Minimum Spanning Trees

Given:-
undirected Graph: $G=(V,E)$ $(u,v) \in E$ (edge)
$w(u,v)$ (weight)

Goal:
Total weight
$w(T) = \sum_{(u,v) \in T} w(u,v)$ is minimized

↓ acyclic Graph ⇒ Tree [Spanning Tree]

Run Time: $O(E \lg V)$ (Kruskal ; Prims)
↑ using binary heap.

Greedy Approach : choice (best at the moment)
⇒ Such strategy is not generally guaranteed to find global optimal soln. to a problem.

Generic MST (G,w) "Generic Algo grows minimum Spanning Tree one edge at a time"
$A \leftarrow \emptyset$
while A doesnot form a spanning tree.
do find an edge (u,v) that is safe for A.
$A \leftarrow A \cup \{(u,v)\}$.
return A.

A : subset of some MST. (initially empty)
G : Given graph. w : edge weight.

" determine an edge (u,v) can be added to A, without violating " A is a subset of MST " → $A \cup \{(u,v)\}$ is also a subset of MST.
Added edge is safe edge.

* No cycle formed

Safe edge rule :

Kruskal's → Set A is a forest
Safe edge added to A is always a least weight edge in graph that connects two distinct components

'Prim's. Set A forms a single tree
Safe edge added to A is always a least weight edge connecting the tree to a vertex not in the tree.

```
MST Kruskal (G, w)
1  A ← ∅
2  for each vertex v ∈ V[G]
3       do Make_Set (v)
4  Sort the edges of E into non decreasing order by weight w
5  for each edge (u,v) ∈ E taken in non dec. order by weight
6       do if Find Set (u) ≠ Find Set (v)
7            then A ← A ∪ {(u,v)}
8                 UNION (u,v).
9  return A
```

working

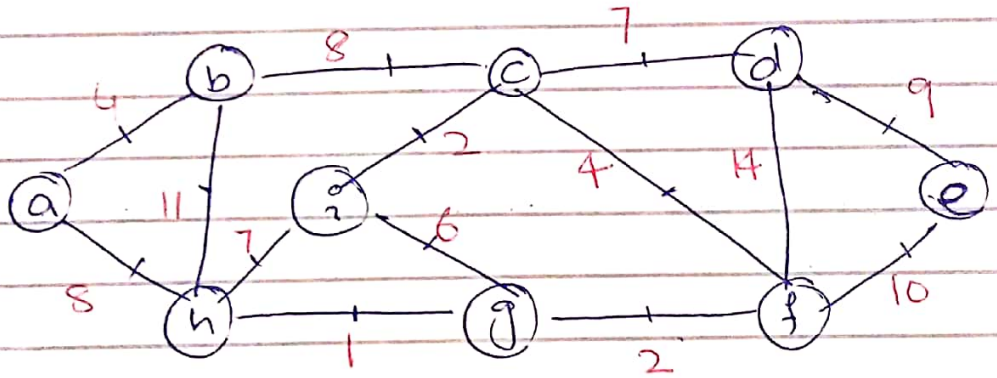
line 1 — 3 initialize set A. to empty set.
create |V| tree (tree containing one vertex)
⇒ # of tree = no. of vertices

line 4 sort edges in terms of weight in increasing order.

line 5-8 pick edge (u,v) ; if 'u' and 'v' (edge endpoints) belong to same tree Yes? donot add the edge to forest

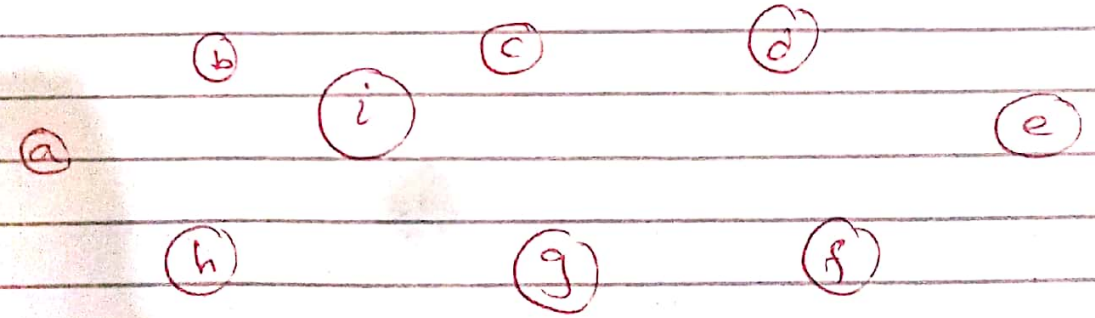
else 'v' and 'u' belong to different trees in forest add the edge and vertices are merged.

