



Today's agenda

↳ Find mid

↳ Floyd cycle

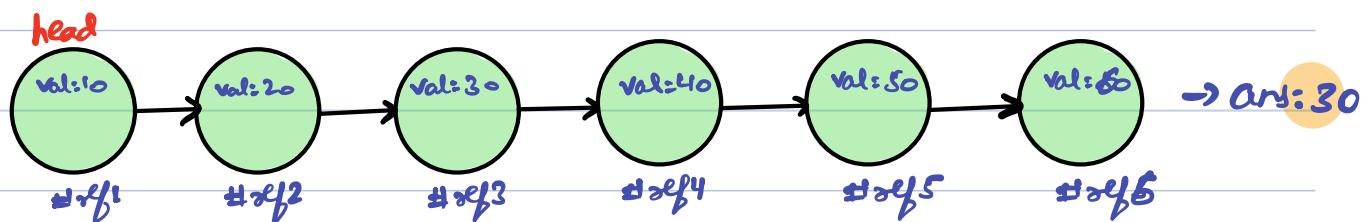
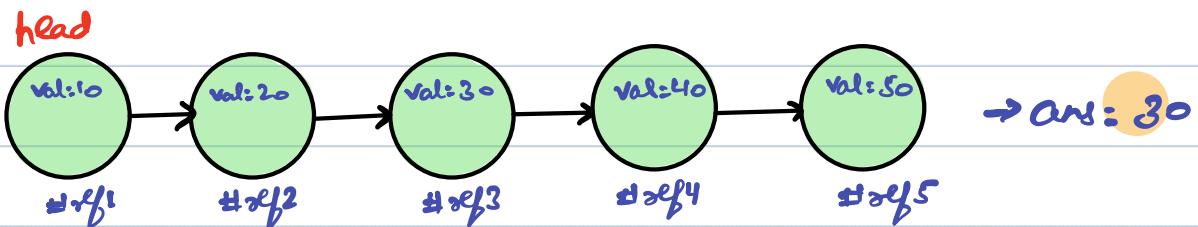
↳ why  $n/2$  in binary search



AlgoPrep



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



Idea1

→ iterate and find out length of LL. Iterate again and return  $\frac{\text{length}}{2}$  element.

