Linked List Code Link:
https://www.interviewbit.com/snippet/0689528ebef0733aaf74/

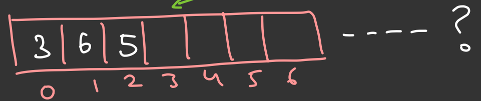
Sunday – 9 PM    Recursion + Doubts    Extra Class.

- Linked list

↳ Data structure to store a list of elements

Recap

arr[7]



↳ define the size of array

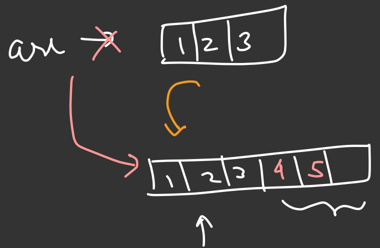
Advantage [Array]

→ Insert at end is very fast O(1)

→ Read any random index in O(1).

- Arraylist – grow on demand
(vector C++)

↓ create a copy of the array with larger size



Continuous ~~arr~~ Memory

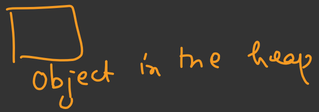
arr[3]

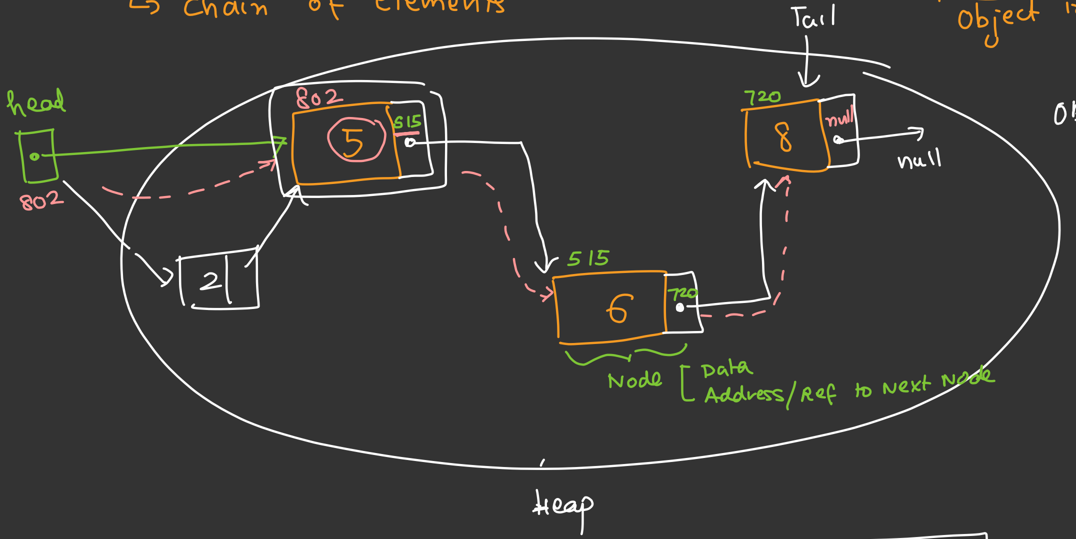
[Arraylist]

+ → grow automatically if needed

# Linked list

↳ chain of elements

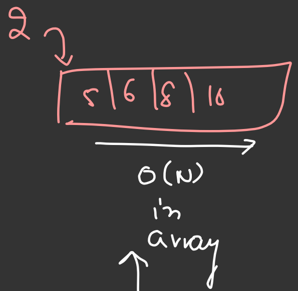




```
Object DT oname
  = new Object DT ();
```

→ Allocating Memory on Demand

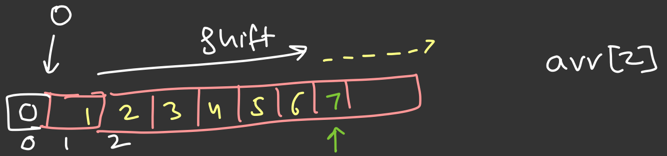
→ ↑ Compile can give memory in any location, it is Not continuous.

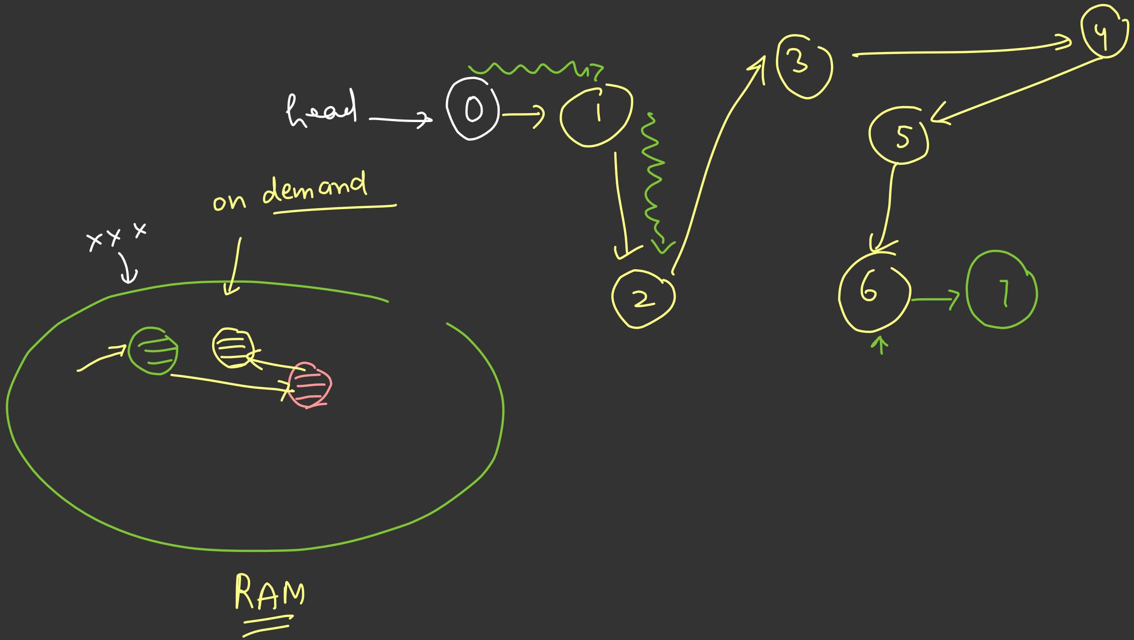


→ Insertion at beginning is very fast O(1)

