23/24-Dec-2021

23 December 2021 20:50

Hash Set:- 1, 2, 3, 4,

get => O(1)

add = O(1)

remove = O(1)

1 2 3 4 5 6

Print common Elements:-

a = [1, 2, 3, 4, 5]

b = [3, 4, 5, 6, 7, 8]

n^2, n, (m×n),

with Hash Set

O(n), O(1)

n, n^2, n log n ✓ O(1)

TC Space

a = [4, 1, 3, 5, 2,]

b = [6, 7, 1, 3, 2]

O(n), n^2, (m+n), 2n

✓

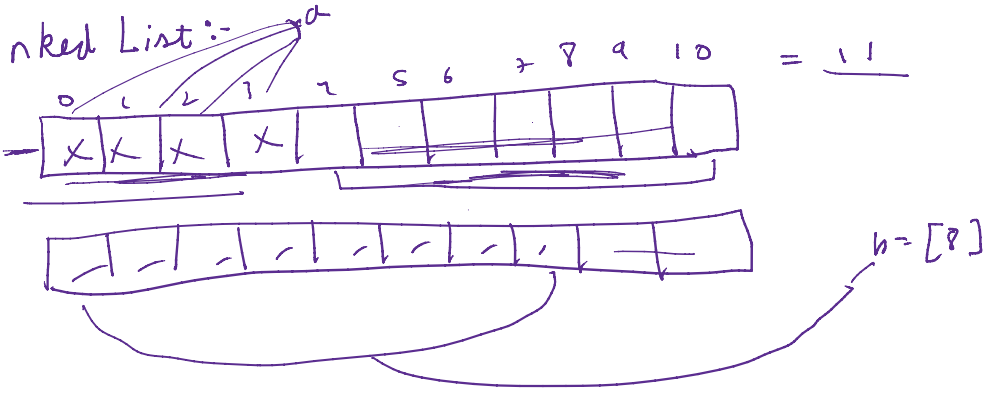
O(n)

Hash Set

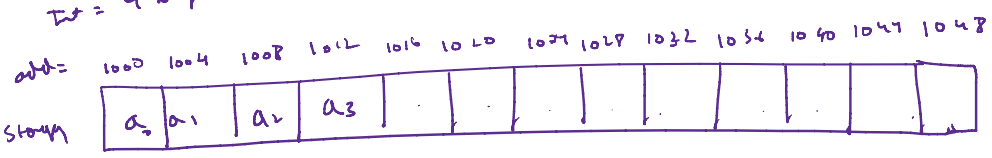
4
1
3
5
2

[1, 3, 2]

Linked List:-



Int = 4 bytes

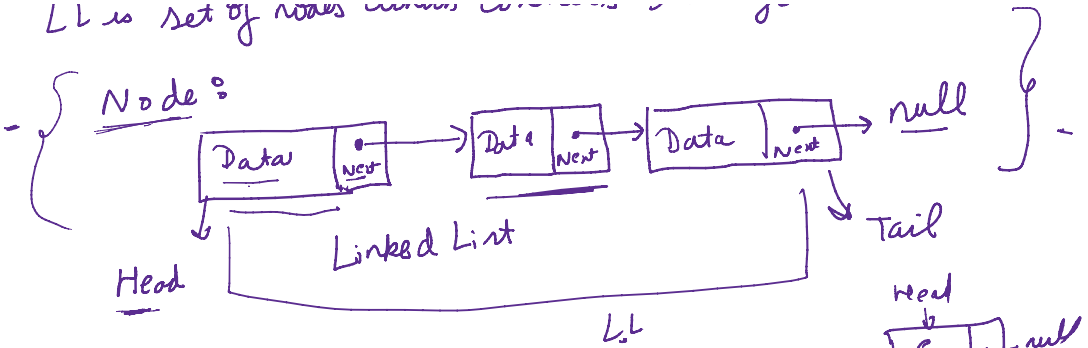


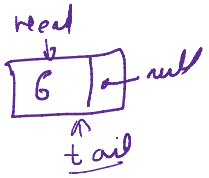
a = new [4]

b = new [10]

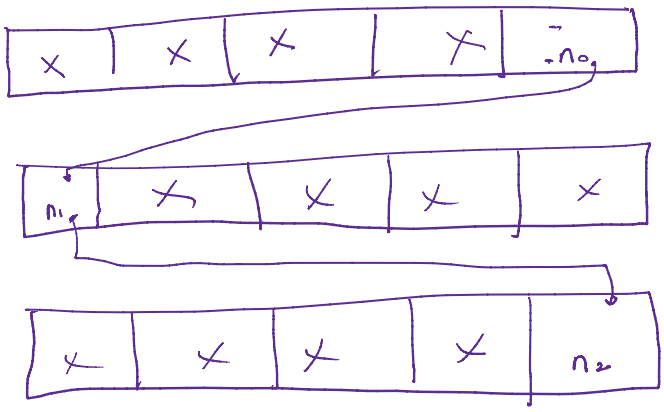
LL is set of nodes which contains the information.

