Madhan Kumar M S
Abhishek Sharma
amit khandelwal
Anuj chandil
Balaji S K
Bhavesh Rathod
Burhan
Dewnash
Gagan Kumar S
Gowtham
Hemant Kumar
Nikhil Pandey
Priyanka Rana
Purusharth A
Rajat Sharma
Rajendra
Sanket Giri
Saurabh Ruikar
Shani Jaiswal
shilpa mamillapalli
Shradha Srivastava
Sneha L
Sridhar Hissaria
Subhashini
Sumit Adwani
Sushant Patil
Suyash Gupta
Vasanth
Vimal Kumar
Yugesh v

Graph — 3

Content

- Challenges in Flipkart's Logistics & Delivery
- Prims Algorithm
- Dijkstra

Full syllabus → 15th May

↓

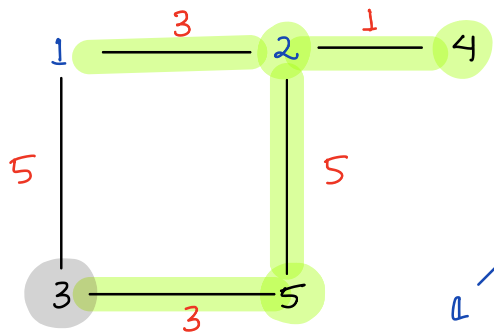
100% PSP list

Suppose Flipkart has N local distribution centers spread across a large metropolitan city. These centers need to be interconnected for efficient movement of goods. However, building and maintaining roads between these centers is costly. Flipkart's goal is to minimize these costs while ensuring every center is connected and operational.

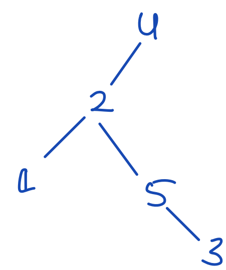
Goal: You are given number of centers and possible connections that can be made with their cost. Find minimum cost of constructing roads between centers such that it is possible to travel from one center to any other via roads.

N = 5

1	3	2
1	5	3
2	1	4
2	5	5
3	3	5

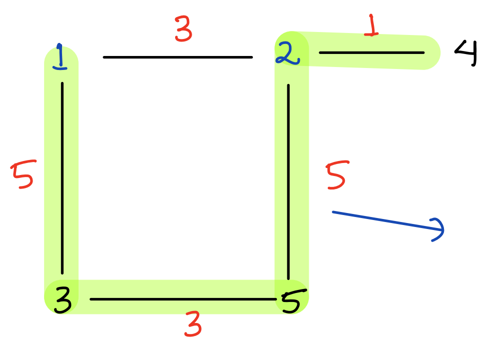


tree



cost = 12

If a graph is connected and has N-1 edges $\longrightarrow$ tree



This is a tree but not MST
cost = 14

Minimum Spanning Tree {MST}

Tree generated from a connected graph, such that

— all nodes are connected

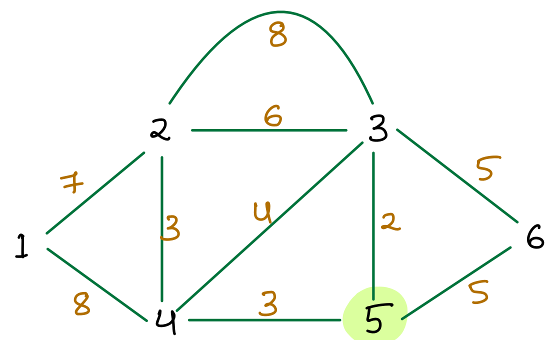
— sum of weights of all selected edges is min

