8-June-2022

DSA-301

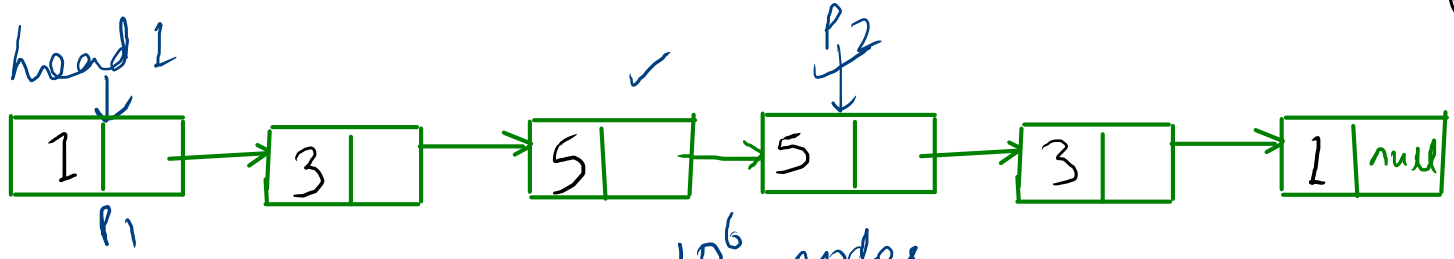
LECTURE # 9

TOPIC: LINKED LISTS - 3

- Palindrome Linked List
- Merge 2 Linked Lists
- Detect cycle in Linked List

Palindrome Linked List

if palindrome → ret True
else → ret False



10^6 nodes

Approaches:

1) Array / stack → Space comple O(N)

2) Mid

3) Reverse the array → O(N)

4) Reversed the link → original head2 O(N) extra space

prev
cur
future

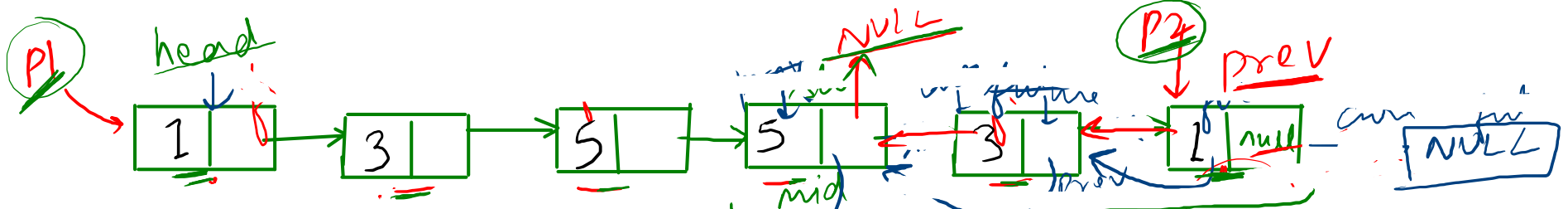
[1 3 5 | 5 3 1]

mid

Step 1) Find middle

Step 2) Reverse the Linked List

Step 3) Comparing using 2 pointers



Step 1) Find mid

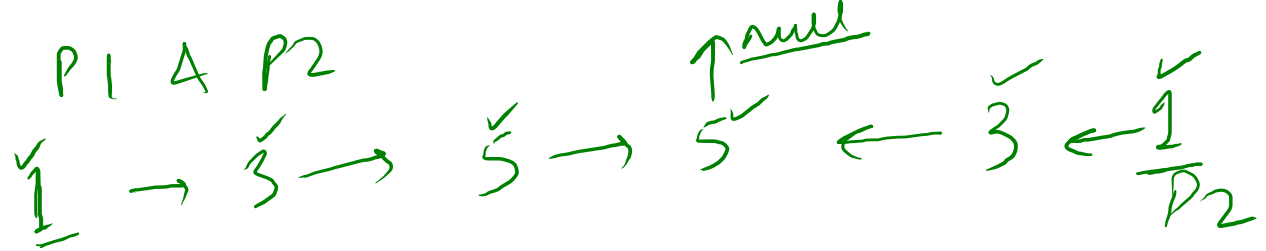
(slow/null) head 2 = slow

Step 2) Reverse the 2nd half (right)

```
var prev = null
" curr = slow  // new head
" future = null
while (curr != null)
{
    future = curr.next
    curr.next = prev
    prev = curr
    curr = future
}
var P2 = prev // head2
```

```
findMid (head)
{
  var slow = head
  var fast = head
  while(f != null && (f.next != null
  {
    → s = s.next
    → f = f.next.next
  }
  ret slow
}
```

