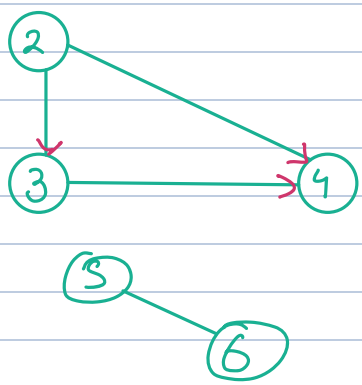
Applications DSU.

① Check if a (undirected) graph is cyclic

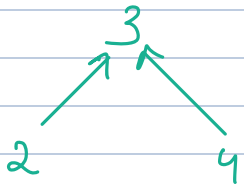
* For all nodes consider them as independent nodes / sets.
* Take union of all sets

if (union(u, v) == False) → cyclic

else → Not cyclic



(2,3)
(3,4)
(4,2)



② Check if a graph is connected or not.

* For all nodes create independent sets
* Take union of each vertex with another { union each edge given to you

if root is same for all nodes
↳ graph is connected

else → Not connected.