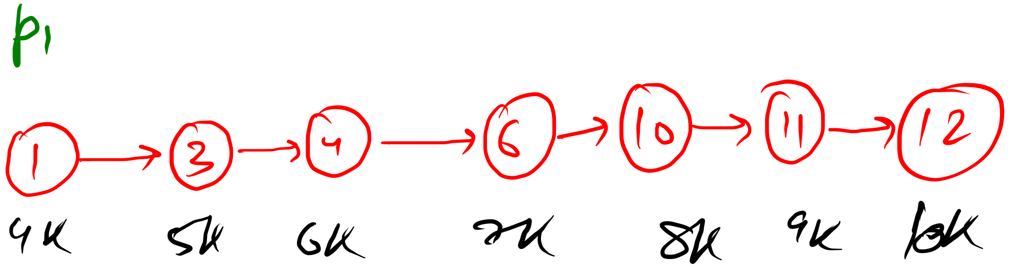
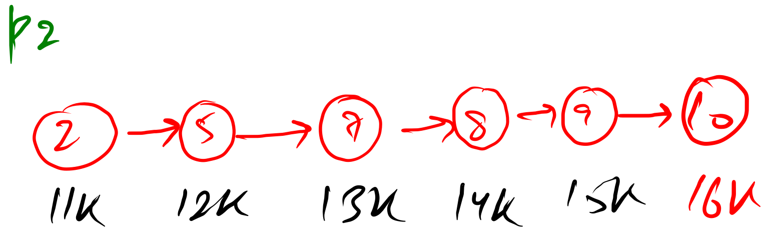
ll1