Step3) compare P1 & P2



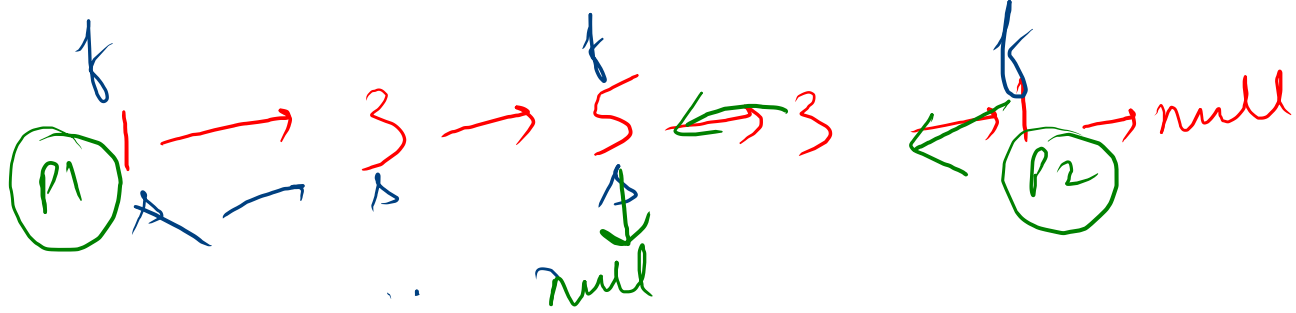
while (P2 != P1 null)

```
{
    if ( P1.data != P2.data)
    {
        return false
    }
    P1 = P1.next
    P2 = P2.next
}
ret true
```

Step1) Mid node

Step2) Reverse

Step3) compare



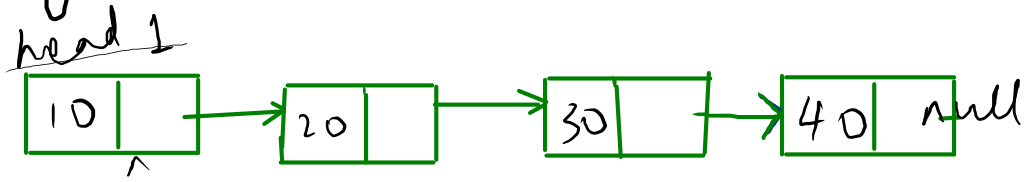
while (P2 != null)

LL_3/isPalinLL.js ×

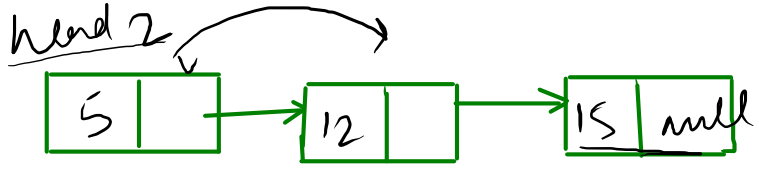
```
36
37 ▼  isPalindrome(){
38      //1) find middle
39      var p1=this.head
40      var s=this.head
41      var f=this.head
42 ▼    while(f!=null && f.next!=null){
43        s=s.next
44        f=f.next.next
45      }
46      //s= mid node
47
48
49      //2)reverse the right half
50      var prev=null
51      var cur= s
52      var future=null
53 ▼    while(cur!=null){
54        future=cur.next
55        cur.next = prev
56        prev=cur
57        cur=future
58      }
59      //new head= prev
60      var p2 = prev
61
62      //3) comparing p1 &p2
```

```
61
62      //3) comparing p1 &p2
63 ▼    while(p2!=null){
64 ▼      if(p1.data != p2.data){
65          return false
66        }
67        p1=p1.next
68          p2=p2.next
69      }
70      return true
71
72    }
73  }
74  l1 = new LinkedList()
75  l1.insertAtEnd(10)
76  l1.insertAtEnd(20)
77  l1.insertAtEnd(30)
78  l1.insertAtEnd(20)
79  l1.insertAtEnd(70)
80  // l1.printLL()
81
82  ans = l1.isPalindrome()
83 ▼ if(ans==true){
84    console.log("Its PAlindrome")
85  }
86 ▼ else{
87    console.log("NOT PAlindrome")
88  }
```

