DELTA Page No. Date:

Minimum Spanning tree

Consider a connected undirected graph $G = (V, E)$ where each edge $(u, v) \in E$ has a weight $w(u, v)$. We wish to find an acyclic subset $T \subseteq E$ that connects all vertices and whose total weight

$$w(T) = \sum_{(u,v) \in T} w(u, v) \text{ is minimum}$$

This tree is acyclic and connects all vertices, and spans graph G, so it is called minimum spanning tree.

Growing a minimum spanning tree

- Assume we have a connected undirected graph $G = (V, E)$ with a weight function $w : E \to R$ and we wish to find a minimum spanning tree for G. The algos used to find MST have a greedy approach.

- Greedy approach is captured by the generic algorithm

→ Maintainance: maintains invariant by adding only safe edges.

Page No.
Date:

Generic-MST (G, w)
1 A ← ∅
2 while A does not form a spanning tree
3 do find an edge (u, v) that is safe for A
4 A ← A ∪ {(u, v)}
5 return A

↳ Termination → A returned is MST

- The algo has a set A maintaining loop invariant that prior to each iteration, A is subset of some MST.
- At each step, a safe edge (u, v) can be added to A without violent the invariant s.t. A ∪ {(u, v)} is also subset of MST.

The algos of Prim + kruskal are used to find this safe edge.

A cut (S, V−S) of an undirected graph (V, E) is a partition of V.

A edge (u, v) ∈ E crosses the cut (S, V−S) if one of the endpoints is in S and other in V−S.

Let G =
weight
that is
cut that

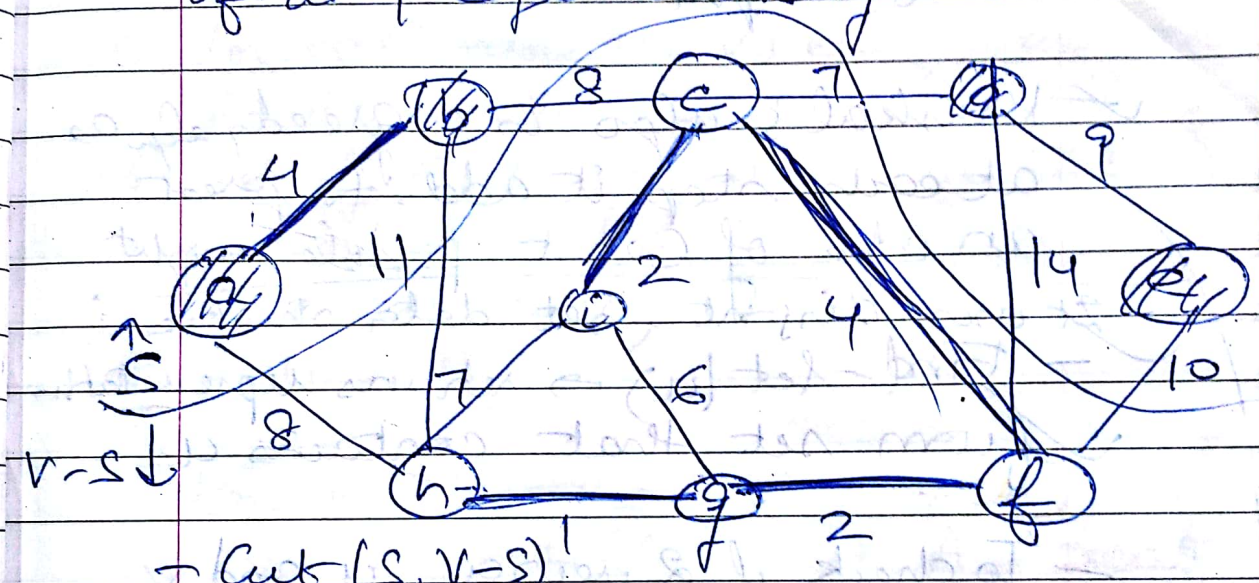
A
f
A
a
of

S
V−S

Let $G=(V,E)$ be connected, undirected graph with weight f 'w' on E. Let A be subset of E that is included in MST, let $(S, V-S)$ be any cut that respects A + let (u,v) be light

A cut respects a set A of edges if no edge in A crosses the cut.

