last

1 2 3 5 6 7 4

pivot

<= >

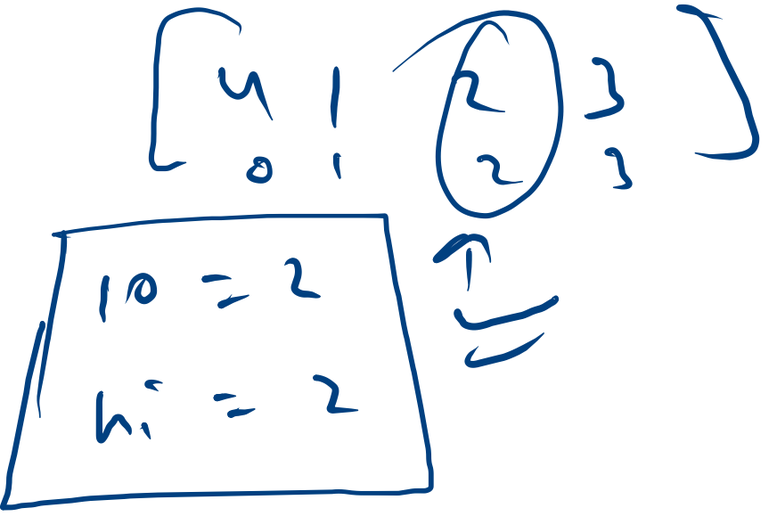
small large

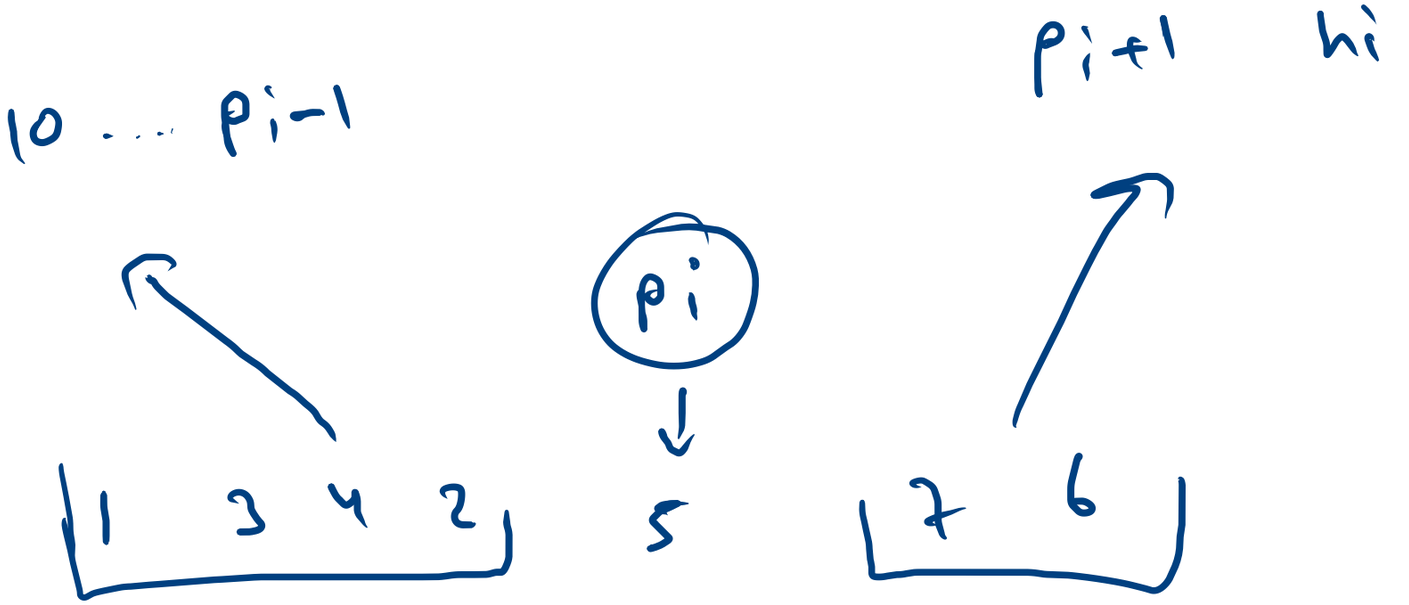
1 3 2 4 7 5 6

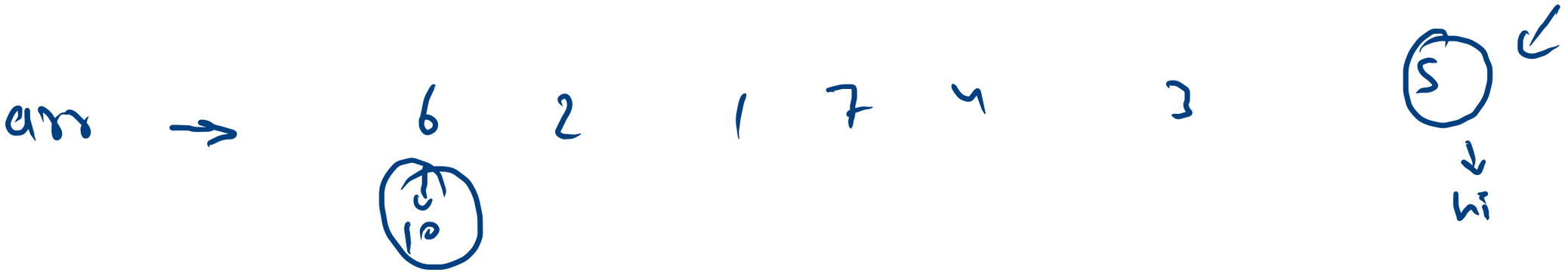
< >

sort sort

1 2 3 4 5 6 7


4 1 2 3
0 1 2 3
lo = 2
hi = 2

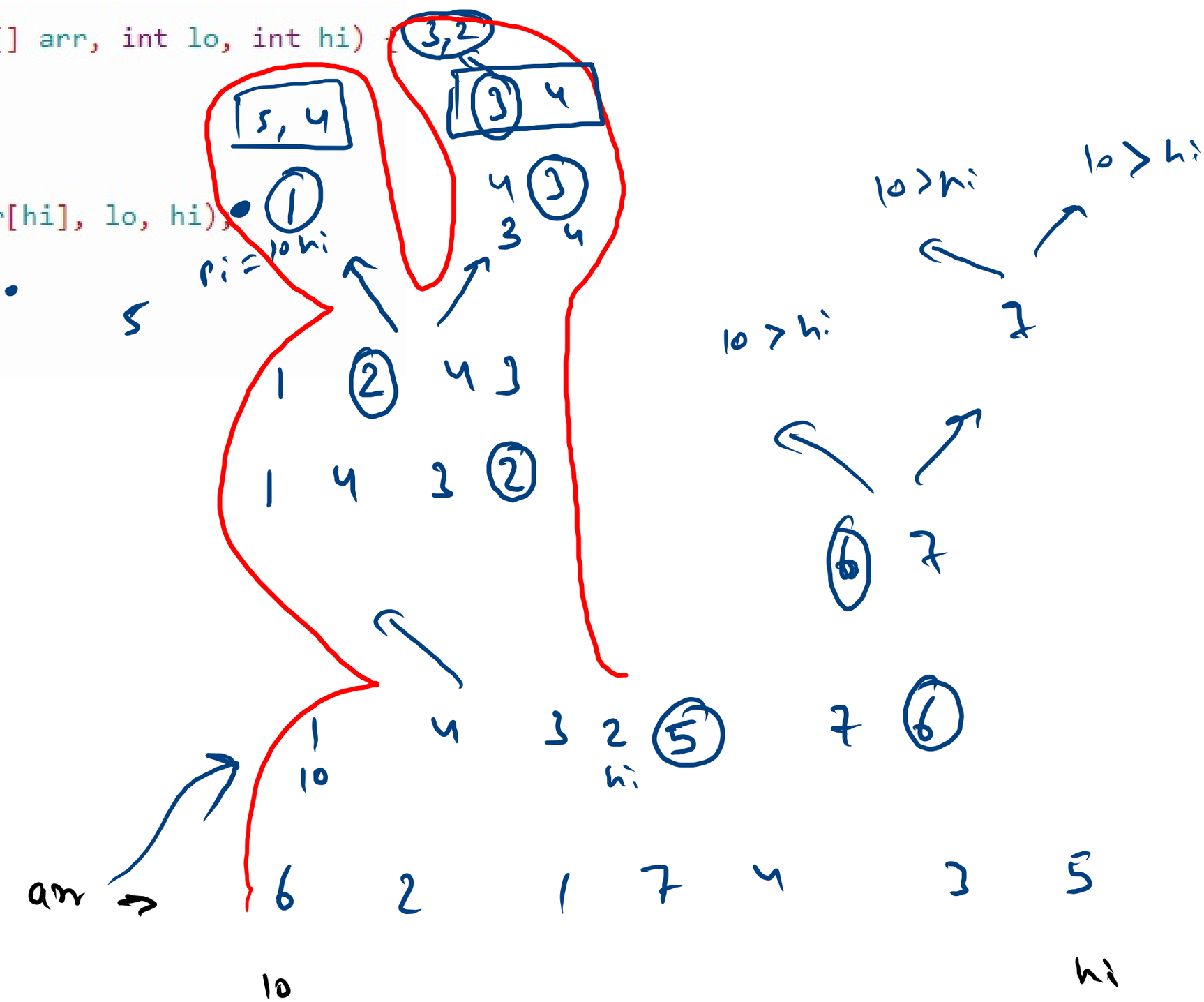

lo pi-1
pi+1 hi
pi
1 3 4 2
5
7 6


arr →
6 2 1 7 4 3
5
lo
hi

```java
public static void quickSort(int[] arr, int lo, int hi) {
    if(lo>hi){
        return;
    }

    int pi = partition(arr, arr[hi], lo, hi);

    quickSort(arr, lo, pi-1);
    quickSort(arr, pi+1, hi);
}
```