head = 4k
tail = 10k
size = 7



ll2

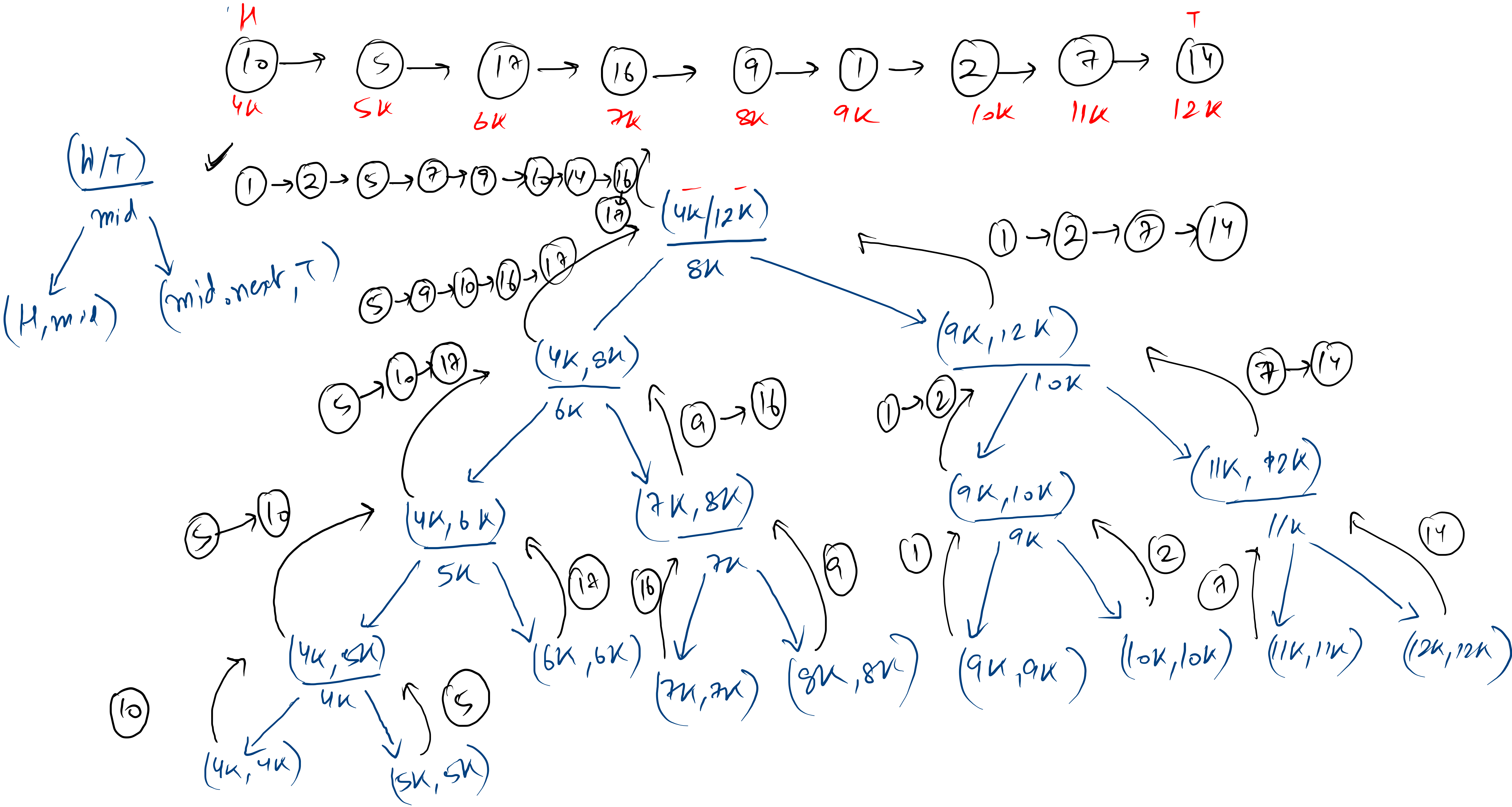
head = 11k
tail = 16k
size = 6

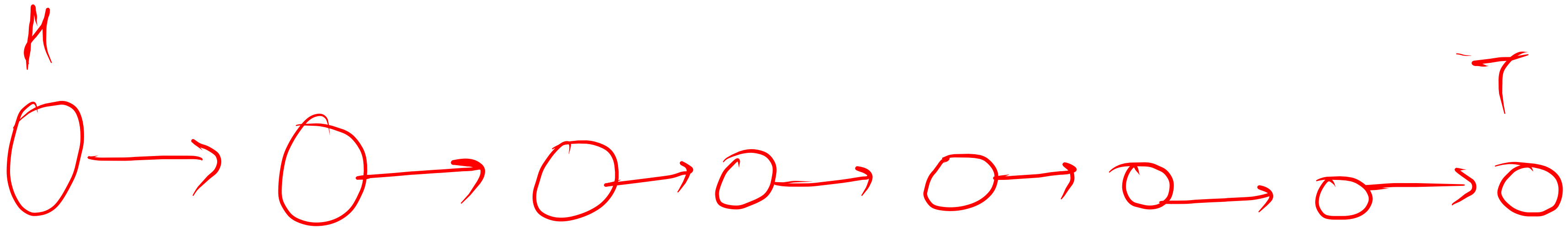


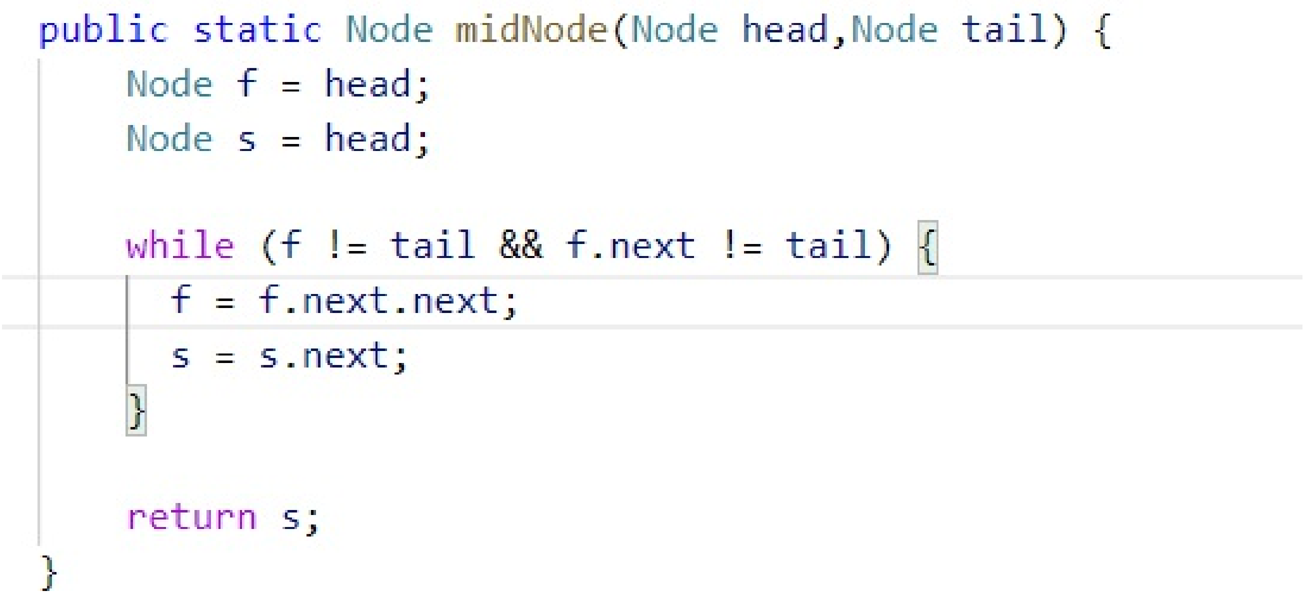
if(pi == null)