An edge is a light edge crossing a cut if its weight is minimum of any edge crossing the cut.



- Cut $(S, V-S)$
- Vertices in set S are shown shaded and those in $V-S$ are shown white
- Edges crossing cut are those connecting black with white vertices. (eg. (b,h), (d,f) etc.
- Edge (d,c) is light edge crossing cut.
- A subset A is shaded, the cut $(S, V-S)$ respects A, since no edge of A crosses the cut.

edge crossing $(S, V-S)$. Then edge (u,v) is safe for A.

DELTA Page No. Date:

Kruskal's algo

- Here set A is a forest.
- It finds a safe edge to add to growing forest by finding all edges that connect any 2 trees in the forest an edge (u,v) of least weight
- Kruskal's algo is a greedy algo as at each step it adds to forest an edge of least possible weight.
- Uses disjoint set data structures:
  - Find-set(u) → returns representation from set that contains u.
  - To check if 2 vertices u and v belong to the same tree by testing whether Find-set(u) equals Find-set(v)
  - Combining of trees is done by union procedure
  - Make-set(v): creates a new set whose only member is pointed to by v.
  - If C_1 & C_2 are 2 trees in forest connected by (u,v). As (u,v) must be light edge connecting C_1 to some other

MST
1 A ←
2 for e
3 do
4 sort order
5 for e nond
6 do
7 then
8 Un
9 retu

- Kru
- Lines |V|
- Line
- Uni
if
sa
(u,
as

tree, so is safe ed

edge

Page No.
Date:

to add to
ding all
only 2 trees
(u,v) of

reedy algo is
to forest
ible weight
structure
ts representation
tains u

u and v
tree by
Set(u)

