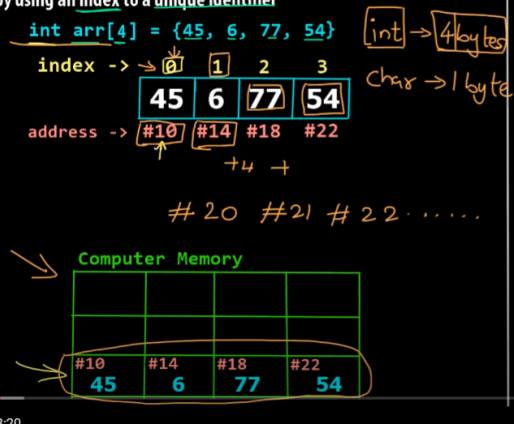
Introduction & Comparison

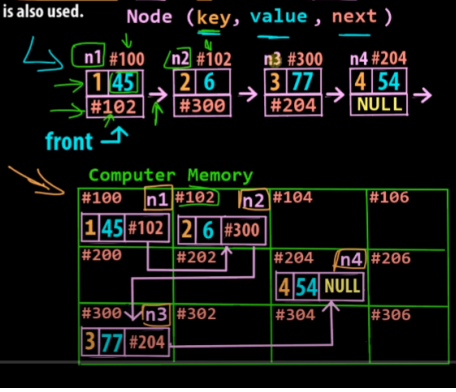
What is an Array ?

An array is a collection of elements of the same type, placed in contiguous memory locations than can be individually referenced by using an index to a unique identifier



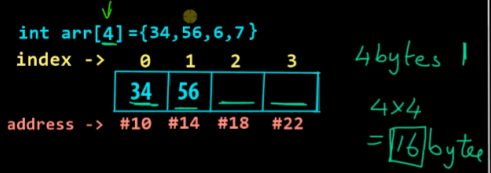
What is a Linked List ?

A linked list is a collection of elements(nodes) that are NOT stored in contiguous memory locations. The nodes are linked to each other using pointers(entity that points ot the next node). A node usually consist of 2 fields namely - data & next. Sometimes a 3rd field called id/name is also used.



1. Size of Data Structure

ARRAYS



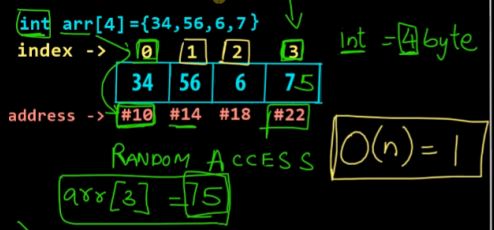
Size is STATIC

LINKED LIST

Size is DYNAMIC

2. Data Acess/Read/Update (Cost)

ARRAYS

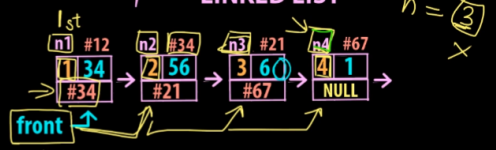


= Base Address + (Index pos x Size of element)

= 10 + (3 x 4)

= 22

LINKED LIST



SEQUENTIAL ACESS

O(n) = n → Linear Time

3) Insertion & deletion of elements (Cost)

ARRAYS

1) Insertion/Deletion at the Beginning -

$O(n) = n$ Linear

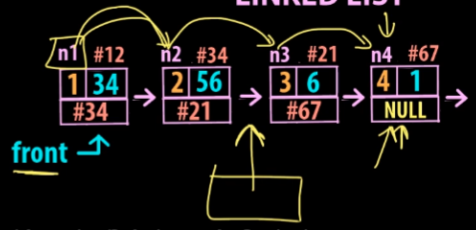
2) Insertion/Deletion at the End -

$O(n) = 1$ Constant

3) Insertion/Deletion at random nth postion -

$\rightarrow O(n) = n$ Linear

LINKED LIST



1) Insertion/Deletion at the Beginning -

$O(n) = 1 \rightarrow$ Constant

2) Insertion/Deletion at the End -

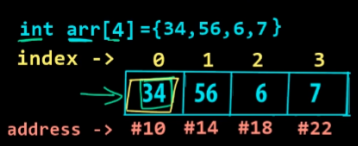
$O(n) = n \rightarrow$ linear

3) Insertion/Deletion at random nth postion -

$O(n) = n \rightarrow$ linear

4) Memory Requirement & Memory Usage/Efficiency

ARRAYS



int = 4 byte

$= 4 \times 4$

$= 16$ byte

arr[100] = 400 byte

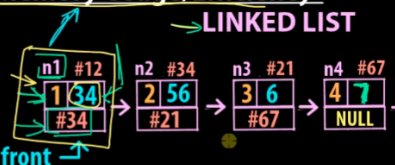
34 | 56 | 6 | 7 | ....

Person obj → 18 bytes

$18 \times 3 = 54$ bytes

Person obj[100] = 1800 byte

LINKED LIST



int = 4byte

8byte + 4

node = 12byte

4 nodes = 12×4 = 48 byte

n1 n2 n3 = 78 byte

26 by

6) Searching Methods

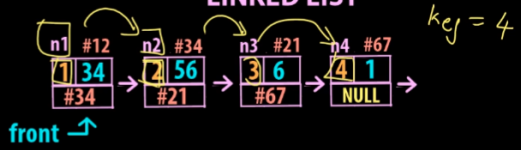
ARRAYS

```
int arr[4]={34,56,6,7}
index ->   0   1   2   3
          | 34 | 56 |  6 |  7 |
address -> #10  #14  #18  #22
```

1) Sequential Search

2) Binary Search.

LINKED LIST



1) Sequential Search

6) Searching Methods

ARRAYS

```
int arr[4]={34,56,6,7}
index ->   0   1   2   3
          | 34 | 56 |  6 |  7 |
address -> #10  #14  #18  #22
```

LINKED LIST

No one Data Structure can be considered as absolute best in all situations. It depends on the requirements/usecases/scenarios