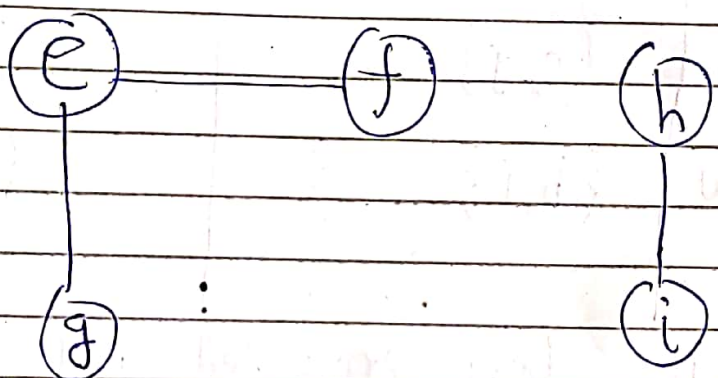
PAGE NO.:
DATE:

* Applications of Disjoint-set Datastructures :

(I) Finding connected components in a graph (undirected)

Let us apply operations of disjoint-set datastructures and find out connected components in an undirected graph.

Let $G=(V,E)$ be an undirected graph as shown below :



We apply algorithm CONNECTED-COMPONENTS(G) :-

```
1) for each vertex v ∈ G.V
2)     MAKE-SET(v)
3) for each edge (u,v) ∈ G.E
4)     if FIND-SET(u) ≠ FIND-SET(v)
5)         UNION(u,v)
```

To check whether vertices are part of the same component or not, we apply the algorithm SAME-COMPONENT(u,v)

```
1) if FIND-SET(u) == FIND-SET(v)
2)     return TRUE
3) else return FALSE.
```

Applying both of the mentioned algorithms on the given graph, we can detect connected components as follows :-

Operation	Edge Processed	Collection of disjoint sets
I) MAKE-SET	Initial sets	{e} {f} {g} {h} {i}
II) FIND+UNION	{e,f}	{e,f} {g} {h} {i}
III) FIND+ UNION	{e,g}	{e,f,g} {h} {i}
IV) FIND+UNION	{h,i}	{e,f,g} {h,i}

All edges have been processed and we get two connected components :- {e, f, g} and {h, i}.

We can find whether two vertices are in the same component or not as follows :-

Check FIND-SET(e) == FIND-SET(f) [TRUE]

So, e and f are in same component.

Check FIND-SET(e) == FIND-SET(h) [FALSE]

So, e and h are in different components.

II) Finding Minimum Spanning Tree using Kruskal's Algorithm :

```
MST-KRUSKAL(G,w)  /* A weighted graph */
1) A = ∅
2) for each vertex v ∈ G.V
3)     MAKE-SET(v)
4) sort the edges of G.E into nondecreasing order by weight w
5) for each edge (u,v) ∈ G.E taken in nondecreasing
       order by weight w
6)     if FIND-SET(u) ≠ FIND-SET(v)
7)         A = A ∪ {(u,v)}
8)         UNION(u,v)
9) return A
```

We want to analyze the running time of Kruskal's algorithm, that uses disjoint-set data structures. But, before analyzing its running time, we need to familiarize ourselves with some basic rules that are followed in the analysis of these data structures.

Rules are as given below :-

1) Running time is calculated in terms of two parameters :-

a) m :- the total number of MAKE-SET, FIND-SET and UNION operations

b) n :- the number of MAKE-SET operations.

2) Since each UNION operation, reduces the number of sets by one, so after n-1 UNION operations, only one set will remain.

3) We assume that the n MAKE-SET operations are the first n operations to be performed.

4) When we apply FIND + UNION operations, a very slow growing function ($\alpha(V)$) gets associated with their running times.

Ex:- FIND + UNION operations combines V vertices and E edges, then the time complexity would be

$$O((V+E)(\alpha(V)))$$

DELUXE
PAGE NO.:
DATE:

So, let us now apply these rules, to compute running time of MST-KRUSKAL algorithm :-

		Time taken
Line (1)	$A = \phi$	$O(1)$
Line (4)	Sorting of edges	$O(E \log E)$
Lines (5 to 8)	"for" loop [FIND + UNION operations]	$O(E)$
Lines (2 to 3)	"for" loop [MAKE-SET operations]	$O(V)$
Line (9)	return A	$O(1)$

Applying rule no. (4), $O(E)$ and $O(V)$ can be integrated (combined), and because of UNION + FIND operations, we get the running time as $O((V+E)(\alpha(V))$, where $\alpha(V)$ is a very slowly growing function.

So, effectively the running time of this algorithm is :-

$$O(1) + O(E \log E) + O((V+E)(\alpha(V)) + O(1)$$

$= O(E \log E) + O((V+E)(\alpha(V))$

Since, G is a connected weighted graph, so $|E| \geq |V| - 1$, so asymptotically V can be ignored with respect to E, so we get :-

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

$= O(E \log V)$

(Since $\alpha(V) = O(\log V)$, because $\alpha(V)$ is very small)

So, the running time is :-

$O(E \log E) + O(E \log V) \longrightarrow (1)$

$= O(E \log E)$.

Observing that, $|E| < |V|^2$, we have

$\log|E| \leq 2 \log|V|$

So, $\log|E| = O(\log V) \longrightarrow (2)$

Putting (2) in (1), we get running time:

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