Merchant Monetary System

Data Structure

Project Supervisor

Mr. Samyan Qayyum Wahla

Group ID ($G11$)

Project Member

Syed Hashir	2021-CS-1
Kabir Ahmed	2021-CS-4
M. Hamad Hassan	2021-CS-33

Department of Computer Science
University of Engineering and Technology, Lahore
Pakistan

Data Strucuture

The following section shows the reason for choosing the data structure in the particular use case with a brief explanation.

Use Case IDs	U01,U02,U03,U04,U05,U06,U07,U08,U09,U010,U11,U12,U13,U14, U15,U16,U17,U18,U21,U22,U23,U24,U25,U26,U30,U31
Data Structure Used	Linked List
Time Complexity	In Worst Case: Search: O(n), Insertion: O(1), Deletion: O(n)
Space Complexity	O(n)

Pseudocode	**Search:**
	LIST-SEARCH(L,k)
	x=L.head
	while x $\neq$ NIL and x:key $\neq$ k
	    x = x.next
	return x
	Insert:
	LIST-INSERT(L, x)
	x.next=L.head
	if L.head $\neq$ NIL
	    L.head.pre = x
	L.head = x
	x.pre = NIL
	Delete:
	LIST-DELETE(L,x)
	if x.pre $\neq$ NIL
	    x.pre.next=x.next
	else L.head D x.next
	if x.next $\neq$ NIL
	    x.next.pre =x.pre

Justification for the use of data structure	In mentioned use case required a linear-dynamic data structure. Doubly LinkedList provides an efficient way to search the specific information from a large amount of data and then compare it with input information to produce the required result. It helps to store and delete the data fastly. It allows you to move back and forth in the list to get the required result.
Available choices	Array List,Hash Table
Comparison	The array list worst and average case time complexity is O(n). It takes contiguous memory. The hash table is best in the average case, but in the worst case time, complexity rise to O(n). It takes contiguous memory for storing the hash function value. In the average and worst case, the linked list insertion and deletion take O(1) time. In the average and worst case, it takes O(n) time for deletion. It did not require contiguous memory allocation.Array list, hash table, and linked list space complexity O(n) are the same.