Merge 2 Linked Lists

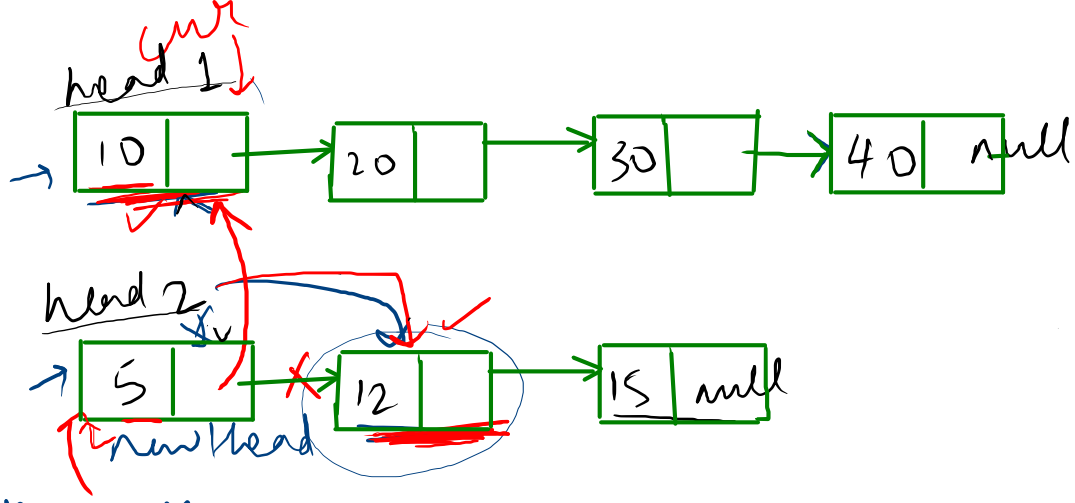


→ N nodes 10^6



→ M nodes 10^5

Result → 5 → 10 → 12 → 15 → 20 → 30 → 40 → null



Step 1) compare (head1 < head2)

var newHead = head2

head2 = head2.next

Step 2) while (head1 != null & head2 != null)

{

if (head1.data < head2.data)

{

cur.next = head1

head1 = head1.next

cur = cur.next

(1) if (head1.data < head2.data)