T.C:  $O(n + n/2) \approx O(n)$  S.C:  $O(1)$

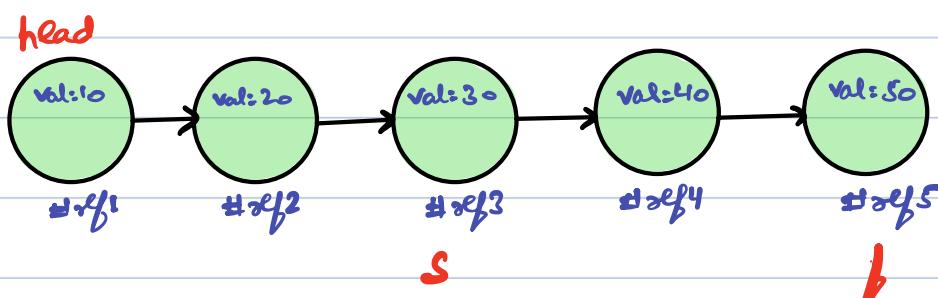
Idea2 → Single iteration

Pritam  $\rightarrow \frac{x}{2} \text{ km}$   
↳ 5 km/hr

Shubh  $\rightarrow x \text{ km}$   
 $\frac{1}{20} \text{ km/hr}$

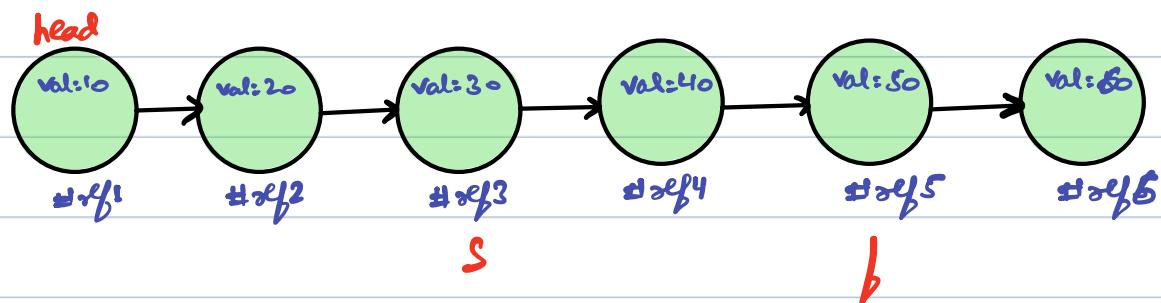


odd



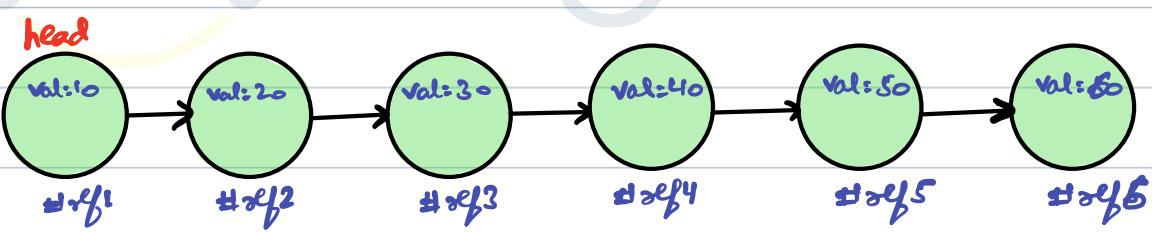
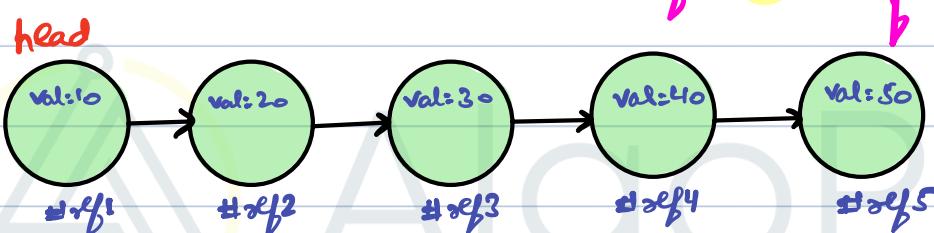
f.next == null  $\rightarrow$  exit the loop  $\rightarrow$  Ans: S.val

even



$f.\text{next}.\text{next} == \text{null} \rightarrow \text{exit} \rightarrow \text{ans} = s.\text{val}$

$f.\text{next}.\text{next}.\text{next} == \text{null}$   
 $f.\text{next} == \text{null}$   
**b**



$f.\text{next}.\text{next} == \text{null}$   
 $f == \text{null}$

$(f.\text{next} == \text{null} \quad || \quad f.\text{next}.\text{next} == \text{null}) \rightarrow \text{exit}$

$(b.\text{next} != \text{null} \quad \& \quad f.\text{next}.\text{next} != \text{null}) \rightarrow \text{run}$



