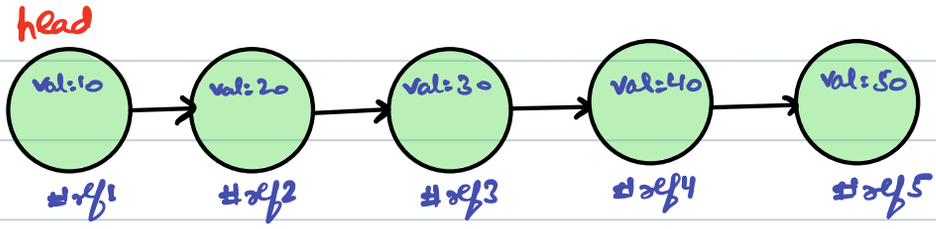
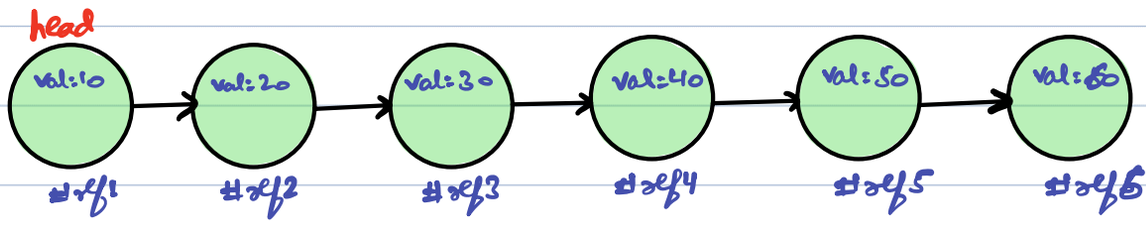
Today's agenda

- Find mid
- floyd cycle
- why n/2 in binary search

Q) Given head of a linked list, find the mid of it.



→ ans: 30



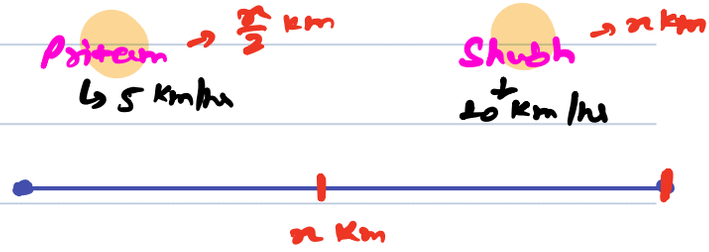
→ ans: 30

Idea1

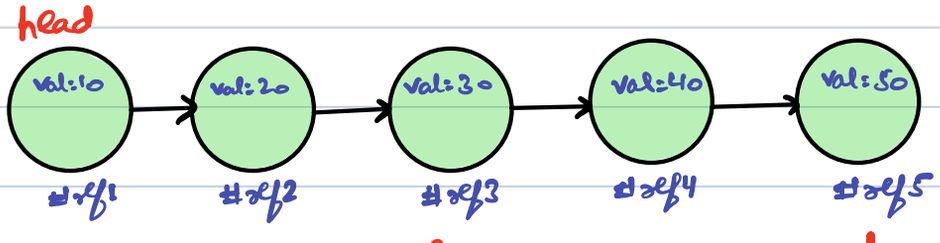
↳ iterate and find out length of LL. Iterate again and return $\frac{length}{2}^{th}$ element.

