Priority Queue

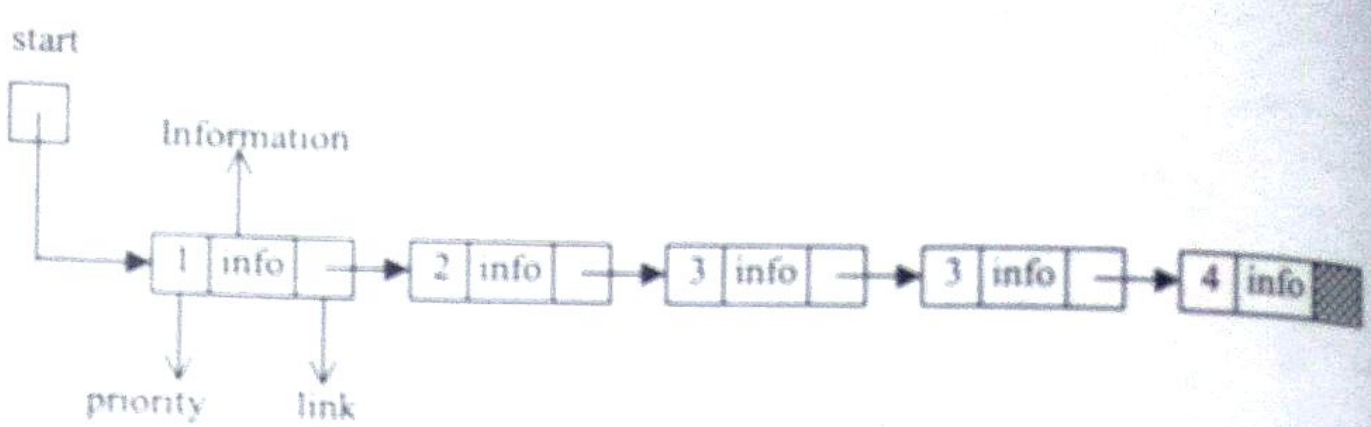
As we have seen earlier, queue is an ordered list of elements in which we can add the element only at one end called rear of the queue and delete the element only at the other end called front of the queue .But in priority queue every element of queue has some priority and based on that priority it will be processed. So the element of more priority will be processed before the element which has less priority.
Suppose two elements have same priority then in this case FIFO rule will follow, means the element which comes first in the queue will be processed first.
In computer implementation, priority queue is used in the CPU scheduling algorithm, in which CPU has need to process those processes first which have more priority.

Linked list implementation of priority queue

Priority queue can be maintained in other ways also but here we are taking linked list approach. First, we have a need to define the structure of elements which will be used fo priority queue-

```
struct pq{
        int priority;
        int data;
        struct pq *link;
        }
```

Here first member of structure is the priority for that element ,second has information an third is link which has address of next element.



Here priority 1 means the highest priority. If priority of an element is 2 then it means it has priority more than the element which has priority 3.

Operation in priority queue

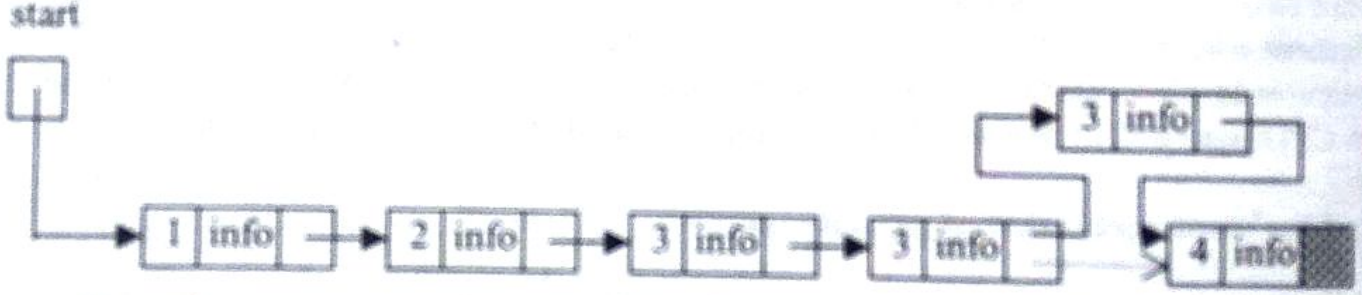
As in queue priority queue has also same two operations ,but in different manner.

