Priority Queue Implementation Using Array

Done By: Rohit Karunakaran **Roll no:** 58
Date : 05-10-2020

Aim: To implement a priority queue using array

Data Structure used : Priority Queue, Array

Algorithms

1. Algorithm for enqueue

Input: An Array implementation of Priority Queue (P_Q[SIZE]), with front pointing to the first element and rear pointing to the last element in and an element E to be inserted into the queue, with a priority P

Output: The Priority Queue with the element E inserted at the end

Data Structure: Priority Queue

Steps:

```
Step 1: if(rear == SIZE) then
        Step 1: print("The queue is full insertion not possible")
        Step 2: exit(1)
Step 2: else
        Step 1: if(rear == -1) then
                Step 1: front ++
        Step 2: EndIf
        Step 3: ++rear
        Step 4: Q[rear].elem = E
        Step 5: Q[rear].priority = P
Step 3: EndIf
```

2. Algorithm for dequeue

Input: An Array implementation of Queue (Q[SIZE]), with front pointing to the first element and rear pointing to the last element in the queue.

Output: The element E which has the lowest priority is removed form the priority queue

Steps

```
Step 1: if(front == -1) then
        Step 1: print("The Queue is empty")
        Step 2: exit(1)
Step 2: else
        Step 1: ptr = front
        Step 2: lowestPriority = Q[front].priority
```

Step 2: while(ptr<=rear)
Step 1: if(Q[ptr].priority<lowestPriority) then
Step 1: lowestPriority = Q[ptr].priority
Step 2: pos = ptr
Step 2: endif
Step 3: ptr++
Step 3: endWhile
Step 4: E = Q[pos].elem
Step 5: While(pos>front) do
Step 1: pos--
Step 2: Q[pos+1] = Q[pos]
Step 6: EndWhile
Step 7:if(front==rear) then
Step 1: front=-1
Step 2: rear = -1
Step 8:else
fornt = front +1
Step 9: endif
Step 3: endif

Description of the Algorithm:

In this algorithm the time complexity of insertion is O(1) while deletion is O(n).

Program code:

```
/* Priority Queue implemetation using dynamic array
 *  Done By : Rohit Karuankaran
 * */


#include <stdlib.h>
#include <stdio.h>
#define SIZE 32


typedef struct priority_queue
{
    int **Q;
    int size;
    int front;
    int rear;
}pqueue;

void initQueue(pqueue *q)
{
    q->size = SIZE;
    q->Q = (int**) malloc(q->size*sizeof(int*));

    for (int i = 0;i<q->size;i++)
        q->Q[i] = (int*)malloc(2*sizeof(int));
```