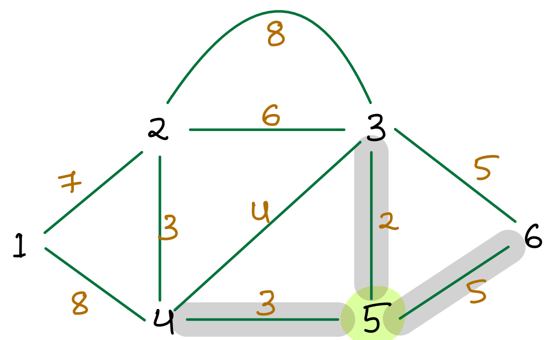
MST Algo

— Kruskal ⟶ DSA 4.2

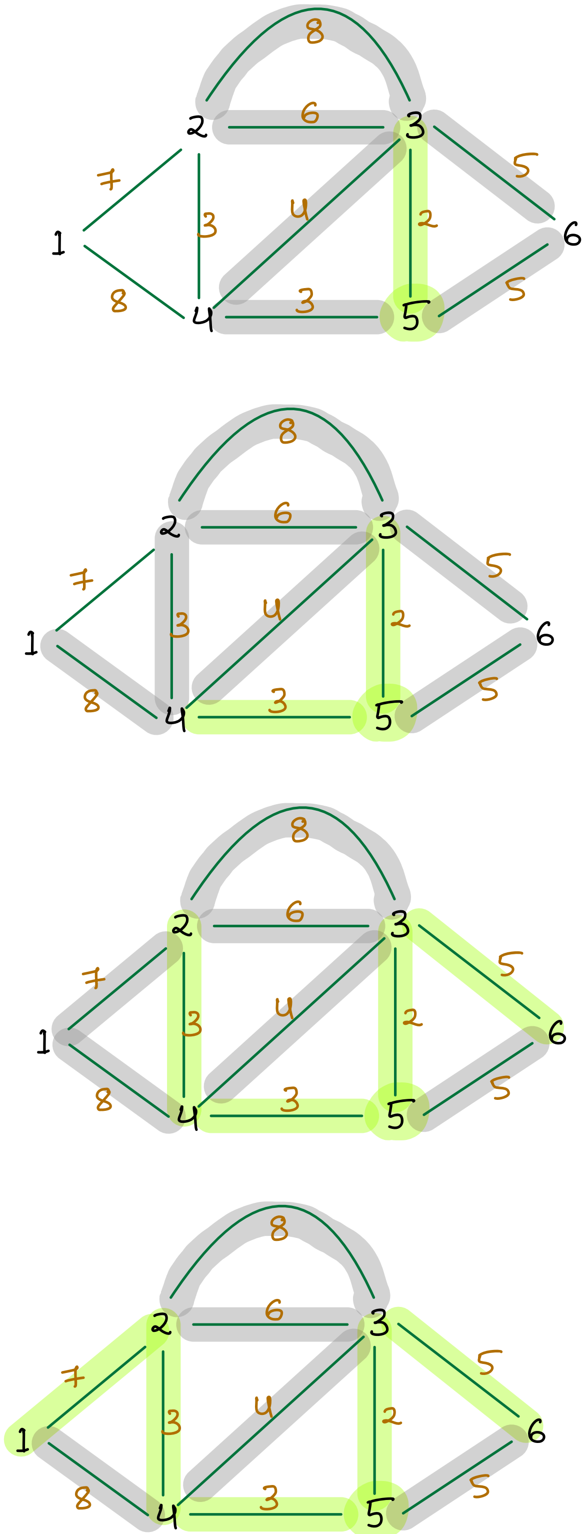
— Prims

Prims Algorithm

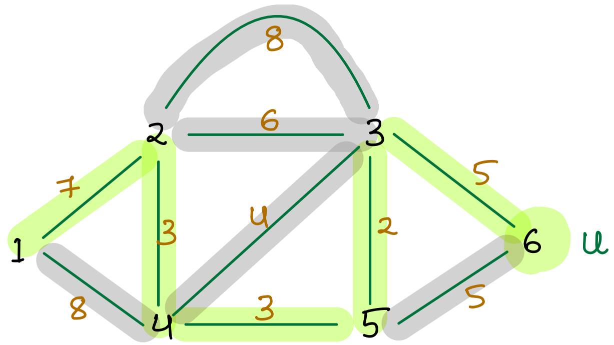




3	2	5
2	3	4
4	3	5
4	4	3
5	5	6
3	5	6
2	6	3
1	7	2
1	8	4
2	8	3



ans = 20



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

w v

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

visited →

T	T	T	T	T	T
1	2	3	4	5	6

N = 5

$1 \xrightarrow{3} 2$
$1 \xrightarrow{5} 3$
$2 \xrightarrow{1} 4$
$2 \xrightarrow{5} 5$
$3 \xrightarrow{3} 5$

