

Linked Allocation of Data with various Algorithms and Future Applications

Mr. Vinay Soni

Lecturer

Department of Computer Engineering

Sigma Institute of Technology Gujarat Technological University

Abstract— This paper will give you the basic understanding for studying Linked Data structure and its algorithms with future applications in various fields.

Key words: Linked List, Linked List Algorithms, Insertion & Deletion in Linked List

7	Searching an element in the Linked List
8	Counting the number of nodes in Linked List

Table 1: list of operations on linked list

I. INTRODUCTION

As stated in Project title We will directly point to the Actual topic and our topic is all about Linked List which is somewhat confusing to some entry level students who are trying to study Data Structures. In brief I will tell you what Data structure is, One should have to store his/her data in specific manner that whenever any other person in future trying to access those data and want to use/edit, He/she can do all the tasks in Space & Time saving manner(Space & Time Complexity must be minimized). So, In this path We are having Linear and Non-Linear types and our topic Linked List is coming under Linear type. We are moving to the point directly.

A. What is Linked List?

Till the introduction of this Linked List We are storing our Data Contiguously one after another. Linked Lists differ in this fashion where We are not saving our data linearly but calling memory to find free space where we can store our data and after every data entered we are asking for next memory location where we can store our data. Two things will be required to create this structure that is data and the next location of that data which we consider as pointer for next data. Combining both terms and we are making a Basic structure for Linked List that is called NODE. NODE contains two parts, one for data and other for location of next data. In this manner we are creating our long Linked List and thousands of data which are linear but are not contiguous.

For example, we are comparing Linked List with an interesting game TREASURE HUNT. We are given a clue to find first data and other data will be found on the base of clue that current data is giving us and we are reaching at the last data by continuously doing the same searching. We are going to understand the whole topic by the same example. We have 7 basic operations to perform on linked list listed below in table. There could be some other operations on Linked List and also to extended version of Linked List that is Doubly Linked List which is not covered in this text.

Sr No.	Operation
1	Insert a node at the beginning of the List
2	Insert a node at the end of the List
3	Insert a node before given node
4	Insert a node after a given node
5	Delete a node from the beginning of the List
6	Delete a node from the end of the List

II. IN DEPTH

A. Insertion

For Insertion We must have space where we can add a new data member. So first of all for any insertion(for first four rows in Table 1) We have to check for the availability in the memory. For our TREASURE HUNT game if We want to add new clue and want to expand our game, we must have space and ideas with us. So we are checking for space and if its available then only we can proceed, say no otherwise.

I am going to write steps here but its not with all the details like actual algorithms.

- 1) Step 1: Check for the availability in memory, If answer is No then STOP else go to next step.
- 2) Step 2: Search for the STARTING POINT of the game, if its not there then our inserted data will be known as STARTING POINT otherwise go to next step.
- 3) Step 3: If STARTING POINT is available then We can add as per our requirements such as
 - If we want to insert at the end then we only have to update last node's next clue part with new location and add new data to new location and update new location's next clue part with end of the list that is NULL.
 - If we want to insert at the beginning then we have to update our new location with STARTING POINT, inserting data there and updating its next clue part with previous STARTING POINT.
 - If we want to insert before any specific location then we have to update next clue part of the location just before our specified location with new location and updating our new location's next clue part with our specified location.
 - If we want to insert after a given location then we have to update our specified location's next clue part with new location and new location's next clue part with next location of our specified location.

In such manner we have inserted new location in our game in which a data part and next clue part is updated in various ways. After that we are going to delete some location from our game that is deleting any NODE from our Linked List. More fun is ahead. Let see.

B. Deletion

Now we are at the stage where We want to clear some location in the game that is deleting something from our Linked List. Deleting from our game is easier than that of inserting something new as always. In simple manner we are

bypassing that location by changing previous location's next clue part. First of all we have to check out that whether the game is having its STARTING POINT or not. If Its not there, We don't have game or location to delete. Come on.. See Step by Step:

- 1) Step 1: Check for STARTING POINT, If Its not there then say STOP else go to next step.
- 2) Step 2: If only STARTING POINT is found and we cannot find any clue in it then we have to clear that STARTING POINT and making our game STOP. Otherwise we have other conditions such as:
 - If we want to delete from end then Searching on next clue part of all locations and if we find any NULL or we can say nothing written in it then we can justify it as last location and deleting it by clearing that location value from the previous location's next clue part.
 - If we want to delete from the front then its too simple. Make second location as STARTING POINT and bypass that old STARTING POINT. Easy.
 - If we want to delete from any intermediate position we can do it in same manner as done in insertion.

C. Searching

Searching from our game is somewhat cross check type thing we are doing. We are checking whether the data we wanted to insert is there or not. Also after deleting the data there is check contains whether its deleted or not. Result FOUND or NOT FOUND will come at the end of the procedure.

We are specifying some data value and checking all locations one by another searching our specified data. If its found we will return FOUND and after whole traversal if we couldn't able to find then the message will be NOT FOUND.

D. Counting

Counting number of locations in our game is important in many aspects so its necessary to do. But its too much easy as we have done it previously.

Simply go through all the locations one by one and find a location in which there is no next clue part that is our last location. Till then increment our counter by one. Done!! we found number of locations.

III. FUTURE USAGES AND CONCLUSION

As we have discussed so far the Linked allocation of data is looking somewhat complex method to store data like TREASURE HUNT game but for computer system Its very much useful to store in such manner so that computer memory is used in efficient manner. There are many usages of this type of structure in various fields, infect in computer physical memory storage, directory need to remember only first address and rest of the part will be self-running. In future technology all the storage and other tasks will be done in linked manner as it is smart and efficient way to store, retrieve and delete anything from our List.

ACKNOWLEDGMENT

I acknowledge my college teacher Swapnil Shah and my colleagues to support me in choosing and understanding this complex topic. Thank you.

REFERENCES

- [1] Antonakos, James L.; Mansfield, Kenneth C., Jr. (1999). Practical Data Structures Using C/C++. Prentice-Hall.
- [2] Collins, William J. (2005) [2002]. Data Structures and the Java Collections Framework. New York: McGraw Hill.
- [3] Cormen, Thomas H. Charles E. Leiserson Ronald L. Rivest Clifford Stein (2003) Introduction to Algorithms
- [4] Data and file Structure using C, Thareja Reema, Oxford University press New Delhi