



# Data Structures and its Applications

---

**Dinesh Singh**

Department of Computer Science & Engineering

# DATA STRUCTURES AND ITS APPLICATIONS

---

## Priority Queue - Implementation

**Dinesh Singh**

Department of Computer Science & Engineering

## Priority Queues - definition

---

- Priority Queue is an extension of queue where every item has a priority associated with it.
- There are two types of priority queue
  - Ascending priority queue
  - Descending Priority queue
- In Ascending Priority queue ,the item with lowest priority is removed
- In descending priority queue, the item with the highest priority is removed.

- Operations in Ascending priority queue
  - `pqinsert(apq,x)` : inserts element x into the priority queue apq.
  - `pqmindelete(apq)` : removes the element with the minimum priority.
- Operations in Descending priority queue
  - `pqinsert(dpq,x)` : inserts element x into the priority queue dpq.
  - `pqmaxdelete(dpq)` : removes the element with the maximum priority.
- The operation `empty(pq)` determines whether the queue is empty or not.

- In alternate definition of a priority queue, the items do not have a priority but the intrinsic ordering of elements determine results of its basic operation.
- In ascending priority queue, items can be inserted arbitrarily, but only the smallest element can be removed.
- In descending priority queue, items can be inserted arbitrarily, but only the largest element can be removed.

- Priority queues can be implemented by
  - Arrays
  - Linked Lists
  - Heap

# Data Structures and its Applications

## Priority Queues- Array Implementation

---



```
struct pqueue
{
    int items[MAXPQ];
    int pty;
}
struct pqueue pq;
```

implementation of ascending priority :

### Items inserted based on their priority

- **Insert** : The item is inserted based on its priority. The queue is ordered in the ascending priority of the items. The item with the lowest priority will be at the first location in the array.
- **Remove** : delete always the first element, Shift the other elements to the left after the deletion

implementation of ascending priority :

### Items inserted arbitrarily

- **Insert** : The item is inserted at the rear of the queue. The queue is therefore unordered.
- **Delete** : The item with the lowest priority is found and removed. The items following the position of the removed element are shifted to the left.



implementation of ascending priority where items are inserted according to the priority

```
void pqinsert(int x,int pty,struct pqueue *pq,int *count)
{
    // x is item to be inserted
    // pty is the priority of the item
    // pq is the pointer to the priority queue
    // count is the number of items in the queue
```

```
    int j;
    struct pqueue key;
    key.data=x;
    key.ptty=pty;
```

# Data Structures and its Applications

## Priority Queues- Array Implementation

implementation of ascending priority where items are inserted according to the priority

$j = *count - 1$ ; // index of the initial position of the element

//compare the priority of the item being inserted with the

//priority of the items in the queue

// shift the items down while the priority of the item being

//inserted is greater than priority of the item in the queue

```
while((j >= 0) && (pq[j].pty < key.pty))
```

```
{
```

```
    pq[j+1] = pq[j];
```

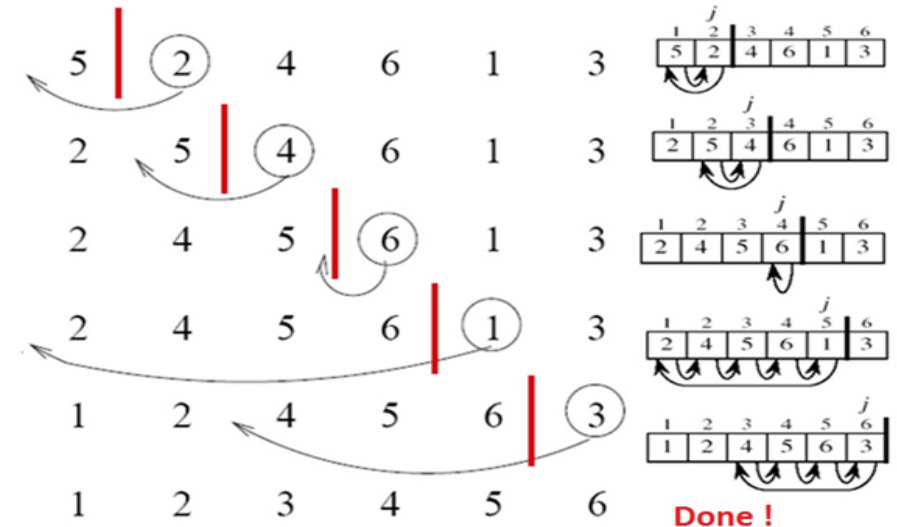
```
    j--;
```

```
}
```

```
    pq[j+1] = key; // insert the element at its correct location
```

```
    (*count)++;
```

```
}
```



### implementation of ascending priority where items are inserted according to the priority

```
struct pqueue pqdelete(struct pqueue *pq,int *count)
{
    // pq is a pointer to the priority queue
    // count is the number of elements in the queue

    int i;
    struct pqueue key;
    // if queue is empty, return a structure with priority -1

    if(*count==0)
    {
        key.data=0;
        key.ptty=-1;
    }
```

implementation of ascending priority where items are inserted according to the priority

```
//delete the first item
//shift the other items to the left
else
{
    key=pq[0];
    for(i=1;i<=*count-1;i++)
        pq[i-1]=pq[i];
    (*count)--;
}
return key; //return the key with the lowest priority
}
```

# Data Structures and its Applications

## Priority Queues- Array Implementation

---



### Implement the following

- Ascending Priority queue where item is inserted at the end and item with the lowest priority is deleted.
- Descending priority queue
- Ascending and Descending Priority queue using a linked list



**THANK YOU**

---

**Dinesh Singh**

Department of Computer Science & Engineering

**dineshs@pes.edu**

**+91 8088654402**