1 Add

2 Delete

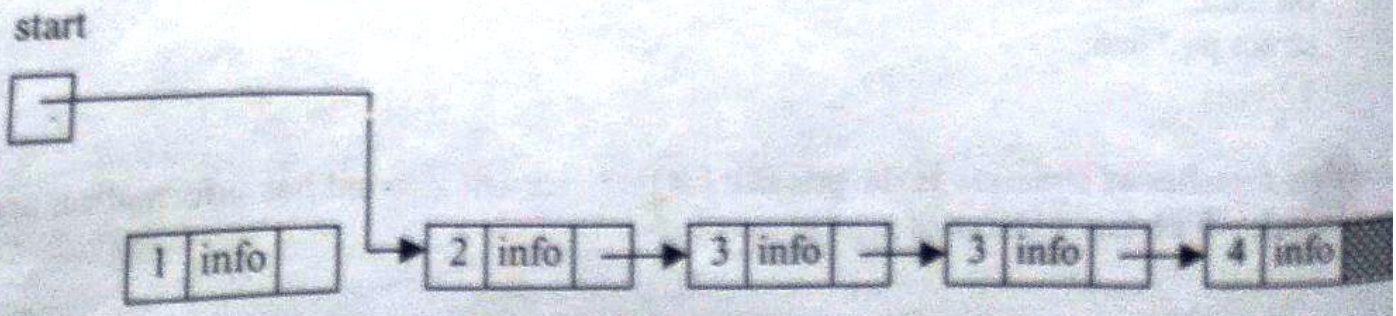
Add operation in Priority Queue

Add operation in priority queue is same as the insert operation in sorted linked list. Here we insert the new element on the basis of priority of element. The new element will be inserted before the element which has less priority than new element.



Delete operation in Priority Queue

Delete operation will be the deletion of first element of list because it has more priority than other elements of queue.



```
/* Program of priority queue using linked list*/
# include<stdio.h>
# include<malloc.h>
struct node
{
        int priority;
        int info;
        struct node *link;
}*front = NULL;

main( )
{
        int choice;
        while(1)
        {
                printf("1.Insert\n");
                printf("2.Delete\n");
                printf("3.Display\n");
                printf("4.Quit\n");
                printf("Enter your choice : ");
                scanf("%d", &choice);

                switch(choice)
                {
                case 1:
                        insert( );
                        break;
                case 2:
                        del( );
                        break;
                case 3:
                        display( );
                        break;
                case 4:
                        exit(1);
                default :
                        printf("Wrong choice\n");
                }/*End of switch*/
        }/*End of while*/
}/*End of main( )*/

insert( )
{
        struct node *tmp,*q;
        int added_item , item_priority;
        tmp = (struct node *)malloc(sizeof(struct node));
        printf("Input the item value to be added in the queue : ");
        scanf("%d",&added_item);
        printf("Enter its priority : ");
```

```
            scanf("%d",&item_priority);
            tmp->info = added_item;
            tmp->priority = item_priority;
            /*Queue is empty or item to be added has priority more than first item*/
            if( front == NULL || item_priority < front->priority )
            {
                    tmp->link = front;
                    front = tmp;
            }
            else
            {
                    q = front;
                    while( q->link != NULL && q->link->priority <= item_priority )
                            q=q->link;
                    tmp->link = q->link;
                    q->link = tmp;
            }/*End of else*/
    }/*End of insert( )*/

    del( )
    {
            struct node *tmp;
            if(front == NULL)
                    printf("Queue Underflow\n");
            else
            {
                    tmp = front;
                    printf("Deleted item is %d\n",tmp->info);
                    front = front->link;
                    free(tmp);
            }
    }/*End of del( )*/

    display( )
    {
            struct node *ptr;
            ptr = front;
            if(front == NULL)
                    printf("Queue is empty\n");
            else
            {
                    printf("Queue is :\n");
                    printf("Priority      Item\n");
                    while(ptr != NULL)
                    {
                            printf("%5d      %5d\n",ptr->priority,ptr->info);
                            ptr = ptr->link;
                    }
            }/*End of else */
    }/*End of display( ) */
```