n=1 → 1

n=2 → 12

n=3 → 123

n=4 → 1234

n=5 → 1 2 3 4 5

n=6 → 1 2 3 4 5 6

small | big

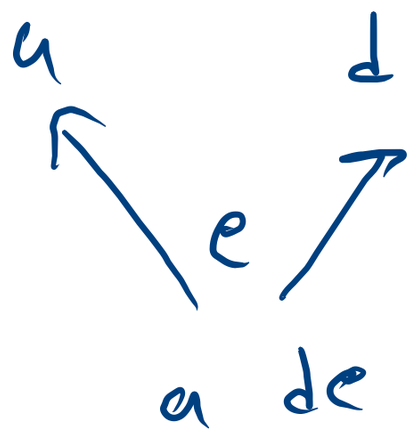
$1+2+3+4+5+\cdots+n$

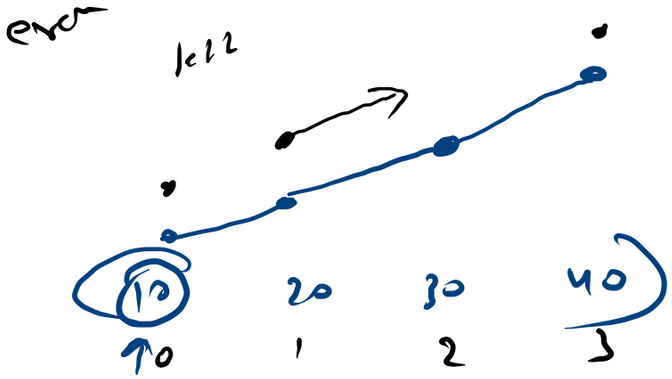
$\frac{n(n+1)}{2} = \frac{n^2+n}{2}$

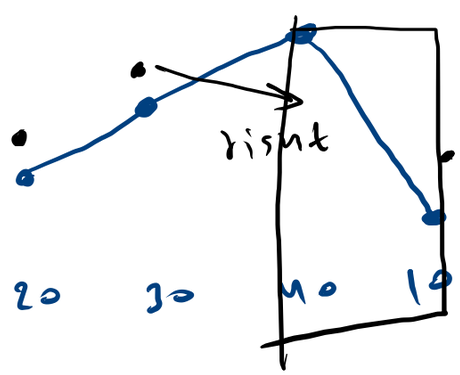
$O(n^2)$

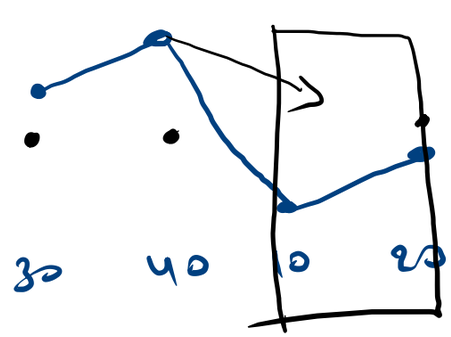
assume halt =

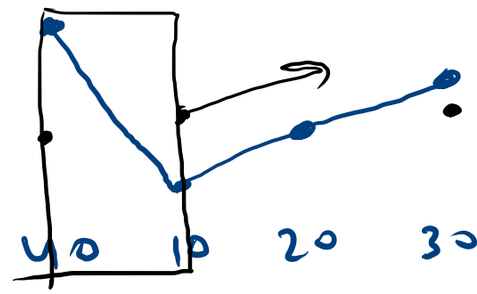
$\Omega(n \log_2 n)$

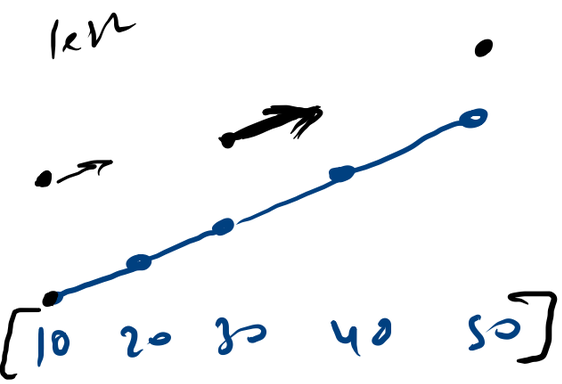


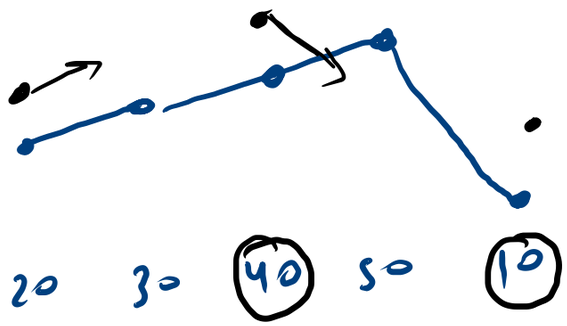




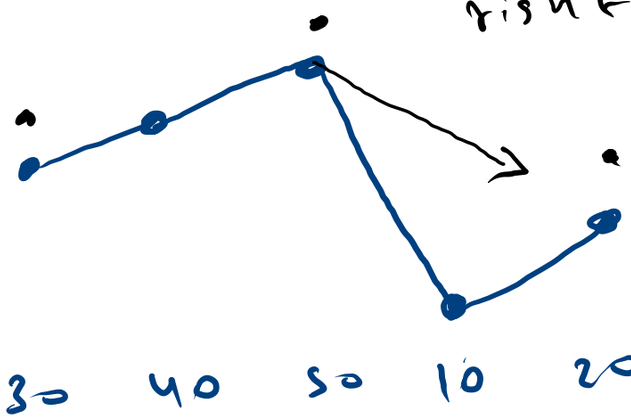






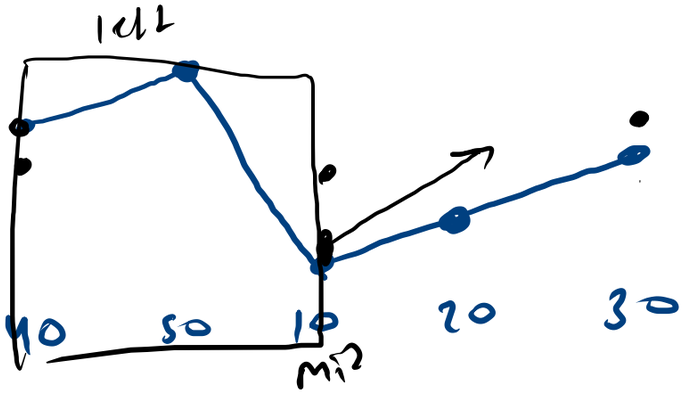


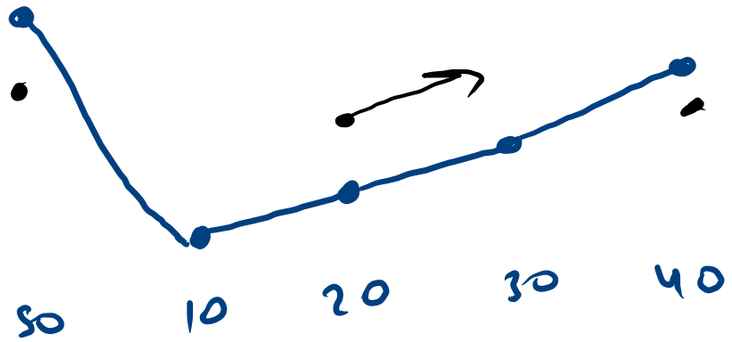
right



right

```
mid
left  lo , mid
right mid+1, hi
```

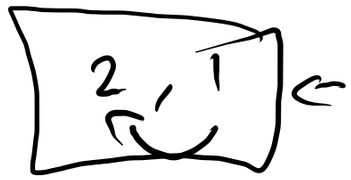




```java
int lo=0;
int hi = arr.length-1;

while(lo<hi){
    int mid = (lo+hi)/2;

    if(arr[mid] < arr[hi]){
        hi = mid;
    }else{
        lo = mid+1;
    }
}
return arr[lo];
```



$\frac{12}{2^0}$ $n = 12$

$\frac{12}{2^1}$ $n = 6$

$\frac{12}{2^2}$ $n = 3$

40 50 10 20 30

10

mid hi

$n = 2$

$\frac{12}{m}$ $n = 1$

m

$\log_2(n)+1$

$O(\log_2(n))$

$n = 2^m$

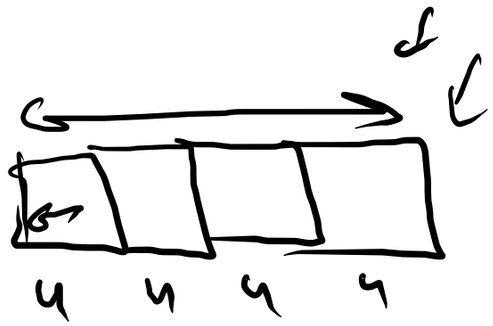
$m = \log_2(n)$

Array

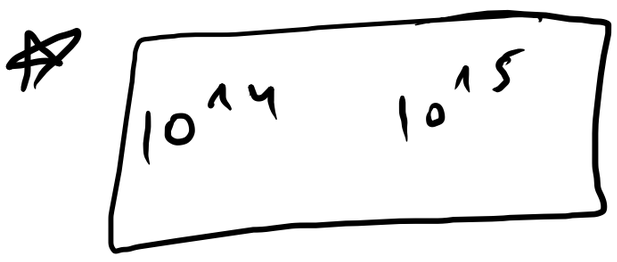
size fixed → ArrayList

continous → LinkedList

new int[4]

