i/p → $\frac{29}{7}$ → 4 ...
↓
ans

$\frac{28}{2}$ → (14)
↓
ans

divisor ) dividend ( Quotient

Remainder

✓ Quotient * Divisor + Remainder = Dividend

⇓

Quotient * Divisor <= Dividend

$\frac{29}{7}$ → [0 ⟷ 29]

search space → [0 ⟶ dividend)

0 ⟵——————⟶ 29

s = 0, e = 29, mid = $\frac{0+29}{2}$ = 14

is 14 a possible answer?

Q x Divisor ⇒ 14 x 7 = (98) == 29 ✗

98 < 29 ✗

(98 > 29) → left me jao

i/p → 2 no. → Divide → Binary search

(a, b)

$\frac{a}{b}$ → Quotient

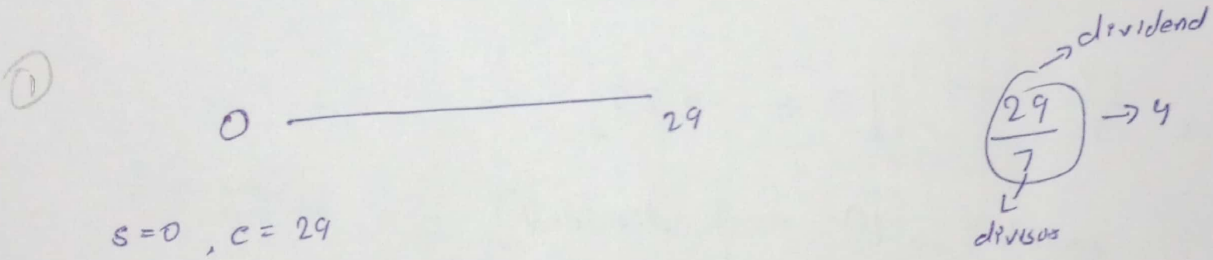
$\frac{28}{7} \geq 4$

divisor ) Dividend ( Quotient

Remainder

Quotient * divisor + remainder = Dividend

Quotient * divisor <= Dividend




s = 0 , e = 29

$mid = \frac{0+29}{2} \Rightarrow 14$

is this a possible answer

Q x divisor <= Dividend

14 x 7 => 98 >= 29

→ ~~false~~

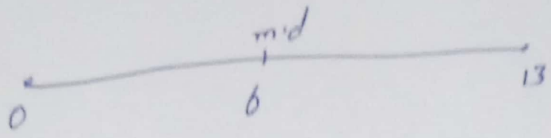
98 >= 29

→ Left → e = mid - 1

e = 14 - 1

e = 13

(2)




s = 0 , e = 13

mid = 6

Q x divis

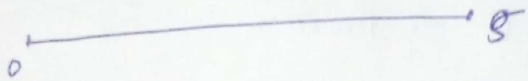
6 x 7 29

42 > 29

↳ left → e = mid - 1

e = 6 - 1 → (5)

(3)




s = 0 , e = 5

mid = 2

Q x divisor <= dividend

2 x 7 <= 29

14 <= 29 → valid ans

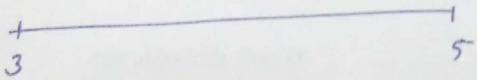
↳ ans store

↳ right me jao

s = mid + 1

s = (3)

(4)




s = 3 , mid = 4

e = 5

Q x divisor <= dividend

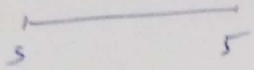
4 x 7 <= 29

28 <= 29 → valid ans

↳ ans store

↳ right me jao

(5)