{

newHead = head1

head1 = head1.next

cur = newHead

}

else

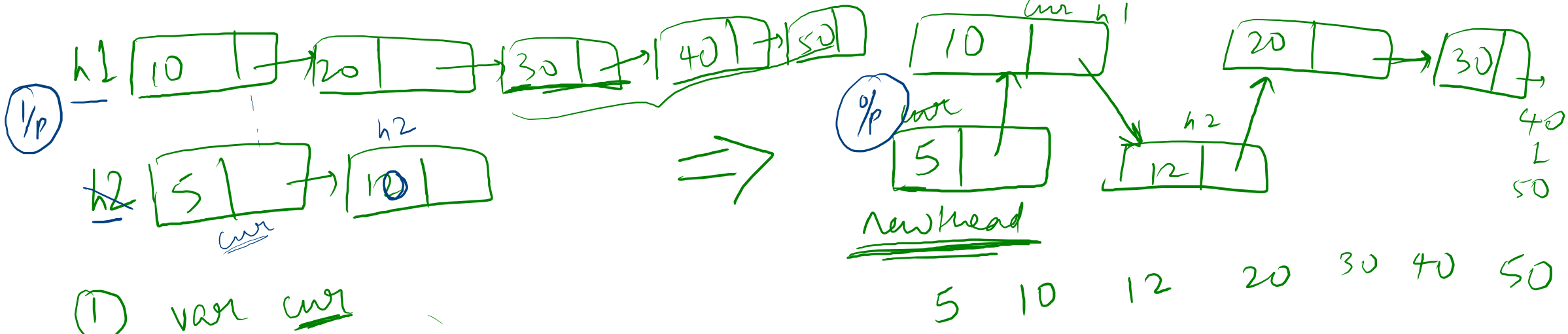
{

newHead = head2

head2 = head2.next

cur = newHead

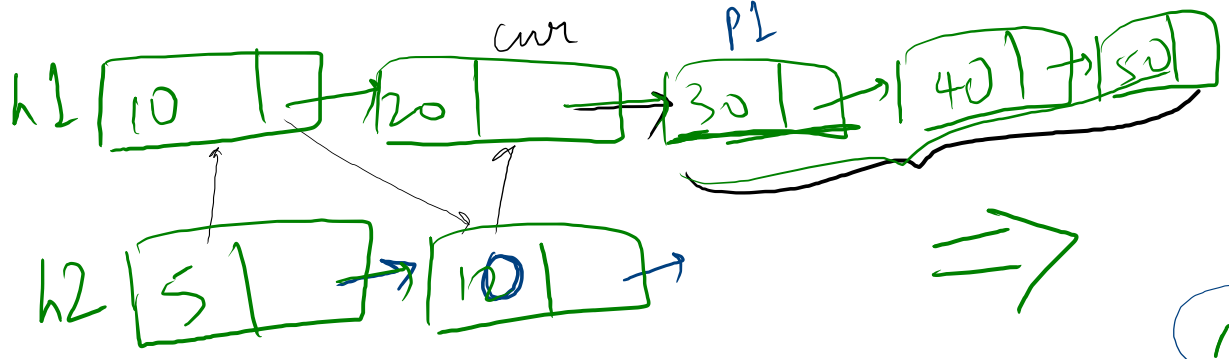
}

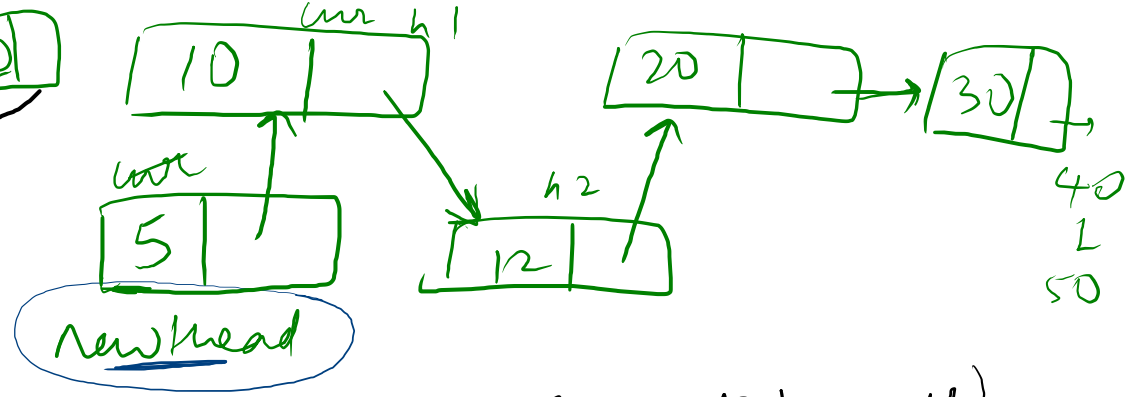


(1) var cur
var newhead

step1 → assigning newhead

step2 → while( )
{ cur → next smallest node
h1.dt < h2.data






```
function mergeLL(h1, h2)
{
    var curr = null
    var newHead = null
    var P1 = h1
    var P2 = h2

(1) if(P1.data < P2.data)
    {
        newHead = P1
        P1 = P1.next
    }   curr = newHead
    else
    {
        newHead = P2
        P2 = P2.next
    }   curr = newHead
```

(2)

```
while(P1 != null && P2 != null)
{
    if(P1.data < P2.data)
    {
        curr.next = P1
        curr = curr.next
        P1 = P1.next
    }
    else
    {
        curr.next = P2
        P2 = P2.next
        curr = curr.next
    }
}
```

