Bellevue University

Differences Between Arrays and Linked Lists

Timothy Jelinek

CS420-T301 Data Structures

6/15/2024

Arrays are collections of similar data types. Applications of arrays include storing and accessing data, sorting, searching, representing matrices, stacks and queues, graphing, and dynamic programming. There are many advantages of using arrays, such as efficient access to elements, fast data retrieval, they are memory efficient, versatile, easy to implement, and compatible with hardware. Arrays allow for accessing any element in the array, no matter how big the array is, in the same amount of time. Arrays are also very versatile and provide the use for any data type, however, you can only use one data type per array. It is easy to create arrays, so they are beginner friendly. There are also disadvantages to using arrays, such as them being a fixed size, having memory allocation issues, insertion and deletion issues, waste space in memory, using only the same data type, and lack of flexibility. Arrays have a set size when they are created, which means new arrays need to be created and have the data copied from the previous array that is having data added onto it, resulting in more time and resources being used.

Linked lists are collections of data with the assistance of nodes. Applications of linked lists include implementing stacks and queues, representing graphs and trees, representing sparse matrices, manipulating polynomials, finding paths in networks, and improving the performance of algorithms that insert and delete items. There are many advantages to linked lists such as, dynamic size, efficient inserting and deleting of elements, being memory efficient, easy to implement, flexible, and easy to navigate. Linked lists allow for the addition or removal of elements when you want to, dynamically. Linked lists are easier to use than trees and graphs, which make them an excellent tool for beginners. They can also be used easily to find specific elements. There are also disadvantages to linked lists including slow accessing time of elements, use of pointers, higher overhead, cache inefficiency, and requiring extra memory. It is unfortunate that accessing elements can be slow in linked lists, as it takes an O(n) operation as opposed to arrays using an O(1) operation. Linked lists also use pointers, which are complex and make debugging and maintenance difficult. They also need extra memory for storing pointers for each node they have, which can add a lot of memory in larger data sets.

After reviewing arrays and linked lists, linked lists are better for adding and removing elements, while arrays are better when having bigger data sets. Efficiency and memory are both priorities when using data structures, and linked lists are better in both aspects as long as there isn’t too much data being stored in them. Finding data in arrays is much quicker, as it doesn’t matter how big an array is, or how far into the array the data type is, it takes the same amount of time to find it, while linked lists are slower by using the O(n) operation. Both arrays and linked lists are very useful, but they have their uses.

Sources:

GeeksforGeeks. (2023, February 21). *Applications, Advantages and Disadvantages of Array*. GeeksforGeeks. <https://www.geeksforgeeks.org/applications-advantages-and-disadvantages-of-array-data-structure/?ref=header_search>

GeeksforGeeks. (2023, April 5). *Applications, Advantages and Disadvantages of Linked list*. GeeksforGeeks. https://www.geeksforgeeks.org/applications-advantages-and-disadvantages-of-linked-list/