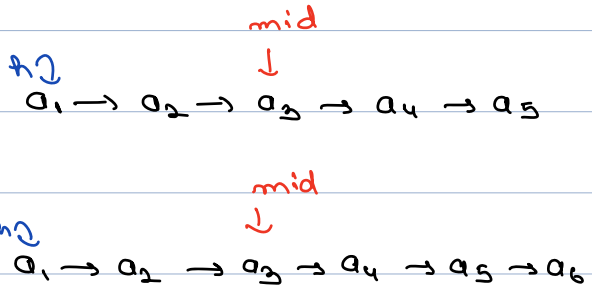
Ques Given a LL, find middle element.

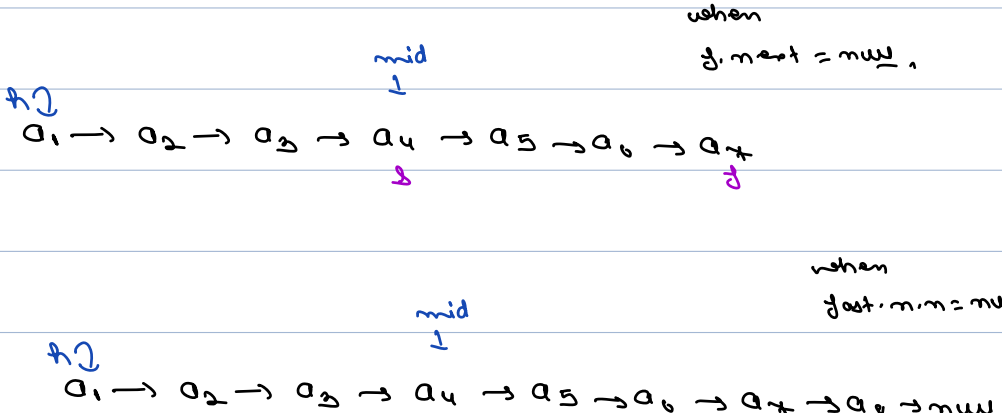
e.g.)



Soln 1:- find size of LL.
↳ Travel till half of it & return.

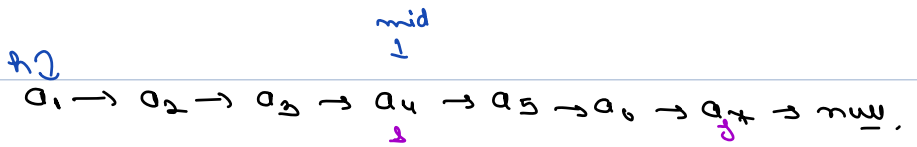
T.C → O(n), S.C → O(1).

Constraint :- Do it in 1 iteration

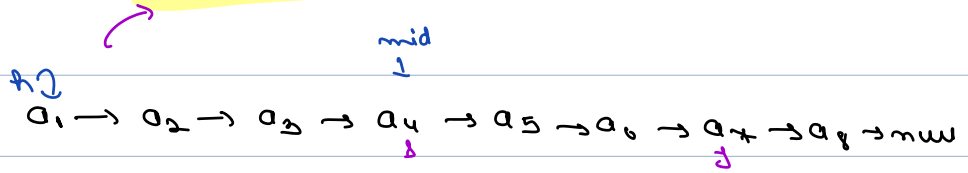


```
Node mid (Node h) {
    if (h == null) { return null; }
    Node s = h;
    Node f = h;
    while (f.next != null && f.next.next != null) {
        s = s.next;
        f = f.next.next;
    }
    return s;
}
```

T.C → O(n)
S.C → O(1).