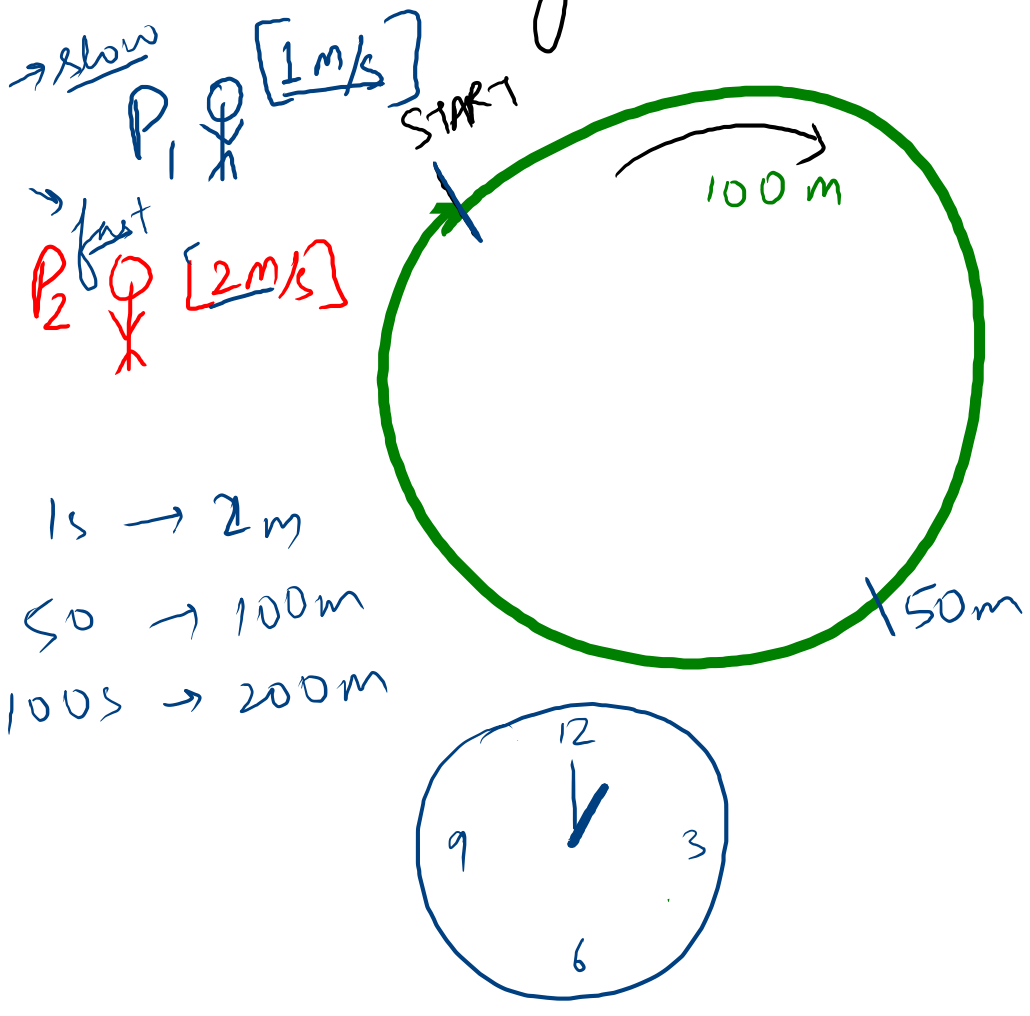
(3)

```
if(P1 != null)
{
    curr.next = P1
}
if(P2 != null)
{
    curr.next = P2
}
```

(4)

```
return newHead
}
```

Detect cycle in Linked List



1s → 1m

50 → 100m

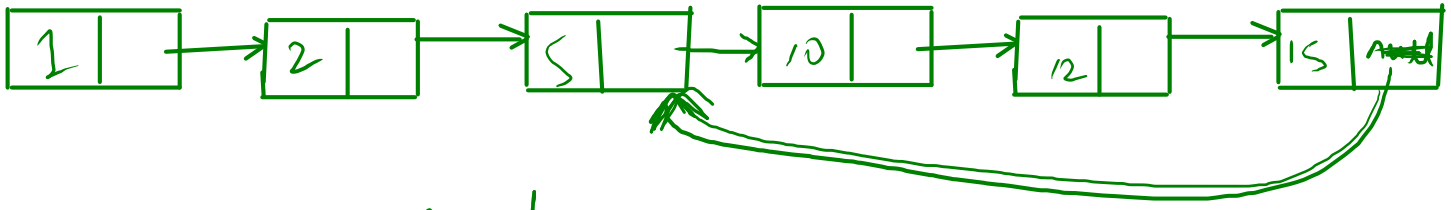
100s → 200m

	P1	P2
50 sec	50m	start
100 sec	start	start

if not cycle LL (ff != null)
slow & fast never meet

if LL has a cycle

if (slow == fast)
ret "Yes"



