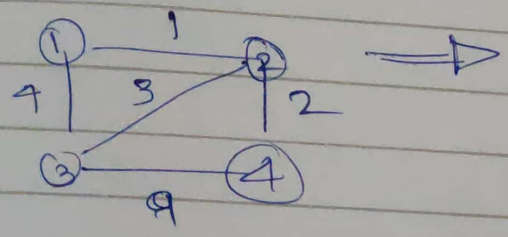
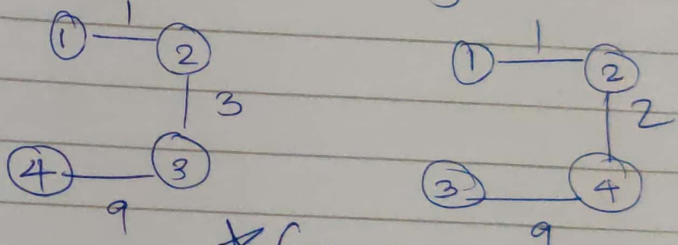
MST
Kruskal
Prims
Dijikstras
Topological Ordering — Bellman Ford

* Spanning trees is a representation of a graph in a tree like structure where all nodes are accessible when traversed from the root / start node
* ~~Doesn't work on unconnected graphs~~.
MST is a case when cost associated is the minimum from the graph



Possible spanning Trees



* (it is an acyclic graph)
etc...

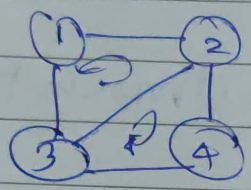
Usually Trend is
E = set of edges
V = set of vertices
G = the graph
S = spanning Tree

Spanning Tree = (V', E')

where $S \subseteq G$
$S = (V', E')$
$V' = V$
$|E'| = |V'| - 1$
no. of edges = no. of nodes - 1
for a spanning tree.

not important

No. of possible spanning trees = $^{|E|}C_{|V|-1}$ - no. of cycles



no. of cycles = 2

(1, 2, 3) (1, 3, 4)

E = 5

V = 4

V - 1 = 3

no of possible spanning trees = $^{5}C_{3}$ - 3

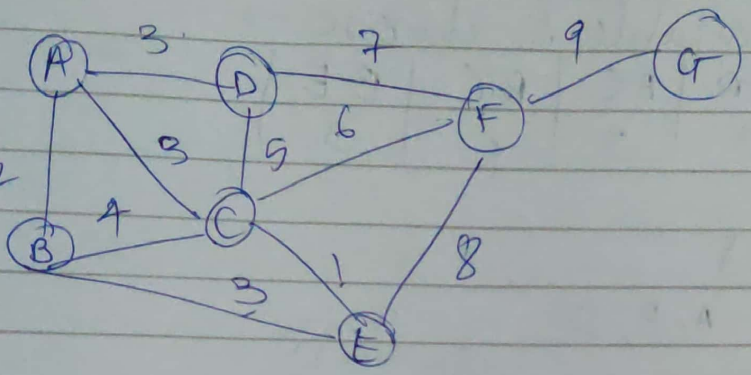
= 5 × 4 - 3

= 9

$O(|E| \log |E|)$

Prim's Algorithm

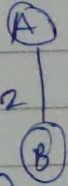
① It is a greedy algorithm
② start at an arbitrary vertice/node
③ add to the visited list
④ choose the smallest edge connected to this starting node & add the next node to visited list, pick nodes only that have not been visited
⑤ Repeat till all done
⑥ Piching is to be done based on available edges of the visited nodes in the spanning tree
⑦ Always makes sure it builds a tree



① lets start at say A
② visited = {A,
③ edges available from nodes in visited
④

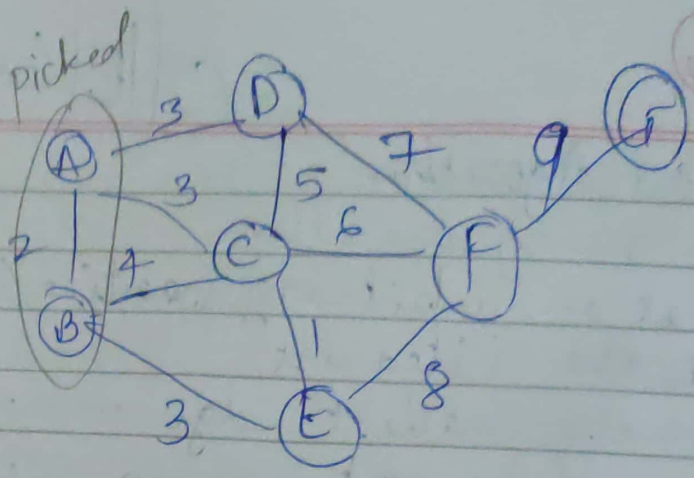
edge	AB	AC	AD
cost	2	3	3

⑤ pick least cost edge i.e AB (2)
add B to visited
⑥ visited = A, B
⑦ spanning tree →

