PROJECT REPORT

Abstract:

Implement indexed priority queue. A binary heap is used to store the elements. Use this indexed priority queue to implement Prim's algorithm to find minimum spanning tree for a given graph. The program then outputs the weight of the minimum spanning tree of the input graph.

Problem Statement:

1. Implement indexed priority queue

2.Use Indexed Priority Queue to calculate MST.

If a element x moves to a new index i, then should call x.putIndex(i) to inform the element of its location in the heap. The function decreasekey will be called with the index of the element whose key has decreased and the heap order needs to be restored at the index.

Methodology:

Implement priority queue using a binary heap. It has following properties.

1.Every node has a value smaller than both of its children.

2.Every level of the heap is full, except for possibly the last level which is filled left to right.

3.Elements can be removed or accessed only at the root node.

4.Beginning of priority queue is smallest element of the queue.

5.Node at index i has left child at 2*i while right child at 2*i+1.

Primary Functions implemented are:

insert(x): To insert element into queue

add(x): To add element into priority queue

isEmpty(): To check whether priority queue is empty

resize(): Resize the priority queue by multiple of 2.

deleteMin(): To delete minimum element from the priority queue.

contains(x): To check whether the element is present in the priority queue.

decreaseKey(x): Restore the heap order property after the priority of x has changed.

min(): To find the minimum element of the priority queue.

percolateUp(i): To restore the heap property which got violated when priority of that element at index i of queue has decreased. We then call putIndex to set the index values for changed element.

percolateDown(i): Used to restore the heap property.

In this we have considered three condition:

1.When size of the priority queue is less than index.

2.When size of the priority queue is equal to 2*i. Which means that node has only one child.

3.When size of the priority queue is greater than 2*i+1.In this case we check which child has minimum value. Then compare that child value with index i. One which has minimum value is the set for i and accordingly priority are updated by calling putIndex for changed elements.

buildHeap() : Used to create a heap.

Indexing:

When a element x changes its value then its index i value changes. We call x.putIndex(i) to inform the element that its location has changed in heap by using functions PercolateUp(i) and PercolateDown(i). Function decreasekey(x) is called with index of the element whose key has decreased using x.getIndex().We need to restore the heap order property after the priority of x has decreased.

Minimum Spanning Tree:

In Prim's Algorithm we first initialize all the vertices value to infinity. Then assign value 0 to source vertex that is u. Remove min element u from the Priority queue Then find all the adjacent vertices v from vertex u .Check the weight value and min weight is updated for each vertex. After doing that call DecreaseKey(v) for the vertex as the weight has changed. And for every changed vertex update the priority value accordingly by calling put Index for that changed vertex. After all the vertices adjacent to u has been processed we again find the minimum element present in the priority queue and the process continues as specified before.

Development Platform:

Operating System: Windows8

Hardware Specification: Intel CORE i5 processor,8GB RAM

Development IDE: Eclipse

Test Cases:

Indexed priority queue implemented on Prim's algorithm for following inputs.

1.50 vertices ,140 edges

2.100 vertices ,284 edges

3.200 vertices,580 edges

4.100000 vertices,299971 edges

Results:

Following table shows the performance of the Prim's implementation as input size changes.

File	Output	Time taken
Prim1.txt	84950	0
Prim2.txt	110419	15
Prim3.txt	153534	15
Prim-ck.txt	3384476	711

Conclusion:

Implement Prim's Algorithm using Binary heap(Indexed Priority Queue) gives better performance than compared to adjacency matrix. Runtime using Binary heap is O(|E|log|V|) where |E| is the number of edges and |V| is the number of vertices. Indexed Priority queue makes sure that whenever we delete the element from queue ,that element has the minimum weight compared to other elements from the queue. While in adjacency matrix graph representation we linearly search an array of weight to find the minimum weight edge which takes O(|V|^2) running time. We can further improve the runtime by implementing Fibonacci heap which has runtime complexity O(|E| + |V| log |V|).

References:

Class notes - Balaji

http://en.wikipedia.org/wiki/Prim%27s_algorithm

Introduction of Algorithms - Thomas Cormen, Charles Leiserson, Ronald Rivest , Clifford Stein