✓
res

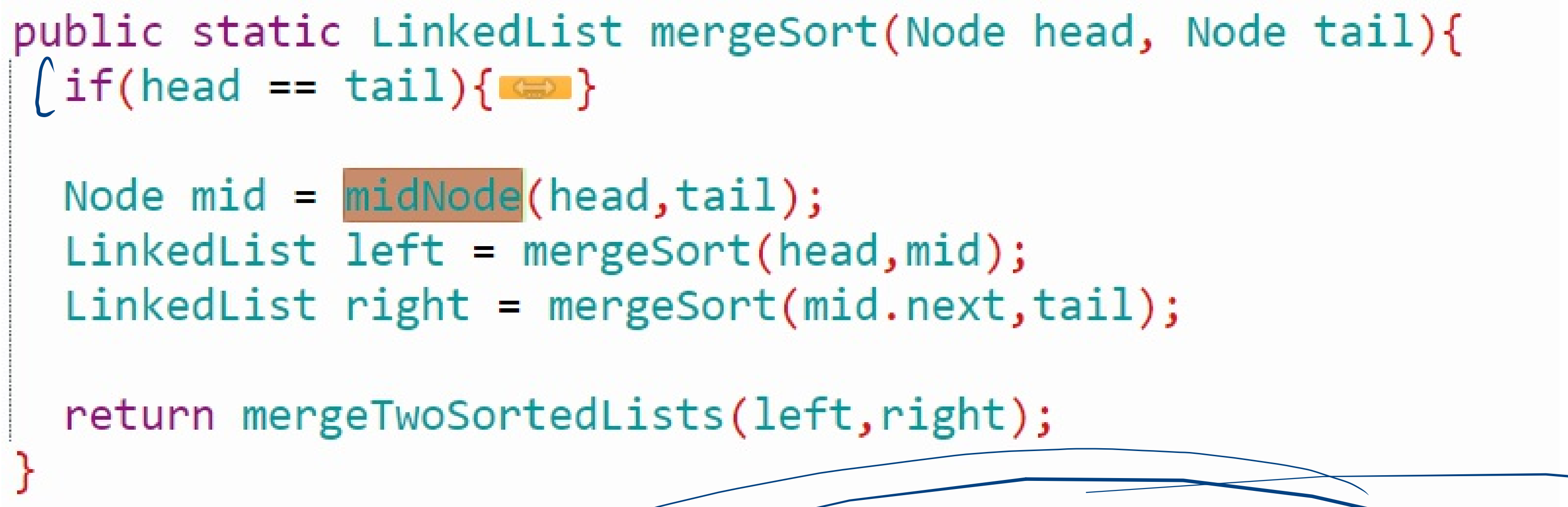
head : n
tail : n
size : 0


H
T
10 → 5 → 17 → 16 → 9 → 1 → 2 → 7 → 14
4k 5k 6k 7k 8k 9k 10k 11k 12k
(H/T)
mid
(H,mid)
(mid.next,T)
1 → 2 → 5 → 7 → 9 → 10 → 14 → 16 → 17
(4k/12k)
8k
5 → 9 → 10 → 16 → 17
1 → 2 → 7 → 14
(4k,8k)
6k
(9k,12k)
10k
5 → 10 → 17
9 → 16
1 → 2
7 → 14
(4k,6k)
5k
(7k,8k)
7k
(9k,10k)
9k
(11k,12k)
11k
5 → 10
17
16
9
1
2
7
14
(4k,5k)
4k
(6k,6k)
(7k,7k)
(8k,8k)
(9k,9k)
(10k,10k)
(11k,11k)
(12k,12k)
10
5
(4k,4k)
(5k,5k)





```
public static Node midNode(Node head,Node tail) {
    Node f = head;
    Node s = head;

    while (f != tail && f.next != tail) {
      f = f.next.next;
      s = s.next;
    }

    return s;
}
```



```
public static LinkedList mergeSort(Node head, Node tail){
  if(head == tail){⇔}

  Node mid = midNode(head,tail);
  LinkedList left = mergeSort(head,mid);
  LinkedList right = mergeSort(mid.next,tail);

  return mergeTwoSortedLists(left,right);
}
```

$T(n) \Rightarrow 2T(\frac{n}{2}) + 2n$

Implement

✓	</>	Is Linked List A Palindrome?	● Easy	10	✓Auth	0	□Public	✓Sol	20
✓	</>	Fold A Linked List	● Easy	10	✓Auth	0	□Public	✓Sol	21
	</>	Add Two Linked Lists	● Easy	10	✓Auth	0	□Public	✓Sol	22
✓	</>	Intersection Point Of Linked Lists	● Easy	10	✓Auth	0	□Public	✓Sol	23