## IIIPseudo 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;

}



AlgoPrep

## Tracing

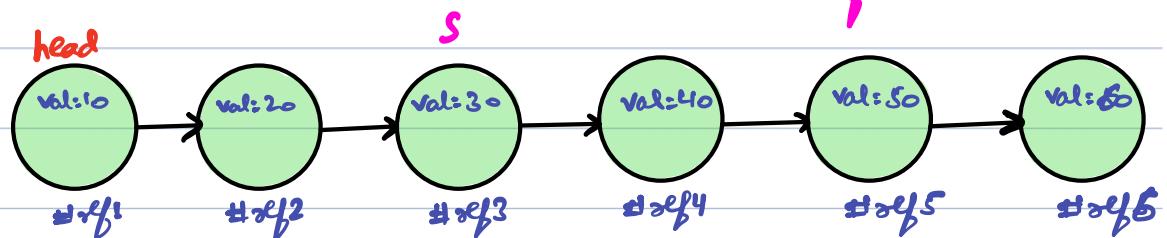


`while (f.next != null && f.next.next != null) {`

`s = s.next;`

`b = b.next.next;`

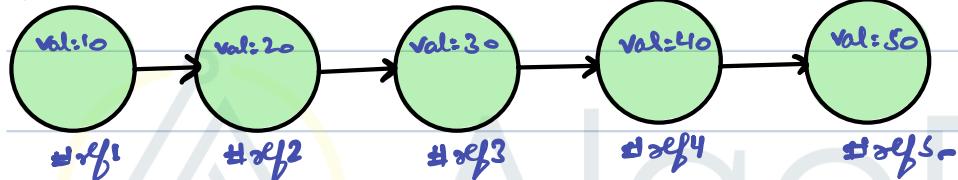
3



`head`

`s`

`b`





## floyd cycle

Q) Given a linked list, Check for cycle & return the Starting Point if exists.