T.C: $O(N + N/2) = O(N)$ S.C: $O(1)$

Idea2 → Single iteration

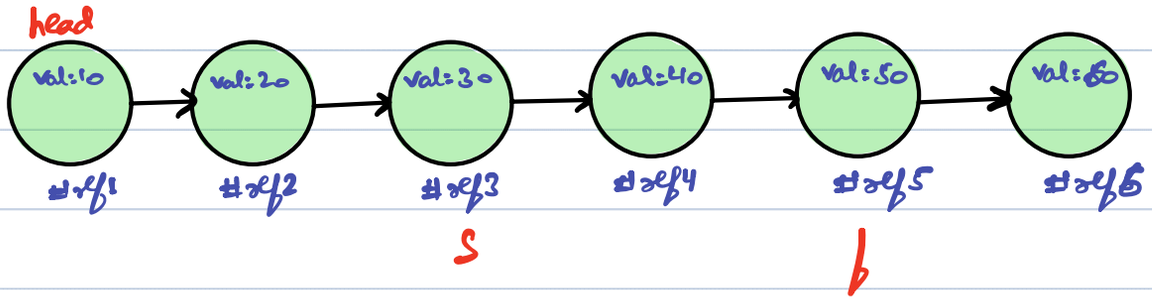


odd



f.next == null → exit the loop → ans: s.val

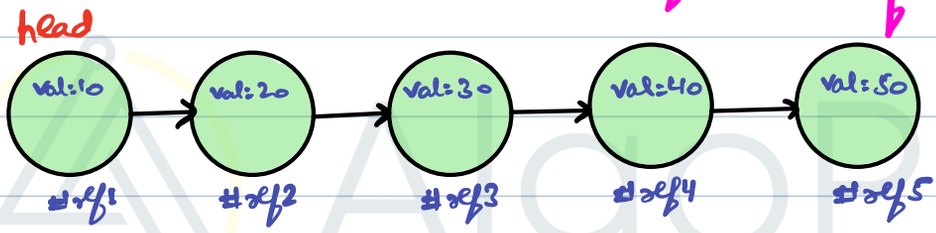
even

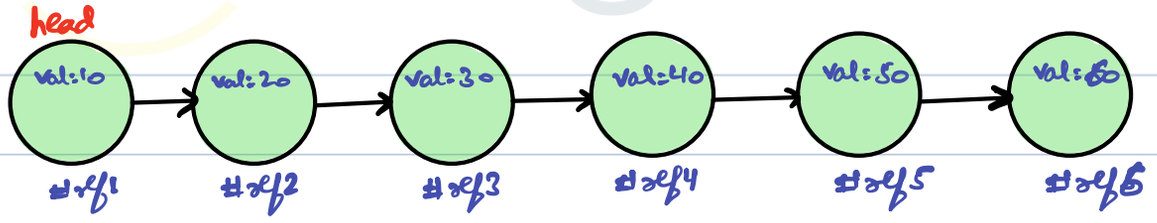


f.next.next == null → exit → ans: s.val

f.next.next.next = null

f.next = null





f.next.next == null

f = null

(f.next == null || f.next.next == null) → exit

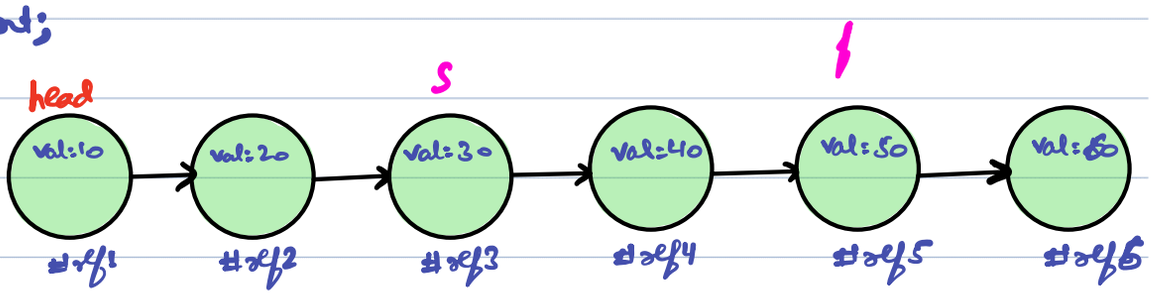
(f.next != null && f.next.next != null) → run