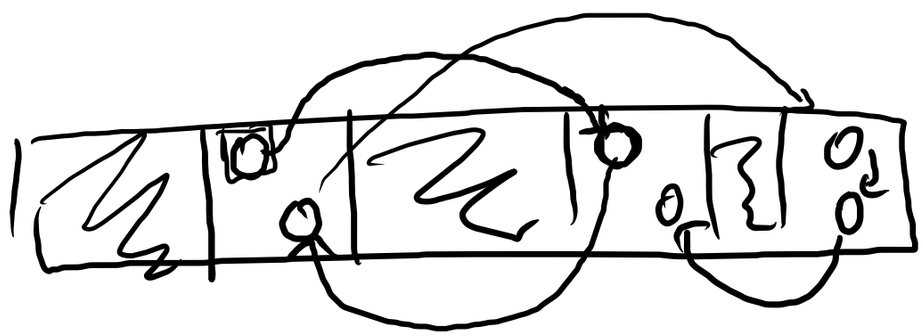


16 byte

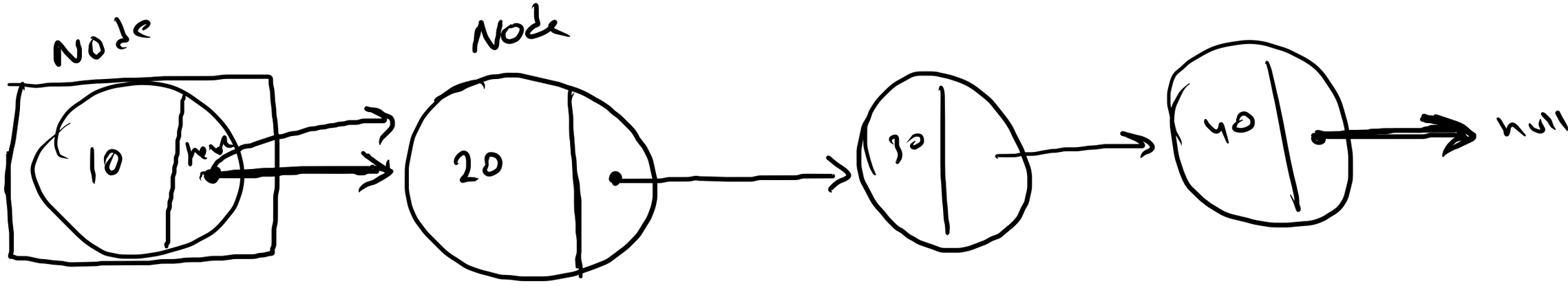


heap

5+5+6 -> 16



data




```
class Node {
    int data;
    Node next;
}
```

data

constant small

```java
public static class Node {
  int data;
  Node next;
}

public static class LinkedList {
  Node head;
  Node tail;
  int size;

  void addLast(int val) {
    // Write your code here
  }
}

public static void testList(LinkedList list) {}
public static void main(String[] args) throws Exception {
  BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
  LinkedList list = new LinkedList();

  String str = br.readLine();
  while(str.equals("quit") == false){
    if(str.startsWith("addLast")){
      int val = Integer.parseInt(str.split(" ")[1]);
      list.addLast(val);
    }
    str = br.readLine();
  }

  testList(list);
```

addLast 10 addLast 20
addLast 30 addLast 40
addLast 50 quit

size = 0

```
Node n = new Node()
n.data = val
tail.next = n
tail = n
size++;
```