done by

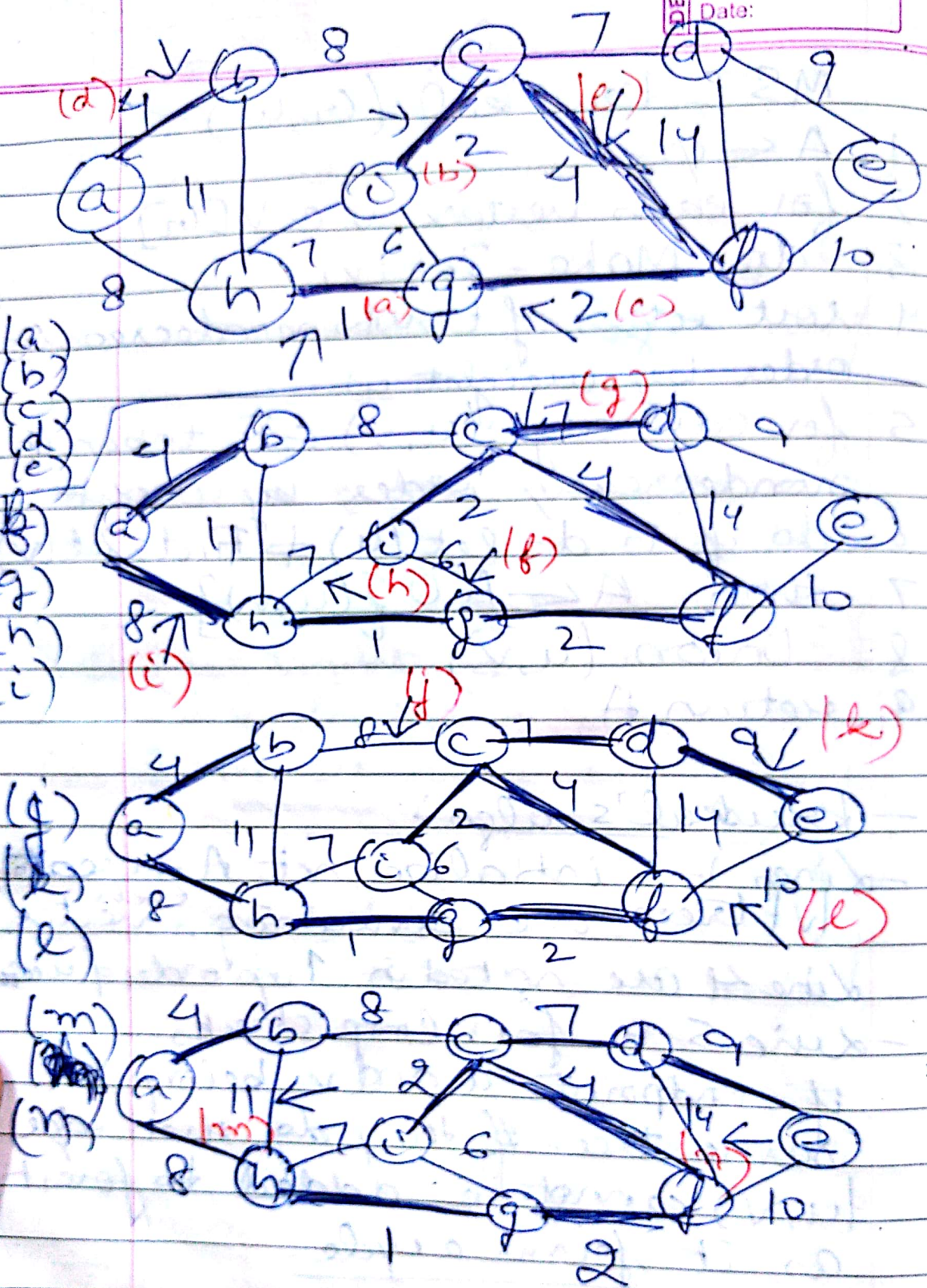
a new set
is pointed

in forest Comp
u, v) must be
G1 to some other

MST_Kruskal (G, w)
1 A ← ∅
2 for each vertex v ∈ V[G]
3 do Make-Set(v)
4 sort edges of E into nondecreasing order by weight w
5 for each edge (u, v) ∈ E, taken in nondecreasing 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

– Kruskal's algo :
– Lines 1–3 initialise set A & create |V| trees one containing each vertex
– Line 4 are sorted in nondecreasing order
– Line 5–8, for loop checks if endpoints u and v belong to same tree. If they do, then edge (u, v) cannot be added to forest as it forms a cycle.

Tree, so (u, v) is safe edge for G1



DELTA Page No.
Date:

Running time

Initialization takes $O(1)$ time
Sorting in line 4 $O(E \lg E)$ time

$O(E)$ find-set + Union operations
+ $|V|$ Make-set operations
Total: $O((V+E)\alpha(V))$ time
where α is very slowly growing function.

As $E \geq V-1$
No disjoint-set operations take
$O(E\alpha(V))$ time.

As $\alpha(|V|) = O(\lg V)$

Total time: $O(E \lg V)$

DELTA Page No. Date:

Prim's algorithm

- Prim's algo has property that the edges in set A always forms a single tree.
- Tree starts from an arbitrary root vertex & grows until tree spans all vertices in V.
- At each step, a light edge is added to tree A.
- Prim's algo is greedy since tree is augmented at each step with an edge that contributes the minimum amt. possible to tree's weight.
- During execution of algo, the vertices that are not in tree reside in a min-priority queue Q based on key field.
- For each vertex v, key[v] is min^m weight of any edge connecting v to a vertex in tree. key[v] = ∞ if there is no such edge.

- $\pi[v]$ names parent of v in tree.

