Name: Veeransh Shah

Reg No: 221070063

Experiment 6

AIM:-

To implement Priority queues using a Linked List..

THEORY:-

The principle of priority queues finds various applications especially in tech and software fields. It can be summarized by giving an example. when there are multiple application downloads

At that time the task with the most weight i.e. of highest priority is executed first compared to others based on priority, after which other applications download proceed. Priority helps us control flow of how things are executed.

In our experiment we have implemented it with the help of a Linked List. Linked List is a form of Physical data structure just like Array whereas Queue is a logical data structure.

This unique combination of data structures has been used to implement Priority queues.

Application wise Linked List tells us what our next procedure is. This can be commonly seen while operating on a webpage. When you click a button it redirects you to a new page as that

button contains the URL of that next page integrated to it. The way in which there is URL which helps us to navigate in a similar way we have address in each node of linked-list which helps us

to navigate to the next element hence navigating through the entire linked-list.

Let's discuss about how our all the data structures and functions are working in this code:-

WORKING OF THE CODE:-

Any linked list comprises of numerous single elements called node.Each node may contain different fields such as the data, address of the next node and even address of the previous node in case of a doubly linked list.Here in the code, we have used singly linked list that is each node contains an address field and a data field.

As seen in the code, functions ‘Insert’, ‘Display’, ‘Max’ and ‘Extract\_Max’ have been declared inside the class ‘PriorityQueue’.

1)Insert:This function creates a new node by allocating it in a memory field.Based on the value of the data given,it compares to the existing data present in the linked list and places the new node in such a way that all the elements arrange themselves in an ascending order.

2)Display:This function displays each element of the linked list by making the temporary variable ‘temp’ traverse till the end.

3)Max:As the ‘Insert’ function arranges the data in the ascending order, the maximum element is present at the end of the linked list,so this function displays the maximum element(i.e. Element at the end by traversing a temporary variable till the end).

4)Extract\_Max:Since the maximum element is present at the last, this function removes the maximum element from the linked list.This is done by removing the address of the last node which is present in the second-last node and making its address as NULL.

In this way, all these functions operate so as to work as a Priority Queue.

Algorithm:

1. Start:
   * Initialize start pointer to NULL.
2. Function create(n):
   * Take input for the total number of elements n in the minimum priority queue.
   * Loop i from 0 to n-1:
     + Allocate memory for a new node r.
     + Take input for the data value a.
     + Set r->data to a and r->next to NULL.
     + If start is NULL:
       - Set start to r.
     + Else:
       - Initialize pointers p, q to traverse the linked list.
       - While traversing the linked list (p not NULL):
         * If r->data is greater than q->data and less than p->data:

Insert r between q and p.

Break from the loop.

* + - * + Else if r->data is less than q->data and q is the start of the list:

Insert r before q.

Set start to r.

Break from the loop.

* + - * + Else:

Move q to p and p to p->next.

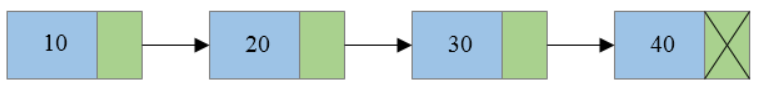
* + - * If r->data is greater than the last element (p->data):
        + Insert r at the end.
      * If r->data is less than the first element (q->data):
        + Insert r at the beginning.

1. Function display():
   * Initialize a pointer m to traverse the linked list, pointing it to start.
   * Output the first element in the minimum priority queue (m->data).
2. Function main():
   * Take input for the total number of elements d in the minimum priority queue.
   * Call the create(d) function to create the minimum priority queue.
   * Call the display() function to output the first element in the minimum priority queue.

EXAMPLE:-

Let the numbers inserted into the linked list be in the sequence:10,40,30,20.

So,the insert function arranges this sequence as:



Upon calling the ‘display’ function, it will output the values as 10,20,30,40 as this is the sequence of the data in the linked list.Now since the linked list is in a sorted manner, when we call the ‘Max’ function, it will return the last element of the linked list.In order to delete the maximum value, we call ‘Extract\_Max’ function which will delete the last value from the linked list and now the maximum value becomes 30.Hence,this linked list works in this fashion as a priority queue.

CONCLUSION:-

In this experiment we have understood how priority queues can be effectively used as one of the ways to sort elements(ascending or descending order). The code helps us to explain how priority queues

(a logical data structure) can we use a linked-list(a physical data structure). To conclude we can say that here we have inserted elements in a linked-list just in a way of how priority queues work.

ScreenShot:

