GREEDY TECHNIQUE

* Greedy approach constructs a solution through a sequence of steps, each expanding a partially constructed solution obtained sofar, until a complete solution to the problem is reached.

* On each step, the choice must be

  feasible – i.e. it has to satisfy the problem's constraints

  locally optimal – i.e. it has to be the best local choice among all feasible choices available on that step.

  irrevocable – i.e. once made, it cannot be changed on subsequent steps of the algorithm.

* Applications are
  - Prim's Algorithm
  - Kruskal's Algorithm
  - Dijkstra's Algorithm &
  - Huffman Trees.

* PRIMS ALGORITHM
  - A spanning tree of a connected graph is its connected acyclic subgraph that contains all the vertices in the graph.

- A minimum spanning tree of a weighted connected graph is its spanning tree of the smallest weight, where the weight of a tree is defined as the sum of the weights on all its edges.

- The minimum spanning tree problem is the problem of finding a minimum spanning tree for a given weighted connected graph.

- Prim's Algorithm
  * Choose a single vertex arbitrarily from the set V & form a subtree
  * On each iteration, attach to it the nearest vertex not in that tree
  * The algorithm stops after all the graph's vertices have been included in the tree
  * If total number of vertices is n, the algorithm requires n-1 iterations

- ALGORITHM Prim(G)

  // Prim's Algorithm for constructing a minimum spanning tree
  // input: A weighted connected graph $G = (V, E)$
  // Output: A minimum spanning tree edges

  $V_T \leftarrow \{V_0\}$

  $E_T \leftarrow \phi$

  for $i \leftarrow 1$ to $|V| - 1$ do

  find a minimum-weight edge $e = (v, u)$ such that v is in V_T & u is in $V - V_T$

$V_T \leftarrow V_T \cup \{u\}$

$E_T \leftarrow E_T \cup \{e\}$

return E_T

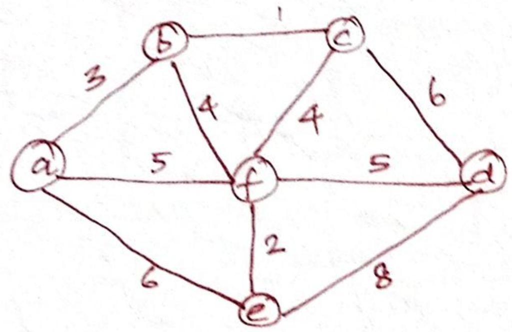
* Attach two labels to each vertex : the name of the nearest tree vertex and the length of the corresponding edge.

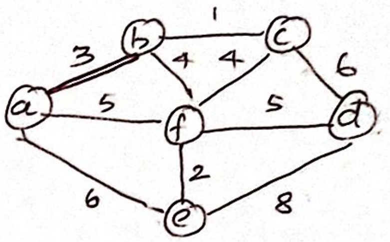
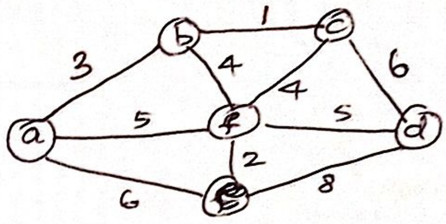
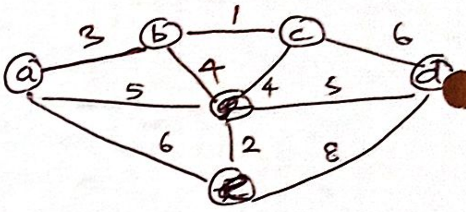
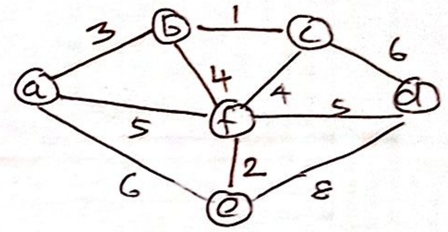
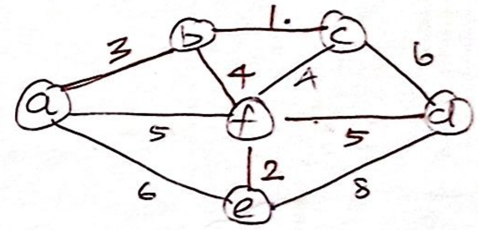
* Vertices that are not adjacent to any of the tree vertices can be indicated with a ∞ label indicated their "infinite distance to the tree vertices & a null label for the name of the nearest vertex

* Vertices can be classified as
  "fringe" – Vertices not in the tree but adjacent to atleast one tree vertex
  &
  "unseen" – vertices not affected by the algorithm

* After choosing the vertex 'u'
  → move u from $V - V_T$ to V_T
  → for each remaining u in $V - V_T$, update labels that are adjacent to u.

* Example



Tree vertices	Remaining Vertices	Illustration
$a(-,-)$	$b(a,3)$, $c(-,\infty)$ $d(-,\infty)$ $e(a,6)$ $f(a,5)$	
$b(a,3)$	$c(b,1)$ $d(-,\infty)$ $e(a,6)$ $f(b,4)$	
$c(b,1)$	$d(c,6)$ $e(a,6)$ $f(b,4)$	
$f(b,4)$	$d(f,5)$ $e(f,2)$	
$e(f,2)$	$d(f,5)$	
$d(f,5)$		

* The efficiency of prim's algorithm depends upon the data structure used for implementing the algorithm
* If the graph is represented by its weight matrix & the priority queue (as unordered array) is used for $V - V_T$ it requires $O(|V|^2)$ as the running time.
* If priority queue is implemented using minheap & ~~it requires~~ graph by its adjacency list it requires $O(|E| \log |V|)$ as the running time.