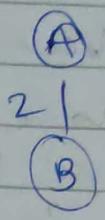


⑧ Repeat from ③ to ⑥ again till all nodes in visited

Graph →



spanning Tree =



visited = A, B

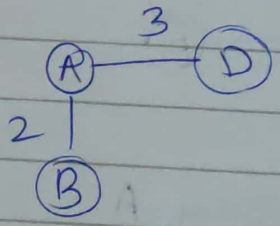
edges available to choose →

edge	AD	AC	BC	BE
cost	3	3	4	3

min( )

visited = A, B, D

spanning tree →



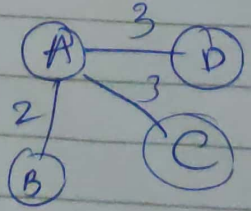
~~ed~~ edges available

edge	AC	BC	DC	BE	DF
cost	3	4	5	3	7

min

visited = A, B, D, C

ST →

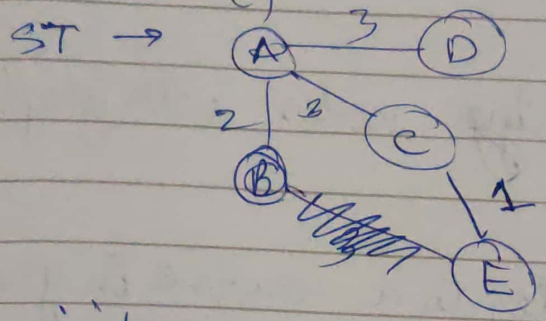


visited = A, B, D, E

edges	BE	DF	CF	CE
cost	3	7	6	1

min()

ST →



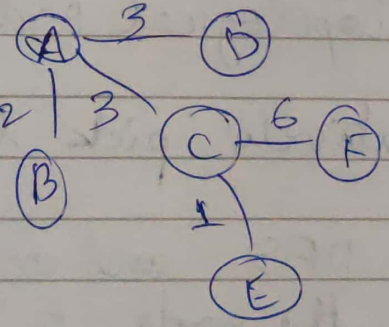
visited = A, B, D, C, E

edges	DF	CF	EF
cost	7	6	8

min

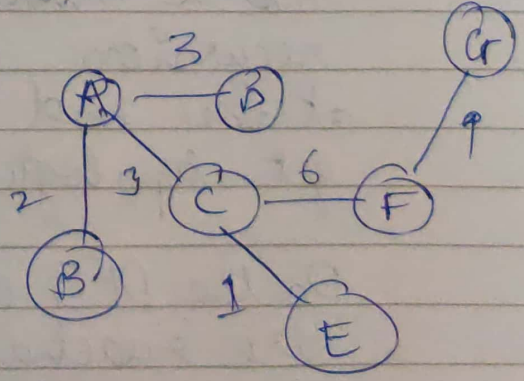
visited = A, B, D, C, E, F

ST →



edges	FG
cost	9

ST →



Total cost = 2+3+3+1+6+9

= 24 (MST cost)

It is a greedy approach as it doesn't

~~answered~~ ★ answered in KRUSKAL,

~~How different Data Structure Affects Prim's?~~

Topological ordering

Fancy way of ordering nodes with graph arrows towards right

Example ① pre requisites to a course in graph Structures

② compiling resources for a program before the actual program.

★ Doesn't work on acyclic graph.
★ Aren't always unique.
★ Every tree has a topological order.

To find it, iteratively pick leaf nodes.

Do a Recursive DFS and on return of recursion, add the node to a stack. at the end pop all elements of stack to get topological ordering.

Do the DFS only on unvisited nodes which are reachable.

Kruskals Algo.

① Pick smallest edge repeatedly to form a tree such that the new edge doesn't create a cycle.

It is a greedy approach based on sorting the edge weights.

Run time $O(E \log E)$ general

$O(e \lg v)$ with binary heap to always get the min edge.

{ Can Run faster with fibonacci heap $O(e + v \lg v)$ if cardinality of edges is smaller than edges. }

Run time is mainly affected by how fast the min edge is given

can also be $O(E \times V)$ if edges are already sorted

Algo →

```
A = ∅
for each (v node) in (graph G)
    MAKE-SET(v)

// sort the edges in ASC order.  // ?

for each (u,v) from sorted list                    // e log v
    if (Findset(u) ≠ Findset(v)
        A = A ∪ {(u,v)}
        UNION(u,v)
return A
```

Bellman Ford. [Fails when there is a cycle with negative weight]

