CSE221 ALGORITHMS

Topic: Minimum Spanning Tree

Prepared by:
Saad Bin Sohan

BRAC University

Email: sohan.academics@gmail.com

GitHub: https://github.com/saad-bin-sohan

Minimum Spanning Tree

Topic: Algorithms to find out Minimum Spanning Tree from a given graph

Spanning Tree

Spanning tree is a tree derived from a graph that contains all the vertices of that graph but will have (n-1) edges.

A graph will be a tree if it fulfills three conditions:

i) All Vertices will have to be Connected somehow

ii) Cycle can't exist

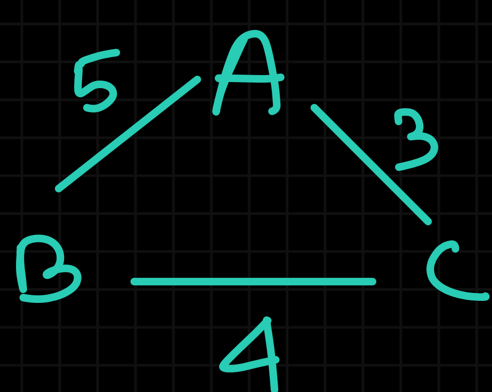
iii) (n-1) edges থাকবে, where n= number of edges making a spanning tree from a graph:

From any given graph, multiple spanning trees can be created. We will find the minimum one.

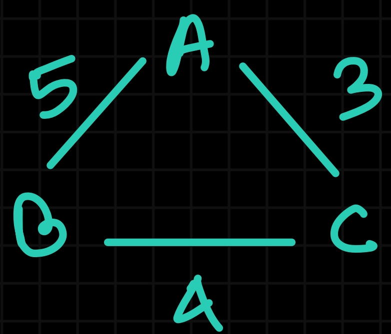
Process:

→ Take all the vertices and $(n-1)$ edges out of all the edges (and make sure no criteria, of a spanning tree, is broken)

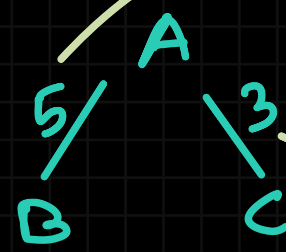
e.g.



From this graph, find the minimum spanning tree soln :

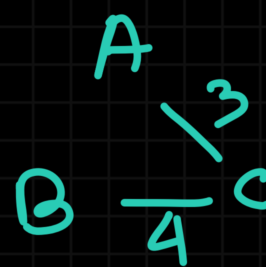


spanning tree-1



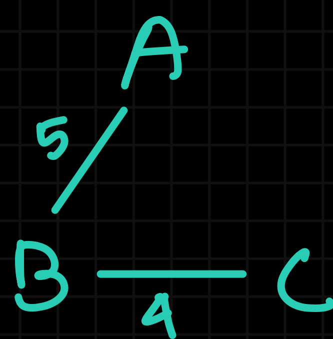
cost = 5+3
= 8

spanning tree-2



cost=3+4
=7

spanning tree-3



cost=5+4
=9

The tree with the minimum cost is called the Minimum Spanning Tree

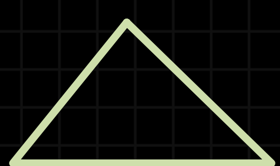
Spanning Tree 2 cost minimum (7). so thats our MST (min spanning tree)

But the above mentioned process is inefficient because it's expensive.

Reason: A complete graph of n vertices has $n^{(n-2)}$ spanning Trees.

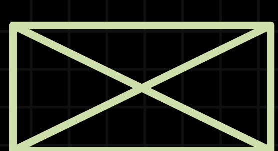
A complete graph is where all the vertices are connected to each other directly

e.g.



3 vertex এর complete graph

$3^{3-2} = 3$ টি spanning ।



4 vertex এর complete graph

$4^{4-2} = 16$ টি spanning tree

5 vertex এর complete graph

$5^3 = 125$ টি spanning tree possible

so we find an algorithm to find MST efficiently

→ Kruskal Algorithm

→ Prim Algorithm

Kruskal Algorithm

Steps:

1) Sort edges in ascending order

2) (n-1) times loop চালাও (can occur more than (n-1) times)

→ Take the minimum edge into the MST

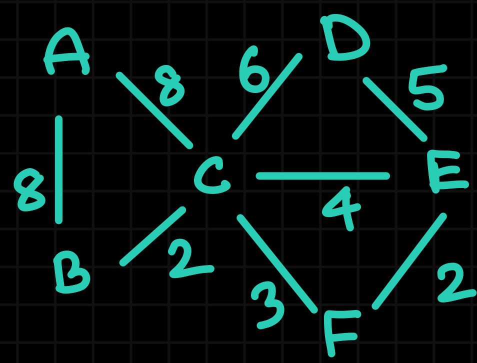
→ Make sure no cycle occurs (Trees have no cycles)

if cycle occurs pick next minimum

→ if (n-1) edges are taken, break the loop

NB: multiple edge এর weight same হলে pick any random one

Que



MST বের করো using Kruskal's Algorithm.

