Data Structure and Algorithm

Assignment No. 1

Singly Link List

|  |  |
| --- | --- |
| *Submitted by:* | *Instructor:* |
| Hermosura, Leigh B. | Engr. Maria Rizette H. Sayo |

August 15, 2025

# Questions

1. What is a singly linked list, and how does it differ from an array?

* A singly linked list is a linear data structure in which each node contains a data field and a reference (or pointer) to the next node in the sequence, which forms a series of connected nodes [1]. The difference between an array and a singly linked list is that in an array, the data are stored in a single block of memory, whereas a singly linked list stores its elements in separate memory locations, with each node linked to the next. This difference affects how elements are accessed: arrays allow direct access to any element by indexing, while singly linked lists need to start from the head node and go through each node sequentially, making access times slower [2].

1. When would you prefer a linked list over an array, and vice versa?

* I would prefer to use a linked list over an array when the data size is unknown and it frequently changes during run time. Arrays struggle with data insertion and deletion due to its fixed size as it’s unable to accommodate new elements or remove gaps. On the other hand, I would prefer arrays when limited use of memory is required, as arrays do not need additional memory to store pointers to reference other elements [3].

1. How are linked lists used in real-world applications (e.g., browser history, undo functionality)?

* Linked lists are used in various real-world applications where dynamic and sequential data handling is required. For example, navigating through browser history in which each tab is linked together, in order to view the previous or the next link. Another is the undo/redo functionality in software such as text editors, graphic design tools, and drawing applications. where each action represents a node in a doubly linked list [4]. Linked lists are also well-suited for implementing sequential data structures like queues, where elements are processed in a first-in, first-out (FIFO) order [5].

# Conclusion

Singly linked lists offer a flexible and efficient way to manage dynamically changing data, especially in scenarios where frequent insertions and deletions are required. Unlike arrays, linked lists do not require contiguous memory and can grow or shrink during runtime without the need for resizing. While they may have slower access times compared to arrays due to their sequential nature, their advantages in memory management and dynamic operations make them ideal for many real-world applications such as navigation history, undo/redo functionalities, and queue implementations. The choice between arrays and linked lists ultimately depends on the specific needs of the application, including memory constraints, data size variability, and operation types.

**References**

[1] GeeksforGeeks, “What is Linked List?,” *GeeksforGeeks*, <https://www.geeksforgeeks.org/dsa/what-is-linked-list/> (accessed Aug. 15, 2025).

[2] BYJU'S, “Difference Between Array and Linked List,” *BYJU'S*, <https://byjus.com/gate/difference-between-array-and-linked-list/> (accessed Aug. 15, 2025).

[3] Stack Overflow, “Array vs single linked list vs double link list,” *Stack Overflow*, <https://stackoverflow.com/questions/48469941/array-vs-single-linked-list-vs-double-link-list> (accessed Aug. 15, 2025).

[4] GeeksforGeeks, “Applications of Linked List Data Structure,” *GeeksforGeeks*, <https://www.geeksforgeeks.org/dsa/applications-of-linked-list-data-structure/> (accessed Aug. 15, 2025).

[5] S. S. Roy, “Why use a linked list instead of an array?,” *Medium*, <https://medium.com/nerd-for-tech/why-use-a-linked-list-instead-of-an-array-f75cdebaad22> (accessed Aug. 15, 2025).