Relaxation

for edges $w(u,v)$ of vertices u & V

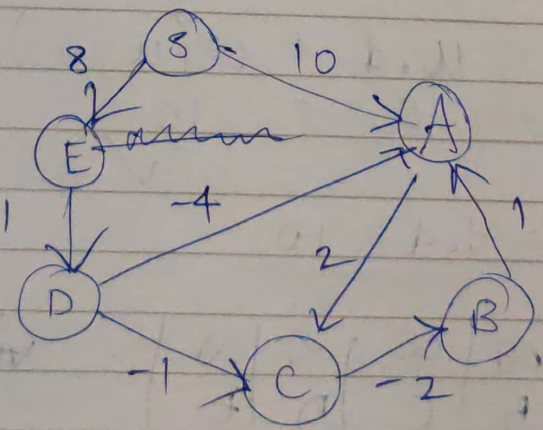
If $d[u] + w(u,v).cost < d[V]$

$\Rightarrow d[V] = d[u] + w(u,v).cost$

means if there is a better way to reach the node, calculate it and replace the old distance to that node.

It takes $|V| - 1$ iterations to find Shortest Path on a weighted directed graph using Bellman-Ford. ~~(Acyclic graphs)~~

This algo allows negative weights.



① initialize all distance from S to other nodes as ∞, S distance to itself is 0

0	∞	∞	∞	∞	∞
S	A	B	C	D	E

S = Starting node.

② Pass ①

now for $u = S$ $V = [A, B, C, D, E, F]$

Check relaxation condition

if $V.d < u.d + w[u,v].cost$

$V.d = u.d + w[u.v].cost$

$u.d$ ↑ distance to u

$w[u.v].cost$ ↑ edge cost

initially d is ∞ for all nodes V

0	∞	∞	∞	∞	∞
S	A	B	C	D	E

w has cost of all edges

Now for, let say $V = A$ & $u = S$.

edge $w(u,v)$ exists with cost 10

$\therefore V.d < u.d + w(u,v)$

$\infty < 0 + 10$ ✓

$\Rightarrow V.d = 10$

0	10	∞	∞	∞	∞
S	A	B	C	D	E

~~visited S, A~~

~~similarly do for all B, C, D, E and solve only if V is reachable with the min~~

~~end of Pass ①~~

~~we get~~

~~0~~	~~10~~	~~10~~	~~12~~	~~9~~	~~8~~
~~S~~	~~A~~	~~B~~	~~C~~	~~D~~	~~E~~

now v = B

0	10	∞	∞	∞	∞
S	A	B	C	D	E

currently we dont have a path to reach B ~~from the ∞~~ using visited nodes = ~~S~~ S, A

move on to v = C no edge available from u

~~visited = S~~

~~we have a path to C using 'A'~~

~~let u = A~~

~~∴ is v.d ≤ u.d + w(u, v)~~

~~∞ ≤ 10 + 2~~

~~v.d = 12~~

~~0~~	~~10~~	~~∞~~	~~12~~	~~∞~~	~~∞~~
~~S~~	~~A~~	~~B~~	~~C~~	~~D~~	~~E~~

v = D no edge,
we have another edge from u = S
for v = E

∴

0	10	∞	∞	∞	8
S	A	B	C	D	E

Now u = A.
now we can Reach C

∴

0	10	∞	12	∞	8
S	A	B	C	D	E

No more edges
let u = B, we have edge BA but we don't know how to reach B, so skip

Still Pass (1)

now u = C

0	10	∞	12	∞	8
S	A	B	C	D	E

we have edge CB = -2

⇒

0	10	10	12	∞	
S	A	B	C	D	E

now u = D

∴ D.d = ∞ skip

u = E

we can reach D with ED = 1

⇒

0	10	10	12	9	8
S	A	B	C	D	E

Pass (1) done

Pass (2)

u = A

no improvement

u = B

no improve

u = C no improve

u = D

DA = -4 improves cost

9-4

0	5	10	12	9	8
S	A	B	C	D	E

DC = -1

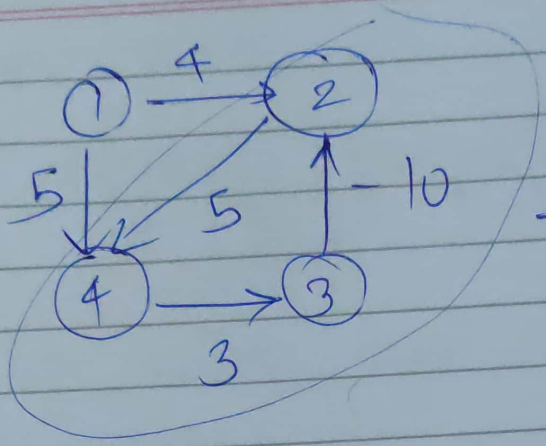
9-1

0	5	10	8	9	8
S	A	B	C	D	E

u = E

no improve

(*) Repeat for Pass (3) and so on till no changes occur ~~add~~ at all in the array



→ it is a cycle

weight = -10+5+3
= -2
< 0

Causes relaxation even after (V-1) iterations which breaks the Bellman ford algo's foundation.

V = no. of nodes