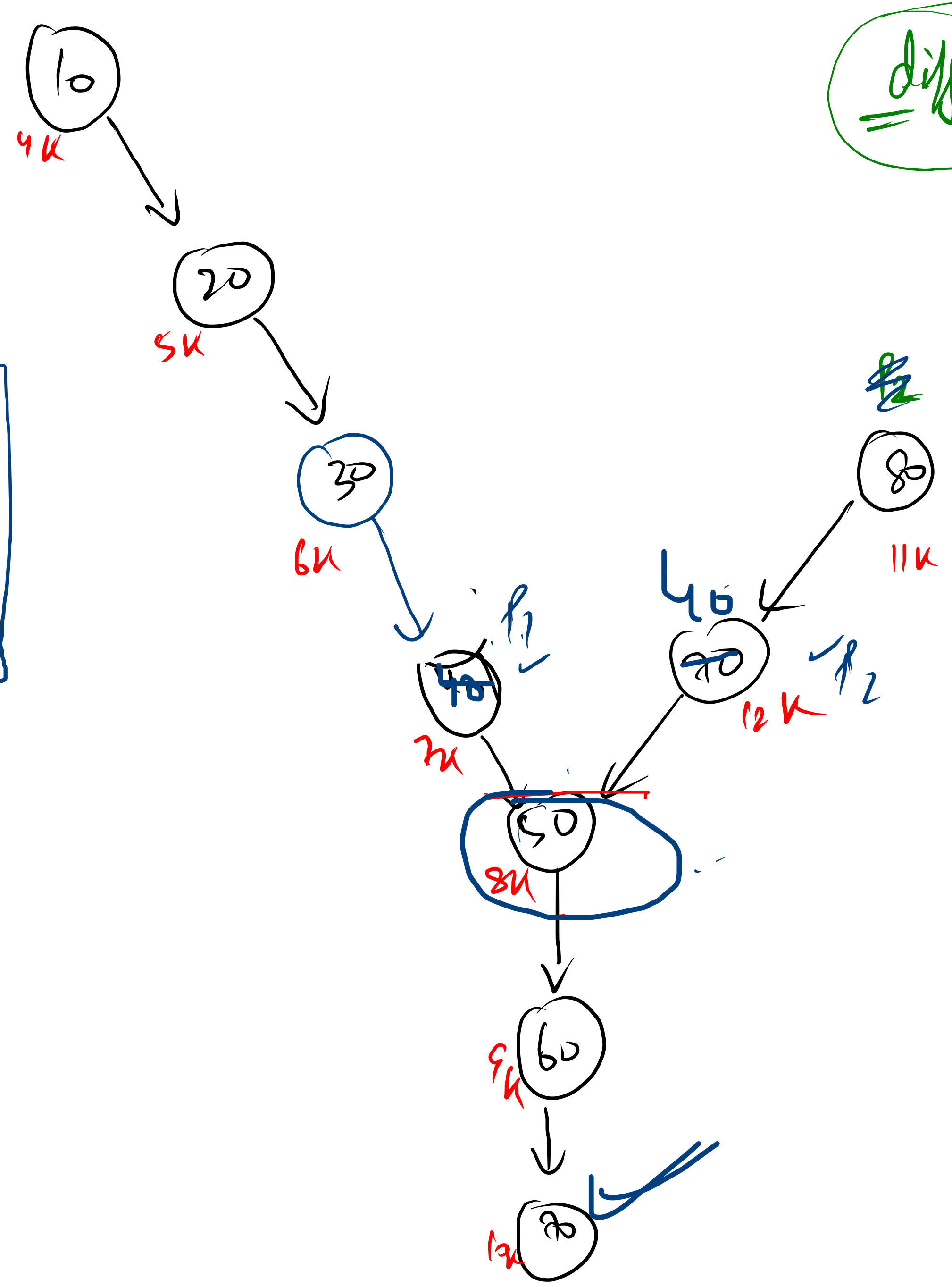
Intersection Point

tt1 one

head = 4k

tail = 10k

Size = 7



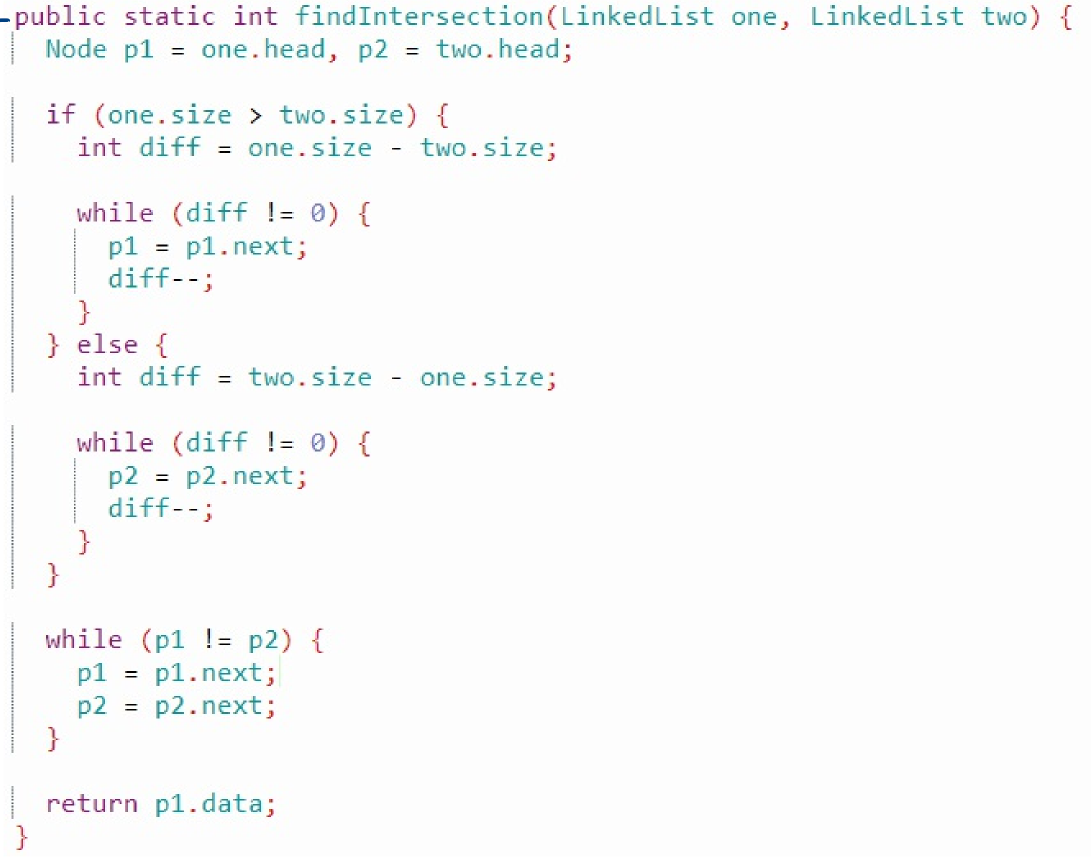
diff = ~~2~~ ~~1~~ 0

tt2 two

head = 11k

tail = 10k

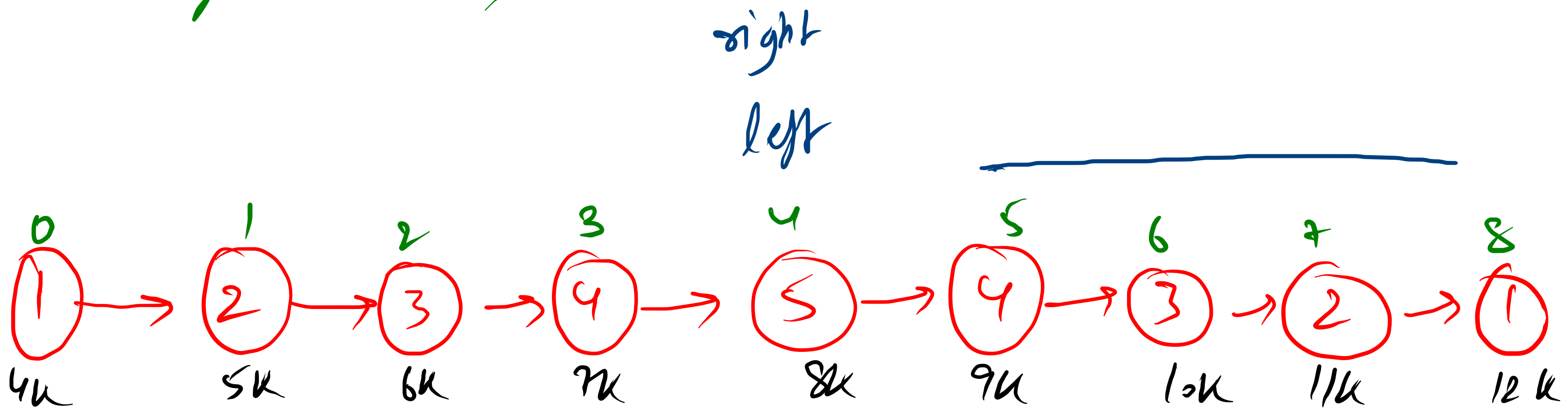
Size = 5



```
public static int findIntersection(LinkedList one, LinkedList two) {
  Node p1 = one.head, p2 = two.head;

  if (one.size > two.size) {
    int diff = one.size - two.size;

    while (diff != 0) {
      p1 = p1.next;
      diff--;
    }
  } else {
    int diff = two.size - one.size;

    while (diff != 0) {
      p2 = p2.next;
      diff--;
    }
  }

  while (p1 != p2) {
    p1 = p1.next;
    p2 = p2.next;
  }

  return p1.data;
}
```