<u>soln:</u>

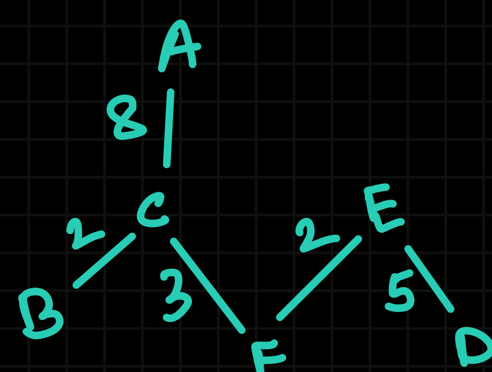
2 (BC), 2 (EF), 3 (CF), 4 (EC), 5 (DE), 6 (CD), 8 (A,C), 8 (AB)

MST cost,

$= 2+3+8+2+5$

$= 20$

∴ 20 এর কম cost এর spanning tree, graph টি থেকে পাওয়া যাবে না।



MST

NB: CF এর পর EC নিলেই তো CFE cycle create হয়ে যেত। AC এর পর AB include করলে তাহলে

যে (n-1) সংখ্যক বা 5টা edge নেওয়া already done. so এখন যত edge ই থাকুক না কেন, আর নিবো না MST তে

Disjoint Set Union

→ Disjoint Set Union method এর মাধ্যমে Kruskal's Algorithm এর কাজ করা যায়।

Process of Disjoint Set Union

→ প্রথমে সবগুলো vertex কে $\underbrace{\text{আলাদা}}_{\text{disjoint}}$ set এ রাখে।

→ then Kruskal এর idea implement

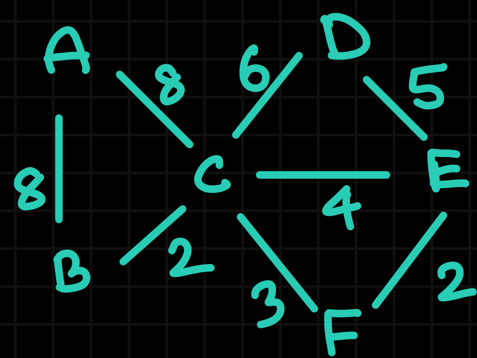
→ sort edges in terms of weight

→ sorted edge গুলো থেকে

loop আকারে edge serially pick করতো

→ each edge MST তে include করতো and নতুন একটা set এ ঐ edge গুলো যে set এর, তাদের Union set নিতো। and included edge গুলোকে calculation থেকে বাদ দিতো।

Que:



MST বের করো

Sol^n:

2 (BC), 2 (EF), 3 (CF), 4 (EC), 5 (DE), 6 (CD), 8 (A,C), 8 (AB)

$S_1 = \{A\}$

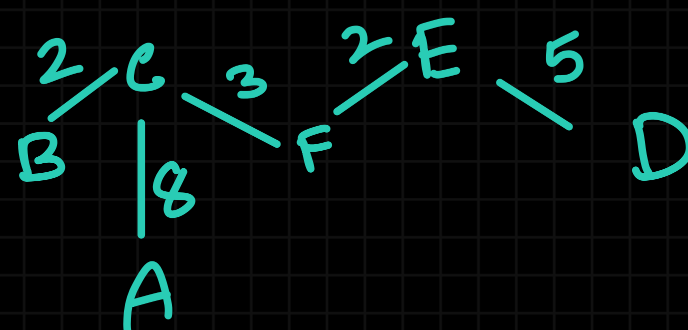
$S_2 = \{B\}$

$S_3 = \{C\}$

$S_4 = \{D\}$

$S_5 = \{E\}$

$S_6 = \{F\}$



$S_7 = S_2 \cup S_3$

$= \{B, C\}$

$S_8 = S_5 \cup S_6$

$= \{E, F\}$

(C, F) এর জন্য → $S_9 = S_7 \cup S_8$

S_3 বা S_6 নিইনি $= \{B, C, E, F\}$

cz ওরা calculation থেকে বাদ।

ওরা এখন যেখানে আছে

(S_7, S_8) তাদের নিলো।

দুইয়ের মাঝে F, C add করতে গিয়ে দেখি

যেহেতু তারা একই set এ (S_9) আছে

আর add করলো না to avoid cycles.

$$S_{10} = S_4 \cup S_9$$

$$= \{D, B, C, E, F\}$$

$$S_{11} = S_1 \cup S_{10} = \{A, B, C, D, E, F\}$$

সব element চলে আসছে। তাই loop break.

