Kruskal's algorithm (Minimum Spanning Tree)

(Finds an edge of least possible weight that connects any two trees in a forest. It is a greedy algorithm, finds MST)

Problem 1:

Node, Point, Vertex, city

Edge, Arc, Bridge Road, Line

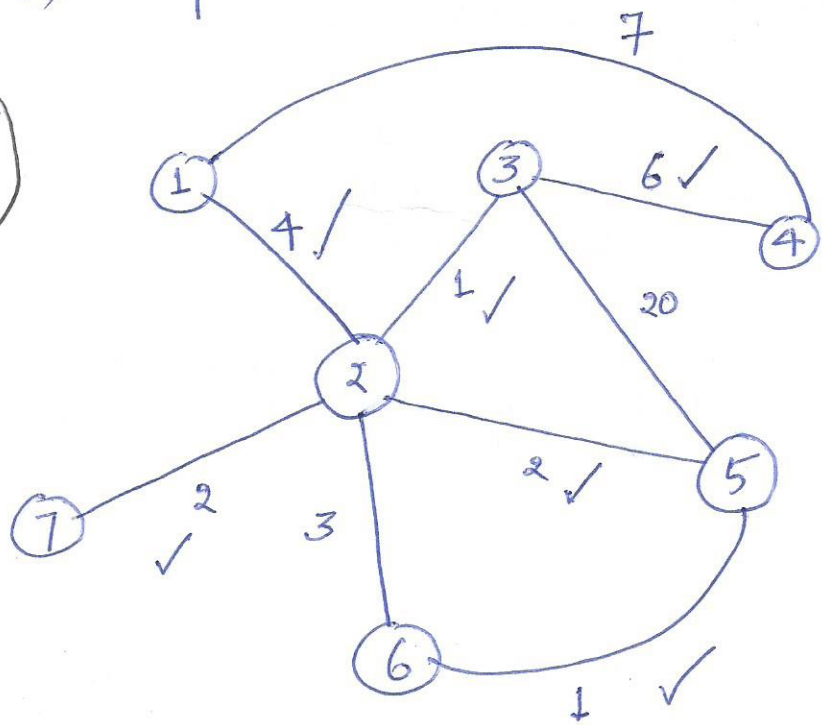
Let there be N nodes in our graph indexed from 1, 2, ..., N.

So, MST has (N-1) edges.

Rules

1) Sort list of edges based on their weight.
2) Include top edge (of the list) that is not included in MST and does not create a cycle.
3) Repeat step 2 until the MST has (N-1) edges

Problem 1



Node A	Node B	Weight
1	2	4
7	2	2
6	2	3
5	3	20
4	3	6
1	4	7
2	5	2
2	3	1

Step 1 Sort list of edges based on weight

&

Step 2:

Node A	Node B	Weight	
5	6	1 ✓	
2	3	1 ✓	Select
2	5	2 ✓	
2	7	2 ✓	
2	6	3 X	← The edge 2-6 is going to create a cycle! So, do not select it.
2	1	4 ✓	
2	4	6 ✓	
3	4	7 X	} These will create cycles! So, do not select it
1	4	20 X	
3	5		

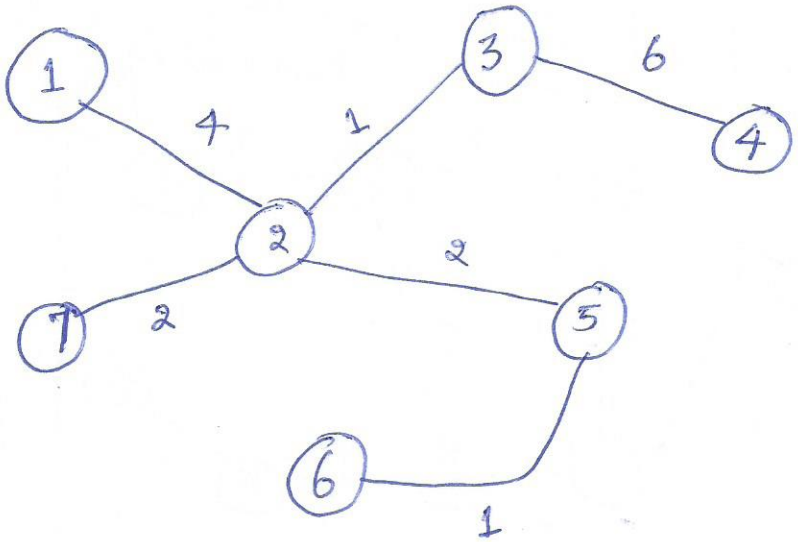
Problem!

MST Edges = 6

So, we have

Step 3:

Problem!