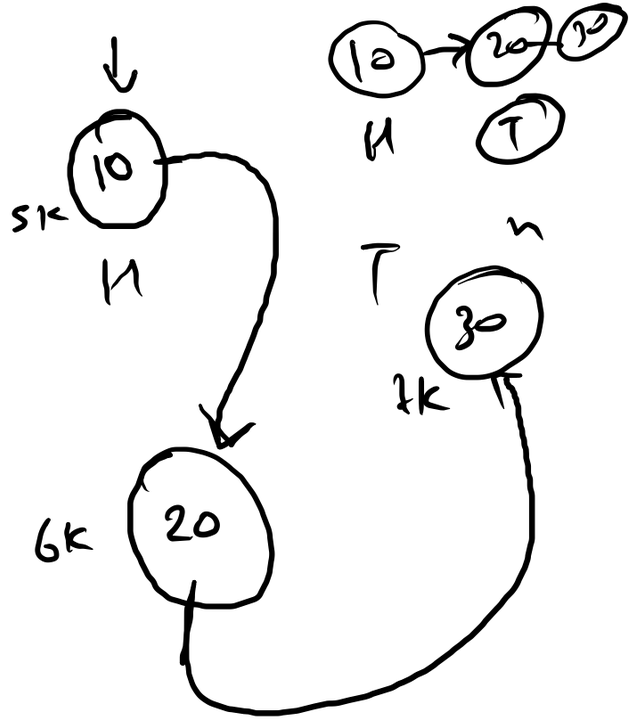
head	null
tail	null
size	0

main

addLast

val	30
this	4k

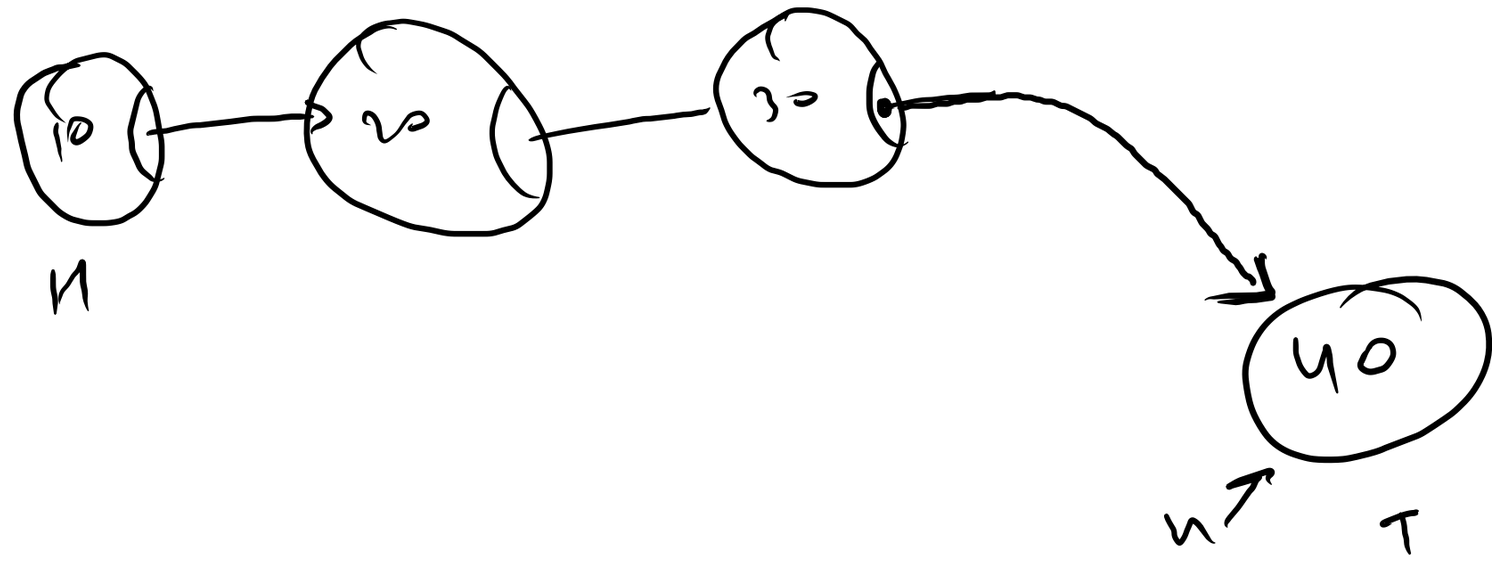
list 4k



head	5k
tail	~~5k~~ ~~6k~~ 7k
size	~~2~~ 3

4k

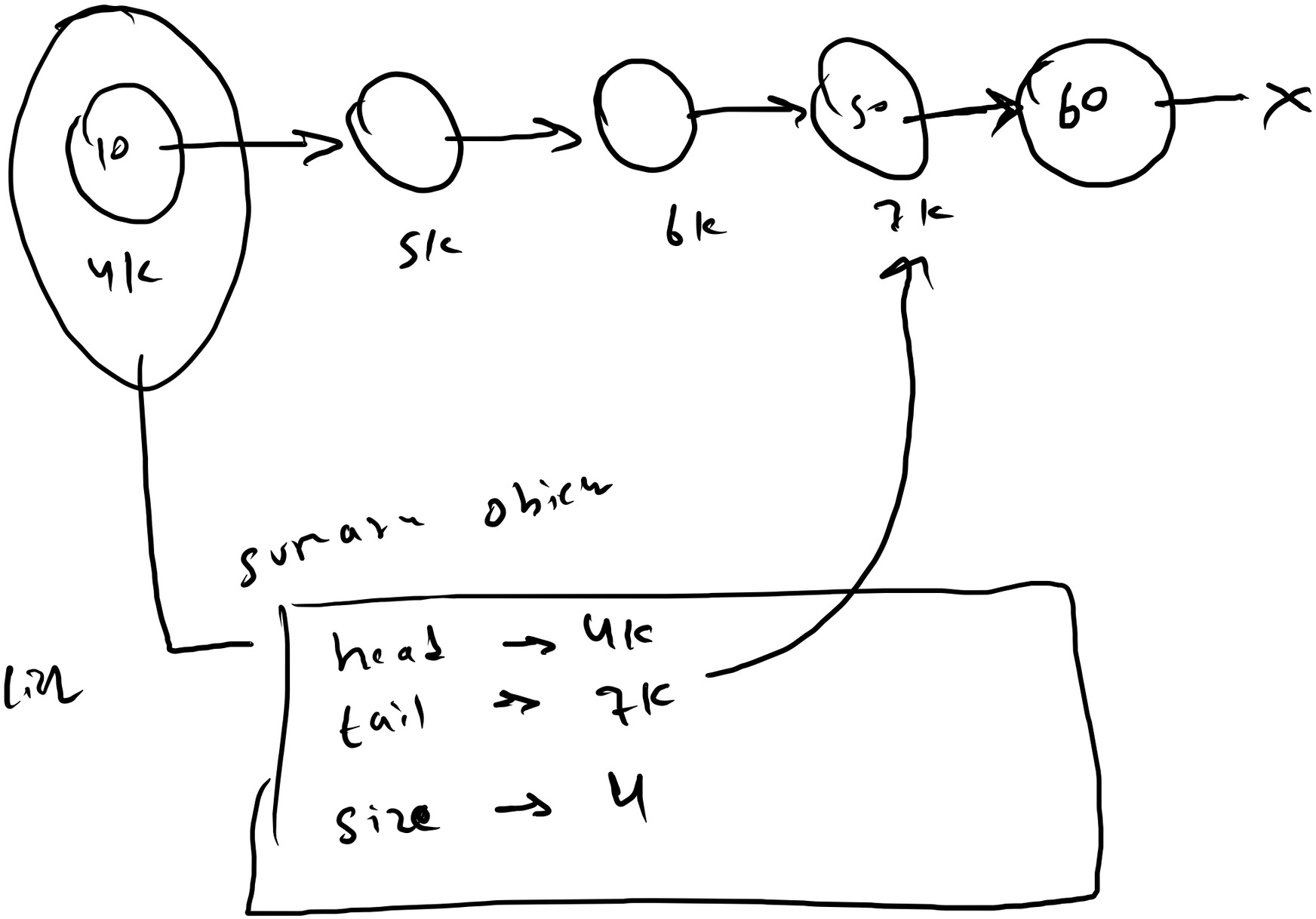
```java
void addLast(int val) {
    Node n = new Node();
    n.data = val;
    if(size == 0){
