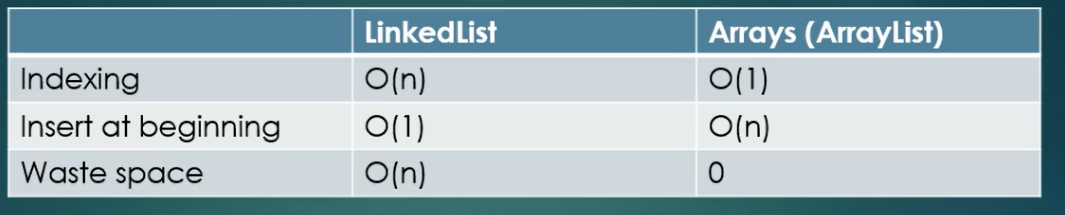
Linked List

1. Each node is composed of a data and a refernce/ link to next node.
2. Used to implement several other common data types like stack and queues
3. Benefit of linked list over array is elements can easily be inserted or removed without reallocation of entire structure.
4. An array has to be declared in source code before compiling and running the program
5. Operations like obtaining last node of the list or finding a node that contains a given data or llocating the place where a new node should be inserted may require sequential scanning of most or all of the list elements

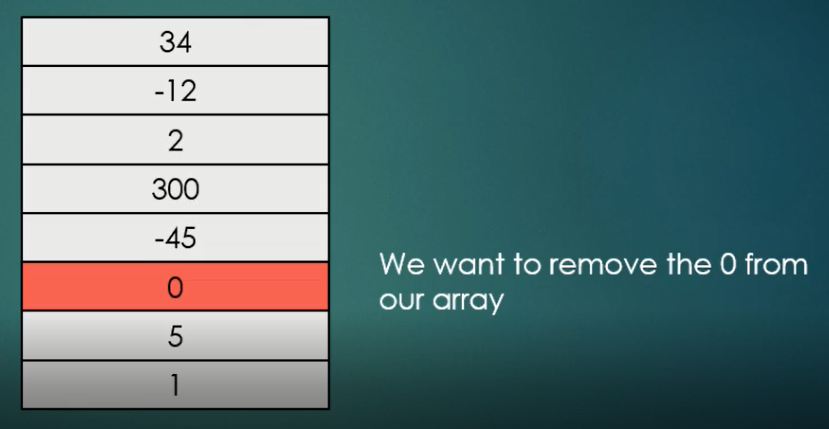
Time Complexity

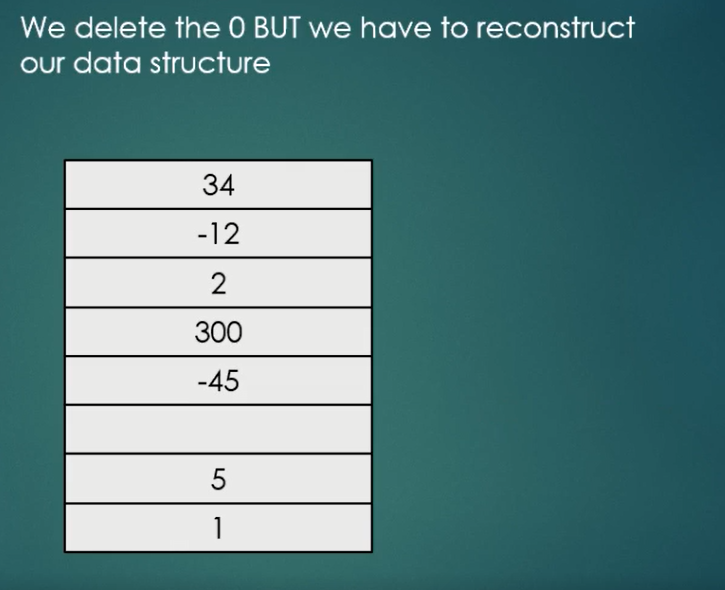


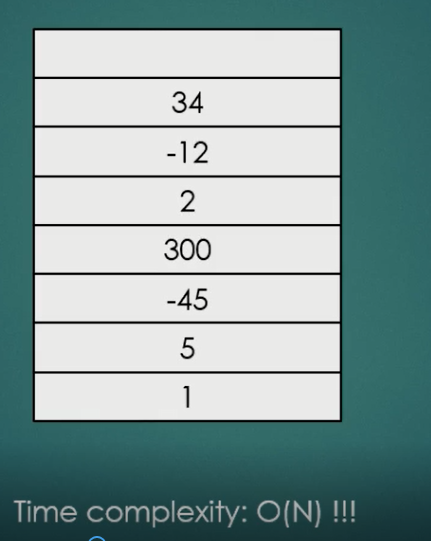
Use linked list if you want to insert/remove elements at the beginning

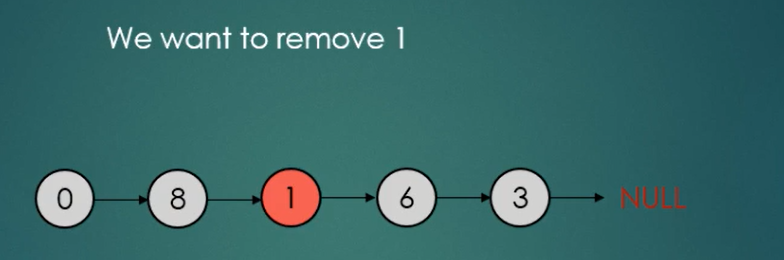
Use linked lists if the size is changing frequently. Otherwise use array

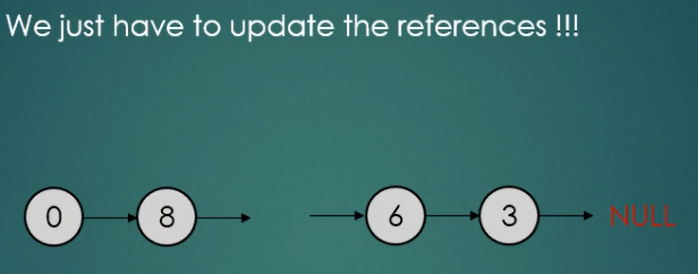
Use arrays is you need random access: it can be done very quickly in O(1) constant time

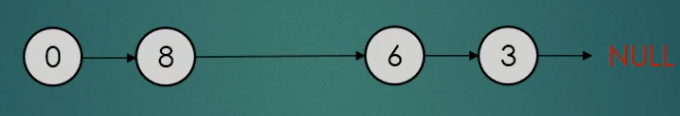


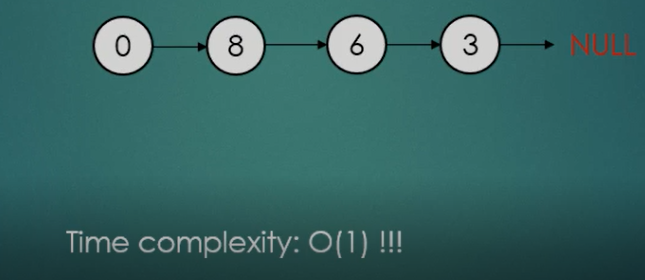












Advantages of Linked List

1. Linked Lists are dynamic data structures
2. It can allocate the needed memory in run time
3. Very efficient if we want to manipulate the first elements
4. Easy implementation
5. Can store items with different sizes: an array assumes every element to be exactly the same
6. It’s easier for a linked list to grow organically. An array’s size nees to be known ahead of time, or re-created when it needs to grow

Disadvantages

1. Waste memory because of references
2. Nodes ina linked list must be read in order from the beginning as linked list have sequential access (array items can be reached via indexes in O(1) time)
3. Difficulties arise in linkde list when it comes to reverse traversing. Singlt linked lists are extremely difficult to navigate backwards. Solution of this problem is double linked list however, memory is wasted in allocating space for a back pointer

Linked List:

Insert/Delete element at beginning: O(1) 🡪 Insert element at beginning and update the reference

Insert/Delete element at end of other position: O(n) 🡪

Linked List traversal: O(n)

Accessing element by value: O(n)

Array

Insert: O(n)

Delete: O(n)

Double Linked List

1. Each node has access to previous and next element