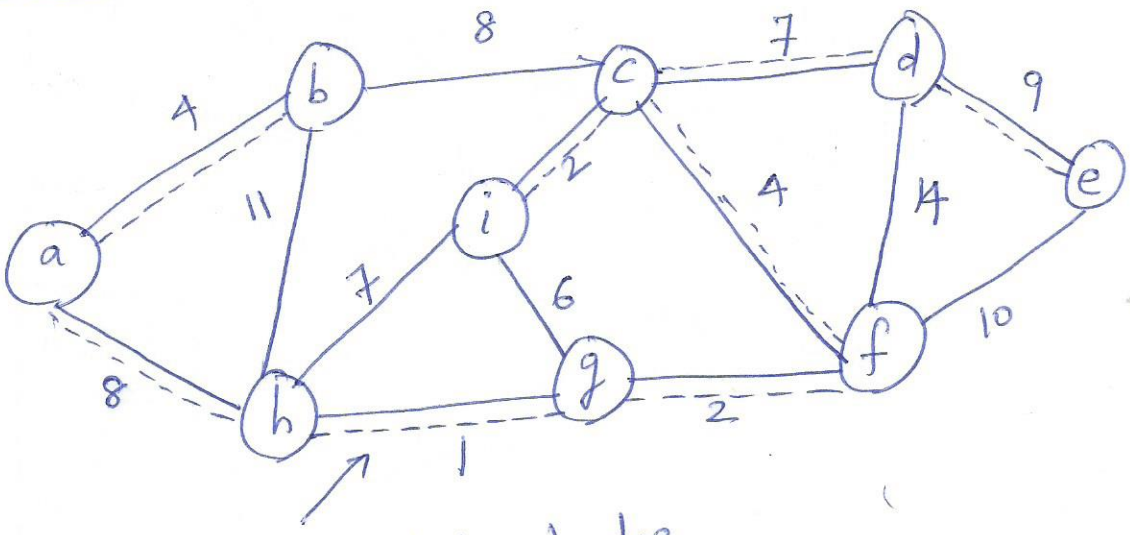


MST is ready!

Problem 1

Node A	Node B	Weight
5	6	1
2	3	1
2	5	2
2	7	2
2	1	4
3	4	6

Problem 2

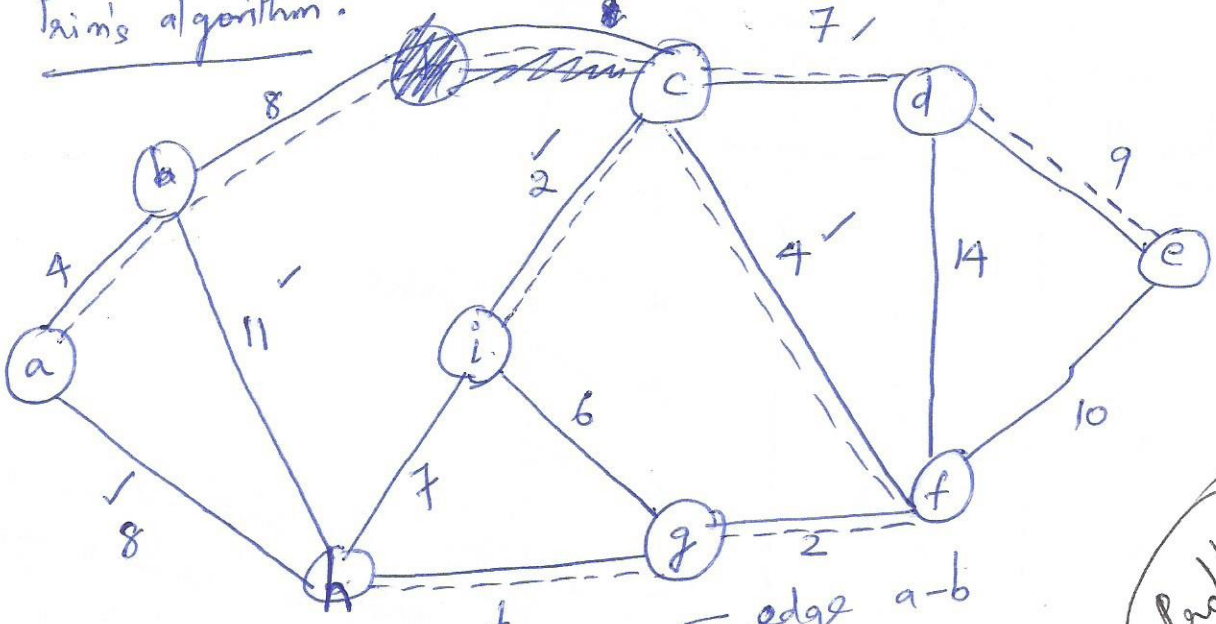


Select least cost edge
Select least cost edge which do not form a cycle

Nok: Choose edges which do not form a cycle

Runs in $O(e \lg v)$ with your binary heaps

Prim's algorithm:



Problem 2

- Start with arbitrary edge — edge a-b / edge a-h
- Pick the least costly edge - choose a to b
- Select edges such that there are no cycles.

Done!

Steps:

1	a-b	4
	b-c	8
2		
	c-i	2
3		
	c-f	4
4		
5	f-g	2
6	g-h	1
7	h-i X	
	c-d	7
8	d-e	9

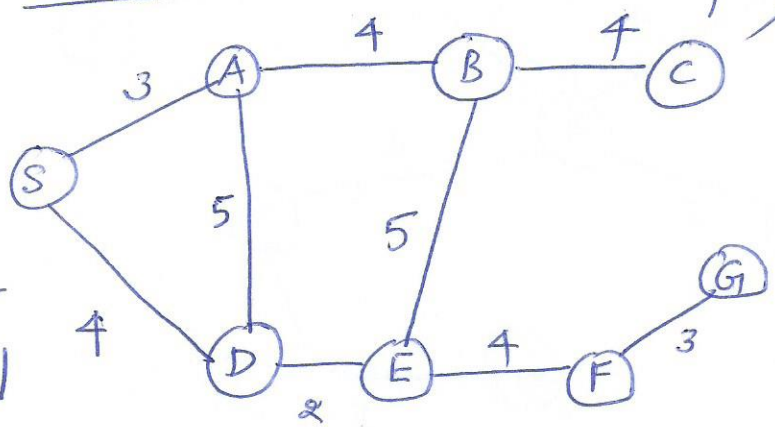
9

Problem 3

Search algorithms:

Problem 3:

Uninformed Search (can lead to backtracking the path)



S → Start

G → Goal