UNIVERSITY OF CALOOCAN CITY
COMPUTER ENGINEERING DEPARTMENT

Data Structure and Algorithm

Laboratory Activity No. 5 (Makeup class)

Singly Linked List

Submitted by:
Maringal, Czer Justine D.

Instructor:
Engr. Maria Rizette H. Sayo

August 11, 2025

I. Objectives

- To equip a clear understanding of the singly linked list data structure, including its components, operations, and implementation in Python.
- To compare its characteristics with arrays
- To analyze its efficiency in terms of time and space complexity
- To demonstrate its practical applications in solving real world computing problems.

II. Methods

Make research about Stingly Linked List:

1. What is a singly linked list, and how does it differ from an array?
2. When would you prefer a linked list over an array, and vice versa?
3. How are linked lists used in real-world applications (e.g., browser history, undo functionality)?
4. Cite your reference/s

III. Results

1. A singly linked list is a linear data structure where each element (called a node) contains two parts:

- Data: the value stored in the node.
- Pointer (next): a reference to the next node in the sequence.

Differences from an Array:

- Memory storage: Linked lists use dynamic memory allocation; arrays use contiguous memory blocks.
- Access time: Arrays allow constant-time random access (O(1)), while linked lists require sequential traversal (O(n)).
- Insertion/Deletion: Linked lists are faster for insertion/deletion in the middle (O(1) if the node is known), arrays require shifting elements (O(n)).

2. I would prefer a linked list over an array when my program requires frequent insertion or deletion of elements, especially in the middle or at the beginning of the sequence. Because linked lists can handle these operations more efficiently without shifting elements.

On the other hand, I would prefer an array when fast random access to elements is important, since arrays allow direct indexing in constant time. Arrays are also suitable when the number of elements is fixed or changes rarely, as this avoids the overhead of dynamic memory management.

3. How are linked lists used in real-world applications?

- Browser history navigation: Moving forward/backward through visited pages.
- Undo functionality in text editors: Storing previous states in a stack-like linked list.
- Music playlists: Navigating forward and backward through songs.
- Dynamic memory management: Operating systems use linked lists to keep track of free and allocated memory blocks.

4. Citations:

Python Software Foundation. (n.d.). Python Data Structures.
https://docs.python.org/3/tutorial/datastructures.html

GeeksforGeeks. (n.d.). Introduction to Linked List.
https://www.geeksforgeeks.org/linked-list-set-1-introduction

IV. Conclusion

The singly linked list is a fundamental data structure. Unlike arrays, it allows efficient insertion and deletion operations without the need for shifting elements or allocating large contiguous memory blocks. While it may not offer the fast random access capability of arrays, its dynamic nature makes it ideal for applications where the size of the data changes frequently. Understanding the strengths and limitations of singly linked lists enables programmers to choose the most appropriate data structure for solving specific problems in both academic and real-world scenarios.