head = 4K
tail = 12K
size = 9

```
static Node left = head;
```

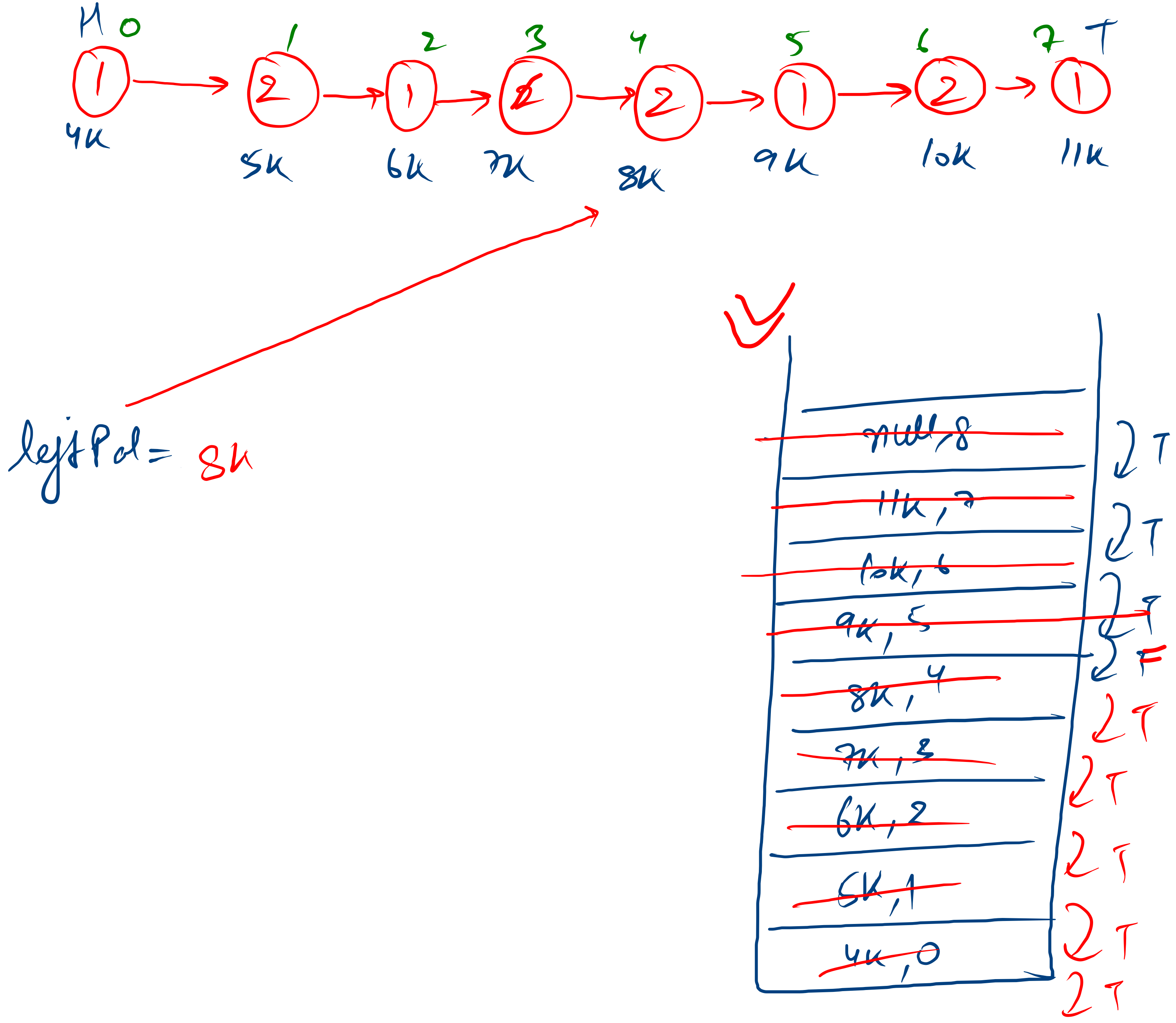


null,9
12K,8 T
11K,7 T
10K,6 T
9K,5 T
8K,4 T
7K,3
6K,2
5K,1
4K,0

```
left = left.next;
```

