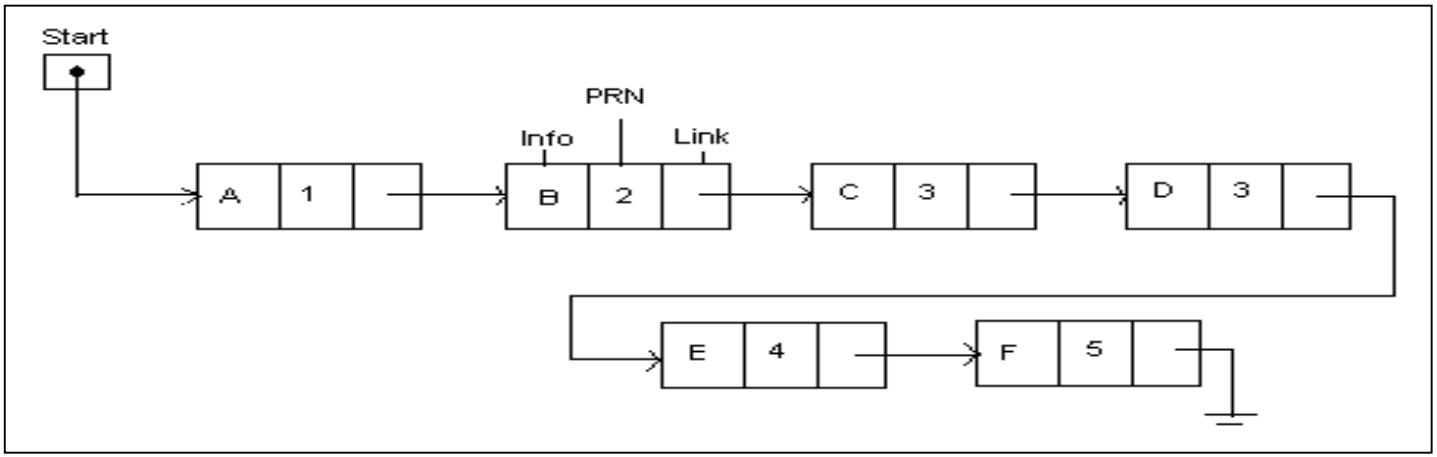
Priority Queue

Priority Queue

- A priority queue is a collection of elements such that each element has been assigned a priority.
- A priority queue supports inserting new priorities, and removing the highest priority.
- Two elements with the same priority are processed according to the order in which they were added to the queue.

Representation of Priority Queue

- Each node in the list has three fields: an information field INFO, a priority number PRN and a link number Link.
- A node X precedes a node Y in the list when X has higher priority than Y or both have the same priority but X was added to the list before Y.



Creation of a Priority Queue

This algorithm creates a priority queue where higher number shows the higher priority.

Algorithm: Create-Priority-Q (Info, Prn, Link, Start, Item, Item_Prn)

1. Set Info[New] := Item and Prn[New] := Item_Prn.
2. If Start = NULL then Start := New and Return.
3. If Prn[Start] < Item_Prn then Set Link[New] := Start and Start := New and Return.
4. Set Save := Start and Ptr := Link[Start]
5. While Ptr ≠ NULL do
6. If Prn[Ptr] = Item_Prn then Link[New] := Link[Ptr] and Link[Ptr]] := New and Return.
7. Else If Prn[PTR]< Item_Prn then Set Link[New]:=Ptr , Link[Save]:= New and Return.
8. Else Ptr := Link[Ptr]
9. Link[Save] := New and Return.
10. Exit

Deletion of a Priority Queue

This algorithm deletes and processes the element with highest priority.

Algorithm: Delete-Priority-Q(Info, Prn, Link, Start, Item)

1. Set Item := Info[Start].
2. Set Start := Link[Start]
3. Process Item.
4. Return.

Complexity

When a priority queue is implemented with an array, the efficiency of the insertion operation will be O(n) if the list is sorted.

Again, when a priority queue is implemented with an array, the efficiency of the deletion operation will be O(n) if the list is unsorted.

Priority Queue Implementation Using Heap

- Using a heap, the execution time efficiency of both the enqueue and dequeue operations in a priority queue is O(logn) because in both insertion and deletion operation we need only to consider a branch of heap and the maximum height of a branch of a complete tree is O(logn).

Insertion Operation

1. Add the new value at the end of the array; that corresponds to adding it as a new rightmost leaf in the tree.

2. Check the new value to the value in its parent. If the parent is smaller, swap the values, and continue this check-and-swap procedure up the tree until the heap-order property holds, or reach to the root.

Deletion Operation

1. Replace the value in the root with the value at the end of the array. Remove that leaf from the tree.

2. Check the new root value to the values in its children. If the value in the current node is less than one of its children, then swap its value with the larger child and continue the process until the heap-order property holds, or reach to a leaf.

Deques

• A deque is a linear list in which data elements can be added or removed at either end but not in the middle.

• This type of queue is also known as dequeue and double-ended queue.

• There are two types of DEQUE.

Input-Restricted Deque – Allow insertions at only one end of the list but deletions at both ends of the list.

Output-Restricted Deque - Allow insertions at both ends but deletions at only one end of the list.

•**Example:**

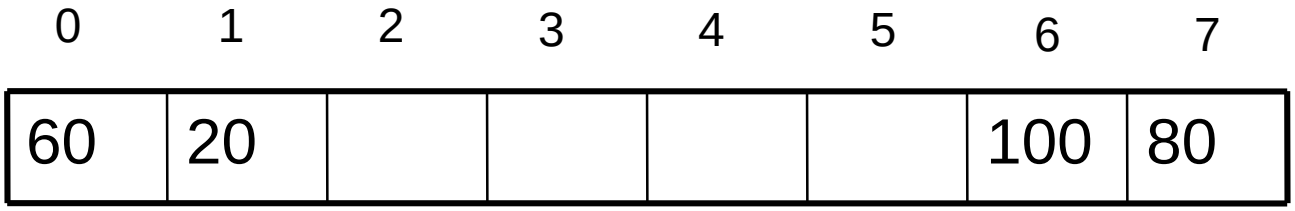


Figure: Example of a Deque

END!!!!