- During the algo, the set A from Generic-MST is kept as:-

$$A = \{(v, \pi[v]) : v \in V - \{r\} - Q\}$$

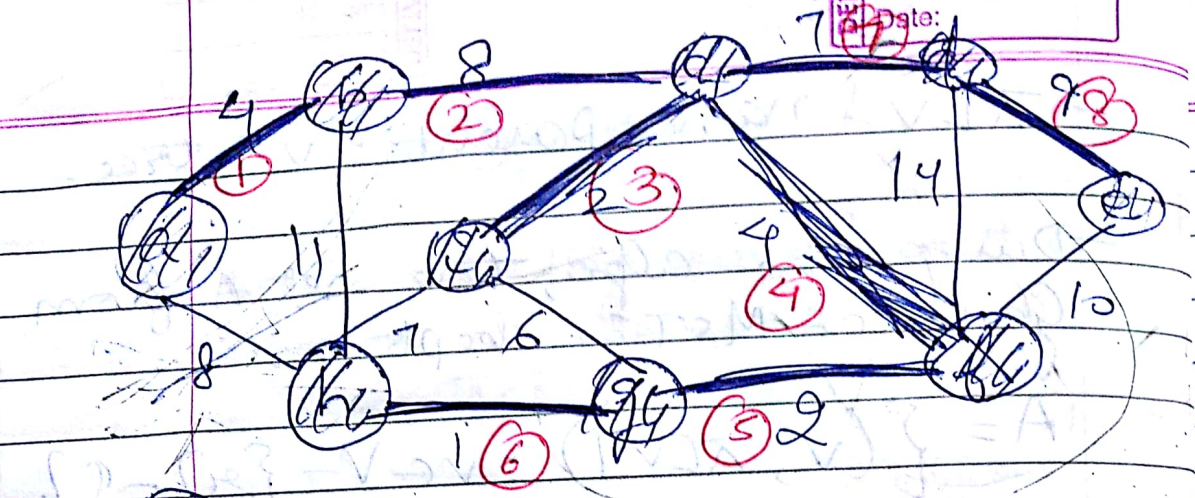
- When algo terminates, Q is empty

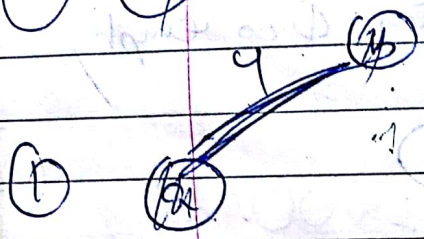
```
MST_Prim(G, w, r)
1  for each u ∈ V[G]                          → O(V)
2     do key[u] ← ∞
3     π[u] ← NIL
4  key[r] ← 0
5  Q ← V[G]
6  while Q ≠ ∅                                O(lg V)
7     do u ← Extract-Min(Q)                   O(V lg V)
8        for each v ∈ Adj[u]                  O(E)
9           do if v ∈ Q and w(u,v) < key[v]
10             then π[v] ← u                  Decrease key
11             key[v] ← w(u,v)                O(lg V)
                                              O(E lg V)
```

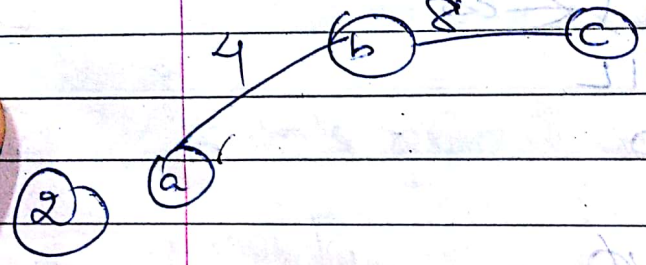
Throughout the algo,

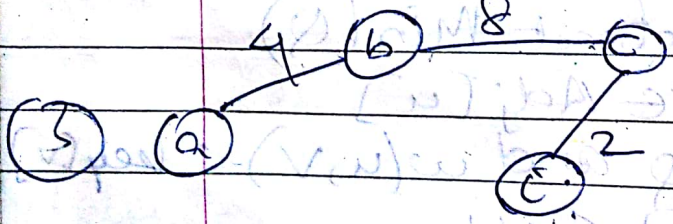
→ V − Q contains vertices in tree being grown.