        head = n;
        tail = n;
        size = 1;
    }else{
        tail.next = n;
        tail = n;
        size++;
    }
}
```



val = 40

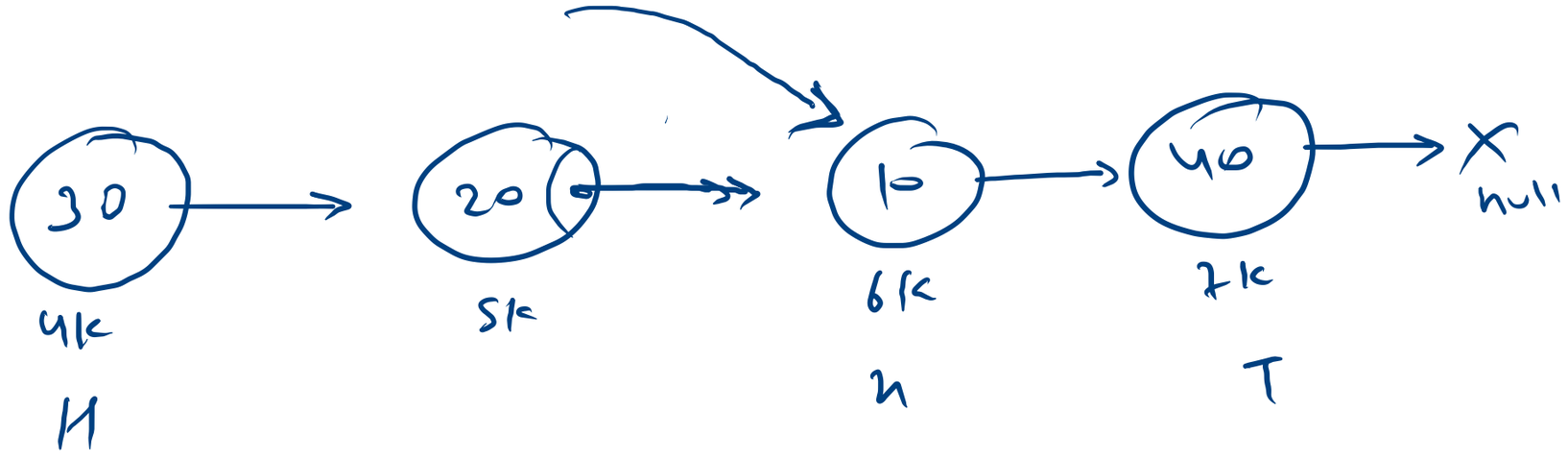
summary object

head	10
tail	~~30~~ 40
size	~~3~~ 4


10
4k
5k
6k
50
7k
60
x
int
60
Linked List
Sumar Object
head → 4k
tail → 7k
size → 4

