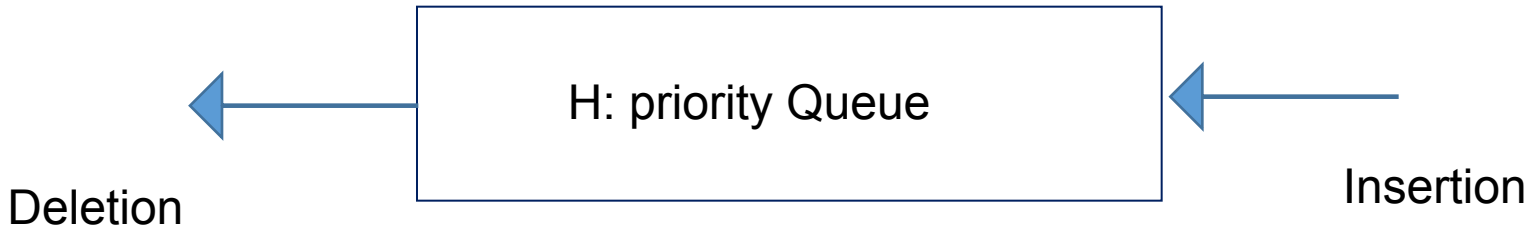
Priority Queues (Heap Trees)

A priority queue is a data structure in which each element is assigned a priority. The priority of the element will be used to determine the order in which the elements will be processed. The general rules of processing the elements of a priority queue are

- An element with higher priority is processed before an element with a lower priority.
- Two elements with the same priority are processed on a first-come-first-served (FCFS) basis.

Basic Model of a Priority Queue



Deletion is performed according to the priority and the list is maintained internally as per the property of the queue

Types of Priority Queues

- **General Priority queue or simple priority queue**

  a) Single Linked List

  b) Multidimensional Array

- **Special Priority Queues**

a) Binary Heap

b) d-heap

c) Leftist Heap

d) Skew Heap

e) Binomial Queues

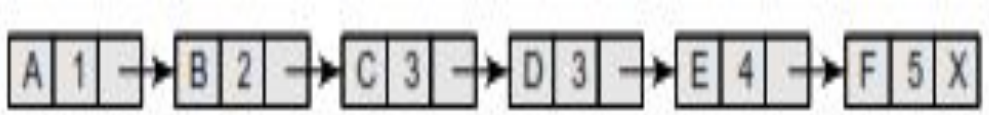
- # General Priority queue or simple priority queue

a) Single Linked List

In the computer memory, a priority queue can be represented using linked lists. When a priority queue is implemented using a linked list, then every node of the list will have three parts:

(a)The information or data part, (b) the priority number of the element, and (c)the address of the next element.

Consider the priority queue shown in the following Fig.



The highest priority element is in the front of the list so that every deletion operation takes same amount of time, But insertion may take different times. In the worst case it is n.

• General Priority queue or simple priority queue

b) Multidimensional Array:

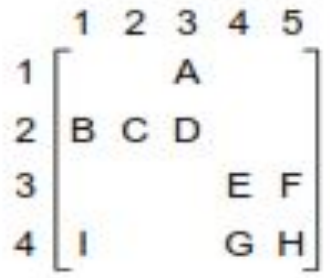
When arrays are used to implement a priority queue, then a separate queue for each priority number Is maintained. Each of these queues will be implemented using circular arrays or circular queues. Every individual queue will have its own FRONT and REAR pointers.

We use a two-dimensional array for this purpose where each queue will be allocated the same amount of space.

Look at the two-dimensional representation of a priority queue given below.

Given the FRONT and REAR values of each queue, the two-dimensional matrix can be formed as Shown in the following Fig.

FRONT	REAR
3	3
1	3
4	5
4	1



FRONT[K] and REAR[K] contain the front and rear values of row K, where K is the priority number.

Binary Heap
(Special Priority Queue)

- **Introduction**
- **Structural Property**
- **Heap order Property**

Operations on Binary Heap (Priority Queue)

- **Insertion**
- **Deletion**
- **Sorting (Heap Sort)**

Algorithm for construction/insertion of Binary Heap Tree

```
Algorithm Insheap( H, N, item)
{  // A heap H with N elements are stored in the array
   N:=N+1, ptr:=N
   While( ptr>1) do
     {  par:=ptr/2
      if (item <=H[par]) then { H[ptr]:=item, return }
       H[ptr]:=H[par]
        ptr:=par
     }
    H[1]:=item
}
```

Call Insheap(H, j, H[j+1]) for j:=1 to N-1 to build the heap tree

Algorithm for Deletion of an item from a Binary Heap Tree

```
Algorithm  Delheap( H, N, item)
{  item:=H[1],  last:=H[N], N:=N-1
  ptr:=1, left:=2, right:=3
   while(right<=N) do
    { if (last>=H[left] && last>=H[right]) then
                                          { H[ptr]:=last, return }
      if ( H[right]<=H[left] )  then  { H[ptr]:=H[left], ptr:=left }
                                else   { H[ptr]:=H[right], ptr:=right}
      left:=2*ptr, right:=2*ptr+1
    }
    if( left=N) then {  if (last<=H[left]) then { H[ptr]:=H[left], ptr:=left }
                        }
      H[ptr]:=last
}
```

Algorithm for Heapsort

```
Algorithm Heapsort(A,N)
{
  for j:=1 to N-1 do
     call Insheap(A, j, A[j+1])
  while(N>1) do
   {
      call Delheap(A, N, item)
         A[N+1]:=item
   }
}
```