s = 5

e = 5

mid = 5

mid x divisor <= dividend

5 x 7 <= 29

35 <= 29 → false

↳ left

Conditions

mid x divisor == divident

↳ mid → finalAns

mid x divisor < divident

↳ ans store

↳ right

mid x divisor > divident

↳ left

T.C => O(Log n)  n → divident

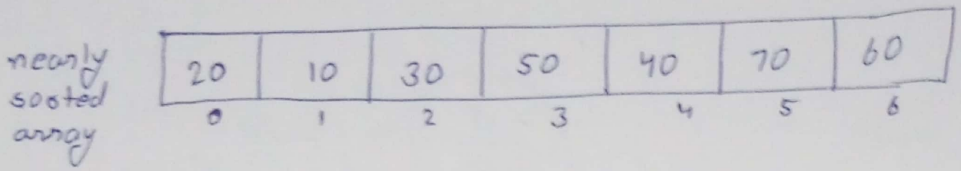
$\frac{+ve}{+ve} \rightarrow +ve$

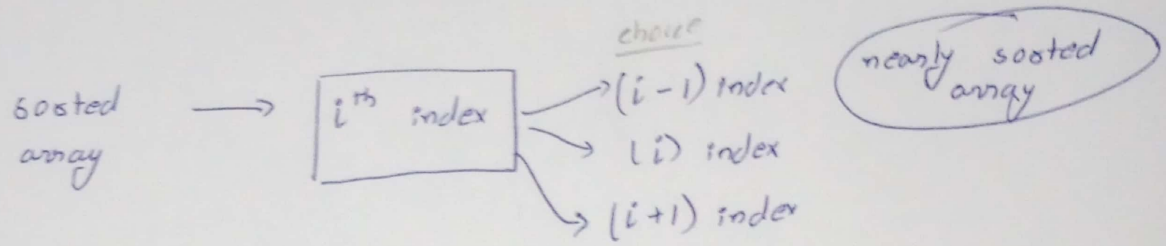
$\frac{-ve}{+ve} \rightarrow -ve$

$\frac{-ve}{-ve} \rightarrow +ve$

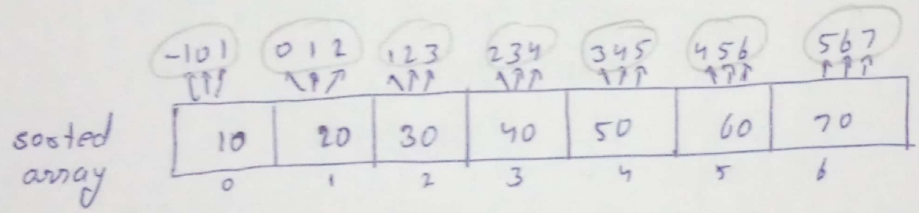
$\frac{+ve}{-ve} \rightarrow -ve$

# Binary search on nearly sorted array




S.A colony (10) NSA 9/10/11



## normal sorted array

→ if (arr[mid] == target)
    return mid

→ if (target > arr[mid])
    ↳ right

- else
    ↳ Left

## nearly sorted array

→ if (arr[mid-1] == target)
    return (mid-1)

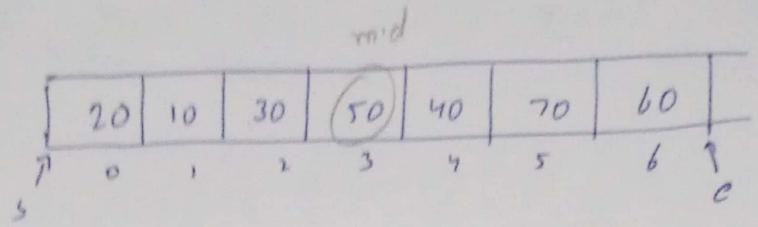
if (arr[mid] == target)
    return mid

if (arr[mid+1] == target)
    return (mid+1)

→ if (target > arr[mid]
    ↳ right

Dry run

(1)




s = 0 , e = 6

mid = 3

arr[mid-1] -> 30 == 70 ✗

arr[mid] -> 50 == 70 ✗

arr[mid+1] -> 40 == 70 ✗

if (target > arr[mid])