Display

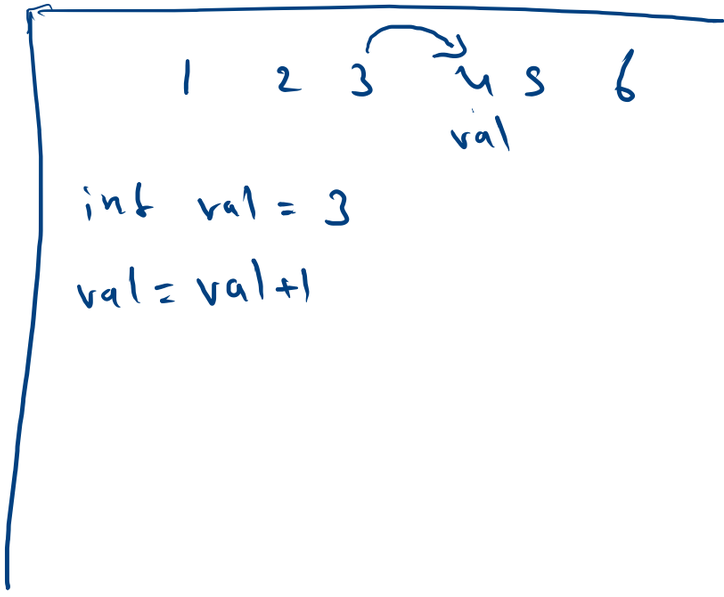
Size

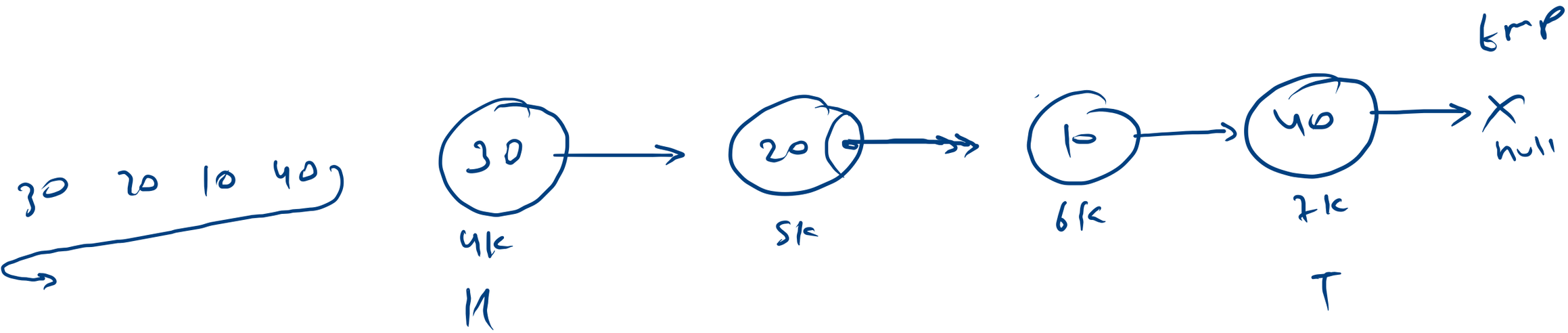


30 20 10 40

h = h.next

head 4k
tail 7k ⇐
size 4





tmp = tmp.next

tmp = H

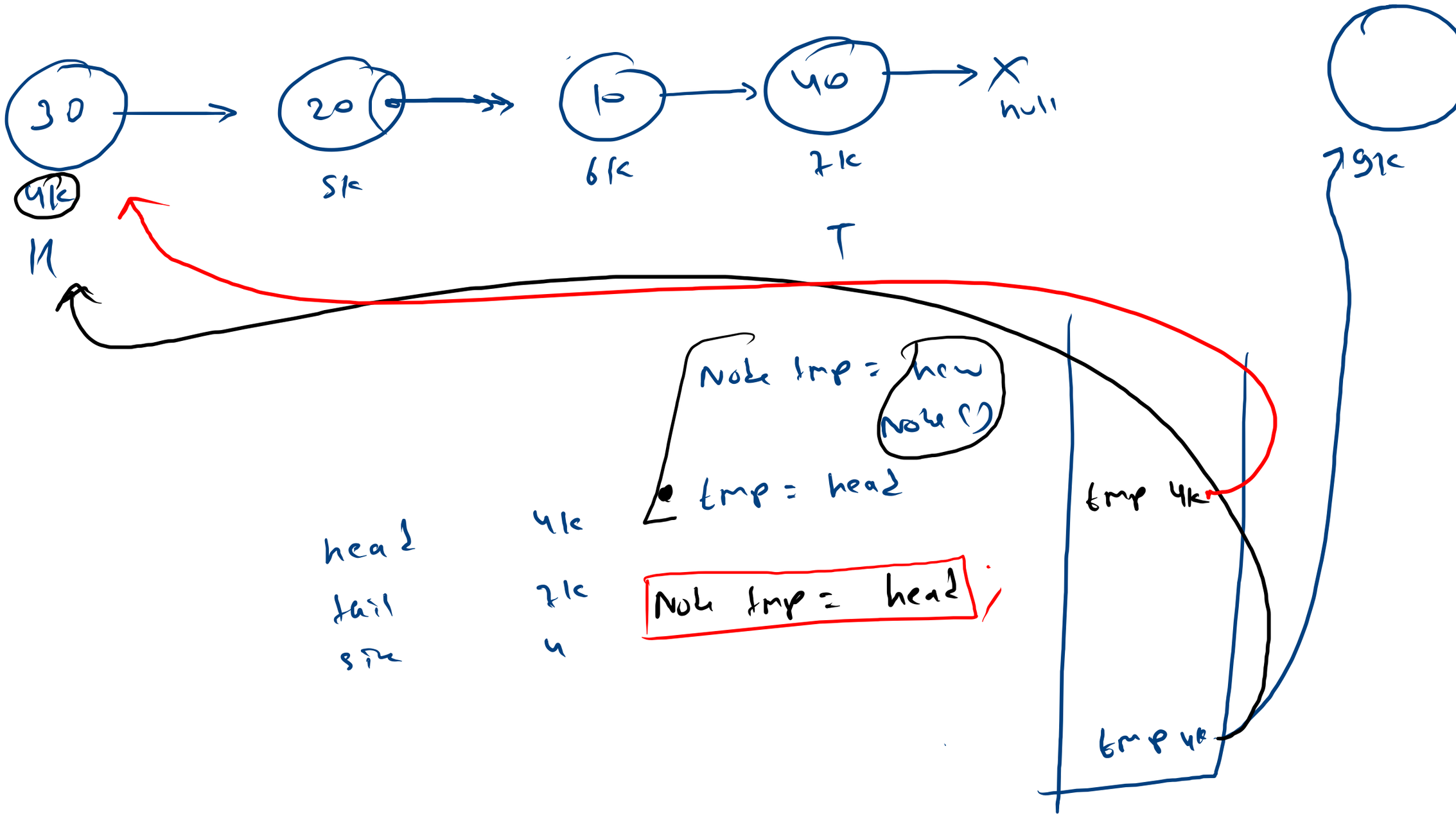
while (tmp != null) {

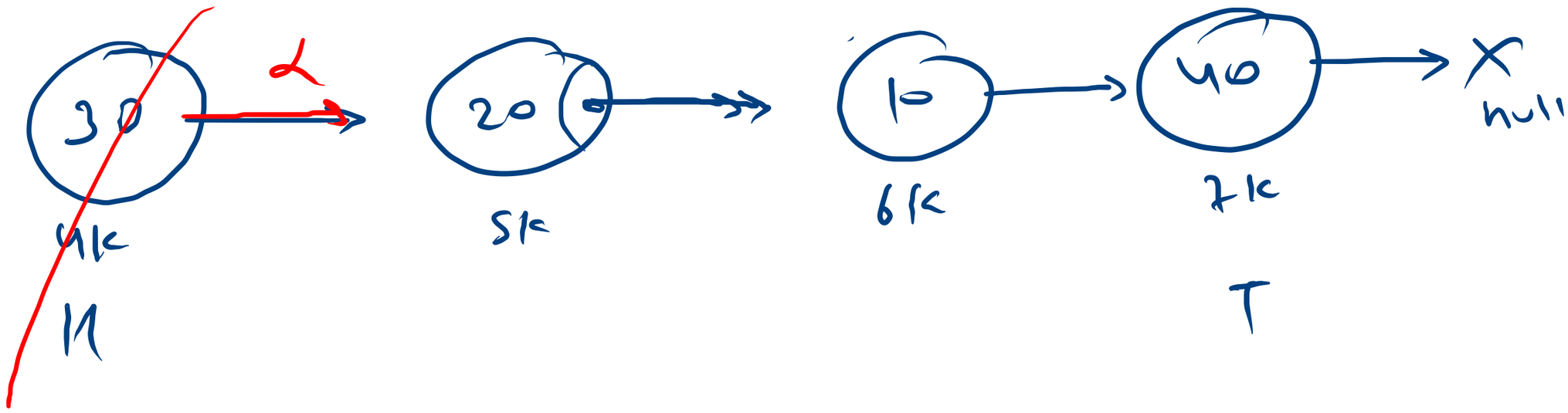
- print (tmp.val
- tmp = tmp.next

} println

head 40k

tail 71k


30
20
10
40
X
null
4k
5k
6k
7k
9k
11
T
Node tmp = new
Node ()
tmp = head
4k
head
tail
size
7k
4
Node tmp = head
tmp 4k
tmp 4k