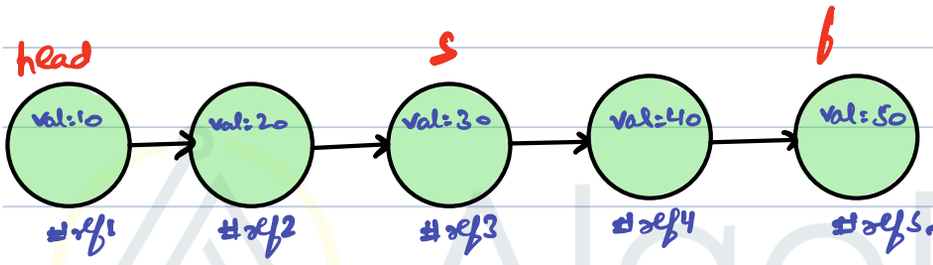
//Pseudo code

```
Node mid (Node head){
    Node s= head;
    Node f= head;

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

    Return s;
}
```

Tracing

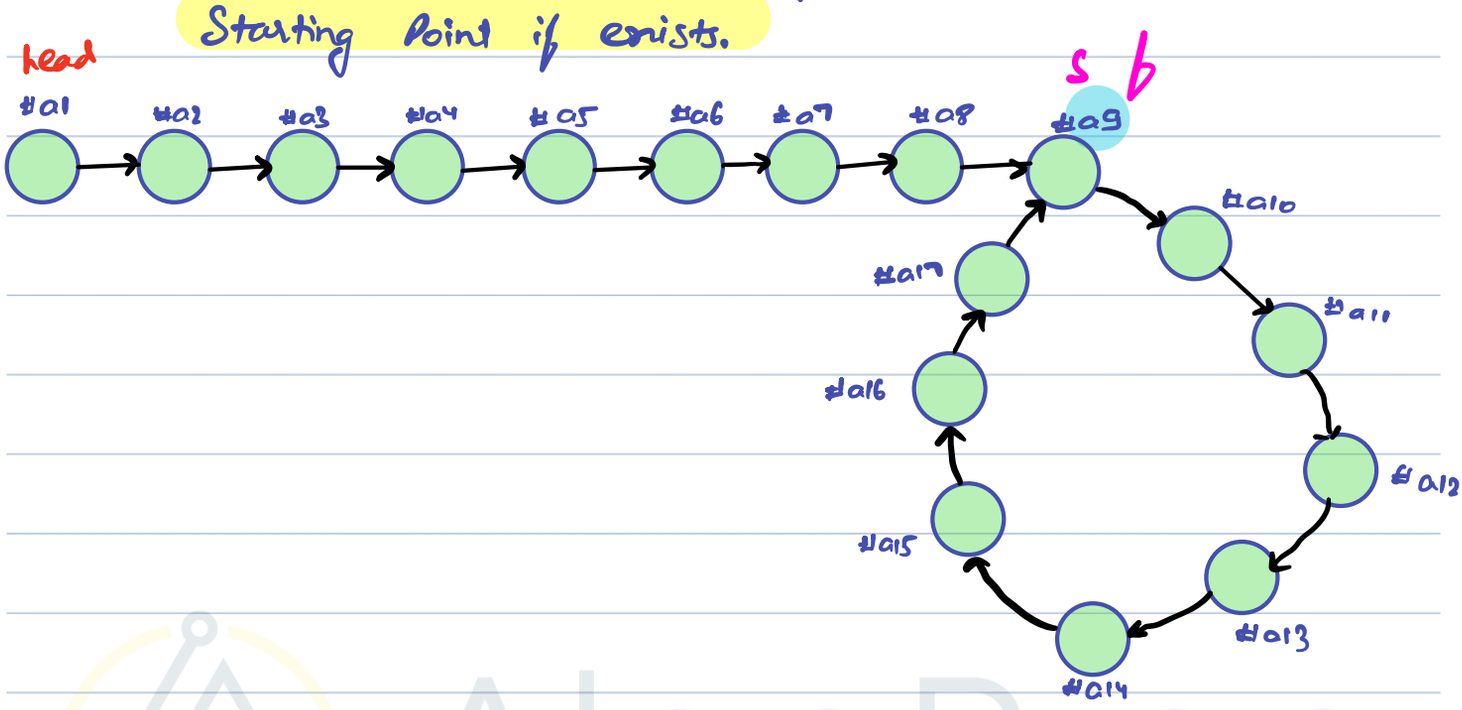
```
while (f.next != null && f.next.next != null ){
    s = s.next;
    f = f.next.next;
}
```





floyd cycle

Q) Given a linked list, check for cycle & return the starting point if exists.