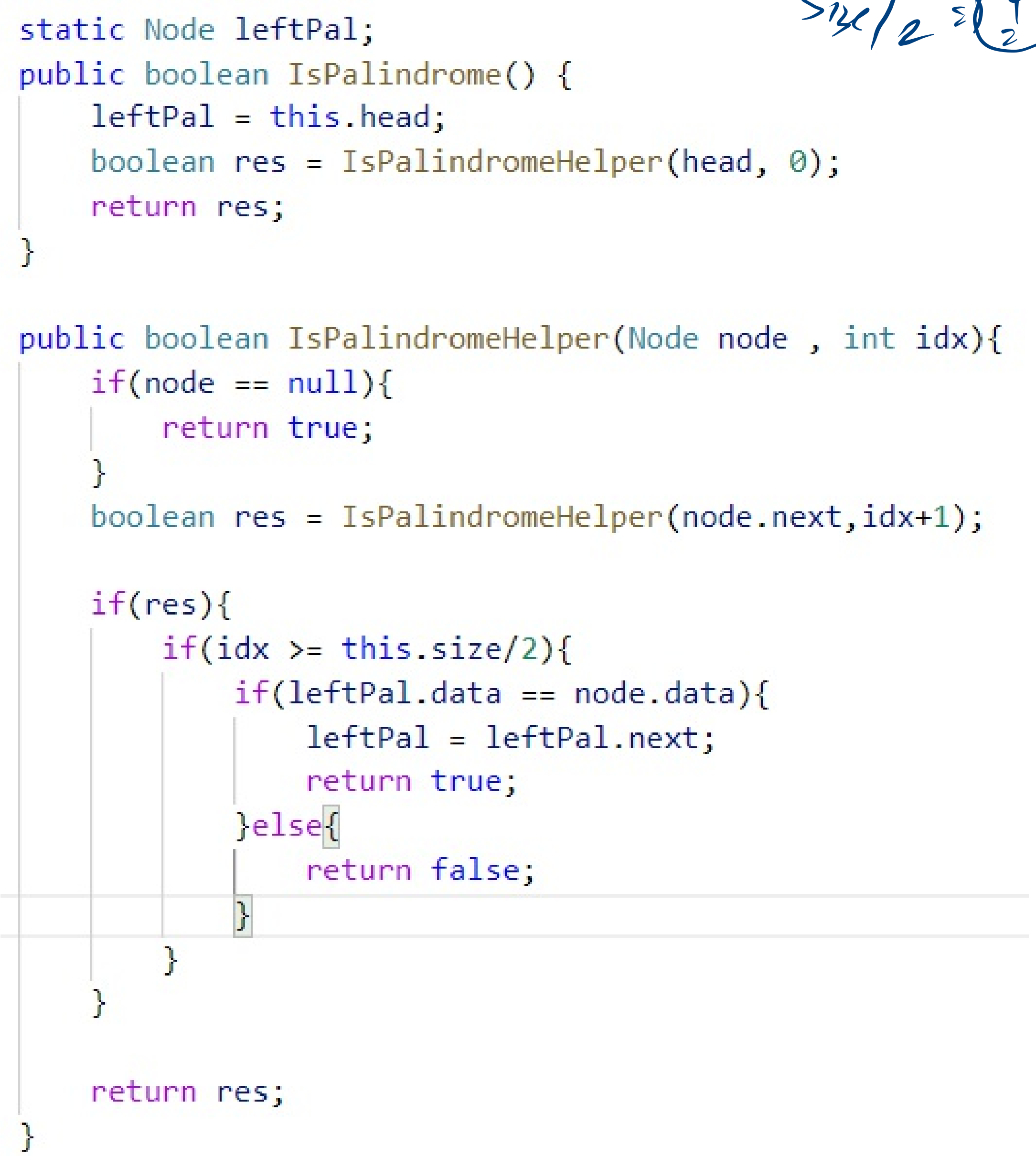
8
=
7
6
5

```
if(res == true) {
  if ( idx > size/2 ) {
    Node right = node;
    if( left.data == right.data) {
      left = left.next;
      return true
    }
  }
}
return false;
```



Size = 8

Size/2 = 4



```
static Node leftPal;
public boolean IsPalindrome() {
    leftPal = this.head;
    boolean res = IsPalindromeHelper(head, 0);
    return res;
}

public boolean IsPalindromeHelper(Node node , int idx){
    if(node == null){
        return true;
    }
    boolean res = IsPalindromeHelper(node.next,idx+1);

    if(res){
        if(idx >= this.size/2){
            if(leftPal.data == node.data){
                leftPal = leftPal.next;
                return true;
            }else{
                return false;
            }
        }
    }

    return res;
}
```