AL

1 → {2, 3km} {3, 5km}
2 → {4, 1km} {5, 5km}
3 → {5, 3km}
4 →
5 →

Options → HM < Integer, AL < Pair > >
→ AL < Edge > [ ]
→ AL < AL < Pair > >
→ AL < AL < int[ ] > >

anyone is fine

Pseudocode

V ⟶ no. of nodes or vertices

E ⟶ total no. of edges

graph ⟶ AL representation

heap ⟶ to store the edges

visited ⟶ false

Build graph O(V+E)

```
// start from node 1
for {v, w} : graph[1] {
        heap.add({w, v})
}
```

weight — w ; neighbor of 1 — v

O(E log E)

```
mst = 0
visited[1] = true

while (!heap.isEmpty()) {
        w, v = heap.remove()        O(log E)
        if (visited[v]) { continue }
        mst += w
        visited[v] = true
        for {nv, nw} : graph[v] {
                if (!visited[nv]) {
                        heap.add({nw, nv})        O(log E)
        }
}

print(mst)
```

O(E log E)

TC: O(V+E + E log E)

SC: O(V+E)

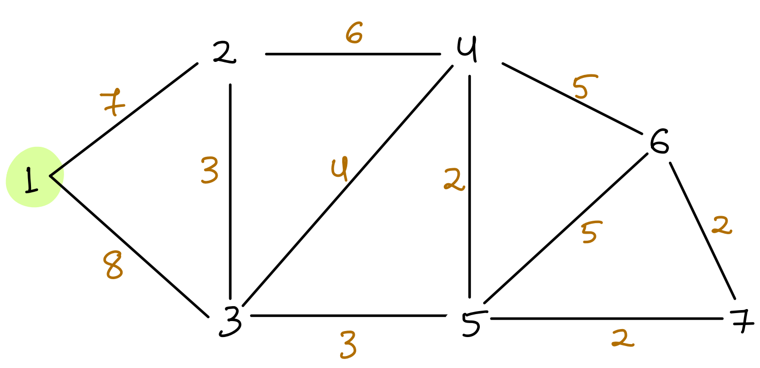
Break

10:30

Dijkstra { single source shortest path }

There are n cities in a country, you are living in city no. 1

Find min distance to reach every other city from 1

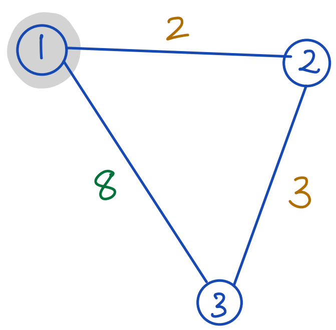


Input

output ⟶ 0 7 . . . .

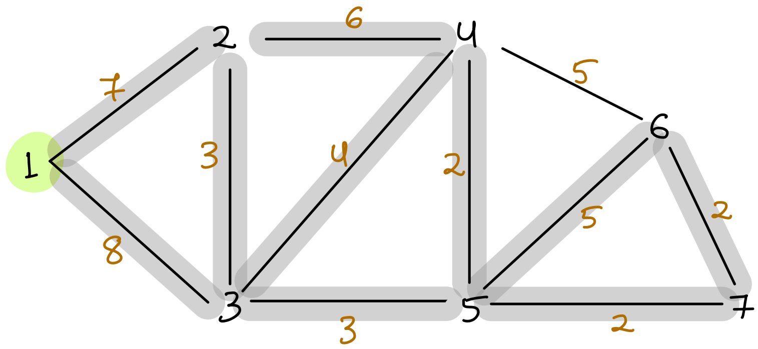
1 2 3 4 5 6 7

Single source shortest path = Dijkstra



$$\underset{8}{d[1-3]} > \underset{5}{d[1-2] + d[2-3]}$$

$$\underbrace{d[1-3]}_{\text{output}} = \underbrace{d[1-2]} + \underbrace{d[2-3]}_{\text{edge weight}}$$