Ideal

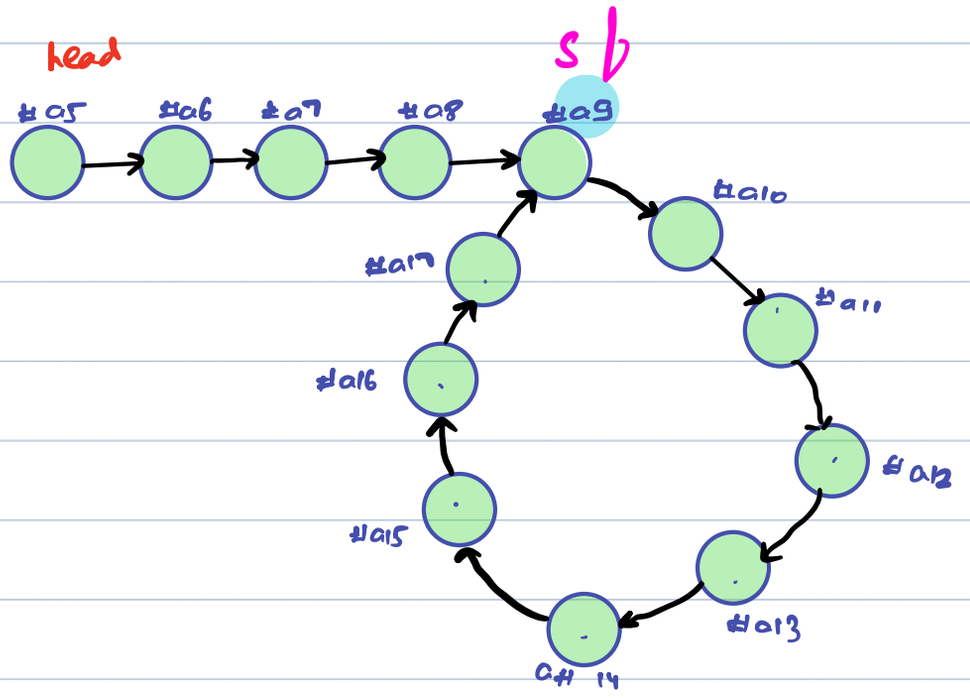
↳ we HashSet (node) T.C: O(N) S.C: O(N)