Graph (G, w).



(1) A = ∅ or { }

(2) |V| ⇒ no. of vertices = 9
Create 9 trees

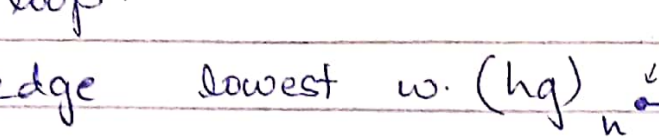


forest of trees.

(3) non-dec. order of edges by weight.
(hg) (gf) (ic) (cf) (ab) (id) (cd) (bc) (ah) (de) (fe) (bh)
(ia) (hi)

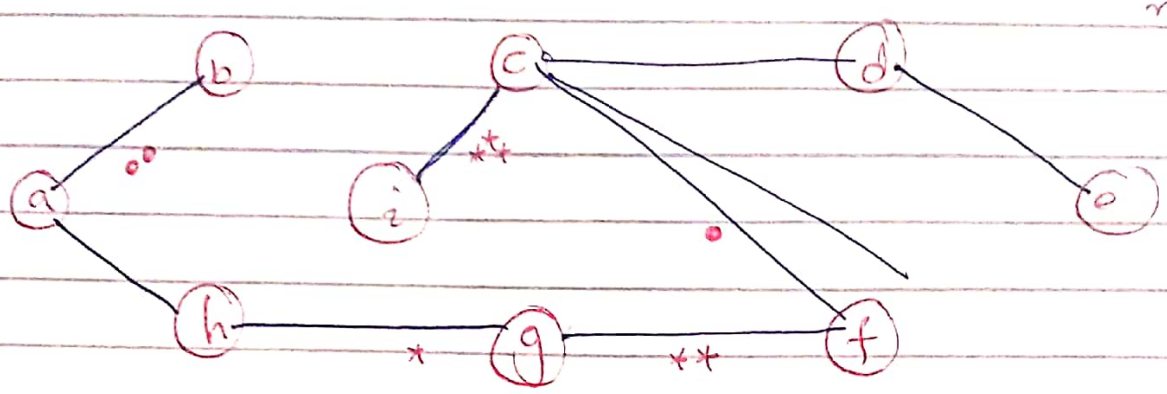
(4) for loop.

pick edge lowest w. (hg) → belong to diff trees

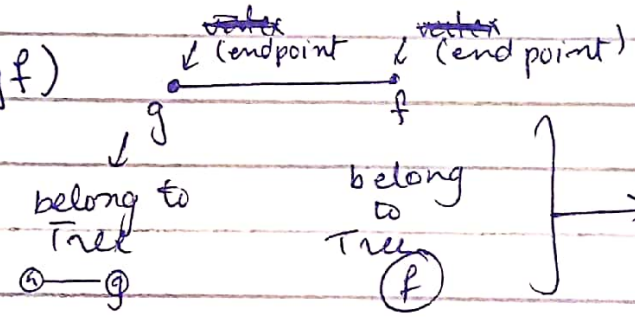


Yes → add.*

merge Vertex {H, G}



(5) repeat edge (gf)



→ diff Trees

add **

Merge Vertex {G, H, f}

edge (ic) → endpoints belong to different trees.

add ***

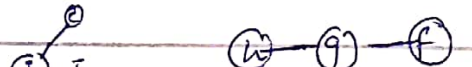
{I, C, G, H, f}

edge (cf)

endpoint belong to diff trees

Yes. {I, C, G, H, f}

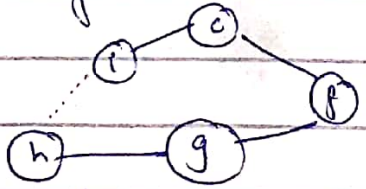
edge (ab)



endpoint diff tree → add. {A, B, I, C, G, H, f}

edge (ig)

belong to same tree.



→ do not add.

Repeat the same for all edges. To get MST. ....

A = {A, B, H, G, f, C, I, ...}

Analysis

Line 1 initiallization. $O(1)$ const time.

Line (2–3) MakeSet op. in for loop $O(V+E)\ a(V))$ — for all vertices

Line 4 $O(E \lg E)$ Sorting edges.

Line 5–8 #5 for all edge $O(E)$
- Find Set
- Union Op.