while (f.next != null && f.next.next != null) {



while (f.next != null && f.next.next != null) {

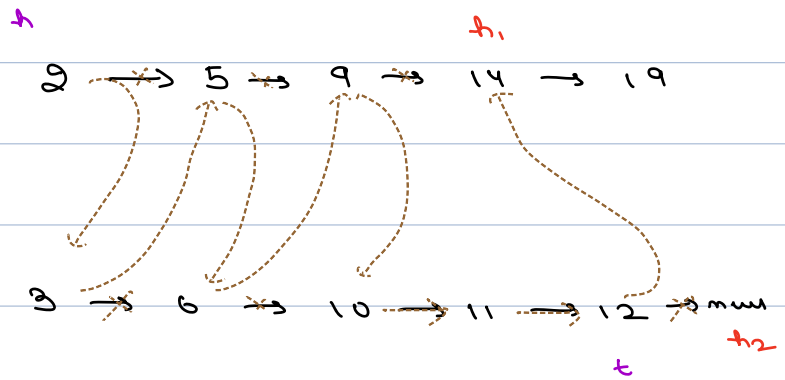
Ques Given two sorted LL, merge them into a single sorted LL.

e.g.) LL1) 2 → 4 → 6 → 8 → 10 (h1)

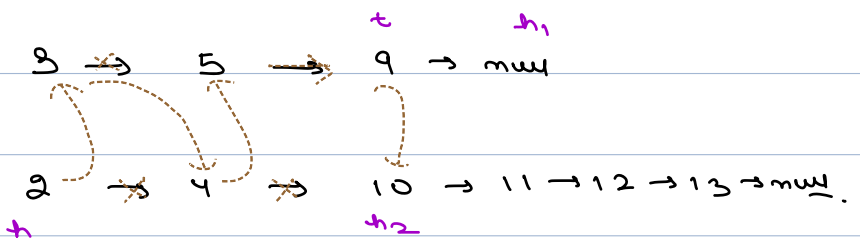
LL2) 1 → 3 → 5 → 7 → 9 (h2)

O/P → 1 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 9 → 10.

e.g



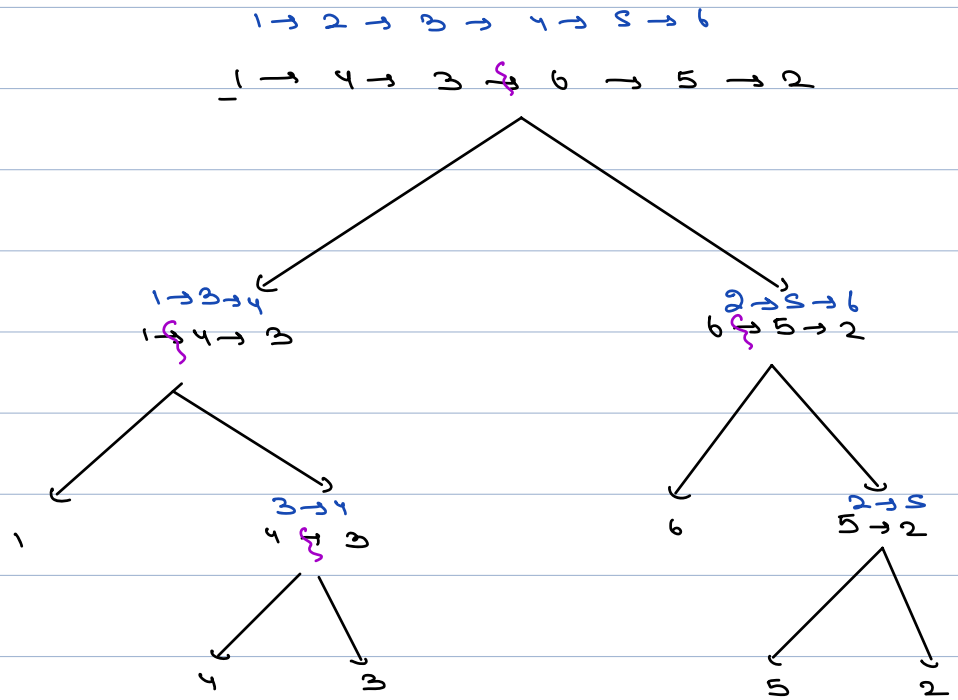
e.g,



```
Node merge (Node h1, Node h2) {
    if (h1 == null) { return h2; }
    if (h2 == null) { return h1; }

    Node h, t;
    if (h1.data < h2.data) { h = h1, t = h1, h1 = h1.next; }
    else { h = h2, t = h2, h2 = h2.next; }

    while (h1 != null && h2 != null) {
        if (h1.data < h2.data) {
            t.next = h1
            t = h1
            h1 = h1.next
        }
        else {
            t.next = h2
            t = h2
            h2 = h2.next
        }
    }
    if (h1 == null) {
        t.next = h2;
    }
    if (h2 == null) { t.next = h1; }

    return h;
}
```