#a1 #a2 ... #a9 ... #a17 → #a9

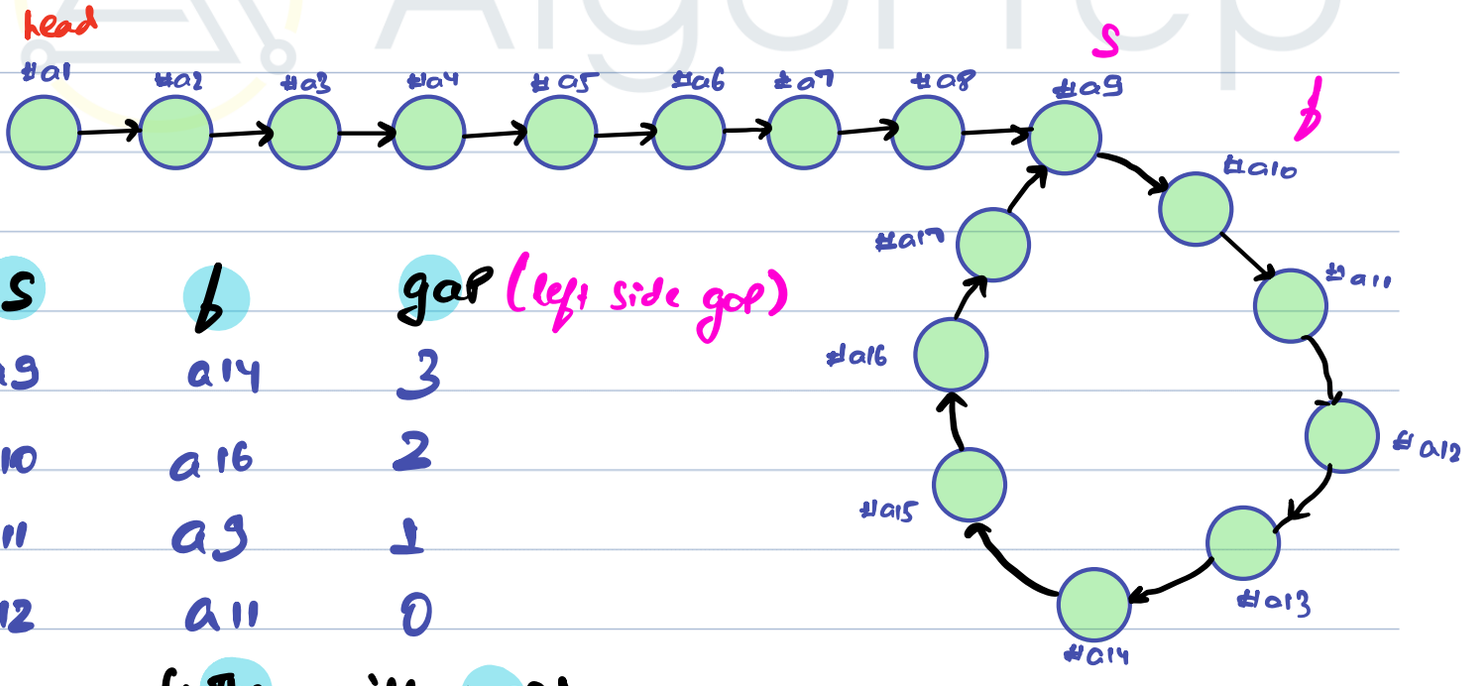
↳ space not allowed.

Ex2:

Chain: 4 Cycle: 9



Idea 2



s	f	gap (left side gap)
a9	a14	3
a10	a16	2
a11	a9	1
a12	a11	0

⇒ They will meet.

obs1: if there is a cycle, s and f will definetly meet as gap is being reduced by 1.

obs2: slow can never complete the cycle.

