* Used for solving optimisation problem
↓
problem that requires either minimum result / maximum result

* Feasible solutions : solutions that satisfy constraints/conditions given in the problem

* If problem demands result should be minimum / need minimum cost solutions, its called Minimisation problem

* A solution thats already feasible, gives minimum cost is called optimal solution
or max
For any problem, there is only 1 optimal solution.
→ according to problem's objective.

* Strategies for optimisation problem:
1. Greedy
2. Dynamic programming
3. Branch + bound.

* Greedy method says that given problem must be solved in stages. In each stage we consider one i/p from given problem and if its feasible, we include it in the solution.
Thus, by including all feasible solns, we get optimal solution

PRIM

Greedy inclusion of nearest vertex to vertices already in the tree.

* PRIM's :

① construct Min span tree through sequence of expanding subtrees

② : no: of iterations = $n-1$

n - no: of vertices

③ Tree generated by Algorithm is obtained as set of edges used for tree's expansion.

④ complexity :

- graph - weight matrix, priority Q - unordered array.

$\Downarrow$

$O(|V^2|)$

- Min heap :- every element <= its children.

Deletion of smallest element, Insertion of new element, changing element's priority. } - $O(\log n)$ $n = V$

- graph - adjacency list, priority Q - Min heap } => $O(|E| \log |V|)$

$\Downarrow$

Because, Algorithm performs

$|V| - 1$ - deletions of smallest element,

$|E|$ - verifications

changes element's priority in a min-heap of size not exceeding $|V|$,

each operation - $O(\log |V|)$

so, $(|V| - 1 + |E|) O(\log |V|) = O(|E| \log |V|)$

$\Downarrow$

Because in connected graph,

$|V| - 1 \le |E|$

KRUSKAL

expanding sequence of subgraphs thats always acyclic but not necessarily connected in intermediate stages of algorithm. ⇒ if cycle creates, simply skip

$\boxed{\text{union-find operation} = O(\log V)}$

using union-find algorithm

Time efficiency $= O(|E| \log |E|)$ / $O(e \log v)$

sorting : $O(e \log e)$

union-operations : bounded above $n-1$

bcz each union Tes subset size atleast by 1, there is only n elements.

$= \boxed{E \log E + E \log V}$

vertices - collection of disjoint subsets of a finite set.

i) makeset (x) - $O(1)$

ii) Find(x) - $O(1)$

iii) union(x,y) - $O(n^2)$

Linked list

Tree.

use only 1 element from disjoint subsets as its representative.

No: of times rep changes = i

size of list $= 2^i \leq N \Rightarrow \boxed{i = \log_2 N}$

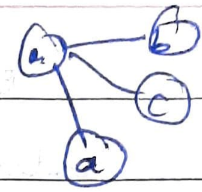
Find - $O(\log n)$

quick union - $O(n + m \log n)$

$n-1$ unions

m finds.

DIJKSTRA'S



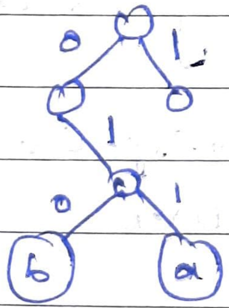
→ Shortest path from a to b, c, d
→ both directed, undirected.
→ Non-negative weights only

COMPLEXITY :

1. Adjacency Matrix – Graph
   unordered array – priority q
   ⇓ $O(|V|^2)$
2. Adjacency list – Graph
   min-heap – PQ
   ⇓ $O(|E| \log |V|)$

HUFFMAN:

→ variable code length.
→ optimal merge pattern.



encoded value of a = 011
Left = 0
Right = 1

→ check which merge result is least value.
→ Tree constructed – Huffman tree.
Manner / Algorithm – Huffman code