VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JNANA SANGAMA", BELAGAVI-590018, KARNATAKA

2023-2024

Design and analysis report on[BCS304] "PRIORITY QUEUE"

Submitted in partial fulfillment of the requirements for the award of the degree of **BACHELOR OF ENGINEERING**

IN

INFORMATION SCIENCE AND ENGINEERING

KAVIRAJ (1AT22IS051)
HARSHAVARDHAN R(1AT22IS041)
HARIPRIYA (1AT22IS037)
KARTHIK K (1AT22IS050)

Under the Guidance of

Dr. Jyoti Metan
Assistant Professor
Dept of ISE
Atria I.T

ATRIA INSTITUTE OF TECHNOLOGY

(Affiliated to Visvesvaraya Technological University)
ASKB Campus, Anand Nagar,
Bengaluru-560024

Department of Information Science and Engineering

CERTIFICATE

Certified that the presentation work entitled **"PRIORITY QUEUE"**

carried out by **KAVIRAJ YA(1AT22IS051),HARSHAVARDHAN R(1AT22ISO41),HARIPRIYA D K(1AT222IS037),KARTHIK K(1AT22IS050)** are Bonafide students of Department of Information Science and Engineering, ATRIA I.T., Bengaluru, in partial fulfilment for the award of Degree of **Bachelor of Engineering** in **Information Science & Engineering** of **Visvesvaraya Technological University, Belagavi,** during the academic year **2023-24**. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the department library. The presentation report has been approved as it satisfies the academic requirements in respect of project work prescribed for the said degree.

Dr.Jyoti Metan
Associate
Professor
Department of ISE
Atria I.T.

Dr. Priti Mishra
Professor & HOD
Department of
ISE Atria I.T.

DECLARATION

We,**KAVIRAJ YA(1AT22IS051),HARSHAVARDHAN R(1AT22ISO41),HARIPRIYA D K(1AT222IS037),KARTHIK K(1AT22IS050)**students of 3rd **semester Bachelor of Engineering, Department of Information Science and Engineering, Atria Institute of Technology, Bengaluru,** would hereby declare that the Design and analysis of priority queue presentation entitled **"Priority queue"** has been carried out by us at **Atria Institute of Technology, Bengaluru,** and submitted in partial fulfillment of the course requirement for the award of degree of **Bachelor of Engineering in Information Science and Engineering** of **Visvesvaraya Technological University, Belagavi,** during the academic year **2023-24.**

We further declare that, to the best of our knowledge and belief, the work embodied in this report has not been submitted to any other university or institution for the award of any other degree.

Place: Bengaluru
Date: 26-02-2024

Signature of the students

KAVIRAJ YA(1AT22IS051)

HARSHAVARDHAN R (1AT22IS041)

HARIPRIYA D K(1AT22IS037)

KARTHIK K(1AT22IS050)

ABSTRACT

- Infix notation is the common arithmetic and logical formula notation, in which operators are written infix-style between the operands they
- act on E.g. A +B
- Infix notation is the notation commonly used in arithmetical and logical formulae and statements. It is characterized by the placement of operators between operands—"infixed operators"—such as the plus sign in 2 + 2.

.

- Postfix notation is the useful form of intermediate code if the given language is expressions.
- Postfix notation is also called as 'suffix notation' and 'reverse polish'.
- Postfix notation is a linear representation of a syntax tree.
- In the postfix notation, any expression can be written unambiguously without parentheses.
- The ordinary (infix) way of writing the sum of x and y is with operator in the middle: x * y. But in the postfix notation, we place the operator at the right end as xy *.
- In postfix notation, the operator follows the operand.

ACKNOWLEDGEMENT

We are grateful to our institution, **Atria Institute of Technology**, for having provided us
with the facilities to successfully complete the Presentation on Design and Analysis of Singly Linked List.

We thank **Dr. Priti Mishra, HOD, ISE** for providing us all the necessary facilities for the successful completion of our Presentation. Deadlines play a very important role in the successful completion of the academic project on time, efficiently and effectively.

We take this opportunity to express our deep sense of gratitude to our guide and coordinators **Dr. Jyoti Metan, Associate Professor, Department of ISE** for their valuable guidance and help throughout the course of the academic mini-project. They have always been patient with us and helped immensely in completing the task on hand. We also thank them for their immense support, guidance, specifications & ideas without which seminar would have been completed without full merit.