$O(E\ a(V))$ — slow growing func and $a(V) = O(\log V) = O(\log E)$

$|E| \geq |V|$

$O(E \log E)$

Total time $= O(1) + O(E \log E) + O(E)$

$\Rightarrow O(E \log E)$

Since $E < |V|^2$ so $\lg E = O(\lg V)$

Total time $= O(E \log V)$

Prims Algo:

```
MST Prims (G, w, r)
 for each u ∈ V[G]
     do key[u] ← ∞
        π[u] ← NIL
 key[r] ← 0
 Q ← V[G]
 while Q ≠ 0
     do ← u ← Extract Min (Q)
        for each v ∈ Adj[u]
            do if v ∈ Q and w(u,v) < key[v]
               then π[v] ← u
                    key[v] ← w(u,v).
```

Working

Line 1–5 (1,2) Set key of each vertex to ∞ except for root 'r'

set (r) (4) key to 0.

(initial vertex)

↓

first vertex to be processed

(3) parent of each vertex is NIL

(5) initialize Min Priority Queue.

Line 6–11 (6) while loop.

(7) identify vertex 'u' incident of lightest edge.

remove this vertex 'u'

add it to the tree vertices in

(8)–(11) update key

and π of every vertex v adjacent to u but not in tree

(u)

Analysis Based on ~~binary~~ min P_Queue.

↳ implemented using binary heap then.

Line 1–5 takes O(V) time. (for all the vertices in Q)

Line 6 while loop executes |V| times.

Line 6–7 under 'top' Extract Min takes O(lgV) time.

Total time O(V lg V)

Line 8–11 for loop executes O(E) times.

length of adj. list = 2|E|

Line (9) constant time (testing cond).

line 11 key operation on min heap (binary)

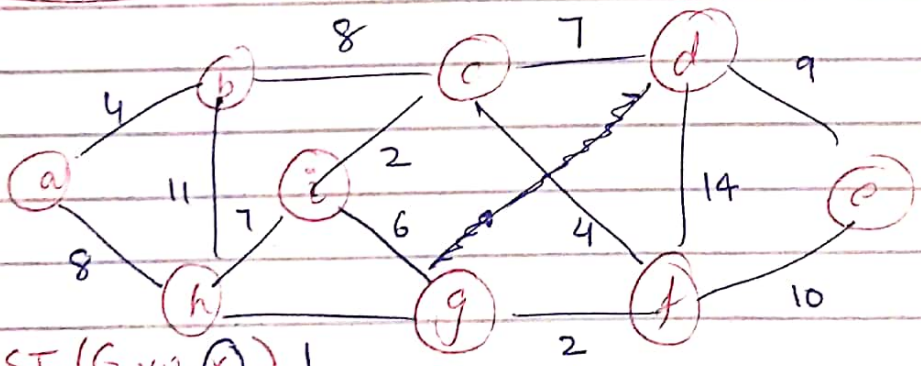
$O(\lg V)$ time

$$O(V) + O(V \lg V + E \lg V) = O(E \lg V)$$

* Prims Algo can be improved using fib Heaps' Fn for P-Q

$O(E + V \log V)$

Execution



MST (G, w, r) 1

line 1 – 3 // for loop (for each vertex)

a initial vertex

key[u] ← ∞

key[a] ← ∞
π[a] ← NILL
key[b] ← ∞
π[b] ← NILL
key[c] ← ∞
π[c] ← NILL
key[d] ← ∞
π[d] ← NILL
key[e] ← ∞
π[e] ← NILL
key[f] ← ∞
π[f] ← NILL
key[g] ← ∞
π[g] ← NILL
key[h] ← ∞
π[h] ← NILL
key[i] ← ∞
π[i] ← NILL

line 4 key[r] ← 0

key[a] ← 0

Line 5 → Q: Min Priority Queue.

{a, b, h, c, i, d, e, f, g}
0 ∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞

Line 6 → for all vertices in Q.

u ← extract Min

u ← a // (min key value)

each v ∈ adj[u]

for each adj vertex. → v = b, h ← adj[a]

do if v ∈ Q and w(u,v) < key[v]
b ∈ Q w(a,b) < key[b]
4 < ∞

π[v] ← u

π[b] ← a // parent of b will be a

key[v] ← w(u,v)

key[b] ← 4.

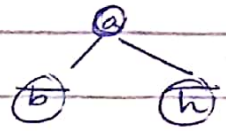
do if v ∈ Q and w(u,v) < key[v].
h ∈ Q w(a,h) < key[h]
True 8 < ∞

π[v] ← u

π[h] ← a // parent of h will be a

key[v] ← w(u,v)

key[h] ← w(u,v)
8



Again extract Min from PQ. {b, c, h, d, e, f, g, i}
4 ∞ 8 ∞ ∞ ∞ ∞ ∞

u ← b (min from Que)
key value.

repeat. Repeat the same steps.