UNIT – I

IMPLEMENTATION OF PRIORITY QUEUES

Priority Queues can be implemented using

- A simple /circular array
- Multi-queue implementation
- Linked list
- Heap tree

Priority queue using an array:

With this representation, an array can be maintained to hold the item and its priority value.

The element can be inserted at the REAR end. The deletion operation will then be performed in either of the 2 following ways:

(a)

- Starting from the FRONT, traverse the array for an element of the highest priority.
- Delete this element from the queue.
- If the deleted element is not the front-most element, shift all its trailing elements after the deleted element one stroke each to fill up the vacant position.

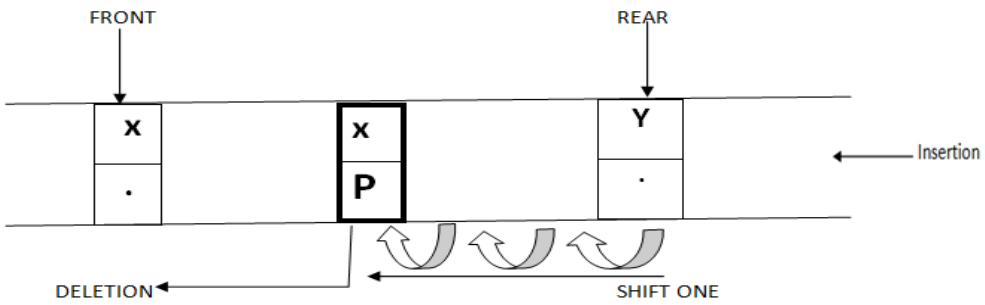

Figure: Deletion operation in an array representation of a priority queue

This is an inefficient implementation as it involves the searching the queue for the highest priority element and shifting the trailing elements after the deletion.

A better implementation is as follows:

(b)

- Add the elements at the REAR end as earlier.
- Use a stable sort algorithm to sort the elements of the queue so that the highest priority element is at the FRONT end.
- When deletion is required, delete element from the FRONT end only.

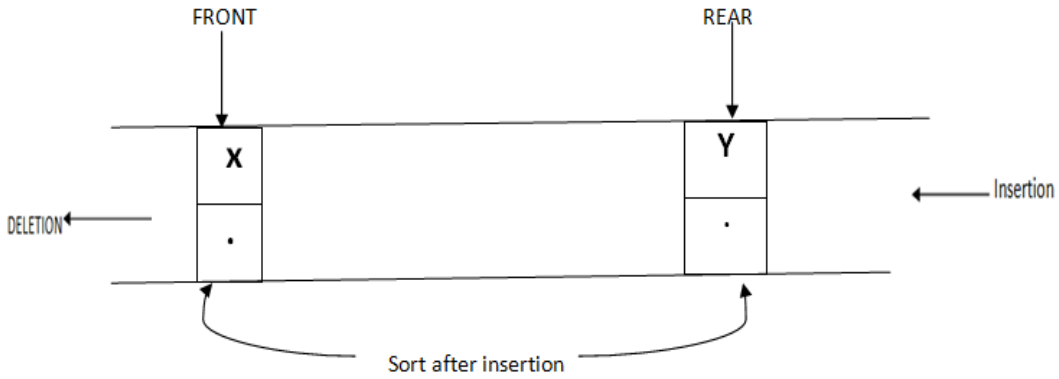

Fig: Another array implementation of a priority queue Multi-queue Implementation:

Multi-queue Implementation:

This implementation assumes N different priority values.

For each priority Pi there are 2 ends Fi and Ri corresponding to the FRONT and REAR ends respectively.

The elements between Fi and Ri are all of equal priority value Pi.

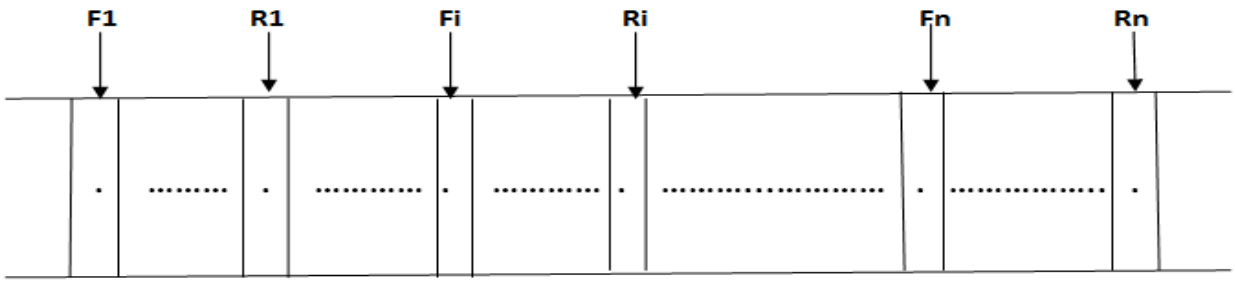

Figure: Multi-queue representation of a priority queue

With this representation, an element with priority value Pi will consult Fi for deletion and Ri for insertion.

But this implementation is associated with a number of difficulties:

- It may lead to a huge shifting in order to make room for an item to be inserted.
- A large number of pointers are involved when the range of priority values is large.
- In addition to the before mentioned, there are 2 other techniques to represent a multi-queue, which are shown in below figures.

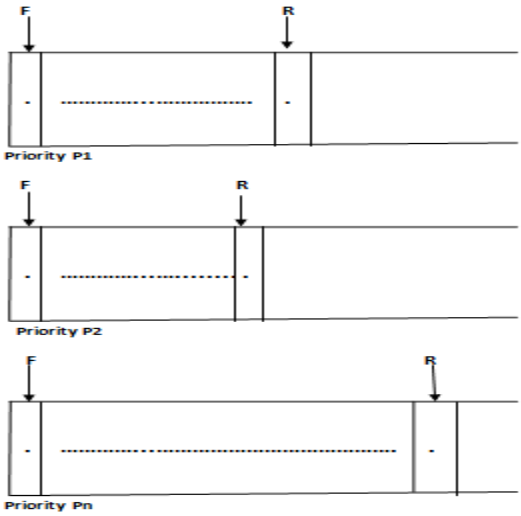

Figure: Multi-queue with simple queues

Priority queue using Linked list: stored and unsorted

- ✓ Performing insertions at front in O(1) and traversing the list which requires O(N) time
- ✓ To delete the minimum, we could insist that the list be kept always sorted: this makes insertions expensive O(N) and delete-mins cheap O(1)

Another way of implementing priority queues would be use a **binary search tree**.

- ✓ This gives an O(log N) average running time for both operations
- ✓ The basic data structure we will use will not require pointers and will support both operations in O(log N) worst case time. The implementations we will use is known as a **binary – heap**

Reference Links

1. https://www.hackerearth.com/practice/data-structures/trees/heapspriority-queues/tutorial/
2. http://pages.cs.wisc.edu/~vernon/cs367/notes/11.PRIORITY-Q.html

Video Links

1. https://www.youtube.com/watch?v=OxhYCLWMdHs

Questions

1. What is a priority queue? Explain operations performed in priority queue.
2. List and explain different ways of representing them.
3. Discuss the insertion and deletion operations in a priority queue.