T.C → O(n+m)

S.C → O(1)

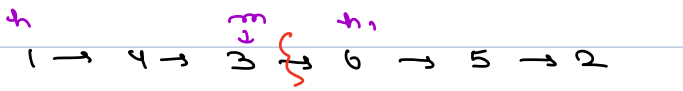

1 → 2 → 3 → 4 → 5 → 6
1 → 4 → 3 → 6 → 5 → 2
1 → 3 → 4
1 → 4 → 3
2 → 5 → 6
6 → 5 → 2
3 → 4
4 → 3
2 → 5
5 → 2

Ques Merge sort on LL

I/p 1 → 4 → 3 → 6 → 5 → 2

O/p 1 → 2 → 3 → 4 → 5 → 6

Step1 :- find middle



h1 = m.next

m.next = null

Step2 :- Call recursion to both of them.

h = mergeSort(h)

h1 = mergeSort(h1)

Step3 :- Merge both the sorted LL.

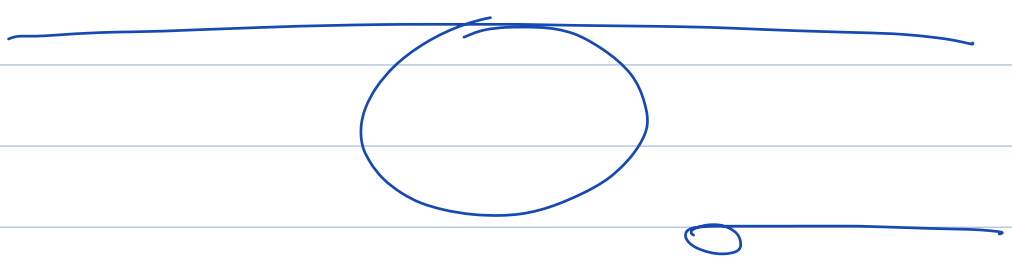
```
                                                    → T(n)
node    mergeSort ( node h) {
  if (h == null || h.next == null) {
      return h;
  }
                                          → n
  node m = middle (h);
  h1 = m.next;
  m.next = null;
                                  → T(n/2)
  h = mergeSort (h)      → T(n/2)
  h1 = mergeSort (h1)
                                → n
  h3 = merge (h, h1);
  return h3;
}
```

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

↳ T.C → $O(n \log n)$

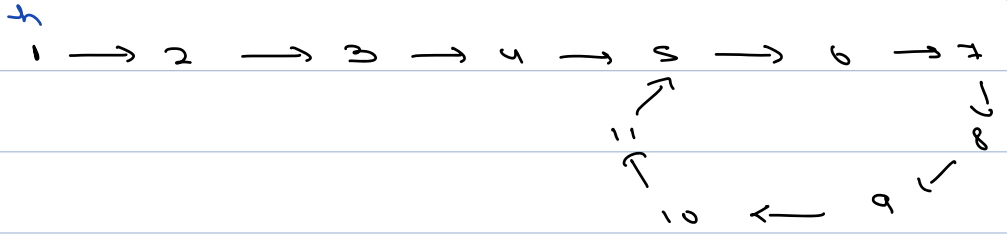
S.C → $O(\log n)$

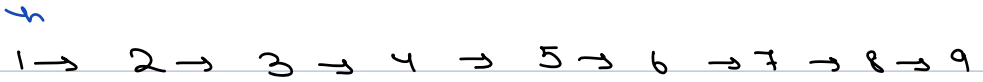
↳ Stack Space.



Ques Detect cycle in a LL.

↳ floyd cycle detection algo.