Last but not least from the Department of Information Science and Engineering, teaching and non-teaching staffs for their constant encouragement, support, patience, and endurance shown during the preparation of this report were remarkable. We also thank the management.

Finally, we thank our parents and friends for their motivation, morale and material support.

KAVIRAJ YA(1AT22IS051)

HARSHAVARDHAN R (1AT22IS041)

HARIPRIYA D K(1AT22IS037)

KARTHIK K(1AT22IS050)

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

- A **priority queue** is a type of queue that arranges elements based on their priority values. Elements with higher priority values are typically retrieved before elements with lower priority values.
- In a priority queue, each element has a priority value associated with it. When you add an element to the queue, it is inserted in a position based on its priority value.

- There are several ways to implement a priority queue, including using an array, linked list, heap, or binary search tree. Each method has its own advantages and disadvantages,
- Priority queues are often used in real-time systems, where the order in which elements are processed can have significant consequences.
- They are also used in algorithms to improve their efficiencies,

CHAPTER 2

PROPERTIES

PROPERTIES :

So, a priority Queue is an extension of the queue with the following properties.

i. Every item has a priority associated with it.

ii. An element with high priority is dequeued before an element with low priority.

iii. If two elements have the same priority, they are served according to their order in the queue.

OPERATIONS

OPERATIONS :

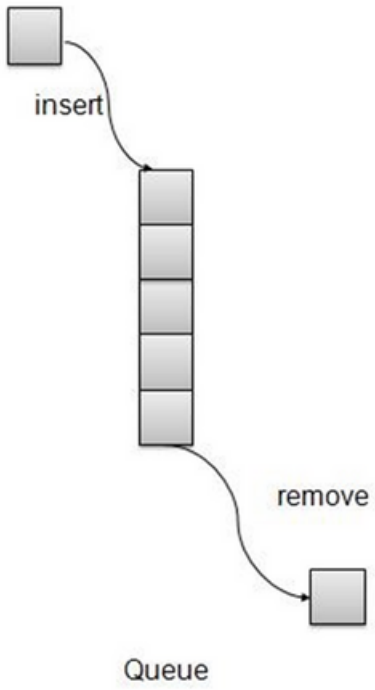
a. Insertion in a Priority Queue

b. Deletion in a Priority Queue

c. Peek in a Priority Queue

A. Insertion in a Priority Queue:

- When a new element is inserted in a priority queue, it moves to the empty slot from top to bottom and left to right. However, if the element is not in the correct place then it will be compared with the parent node. If the element is not in the correct order, the elements are swapped. The swapping process continues until all the elements are placed in the correct position.



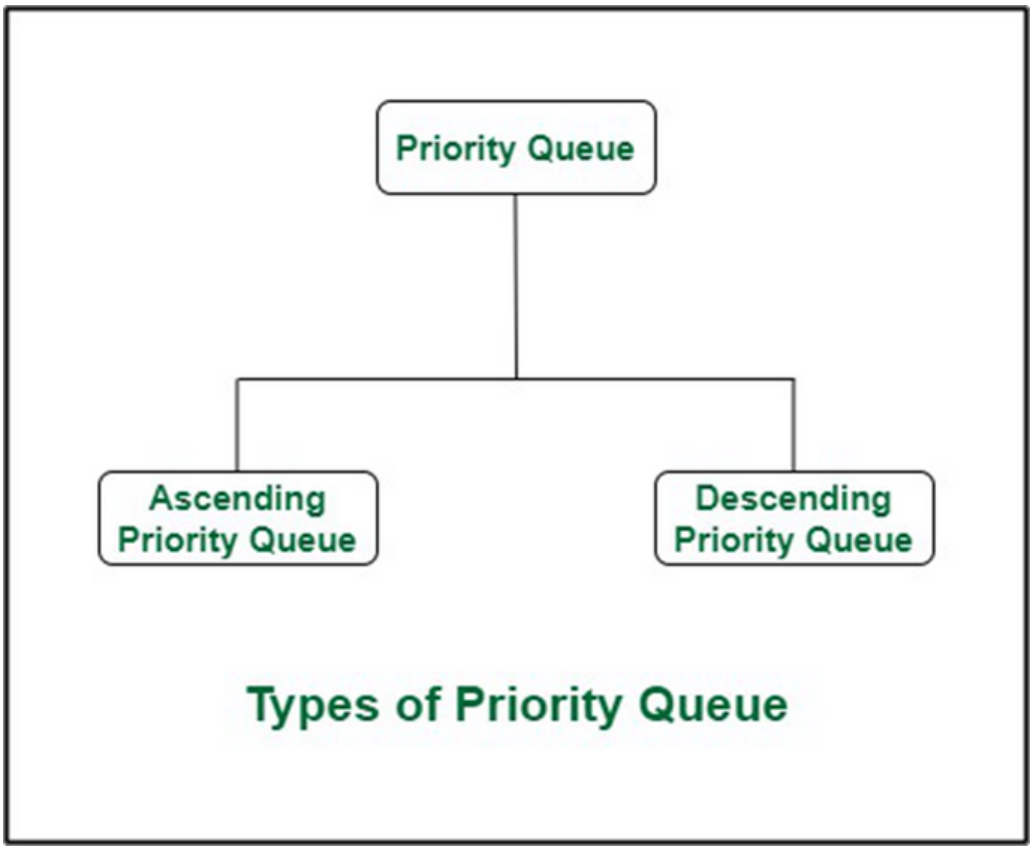
B. Deletion in a Priority Queue :