LL is set of nodes



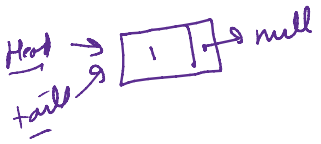


L.L

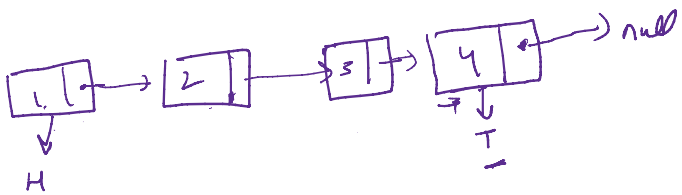



```
LinkedList {
  class Node {
    int data,
    Node next
  }
  Node tail;
  Node head;
  int size;
}
```

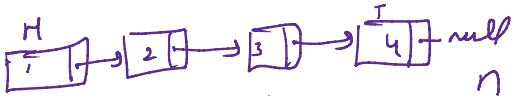
```
addLast(data) { O(1),
  node = new Node(data)
  if(tail == null) {
    tail = head = node;
  } else {
    tail.next = node;
    tail = node
  }
```



data = 4

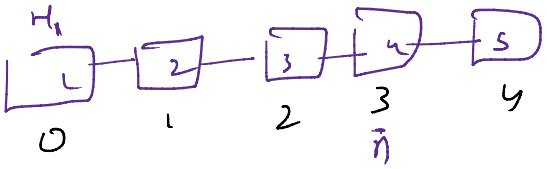



```
PrintLL() { O(n)
  Node node = head;
  while(node != null) {
    print(node.data);
  }
```



```
while( node != ... )
print (node.data);
node = node.next
}
}
```

```
getAt (idx) {
i=0 ;
node = head;
while (i < idx) {
i++;
Node = node.next;
}
return node
}
```

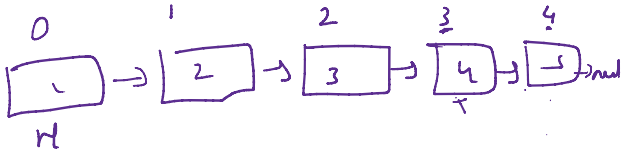


O(n)

i = 0 1 2 3

idx = 3

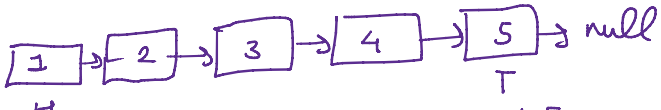
```
removeLast (){ O(n)
node = getAt (size - 2);
node.next = null
tail = node
size--;
}
```



size = 5

```
addFirst (data) {
node.next = head
head = node
this.size++;
}
```

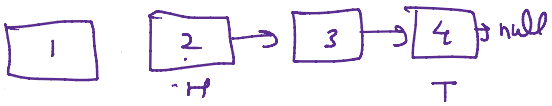
data = 1



size = 5

O(1)

```
removeFirst ( ) {
if(size == 0 ) {
}
val = head;
head = head.next;
this.size--;
res.next = null;
```



O(1)

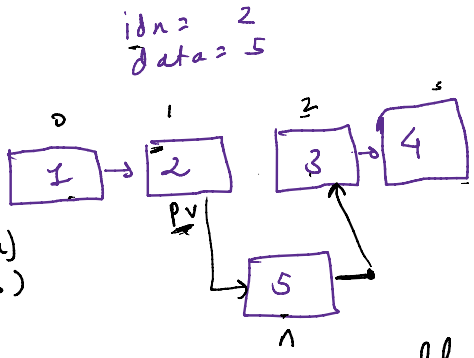
size = 3

res = 1

```
this.size--;
res.next = null;
return res;
}

addAt(idx, data) {
if(idx<0 || idx > size)
  invalid
} if(idx == 0) addFirst(data)
else if(idx == size) addLast(data)
prevNode = getAt(idx-1);
node = (data);
node.next = prevNode.next ✓
prevNode.next = node;
size++;
```

idx = 2
data = 5

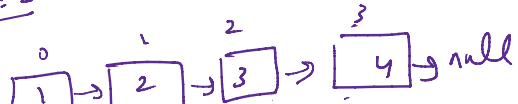


1 → 2 → 3 → 4 → null
0 1 2 3
size = 4

O(n)

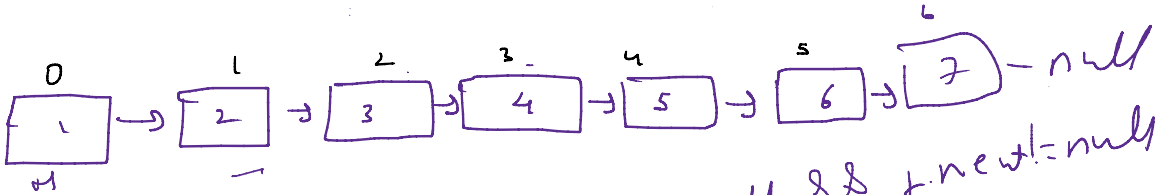
```
removeAt(idx) {
if(size==0) Empty
if(idx<0 || idx≥size) {
  Invalid Op.
}
else if(idx == 0) removeFirst();
else if(idx == size-1) removeLast();
else {
  prev = getAt(idx-1);
  res = Pv.next;
  Pv.next = Pv.next.next;
  res.next = null
  size--;
  return res;
}
```

idx=2



size 4

res = 3




```
getMid() {
  s = head
  f = head
  while(f!=null && f.next!=null) {
    s = s.next
    f = f.next.next
  }
  return s;
}
```

```
f = head
while (f.next != null && f.next.next != null)
        s = s.next
        f = f.next.next;
}
return s;
}
```

reverse a LL (Data Reversal) size ↙ $T(n) = O(n^2)$ → HINI