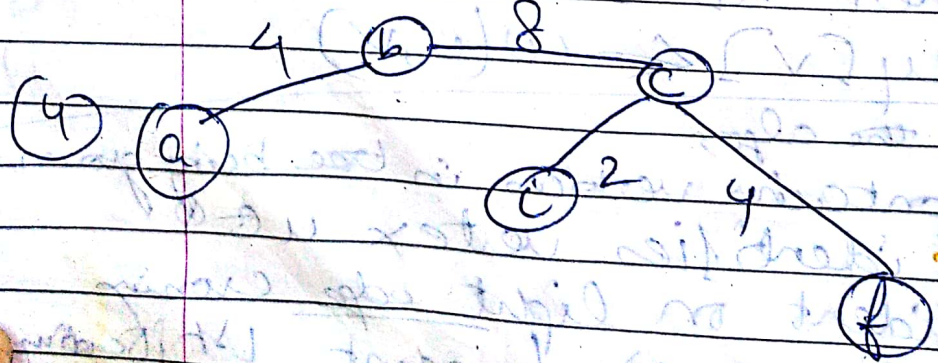
→ Line 7 identifies vertex u ∈ Q incident on light edge crossing cut (V − Q, Q) (except initially where u = r due to line 4)

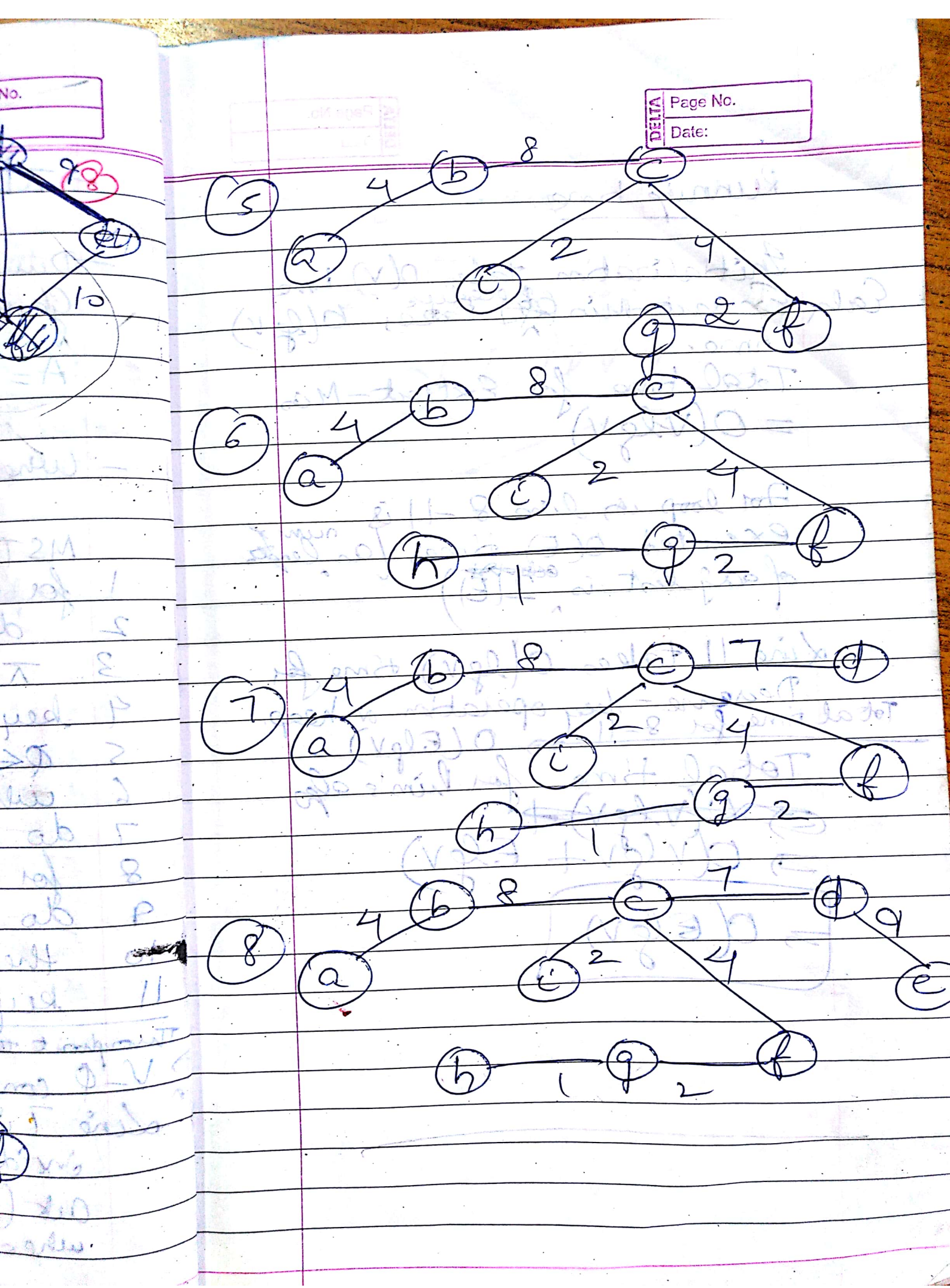












DELTA Page No. Date:

Running time

Initialization takes $O(V)$ time.
Each Extract-Min procedure takes $O(\lg V)$ time.
Total time for Extract-Min $= O(V \lg V)$

For loop in line 8-11 is executed $O(E)$ times (as sum of length of adj list is $2(E)$)

Line 11 takes $O(\lg V)$ time for Decrease-key operation on heap.

Total time for 8-11 $\Rightarrow O(E \lg V)$

Total time for Prim's algo

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

$\Rightarrow O(E \lg V)$

DELTA | Page No. | Date:

Fractional knapsack problem

This problem is similar to 0-1 knapsack problem but here we can take fraction of objects or item.

Here n is the no. of objects or item in the store where each item i has weight w_i and worth v_i and capacity of knapsack is m.