cycle

visited

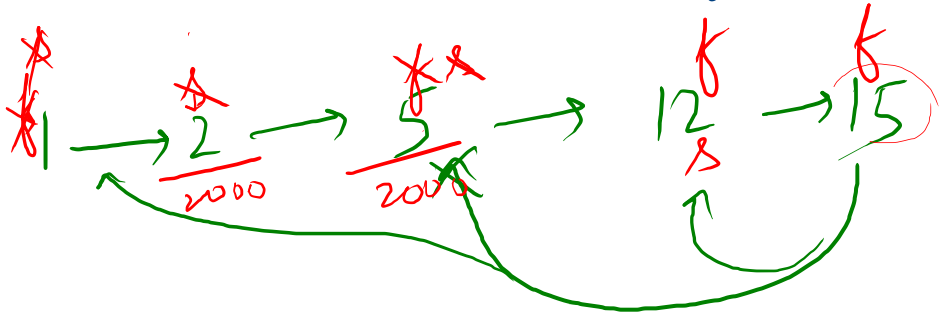
Node
{ data ✓
next ptr ✓
visited

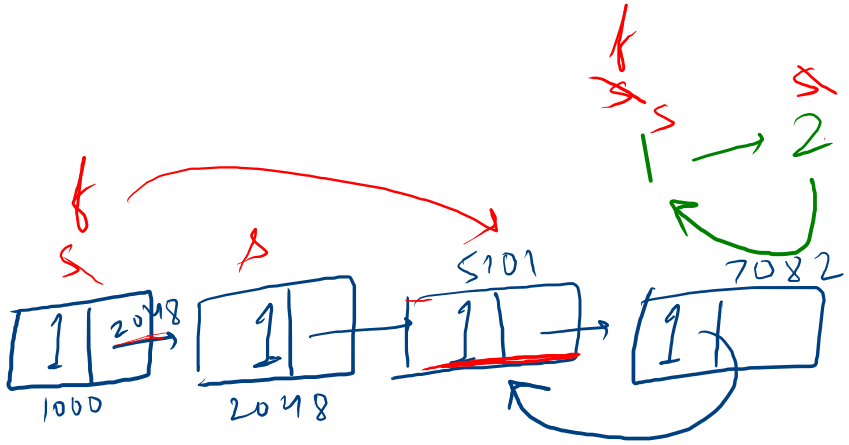
10^6

Floyd cycle-detection

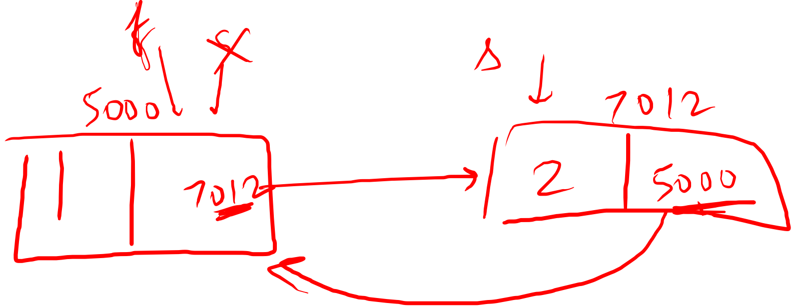
tortoise & Hare

(slow) (fast)






```
function isCycle(head)
{
    if(head==null)
        ret false
    s = head
    f = head
→   while(f!=null && f.next!=null)
    {
        f = f.next.next
        s = s.next
        if(f==s)
            ret TRUE
    }
    ret false
}
```



Node
{
data
next → add of next Node
}

Read Only

s= obj <x12 - - - >
f= < >

```
while( )
{
  s = s.next
  f = f.next.next
  if(s == f)
  { ret TRUE
    console.log(s.next)
    (f.next
  }
}
```