head

#a1

#a2

#a3

#a4

#a5

#a6

#a7

#a8

s b

#a9

#a10

#a11

#a12

#a13

#a14

#a15

#a16

#a17

#a18

#a19

#a20

#a21

#a22

#a23

#a24

#a25

#a26

#a27

#a28

#a29

#a30

#a31

#a32

#a33

#a34

#a35

#a36

#a37

#a38

#a39

#a40

#a41

#a42

#a43

#a44

#a45

#a46

#a47

#a48

#a49

#a50

#a51

#a52

#a53

#a54

#a55

#a56

#a57

#a58

#a59

#a60

#a61

#a62

#a63

#a64

#a65

#a66

#a67

#a68

#a69

#a70

#a71

#a72

#a73

#a74

#a75

#a76

#a77

#a78

#a79

#a80

#a81

#a82

#a83

#a84

#a85

#a86

#a87

#a88

#a89

#a90

#a91

#a92

#a93

#a94

#a95

#a96

#a97

#a98

#a99

#a100

#a101

#a102

#a103

#a104

#a105

#a106

#a107

#a108

#a109

#a110

#a111

#a112

#a113

#a114

#a115

#a116

#a117

#a118

#a119

#a120

#a121

#a122

#a123

#a124

#a125

#a126

#a127

#a128

#a129

#a130

#a131

#a132

#a133

#a134

#a135

#a136

#a137

#a138

#a139

#a140

#a141

#a142

#a143

#a144

#a145

#a146

#a147

#a148

#a149

#a150

#a151

#a152

#a153

#a154

#a155

#a156

#a157

#a158

#a159

#a160

#a161

#a162

#a163

#a164

#a165

#a166

#a167

#a168

#a169

#a170

#a171

#a172

#a173

#a174

#a175

#a176

#a177

#a178

#a179

#a180

#a181

#a182

#a183

#a184

#a185

#a186

#a187

#a188

#a189

#a190

#a191

#a192

#a193

#a194

#a195

#a196

#a197

#a198

#a199

#a200

#a201

#a202

#a203

#a204

#a205

#a206

#a207

#a208

#a209

#a210

#a211

#a212

#a213

#a214

#a215

#a216

#a217

#a218

#a219

#a220

#a221

#a222

#a223

#a224

#a225

#a226

#a227

#a228

#a229

#a230

#a231

#a232

#a233

#a234

#a235

#a236

#a237

#a238

#a239

#a240

#a241

#a242

#a243

#a244

#a245

#a246

#a247

#a248

#a249

#a250

#a251

#a252

#a253

#a254

#a255

#a256

#a257

#a258

#a259

#a260

#a261

#a262

#a263

#a264

#a265

#a266

#a267

#a268

#a269

#a270

#a271

#a272

#a273

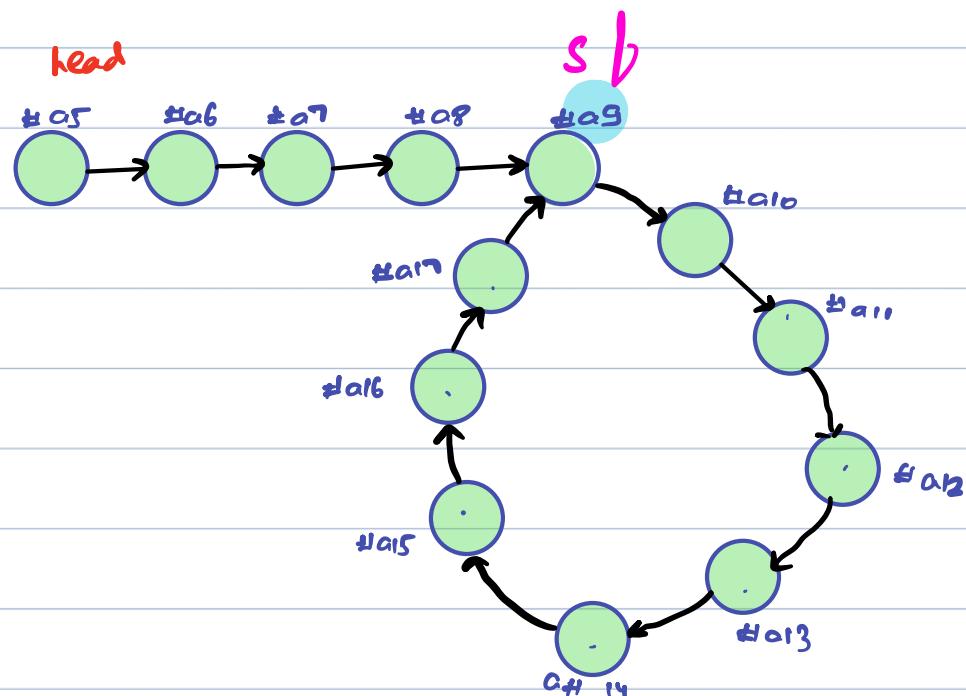
#a2



Idea 2:

Chain = 4

Cycle = 9



Idea 2

head

#01

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

S

b

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 b will definitely meet as gap is being reduced by 1.

**Obs2:** slow can never complete the cycle.



## 11 Pseudo Code

```
node RemoveCycle (node head) {
```

```
    node s = head;
```

```
    node f = head;
```

```
    while ( cond ) {
```

```
        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;
```

```
}
```

