**PROJECT REPORT**

**Project By:**

Name: Swapnil Shah

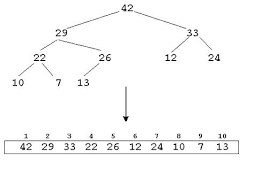
NetId: sjs140230

**1.ABSTRACT**

The project was to implement Indexed priority Queue which supports functions like - Insert, DeleteMin, DecreaseKey. This implementation was to be used to implement Prims Minimum Spanning tree algorithm.

**2.Priority Queue Definition**

A priority queue is an abstract data type which is like a regular queue or stack data structure, but where additionally each element has a "priority" associated with it. In a priority queue, an element with high priority is served before an element with low priority.



The priority queue is implemented using a binary min heap. The above representation shows a min heap representation of the priority queue.

The element at position zero is left empty for coding efficiency and the fact that element at position zero is empty is used many times in code.

**3. Problem Statement**

Implement indexed priority queues.

A binary heap is used to store the elements. If an element x moves to a new index i, then you must 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 that index.

Implement Prim's algorithm using this indexed priority queue.

**4 . Implemented Functions**

The following functions are implemented in PriorityQueueIndexed File:

Insert: Used to insert a value in priority queue.

DeleteMin: Remove the element with minimum index

DecreaseKey: Change the index of a value in priority queue and reposition it in the queue.

PercolateUp: Helper function that takes a light weight element up the priority queue.

PercolateDown: Helper function that taes a heavy weight element down the priority queue.

BuildHeap: Create a heap. Array may violate heap order in many places. This function is used to restore heap order.

isEmpty: Helper function to find if priority queue is empty.

**5. System Configuration**

Processor: Intel Core I7 (Second Generation)

RAM: 6 GB

HDD: 750 GB

IDE : Eclipse

**6. Test Cases and Comparison of result**

The sample input were run on the program which implements Minimum Spanning tree using Indexed priority queue and the following running time were observed.

|  |  |  |
| --- | --- | --- |
| No of Nodes | No of Edges | Running Time |
| 9 | 14 | 3 |
| 50 | 140 | 5 |
| 100 | 284 | 6 |
| 200 | 580 | 10 |
| 100000 | 299971 | 345 |

All time values indicated are in miliseconds.

**7. Conclusion**

By above result, we can conclude that Indexed priority queue can be used to implement graph algorithms efficiently. The running time of the above algorithm is ~O(E log V) where E is number of edges and V is number of vertices.

**8. References**

http://en.wikipedia.org/wiki/Priority\_queue

http://algs4.cs.princeton.edu/24pq/