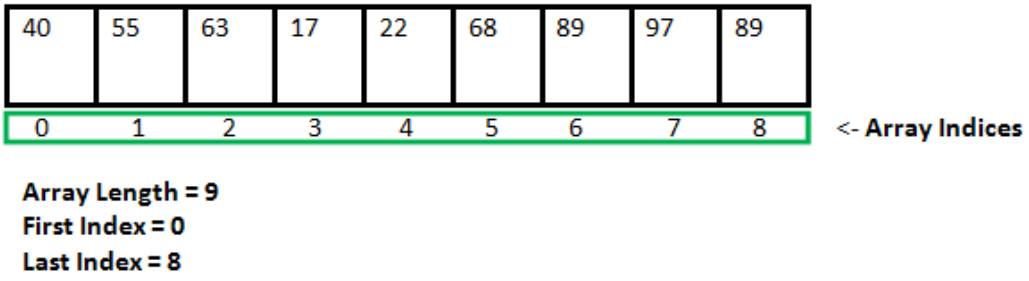
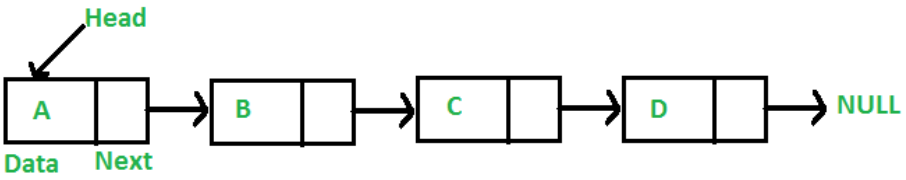
Linked List vs Array

Arrays store elements in contiguous memory locations, resulting in easily calculable addresses for the elements stored and this allows faster access to an element at a specific index. Linked lists are less rigid in their storage structure and elements are usually not stored in contiguous locations, hence they need to be stored with additional tags giving a reference to the next element. This difference in the data storage scheme decides which data structure would be more suitable for a given situation.



Data storage scheme of an array



Data storage scheme of a linked list

Major differences are listed below:

- **Size:** Since data can only be stored in contiguous blocks of memory in an array, its size cannot be altered at runtime due to the risk of overwriting other data. However, in a linked list, each node points to the next one such that data can exist at scattered (non-contiguous) addresses; this allows for a dynamic size that can change at runtime.
- **Memory allocation:** For arrays at compile time and at runtime for linked lists. but, a dynamically allocated array also allocates memory at runtime.

- **Memory efficiency:** For the same number of elements, linked lists use more memory as a reference to the next node is also stored along with the data. However, size flexibility in linked lists may make them use less memory overall; this is useful when there is uncertainty about size or there are large variations in the size of data elements; memory equivalent to the upper limit on the size has to be allocated (even if not all of it is being used) while using arrays, whereas linked lists can increase their sizes step-by-step proportionately to the amount of data.
- **Execution time:** Any element in an array can be directly accessed with its index; however in the case of a linked list, all the previous elements must be traversed to reach any element. Also, better cache locality in arrays (due to contiguous memory allocation) can significantly improve performance. As a result, some operations (such as modifying a certain element) are faster in arrays, while some others (such as inserting/deleting an element in the data) are faster in linked lists.
- **Insertion :** In array insertion operation takes more time but in linked last these operations are fast. For example, if we want to insert an element in array at end position in array and array is full then we copy the array into another array and then we can add a element whereas if linked list is full then we find the last node and make it next to the new node

ARRAY	LINKED LISTS
1. Arrays are stored in contiguous location.	1. Linked lists are not stored in contiguous location.
2. Fixed in size.	2. Dynamic in size.
3. Memory is allocated at compile time.	3. Memory is allocated at run time.
4. Uses less memory than linked lists.	4. Uses more memory because it stores both data and the address of next node.
5. Elements can be accessed easily.	5. Element accessing requires the traversal of whole linked list.
6. Insertion and deletion operation takes time.	6. Insertion and deletion operation is faster.

Array vs Linked List

Following are the points in favour of Linked Lists.

1. The size of the arrays is fixed: So we must know the upper limit on the number of elements in advance. Also, generally, the allocated memory is equal to the upper limit

irrespective of the usage, and in practical uses, the upper limit is rarely reached.

2. Inserting a new element in an array of elements is expensive because room has to be created for the new elements and to create room existing elements have to be shifted.

   For example, suppose we maintain a sorted list of IDs in an array id[ ].

   id[ ] = [1000, 1010, 1050, 2000, 2040, …..].

   And if we want to insert a new ID 1005, then to maintain the sorted order, we have to move all the elements after 1000 (excluding 1000).

   Deletion is also expensive with arrays unless some special techniques are used. For example, to delete 1010 in id[], everything after 1010 has to be moved.

So Linked list provides the following two advantages over arrays

1. Dynamic size
2. Ease of insertion/deletion

Linked lists have the following drawbacks:

1. Random access is not allowed. We have to access elements sequentially starting from the first node. So we cannot do a binary search with linked lists.
2. Extra memory space for a pointer is required with each element of the list.
3. Arrays have better cache locality that can make a pretty big difference in performance.
4. It takes a lot of time in traversing and changing the pointers.
5. It will be confusing when we work with pointers.

Array have the following drawbacks:

1. Array is static in nature. Once the size of array is declared then we can't modify it.
2. Insertion and deletion operations are difficult in array as elements are stored in contiguous memory location and the shifting operations are costly.
3. The number of elements which have to stored in array should be known in advance.
4. Wastage of memory is the main problem in array. If the array size is big the less allocation of memory leads to wastage of memory.