head : 4K

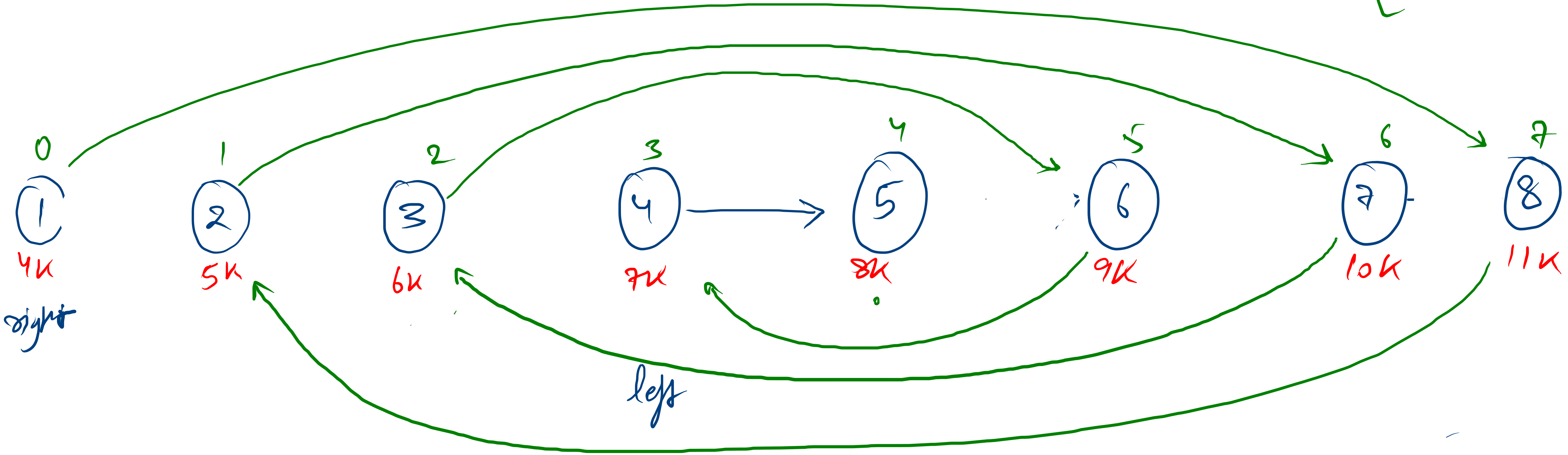
tail : ~~11K~~ 8K

size : 8

```
Node nbr = left.next;
left.next = right
right.next = nbr
left = nbr
```

idx > size/2

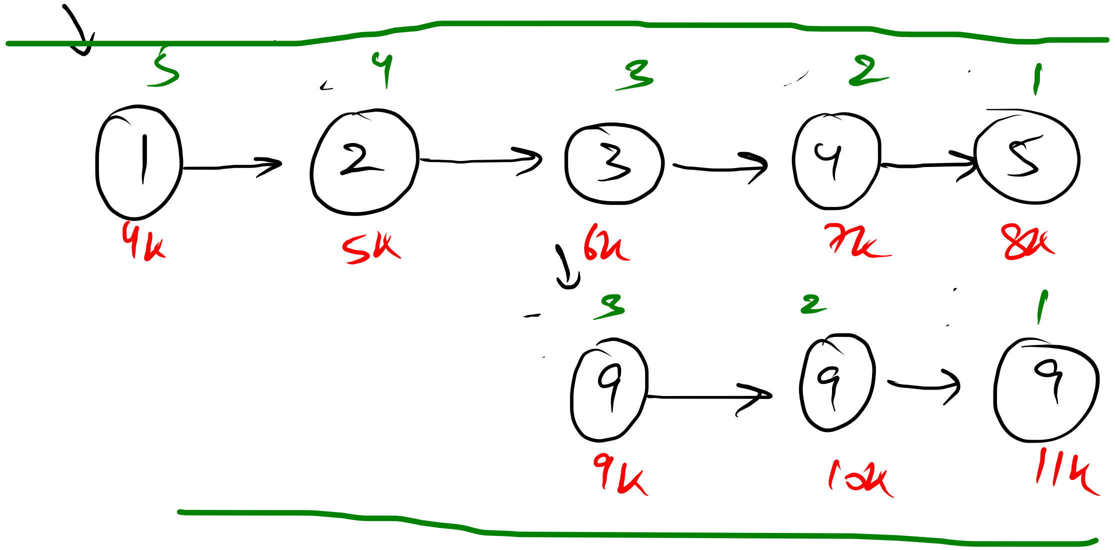
```
idx == size/2
    tail = right.
    tail.next = null
```

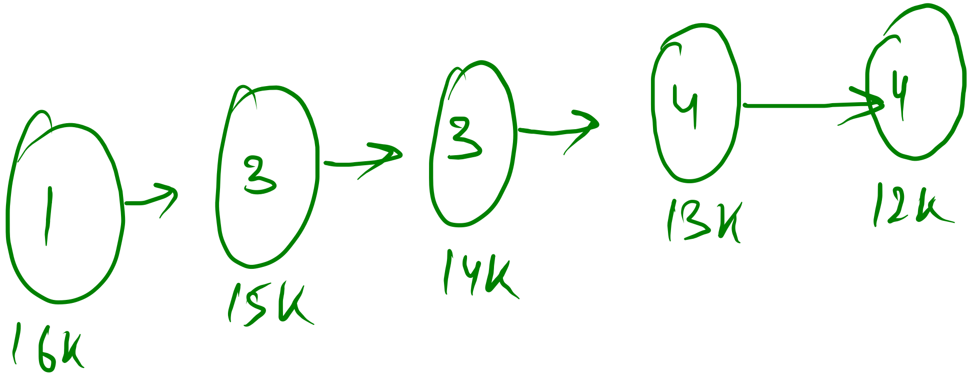


1 → 8 → 2 → 7 → 3 → 6 → 4 → 5

H: 4k
T: 8K
S: 5

H: 9K
T: 11k
S: 3





H : 16k
T : 12K
S : 5

99K

(n1.data + ~~n2.data~~ n2.data + carry) → Sum = 1
digit = 1
carry = 0

return type int

~~null, null, 0, 0, 9k~~

~~8K, 11k, 1, 1, 99K~~ 1

~~7K, 10K, 2, 2, 99K~~ 1

~~6K, 9K, 3, 3, 99K~~ 1

~~5K, 9k, 4, 3, 99K~~ 0

~~4k, 9k, 5, 3, 99k~~ 0

n1, n2, p1, p2, res