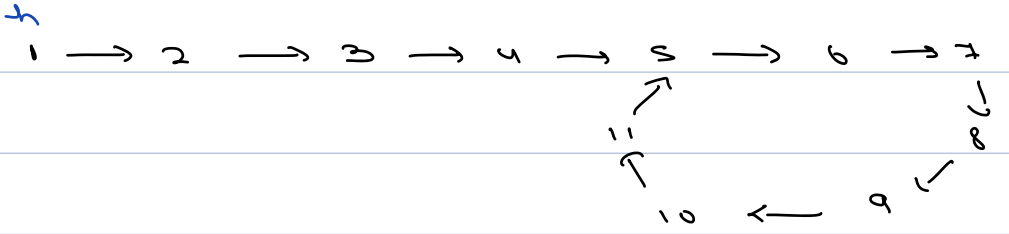


idea :- put a temp at the head.

if temp reaches null (temp = temp.next)

not a cycle

cycle



idea 2 :- HashSet < node >.

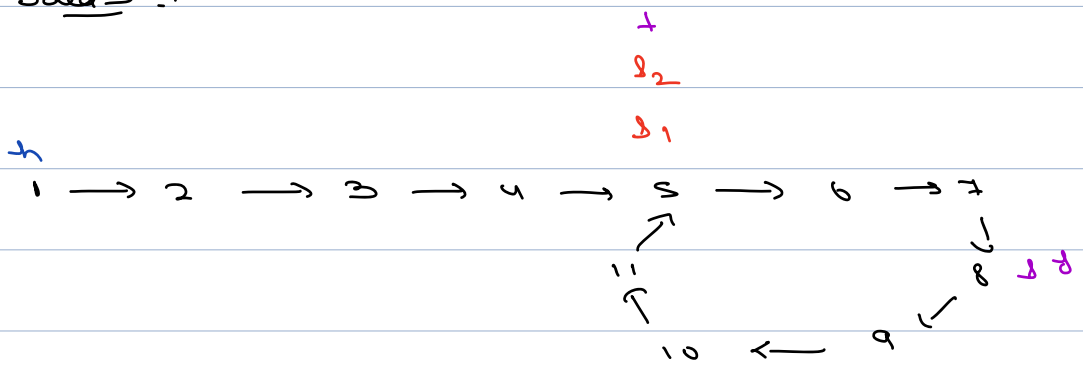
idea :- Iterate on LL, keep on storing nodes. If you reach null ↓ no cycle

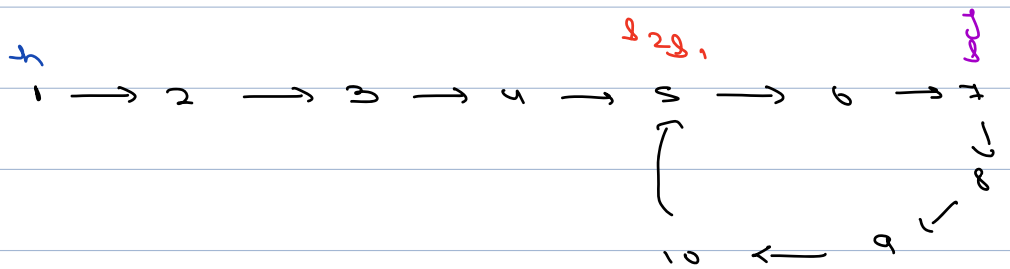
T.C → O(n)

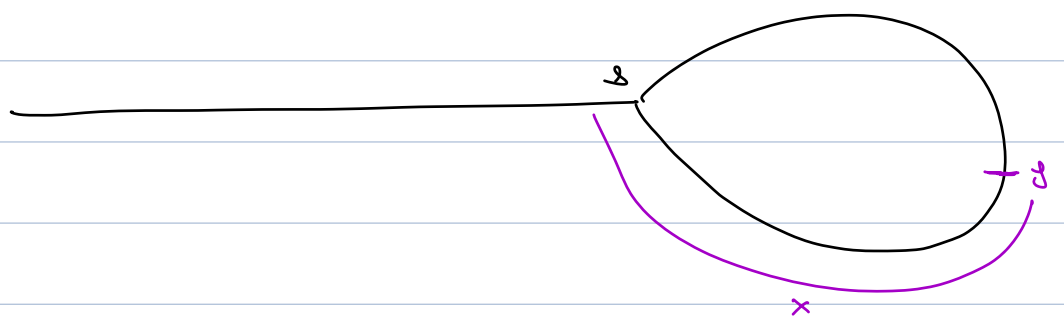
S.C → O(n)