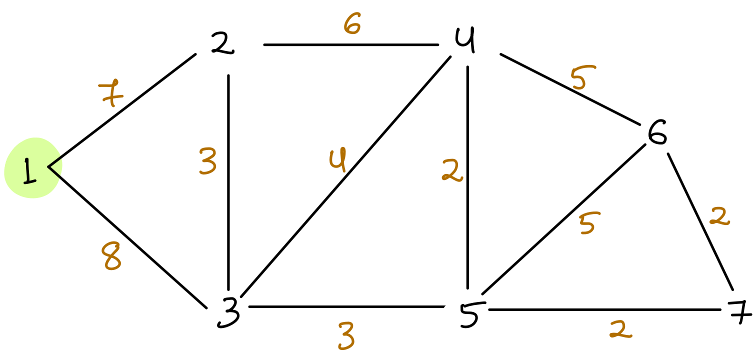


~~(16 km, 6)~~ ~~(13 km, 7)~~

$\boxed{\text{~~(7 km, 2)~~ ~~(8 km, 3)~~ ~~(13 km, 4)~~ ~~(11 km, 5)~~ ~~(12 km, 4)~~}}$

distance to city x from city 1

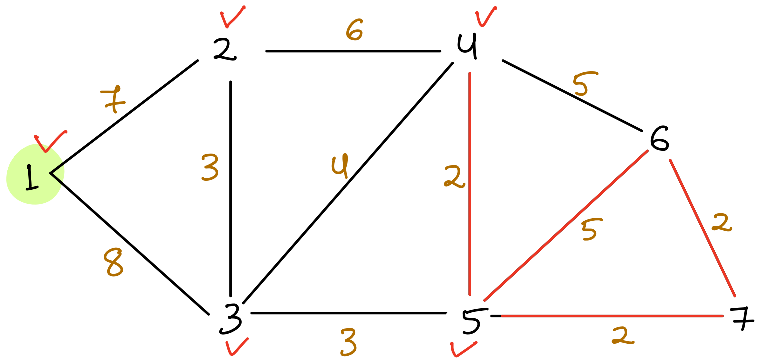
distances	1	2	3	4	5	6	7
	0	~~∞~~ 7	~~∞~~ 8	~~∞~~ ~~13~~ 12	~~∞~~ 11	~~∞~~ ~~16~~ 15	~~∞~~ 13



1	2	3	4	5	6	7
0	7	8				

min heap $\longrightarrow$ ~~(10 km, 3)~~, (13 km, 4), (11 km, 5)

Note : Normal BFS doesn't provide the correct ans ∵ edge weights are different.



~~(16 km, 6)~~
~~(13 km, 7)~~
~~(15 km, 6)~~

distances							
	0	~~∞~~ 7	~~∞~~ 8	~~∞~~ ~~13~~ 12	~~∞~~ 11	~~∞~~ ~~16~~ 15	~~∞~~ 13
	1	2	3	4	5	6	7

Pseudocode

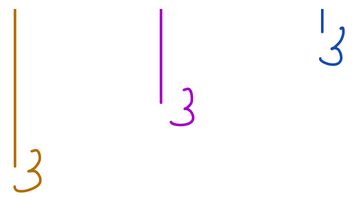
TC: O(V+E)

```
graph ⟶ AL representation   } TC: O(V+E)
heap ⟶
d[N+1] = ∞ ...... // all values as inf }
d[start] = 0                            O(V)

        distance         city
heap.add ( {0, start} )   } O(1)

                                         O(E log E)
while (!heap.isEmpty()) {
    dist, city = heap.remove()

    for {wei, ncity} : graph[city] {
        ndist = dist + wei
        if (d[ncity] > ndist) {
            d[ncity] = ndist
            heap.add ({ndist, ncity})
```



return d

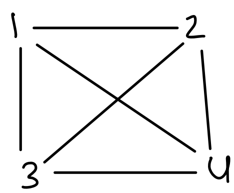
TC: $O(V + E + E\log(E))$ $\longrightarrow$ correct

$E\log(V)$ $\longrightarrow$ correct

In the worst case no. of edges $\longrightarrow$ V^2

replace $E\log(E)$ $\longrightarrow$ $E\log(V^2)$

$2E\log(V)$



$${}^nC_2 = \frac{n(n-1)}{2} = \approx O(n^2)$$

Doubt Session

$N = (V + E)$

$D \longrightarrow O(N \log N)$