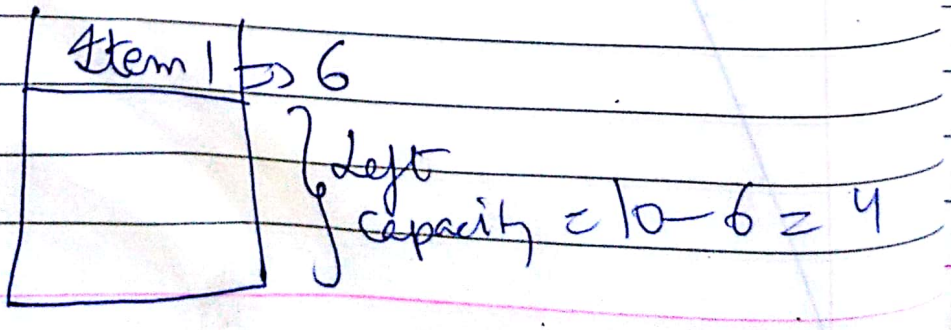
Eg. $(v_1, v_2, v_3) = (8, 5, 5)$
$(w_1, w_2, w_3) = (6, 5, 5)$
$m = 10$

If we use greedy technique we find $\frac{v_i}{w_i}$ ratio

$\frac{v_1}{w_1} = \frac{8}{6} = 1.3$

$\frac{v_2}{w_2} = \frac{5}{5} = 1$

$\frac{v_3}{w_3} = \frac{5}{5} = 1$



Item 1	⇒ 6
Item 2	⇒ 4 (left capacity) / 5 (weight of item 2)

Total weight = $6 + \frac{4}{5} \times 5 = 10$

Total profit = $8 + \frac{4}{5} \times 5 = 12$

Answer is :- $(1, \frac{4}{5}, 0)$ (Item 1, Item 2, Item 3)

Greedy_Knapsack(m, n)

▷ x[1..n] is solution vector

```
for i ← 1 to n
do x[i] ← 0.0
U ← m
for i ← 1 to n
do if w[i] > U
   then break
   x[i] ← 1.0
   U ← U − w[i]
if i ≤ n
then x[i] ← U / w[i]
```

DELTA Page No. Date:

- We assume that objects are sorted in decreasing order of worth to weight ratio.
- Variable U is left capacity of knapsack.

Running time of this algo is $\Theta(n)$

Total running time of fractional knapsack is $\Theta(n \log n)$

(time taken to sort using merge sort.)