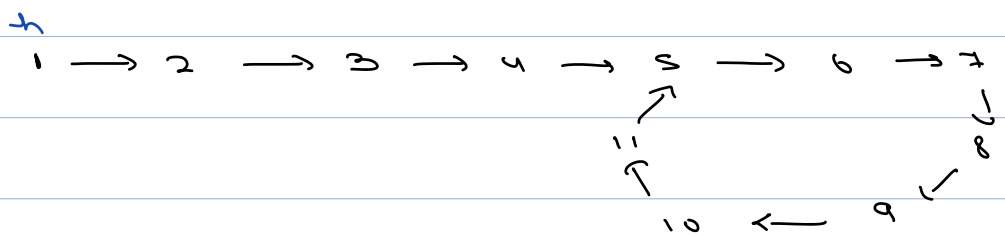
if address starts repeating, there is a cycle.

Idea 2 :-


1
2
3
4
5
6
7
8
9
10
11
slow
fast


1
2
3
4
5
6
7
8
9
10
slow
fast

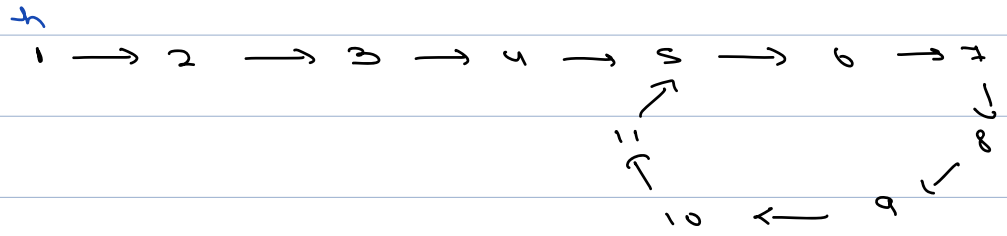

x
fast

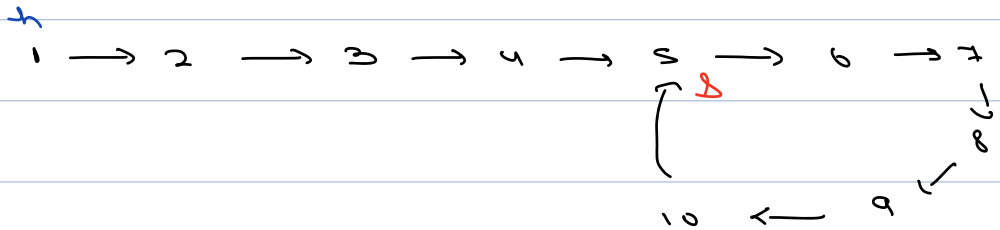

1
2
3
4
5
6
7
8
9
10

why will they intersect ?

when slow pointer enters the cycle, let's say the distance between slow and fast is X. Now we observe that after every step distance will reduce by one so eventually they will meet .

Ques find Starting Point of a cycle.





Start of cycle

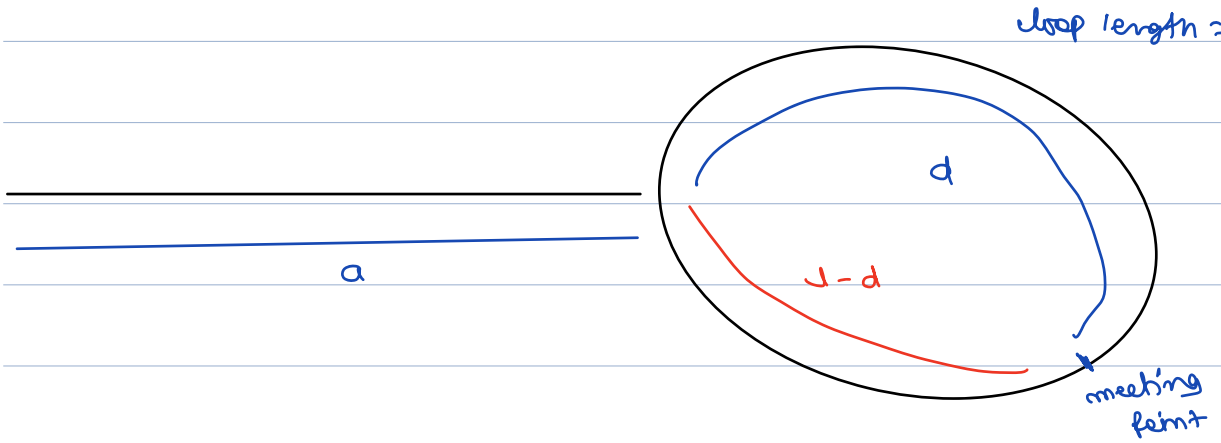
Take two pointers s_1 & s_2.