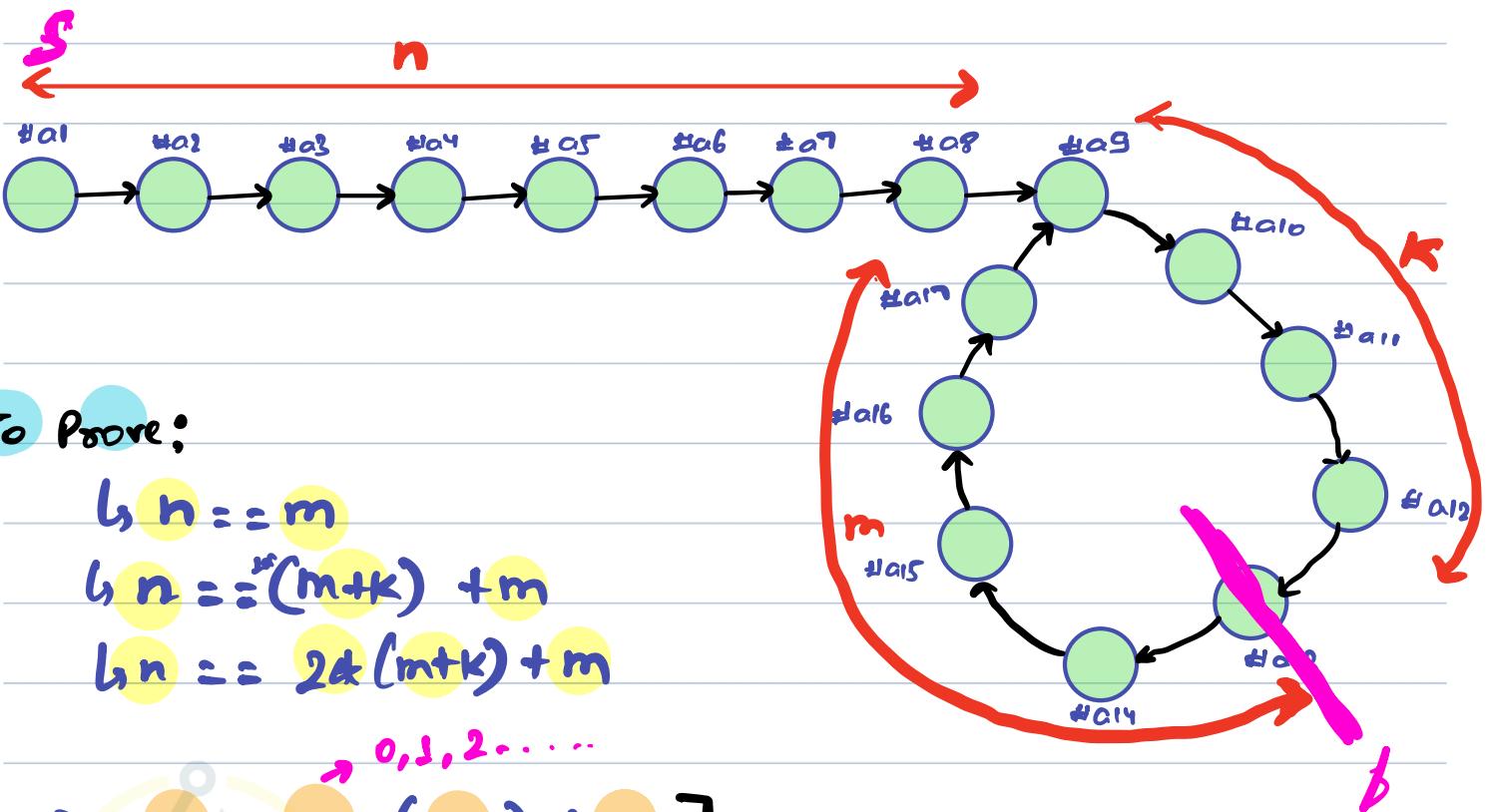
```
return s;
```

```
}
```

T.C:  $O(n)$

S.C:  $O(1)$

Why?



Proof:

$$\text{Slow} = n+k$$

$$\text{Fast} = n + c * (k+m) + k$$

$$2 * \text{Slow} = \text{Fast}$$

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

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

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

$$n+k = (c-1) * (k+m) + \cancel{(k+m)}$$

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

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



bonus

→ why mid

binary search



↳ rejects 1 Part

$$n \rightarrow n/2 \rightarrow n/4 \dots 1$$

→

$$1k \log_2 n$$

↑  
better

better than

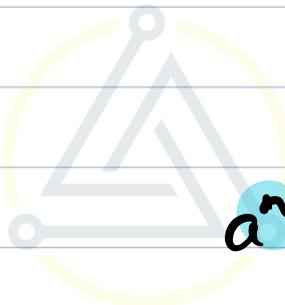


↳ rejects 2 Part

$$n \rightarrow n/3 \rightarrow n/9 \dots 1$$

$$2k \log_3 n$$

↓  
worse



$$a^n = a^{n/2} \times a^{n/2}$$

better than

$$a^n = a^{n/3} \times a^{n/3} \times a^{n/3}$$