- As you know that in a max heap, the maximum element is the root node. And it will remove the element which has maximum priority first. Thus, you remove the root node from the queue. This removal creates an empty slot, which will be further filled with new insertion. Then, it compares the newly inserted element with all the elements inside the queue to maintain the heap invariant.

C. Peek in a Priority Queue:

- This operation helps to return the maximum element from Max Heap or the minimum element from Min Heap without deleting the node from the priority queue.

Types of Priority Queue:



Ascending Order Priority Queue:

- As the name suggests, in ascending order priority queue, the element with a lower priority value is given a higher priority in the priority list.
- For example, if we have the following elements in a priority queue arranged in ascending order like 4,6,8,9,10. Here, 4 is the smallest number, therefore, it will get the highest priority in a priority queue and so when we dequeue from this type of priority queue, 4 will remove from the queue and dequeue returns 4.

Descending Order Priority Queue:

- The root node is the maximum element in a max heap, as you may know. It will also remove the element with the highest priority first. As a result, the root node is removed from the queue. This deletion leaves an empty space, which will be filled with fresh insertions in the future. The heap invariant is then maintained by comparing the newly inserted element to all other entries in the queue.

Implementation of Priority Queue:

Priority queue can be implemented using the following data structures:

- Arrays
- Linked list
- Heap data structure
- Binary search tree

Implementation using array:

- A simple implementation is to use an array of the following structure.

```
struct item {
        int item;
        int priority;
}
```

- enqueue(): This function is used to insert new data into the queue.
- dequeue(): This function removes the element with the highest priority from the queue.
- peek()/top(): This function is used to get the highest priority element in the queue without removing it from the queue.

Implementation using linked list:

- In a LinkedList implementation, the entries are sorted in descending order based on their priority. The highest priority element is always added to the front of the priority queue, which is formed using linked lists. The functions like **push()**, **pop()**, and **peek()** are used to implement a priority queue using a linked list and are explained as follows:
- **push():** This function is used to insert new data into the queue.
- **pop():** This function removes the element with the highest priority from the queue.
- **peek() / top():** This function is used to get the highest priority element in the queue without removing it from the queue.

Implementation using heaps:

Binary Heap is generally preferred for priority queue implementation because heaps provide better performance compared to arrays or LinkedList. Considering the properties of a heap, The entry with the largest key is on the top and can be removed immediately. It will, however, take time O(log n) to restore the heap property for the remaining keys. Operations on Binary Heap are as follows:

- **insert(p):** Inserts a new element with priority p.
- **extractMax():** Extracts an element with maximum priority.
- **remove(i):** Removes an element pointed by an iterator i.
- **getMax():** Returns an element with maximum priority.
- **changePriority(i, p):** Changes the priority of an element pointed by **i to p**.

Applications:

- CPU Scheduling
- Graph algorithms like , Prim's Minimum Spanning Tree, etc.
- Stack Implementation
- All queue applications where priority is involved.
- Data compression in Huffman code
- Event-driven simulation such as customers waiting in a queue.
- Finding Kth largest/smallest element.

Advantages:

- It helps to access the elements in a faster way.
- The ordering of elements in a Priority Queue is done dynamically.
- Efficient algorithms can be implemented.
- Included in real-time systems.

Disadvantages:

- High complexity..
- High consumption of memory.
- It is not always the most efficient data structure.
- At times it is less predictable