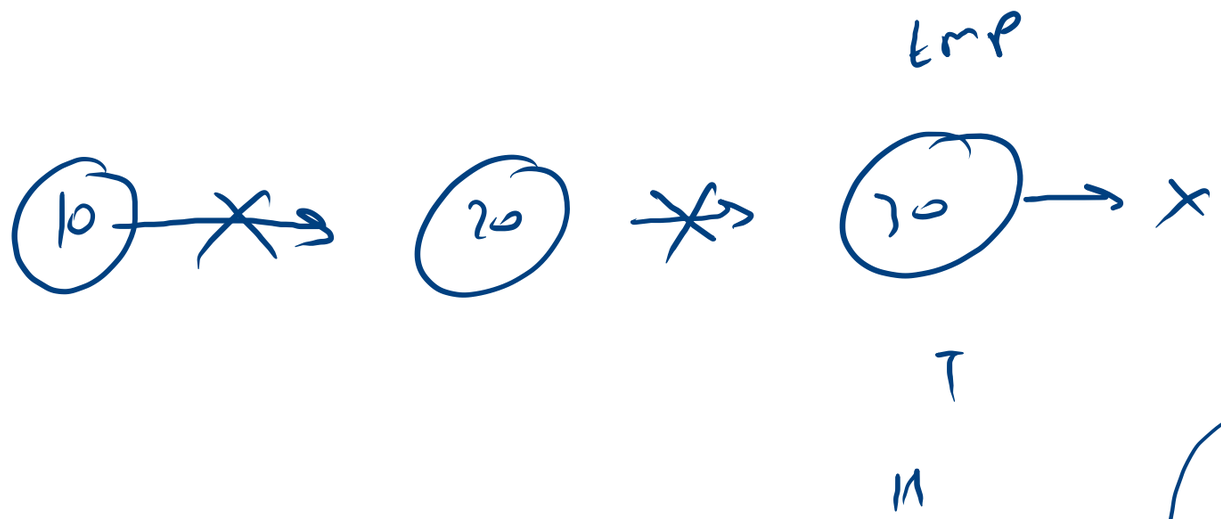


n.next = null

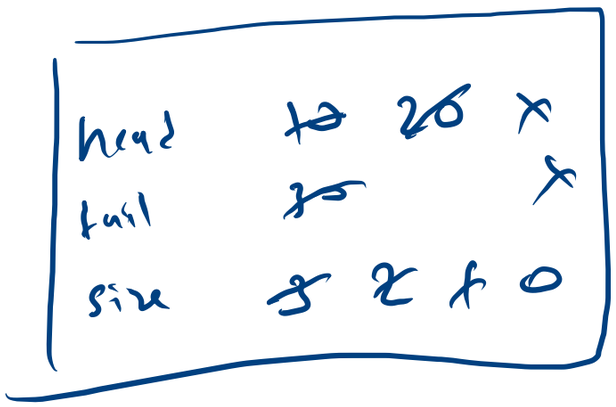
summary

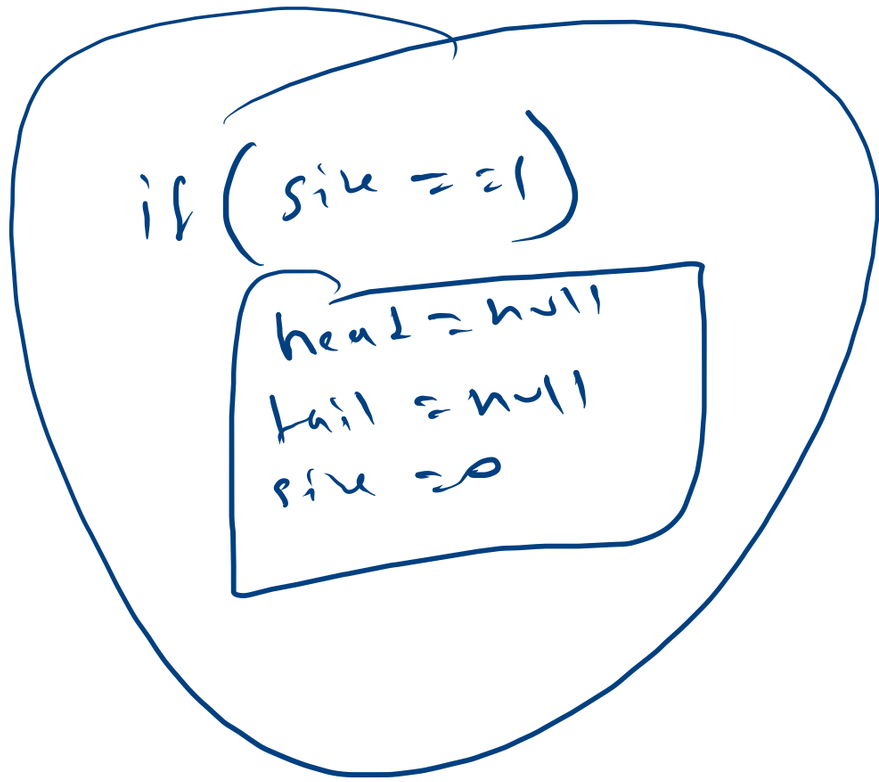
head	4k
tail	7k
size	4

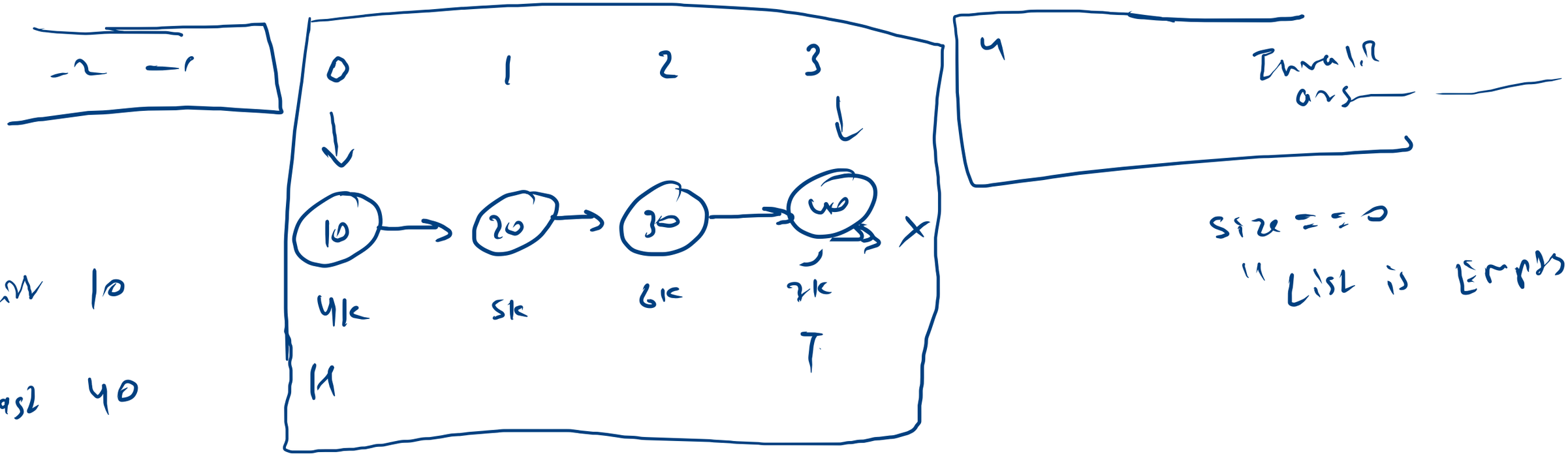
removeFirst



- Node tmp = head.next
- head.next = null
- head = tmp
- size --;







getAt 10

getLast 40

getAt(i)

0

(2)