$s_1 \rightarrow$ head ;

$s_2 \rightarrow$ Intersection of s & f .

move both the pointers one step at a time. The point there where they will meet will be start of the cycle.

```
bool detectFindRemoveCycle(Node h){

    Node s = h;
    Node f = h;
    bool isCycle = false;                    // O(1)
    while (f != null && f.next != null){
        s = s.next;
        f = f.next.next;
        if (s == f){
            isCycle = true;
            break;
        }
    }
    if (isCycle == false){
        return false
    }

    s1 = h, s2 = f;                          // O(n)
    while (s1 != s2) { s1 = s1.next, s2 = s2.next}

    Node t = s1;                             // O(n)
    while (t.next != s1) {
        t = t.next;
    }
    t.next = null;
    return null;
}
```



$ds = a + d$

$df = a + c*l + d$

$df = 2 * ds$

$\Rightarrow a + cl + d = 2a + 2d$

$cl = a + d$

$\Rightarrow a = cl - d$

$\Rightarrow a = cl - d + l - l$

$\Rightarrow a = cl - l + l - d$

$\Rightarrow a = l(\underbrace{c-1}_{x}) + l - d$

$\Rightarrow a = xl + l - d$

help (int [] arr) {

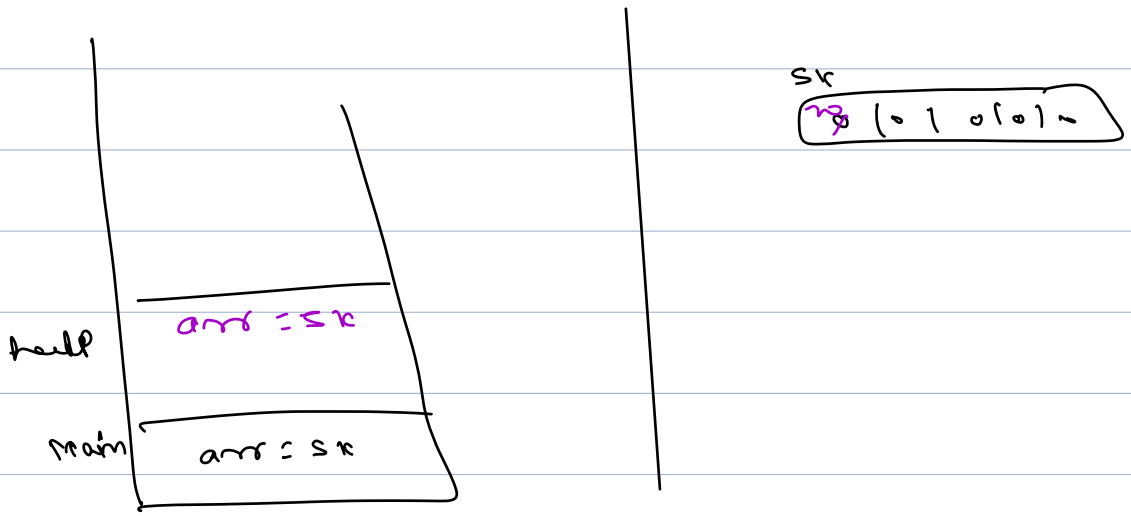
arr[0] = 20

}

3

main()

```
int [] arr = new int [5];
help(arr);
```



fn (...)

```
help ( int [] arr) {
    arr = new int [10];
}

main()
{
    int [] arr = new int [5];
    help(arr);
}
```