→ ———— end in O(1)

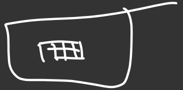
⇒ Disadv : You can't acess ith element directly. ~~O~~(1)

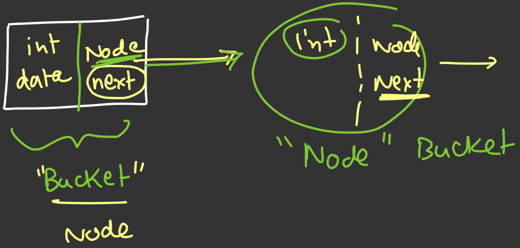




get(i) → O(N)
in
linkedlist

SSD/Hard-drive (Defrag)



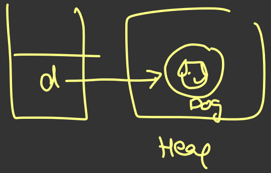


## OOPS

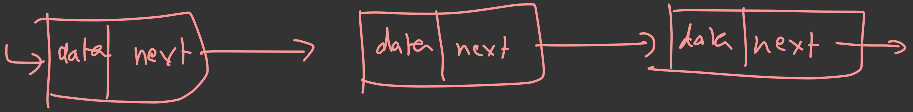
Dog d = new Dog()

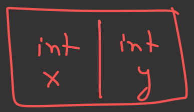
obj reference

object in heap



```
class Node {

        int data,
        Node next,  store the address of next Node

}
```





Bucket

Pair

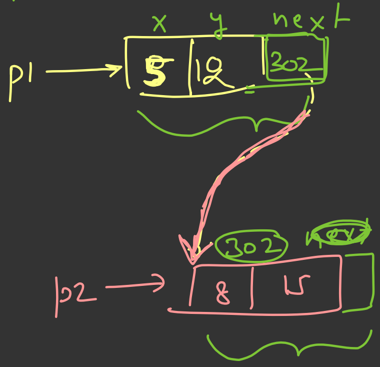
class → is a blue print

```
class Pair{
        int x;
        int y;
        Pair next,
```

3

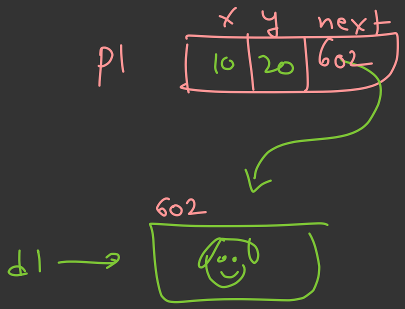
```
Pair p1 = new Pair();
        p1. x = 5
        b1. y = 12

Pair p2 = new Pair();
        p2. x = 8
        p2. y = 15
```

$\boxed{\text{p1. next = p2}}$



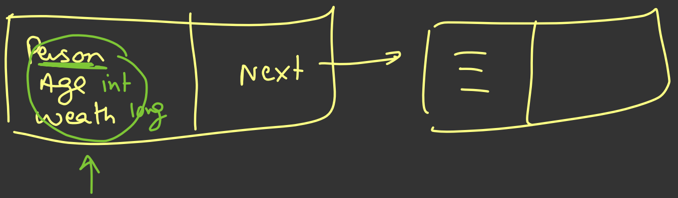
=

```
Pair p1 = new Pair

    p1.x = 10
    p2.y = 20

p1.next = d1;
```



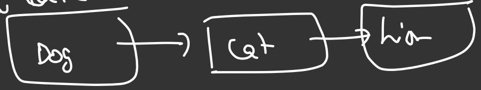
```
Dog d1 = new Dog();
```

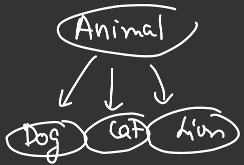
```
class Pair{
        int x, y;

        Dog next;
  }
```



```
Node{
    Animal a = new Dog();
               new Cat()
}
```

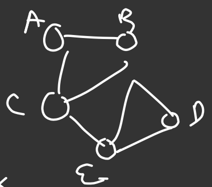
Adv↑





- Linkedlist <Int>
- Linkedlist < persons >
- Linkedlist < person, priority >

## Graph Problems



Adj list

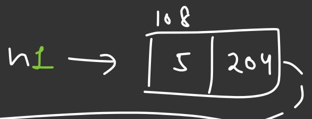
↓

Linked list /

```
class Node {
    int data;

    Node next;   // obj Ref.

    Node (int d) {
        data = d;
        next = null;
    }
}
```

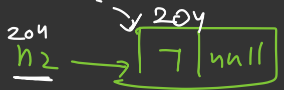
No Return Type

→ Called once when the obj is created



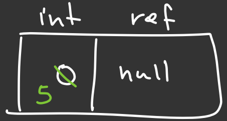
Node n1 = new Node(5);

Node n2 = new Node(7);



n1.next = n2

10.05
To
TRy



without constructor →

```
Node n = new Node();
⇒ n1.data = 5
```

with Constructor ←

```
Node n = new Node(5);
```