Prim Algorithm

Key difference with Kruskal Algo:

→ Prim doesn't sort all the edges at first step like Kruskal

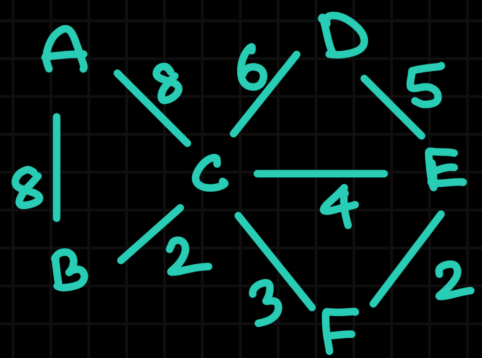
Process:

→ pick a random vertex

→ start loop

1) Take the minimum edge from start vertex
2) If cycle occur, go for the next step
3) (n-1) edge নেওয়া হয়ে গেলে loop break

Que



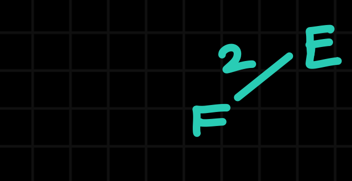
MST বের করো using Prim's Algorithm.

soln:

lets start with a random vertex E

E থেকে যাওয়া possible → 2(EF), 4(EC), 5(ED)

possible গুলোর মধ্যে minimum cost EF (2) এবং।
MST তে EF add করলো, then calculation থেকে
বাদ দিলো।

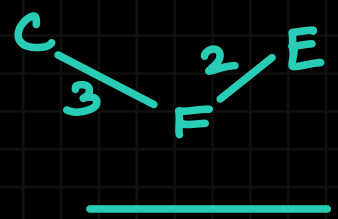


MST

then F থেকে possible (but not in MST already) edge বের করি।

F → 3 (FC)

এবার E এবং F both vertex এর সবগুলো possibility থেকে minimum cost এর edge include করলো MST তে



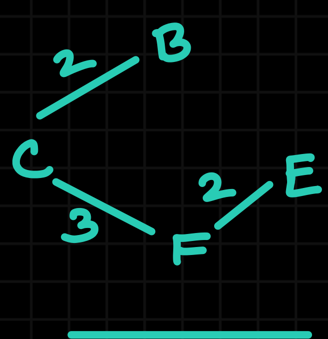
MST

C → 2 (CB), 6 (CD)

E → 4 (EC), 5 (ED)

F →

lowest হলো 2 (CB)



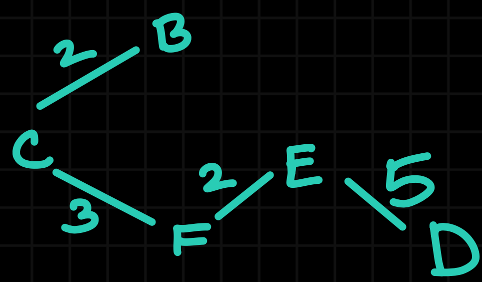
MST

E → 4 (EC), 5 (ED)

C → 6 (CD)

B → 8 (AB)

4(EC) add করা যাবে না কেন cycle not allowed.
so we add 5 (ED) এবং calculation থেকেও 4(EC) বাদ দিলো



MST

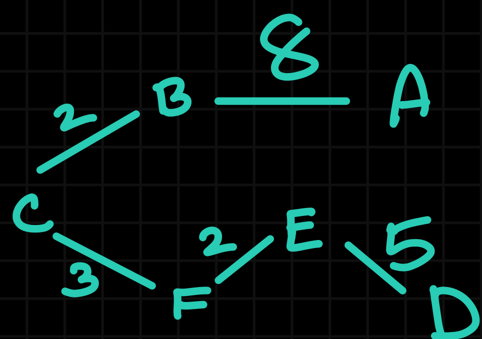
E →

F →

C → 6 (CD)

B → 8 (BA)

D →



(Ans)

MST

#Kruskal / Prim এর result same.

Important questions on difference between Kruskal & Prim's Algo

Que-1: Same graph এ both algo apply করলে কি always same MST আসবে?

Ans: NO. bcz same graph এ একাধিক edge এর weight same হলে since each algo chooses any random one first.

Yes (কখন → যখন graph টাতে each edge has different weight

Que-2 : Same graph এর জন্য both Algo তে MST cost কি always same হবে?

Answer: Yes. Same. Always whatsoever

Que-3: Disconnected graph থেকে MST পাওয়া যায় কী?

Answer: No. bcz disconnected graph can't be a tree.

Que-4: Unweighted graph এ MST কোনটা?

Answer-4: All of the possible spinning trees are MST.
Bcz each edge carry same weight.
so all ST has same cost.

Que-5: Dense graph and Sparse Graph- কোনটার জন্য Kruskal vs Prim কোনটা efficient?

Ans-5: Sparse ⇒ Kruskal (reason: Kruskal যেহেতু শুরুতেই সব sort করে, dense হলে sorting all at once টা inefficient)

Dense → Prim