NORTH SOUTH UNIVERSITY

Department of Electrical and Computer Engineering

Assignment – 04

Name	: Joy Kumar Ghosh
Student ID	: 2211424 6 42
Course No.	: CSE 225
Course Title	: Data Structures and Algorithm
Section	: 06
Date	: 05 April 2023

Ans. to the ques. no. 01

A linked list can be implemented to behave as a Stack or Queue by imposing certain restrictions on its operations. Here are restrictions for the Add and Remove processes for both a Stack and a Queue,

Stack:

In a linked list, we can add new items at either the list's beginning or the end. For, Stack, if we add (Push) a new item at the beginning of the list, we must remove (POP) the item from the beginning. Also, if ~~you~~ we add a new item at the end, then we must remove the item from the back. That means First in, First out or Last in, Last out.

Queue:

In Queue, if we add a new item at the beginning of the list, we must remove the item from the end. As well as, if we add a new item at the end of the list,

we must remove the item from the beginning. That means First in, Last out or Last in, First out.

By imposing these restrictions, a linked list can be used to implement either a Stack or a Queue data structure.

Ans. to the ques. no.02

Array List and Linked Lists are data structures that can store collections of elements. However, they have different characteristic and are better suited for different types of operations.

Memory allocation: An Array List allocates a continuous memory block for its elements, while a Linked List stores its elements in non-continuous blocks. Array size is defined when declared, but the linked list is undefined and unlimited untill memory goes full.

Insertion and deletion: Insertion and deletion operations are faster in a Linked List than in an Array List. In a Linked List, insertion and deletion are performed constantly by modifying a few pointers. At the same time, in an Array List, these operations required shifting all the elements in the array after the insertion or deletion point. Specially in sorted Array its so lengthy process.

Random Access: Random access, which accesses elements at a specific index, is faster in an Array List than in a Linked List. In a Linked List, we must traverse individually until the desired element is found.

Memory usage: A Linked List uses more memory than an Array List due to the overhead of storing pointers to the next and previous elements.

In summary, Array Lists are better suited for operations

that required random access and have a fixed size. At the same time, Linked Lists are better suited for operations that require the insertion and deletion of elements and for ~~sui~~ situation where memory wage is not a significant concern.

Ans. to the ques. no. 03

In implementing Queue wing an Array data structure, it is impossible to define the empty or full states because the value of front and rear are the same for both situations. That's why we reserved an index as the front element.

So, if we found

front == rear

that means the array is empty,

and if we found

(rear + 1) % maxQue == front,

that means the array is full.

Ans. to the ques. no. 04

1. Searching an item in a sorted array:

The worst-case running time for searching an item in a sorted array is $O(\log n)$. Because the binary search algorithm can eliminate half of the remaining elements in each iteration, resulting in logarithmic time complexity.

2. Inserting an item into a sorted array:

The worst-case running time for inserting an item into a sorted array is $O(n)$. Because in the worst-case scenario, the insertion requires shifting all the elements to the right of the insertion point by one position.

3. Inserting an item into a Queue:

The worst-case running time for inserting an item into a Queue is $O(1)$, which means that it takes constant time because inserting an item into a Queue involves

adding an element at the end of the Queue, this can be done in constant time.

4. Remove an item from a Stack:

The wrost-case running time for removing an item from a Stack is also O(1). As Queue, here in a Stack we just need to remove the topmost item, which can be done in constant time regardless of the size of the Stack.

Source of help: Lecture Slide & some from chatGPT