Assignment - II

Ques1. Explain Quick sort Algorithm? Also sort the following elements using quick-sort algorithms
5, 7, 9, 4, 10, 2, 8, 1

Ans: Quick sort is a divide and conquer sorting algorithm

Steps:

1) Choose a pivot → select an element from array (first, last, middle, or random element).
2) Partitioning → Rearrange array such that:
   - All elements smaller than pivot are placed before it.
   - All elements greater than pivot are placed after it.
3) Recursive sort → Recursively apply quick sort to the left and right partitions until the array is sorted.

Time complexity:

- Best case : $O(n \log n)$ (when pivot divides well)
- worst case : $O(n^2)$ (when pivot is always smallest / largest)
- Avg. case : $O(n \log n)$

Example : Sort [5, 7, 9, 4, 10, 2, 8, 1]
we'll use last element as pivot

→ Step-I : Initial array
[5, 7, 9, 4, 10, 2, 8, 1]
Pivot = 1

- After partition → [1, 7, 9, 4, 10, 2, 8, 5]
- Left part = [ ], Right part = [7, 9, 4, 10, 2, 8, 5]

Date
Page

↳ Step-2 : Apply quick sort on right part
[7, 9, 4, 10, 2, 8, 5]
Pivot = 5
• After partition → [4, 2, 5, 10, 9, 8, 7]
• left part = [4, 2], Right part = [10, 9, 8, 7]
so far [1, 2, 4, 5, 10, 9, 8, 7]

↳ Step-3 : Sort left [4, 2]
Pivot = 2
• After partition → [2, 4]
So far → [1, 2, 4, 5, 10, 9, 8, 7]

↳ Step-4 : Sort right [10, 9, 8, 7]
Pivot = 7
• After partition → [7, 9, 8, 10]
• left part = [ ], right part = [9, 8, 10]

↳ Step-5 : Sort [9, 8, 10]
Pivot = 10
• After partition → [9, 8, 10]
• left = [9, 8] , right = [ ]

Now [9, 8] → pivot 8 → [8, 9]

Final sorted array : [1, 2, 4, 5, 7, 8, 9, 10]

Date
Page

Ques: Write Prim's Algorithm for finding the minimum spanning tree. Execute the algorithm for the given below graph. Consider a as starting vertex.

Ans: In Prim's algorithm, grow a tree from any start vertex. At each step, add the cheapest edge that connects a visited vertex to an unvisited vertex.

Pseudocode (adjacency list, weights w):

1) For all vertices v : key[v] = ∞, parent [V] = NIL.
2) Pick a start s . Set key [s] = 0.
3) While some vertex remains unpicked :
- Choose the unpicked vertex u with minimum key[u]
- Mark u picked (add to MST)
- For each neighbour v and u with edge weight w(u, v)
  - If v unpicked and w(u, v) < key [v]
    key [v] = w(u, v), parent [v] = u

4) The MST edges are {(v, parent [v]) | parent[v] ≠ NIL}

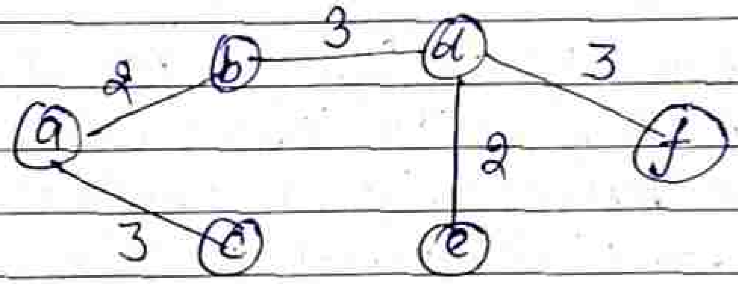
Applying Prim's Algo to given graph (start a)

Edges (undirected from figure) :

a-b : 2 , a-c : 3 , b-c : 5, b-d : 3 , b-e : 4,
c-e : 4, d-e : 2, d-f : 3 , e-f : 5

Initialize : key[a] = 0, other = ∞

Iter	Picked u	Edge added
1	a	–
2	b	a-b (2)
3	c	a-c (3)
4	d	b-d (3)
5	e	d-e (2)
6	f	d-f (3)



Total MST weight = 2+3+3+2+3
= 13

Ques 3. Explain greedy method with example.

Ans : The Greedy method is a problem-solving approach where decisions are made step-by-step, always choosing the options that seems (best) optimal at current step, with the hope that this leads to global optimum.

- It works well when the greedy choice property and optimal substructure property hold :

1) Greedy choice property.
2) Optimal substructure

General steps in greedy Algorithm

1) Initialize sol^n set as empty.
2) At each step, select best possible choice according to some criterion.
3) Check feasibility → if choice is valid, add it to the solution.
4) Repeat until sol^n is complete.

Example : Fractional Knapsack Problem

Item	Value	Weight	Value/Weight
I_1	60	10	6
I_2	100	20	5
I_3	120	30	4

Greedy choice : Pick item with highest value / weight first.

Steps :

- Take all of I_1 (10 weight, value = 60)
- Take all of I_2 (20 weight, value = 100)
- left capacity = 20 → take 20/30 of I_3 = value 80

Total value = 240