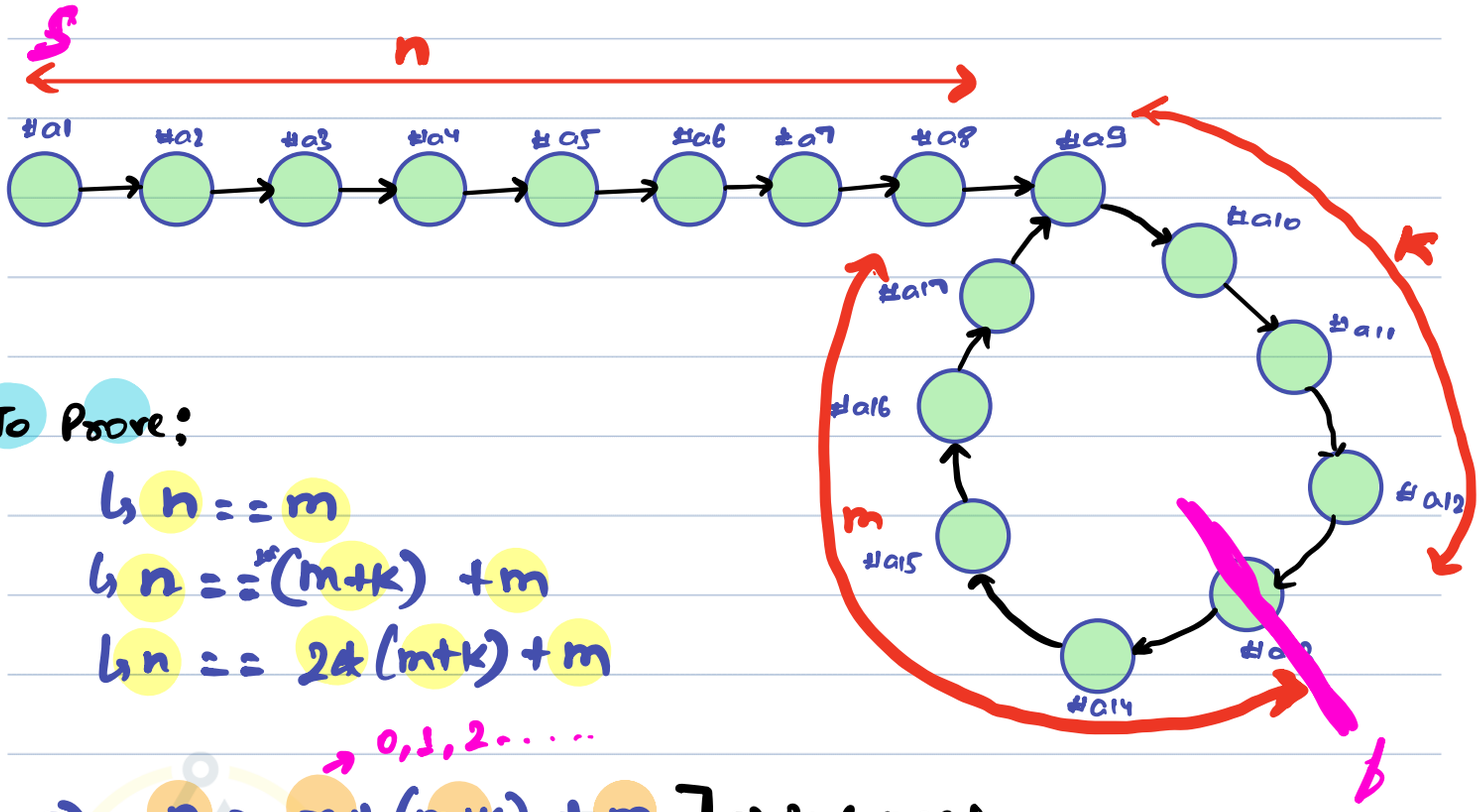
//Pseudo Code

```
Node RemoveCycle (Node head) {

    Node s = head;
    Node f = head;

    while ( condn ) {
        s = s.next;
        f = f.next.next;
    }

    if ( f == null ) { //No cycle
        return null;
    }

    s = head;

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

    return s;
}
```

T.C: O(N)

S.C: O(1)

why?



To Proove:

$\hookrightarrow n == m$

$\hookrightarrow n == x(m+k) + m$

$\hookrightarrow n == 2k(m+k) + m$

$\Rightarrow n = x(m+k) + m$ ($x = 0, 1, 2, \ldots$) → 2nd meet

$n = c'(k+m) + m$ → 1st meet

Proof:

$slow = n+k$

$fast = n + c(k+m) + k$ (with c)

$2 * slow = fast$

$2(n+k) = n + c(k+m) + k$

$2(n+k) - (n+k) = c(k+m)$

$n+k = c(k+m)$

$n+k = (c-1)(k+m) + (k+m)$

$n = (c-1)(k+m) + m$

$n = c'(k+m) + m$

bonus → why mid

binary search

reject 1 part

$N \to N/2 \to N/4 \cdots 1$

→ $1 * \log_2 N$ better than $2 * \log_3 N$

↓ better

ternary search

reject 2 part

$N \to N/3 \to N/3 \cdots 1$

$2 * \log_3 N$

↓ worse

$a^n = a^{n/2} * a^{n/2}$

better than

$a^n : a^{n/3} * a^{n/3} * a^{n/3}$