Practical No. 4

Aim:- Finding the cost of minimal spanning tree

Theory:- A minimum spanning tree (MST) or minimum weight spanning tree for a weighted, connected, undirected graph is a spanning tree with a weight less than or equal to the weight of every other spanning tree.

In Kruskal's algorithm, sort all edges of the given graph in increasing order. Then it keeps on adding new edges and nodes in the MST if the newly added edge does not form a cycle. It picks the minimum weighted edge at first and the maximum weighted edge at last. Thus we can say that it makes a locally optimal choice in each step in order to find the optimal solution. Hence, this is a Greedy Algorithm.

Steps for finding MST using Kruskal's Algorithm:-

1. Arrange the edge of G in order of increasing weight
2. Starting only with the vertices of G and proceeding sequentially add each edge which does not result in a cycle, until (n-1) edges are used.
3. EXIT

MST-KRUSKAL (G, w)

1. $A \leftarrow \emptyset$
2. for each vertex $v \in V[G]$
3. do MAKE-SET (v)
4. sort the edges of E into non decreasing order by weight w
5. for each edge $(u, v) \in E$, taken in non decreasing order by weight
6. do if FIND-SET (u) $\neq$ if FIND-SET (v)
7. then $A \leftarrow A \cup \{(u, v)\}$
8. UNION (u, v)
9. return A

Analysis: Where E is the no. of edges in the graph and V is the no. of vertices, Kruskal's algorithm can be shown to run in $O(E \log E)$ time, or simply, $O(E \log V)$ time, all with simple d structures. These running times are equivalent because:

- E is at most V^2 and $\log V^2 = 2 \times \log V$ is $O(\log V)$.
- If we ignore isolated vertices, which will each then components of the minimum spanning tree, $V \leq 2E$, so $\log V$ is $O(\log E)$

Thus the total time is

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

Conclusion :- Hence, we have studied how to find the cost of minimal spanning tree.

A+

9/8/2024

Practical No. 5

Aim :- Write a program of minimum spanning tree using prim's algorithm.

Theory:- Prim's minimal spanning tree algorithm is one of the efficient methods to find the minimum spanning tree of a graph. A minimum spanning tree is a sub graph that connects all the vertices present in the main graph with the least possible edges and minimum cost (sum of weights assigned to each edge). The algorithm, similar to any shortest path algorithm, begins from a vertex that is set as a root and walks through all the ~~vertex~~ vertices in the graph by determining the least cost adjacent edges.

Prim's Algorithm :- To execute the prim's algorithm, the inputs taken by the algorithm are the graph G {V, E}, where V is the set of vertices and E is the set of edges, and the source vertex S. A minimum spanning tree of graph G is obtained as an output.

Algorithm :-

1) Declare an array visited [] to store the visited vertices and firstly, add the arbitary

root, say S, to the visited array.

2) Check whether the adjacent vertices of the last visited vertex are present in the visited [] array or not.

3) If the vertices are not in the visited [] array, compare the cost of edges and add the least cost edge to the output spanning tree.

4) The adjacent unvisited vertex with the least cost edge is added into the visited [] array and the least cost edge is added to the minimum spanning tree output.

5) Steps 2 and 4 are repeated for all the unvisited vertices in the graph to obtain the full minimum spanning tree output for the given graph.

6) Calculate the cost of the minimum spanning tree obtained.

Practical No. 6

Aim :- Implementation of Huffman code.

Theory :- Huffman Coding :-

Huffman Coding is a technique of compressing data to reduce its size without losing any of the details. It was first developed by David Huffman. Huffman coding is generally useful to compress the data in which there are frequently occurring characters.

Huffman coding Algorithm :-

Create a priority queue Q consisting of each unique character.
sort then in ascending order of their frequencies.
for all the unique characters:
    Create a newNode
    extract minimum value from Q and assign it to leftchild of newNode.
    extract minimum value from Q and assign it to rightchild of newNode.
    calculate the sum of these two minimum values and assign it to the value of newNode
    insert this newNode into the tree.

return rootNode.

Huffman Coding Complexity :-

The time complexity for encoding each unique character based on its frequency is $O(n \log n)$.

Extracting minimum frequency from the priority queue takes place $2*(n-1)$ times and its complexity is $O(\log n)$. Thus the overall complexity is $O(n \log n)$.

Huffman Coding Applications :-

1) Huffman Coding is used in conventional compression for formats like GZIP, BZIP2, PKZIP, etc.

2) For text and fax transmissions