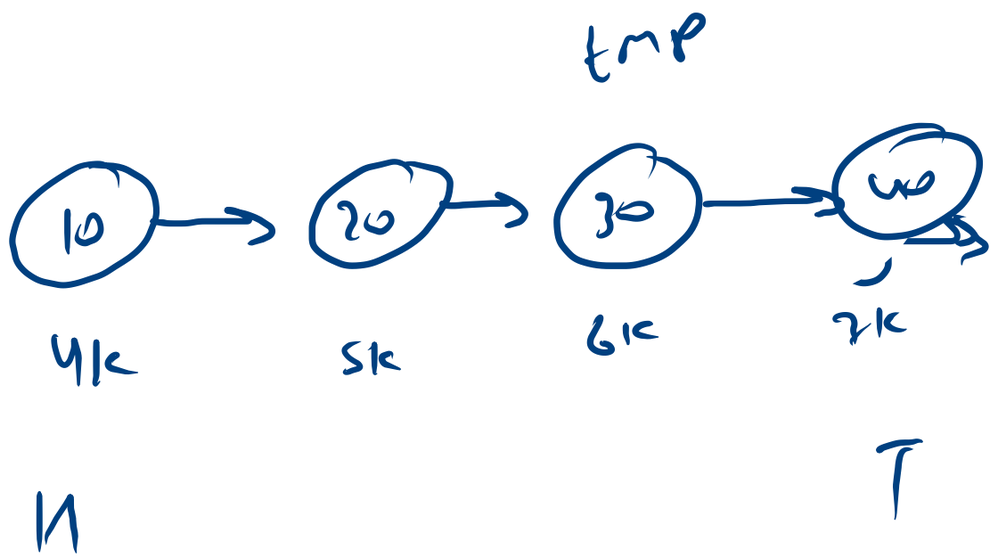
head → 41k

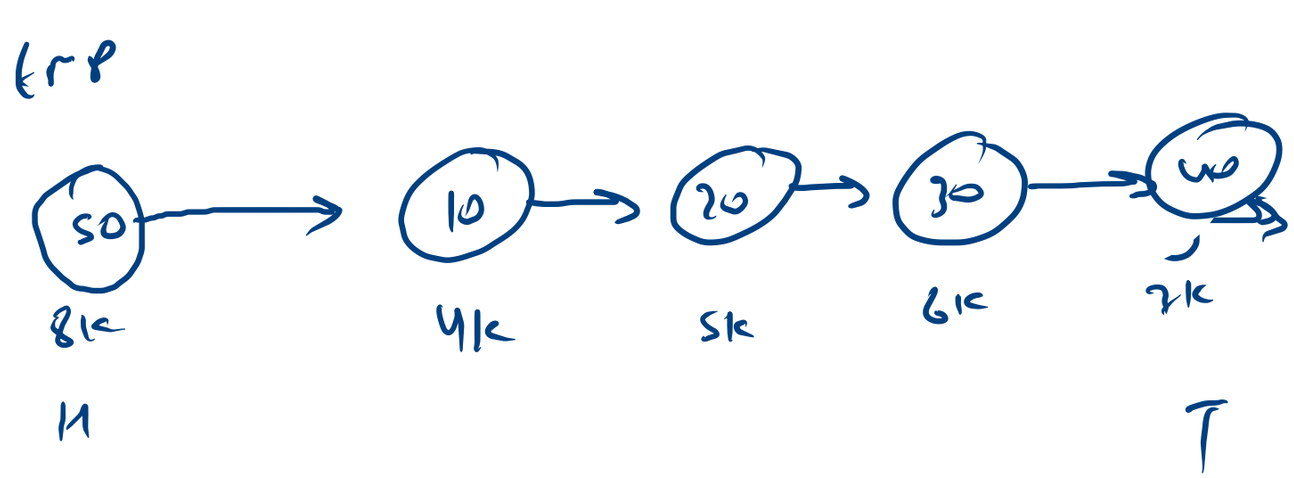
tail → 71k

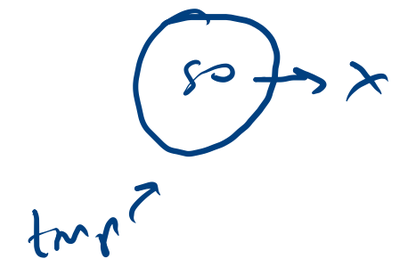
size → 4

ind = 0 + 2

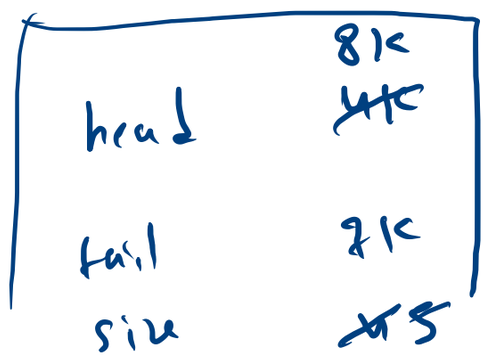
i = 2

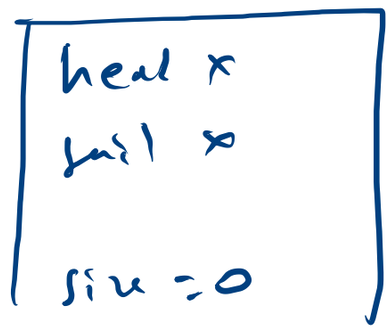






addFirst(50)






```
Node temp = new Node();
temp.data = val;
temp.next = head;
head = temp;
size ++;
```