NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES

DATA STRUCTURE PROJECT REPORT

VISUALISING DATA STRUCTURE

Team Members	1. Muhammad Hamza (21K-3815) 2. Huzaifa Awan (21K-3835) 3. Zayn Khan (21K-3828)
Course Instructor	Ms. Sobia Iftikhar
Submission Date	08-December-2022

Problem Statement:

As a student, I have faced difficulty understanding data structure concepts that how sorting, linked lists, graphs, and other structures. What's the behind story of all in order to accomplish the task? So in order to understand it easily with the input of the data of their own we have come to the solution of creating GUI based JAVA application that helps the user to understand how the data structures work.

Description about used Data Structure:

Sorting:

We have created a template using JAVA swings library to graphically representing the bubble sort algorith on the random integers . The use of array of size 10 with the threads along with the color to recognise how the sorting is done , how the elements are swapping inside the array. The use of threads allow the application to several seconds in order that the user can easily identified how the sorting is done.

Bubble Sort :

Time complexity = O(n^2)

Space complexity = O(n)

Insertion Sort:

Time complexity = O(n^2)

Space complexity = O(n)

Selection Sort :

Time complexity = O(n^2)

Space complexity = O(n)

Stack:

Using the JAVA inbuild GUI libraries we have created template as an array based where user can input the value and it stores inside the array visible to the user. Stacks in Data Structures is a linear type of data structure that follows the LIFO (Last-In-First-Out) principle and allows insertion and deletion operations from one end of the stack data structure, that is top.

Push button allows user to input the value to the first index.

Time complexity = O(1)

Space Complexity = O(n)

Pop button allows the user to delete the last index value.

Time complexity = O(1)

Space Complexity = O(n)

Peek button shows the user the last index value of the array.

Time complexity = O(1)

Space Complexity = O(n)

Queue:

A queue is defined as a linear data structure that is open at both ends and the operations are performed in First In First Out (FIFO) order. We define a queue to be a list in which all additions to the list are made at one end, and all deletions from the list are made at the other end.

Queue is used by the help of array and is represented using JAVA swings library for graphical purpose.

Peek button shows the element at the front of the queue .

Time complexity = O(1)

Space complexity = O(n)

Enque button allows the user to input the value at the rear of queue.

Time complexity = O(1)

Space complexity = O(n)

Deque allows the user to delete the element from the front of the queue.

Time complexity = O(1)

Space complexity = O(n)

Linked List

Singly Linked List:

A linked list is the most sought-after data structure when it comes to handling dynamic data elements. A linked list consists of a data element known as a node. And each node consists of two fields: one field has data, and in the second field, the node has an address that keeps a reference to the next node.

Using Java Graphical library allow easily create interface for the user to understand how link list works.

Add at first button allows the user to enter element at the head of the list .

Time complexity = O(1)

Space Complexity = O(n)

Add at last button allows the user to enter the element at the last of the list.

Time complexity = O(n)

Space Complexity = O(n)

Add at Position button allows the user to enter the element at the position in the list

Time complexity = O(n)

Space Complexity = O(n)

Remvoe at first button allows the user to delete an element from the head of the list and update the head of the list.

Time complexity = O(1)

Space Complexity = O(n)

Remove at last button delete the last element of the list and updating the list.

Time complexity = O(n)

Space Complexity = O(n)

Remove by position button allows the user to enter the position from where to delete and delete the element and update the list.

Time complexity = O(n)

Space Complexity = O(n)

Doubly Linked List

a doubly linked list is a linked data structure that consists of a set of sequentially linked records called nodes. Each node contains three fields: two link fields (references to the previous and to the next node in the sequence of nodes) and one data field.

Using Java Graphical library allow easily create interface for the user to understand how link list works.

Add at first button allows the user to enter element at the head of the list .

Time complexity = O(1)

Space Complexity = O(n)

Add at last button allows the user to enter the element at the last of the list.

Time complexity = O(n)

Space Complexity = O(n)

Add at Position button allows the user to enter the element at the position in the list

Time complexity = O(n)

Space Complexity = O(n)

Remvoe at first button allows the user to delete an element from the head of the list and update the head of the list.

Time complexity = O(1)

Space Complexity = O(n)

Remove at last button delete the last element of the list and updating the list.

Time complexity = O(n)

Space Complexity = O(n)

Remove by position button allows the user to enter the position from where to delete and delete the element and update the list.

Time complexity = O(n)

Space Complexity = O(n)

Graph Traversal:

Graphs in data structures are non-linear data structures made up of a finite number of nodes or vertices and the edges that connect them. Graphs in data structures are used to address real-world problems in which it represents the problem area as a network like telephone networks, circuit networks, and social networks.

Java gui library allows the creation of nodes and the color representing if the node is visited or not .More over the connection between nodes are created using the matrix screen where user checkbox the nodes which they want to have connection of .

DFS button algorithm traverse the graph . It starts at the rrot node and examine each node using stacks data structure and if it is visited or not as far as possible before backtracking

Time complexity = O(V+E)

Space Complexity = O(n)

BFS button uses algorithm traverse the root node and explores all the neighboring nodes storing them in the queue . It select the node define by the user and explores all the unexplored nodes.

Time complexity = O(V+E)

Space Complexity = O(n)

The efficiency of the Project:

The project does not require any sort of high processor or storage space , it can easily run without any internet connection and the user can easily run it without need of any prior knowledge . The project doesnot require any external database or cloud server which results in application to be run at minimal cost and time .

The overall time complexity of the project is O(n).

The overall space complexity of the project is O(n).

System vs External existing System:

The project has been inspired from the webpage https://visualgo.net/en.

The difference between our system and the webpage is that the webpage for accessing needs internet connection and for graphics and traversing purpose it requires some amount of RAM and Processor to run easily. Whereas , our project does not need any internet connection and minimal amount of ram and processor for running as it is a Desktop application.

The space and time complexity as compared to the other project is almost the same as both work on O(n) average time and space principal.