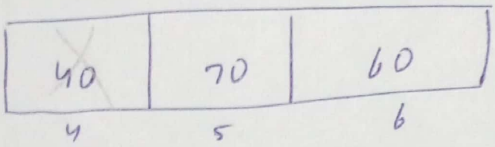
70 > 50

↳ right -> s = mid+1 ~~+1~~

catch

s = mid + 2

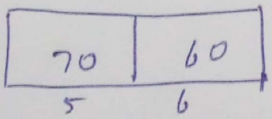
(2)




s = 4 , e = 6

mid = 5

arr[mid-1] =>

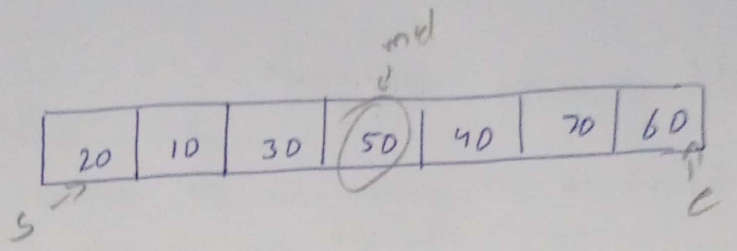



s = 5, e = 6

arr[mid-1]

70 == 70

↳ True




target = 20

s = 0 , e = 6

mid = 3

arr[mid-1] → 30 == 20 ✗

arr[mid] → 50 == 20 ✗

arr[mid+1] → 40 == 20 ✗

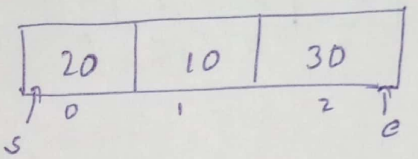
if (target > arr[mid])

20 > 50 → false

↳ Left ⇒ e = mid-1

catch

e = mid-2




s = 0 , e = 2

mid = 1

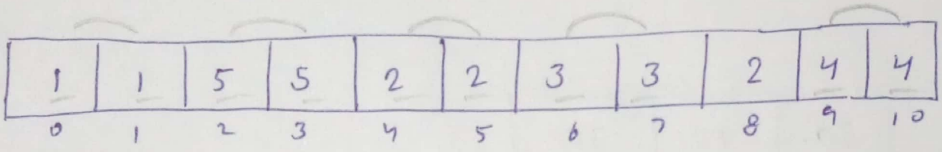
arr[mid-1] → 20 ==

Imp

# Find the odd occuring element

find element that occurs odd times

- → all elements → even no. of times occur except one → odd
- → all repeating no. → pair repeat & pair are not repeated
- → ek baar me koi bhi no. 2 se jada baar nhi aa skta




## Approach

(A) XOR → O(n)

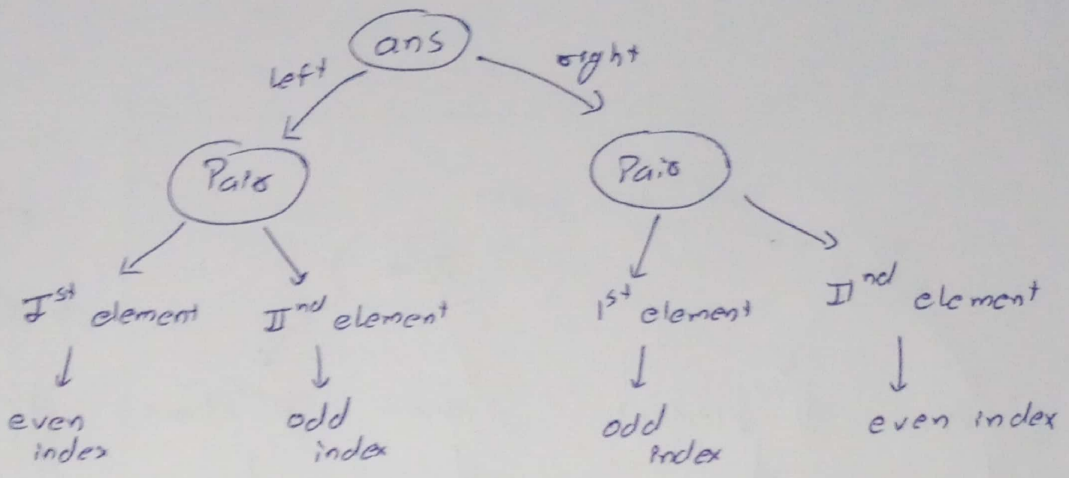
(B) Count → map

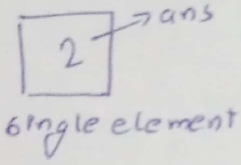
(C) sorting → 3 times

**Binary search**

- → classical
- → Search space
- → Predicate function
- → Index ↓ Logic

## Observation




Guarantee h

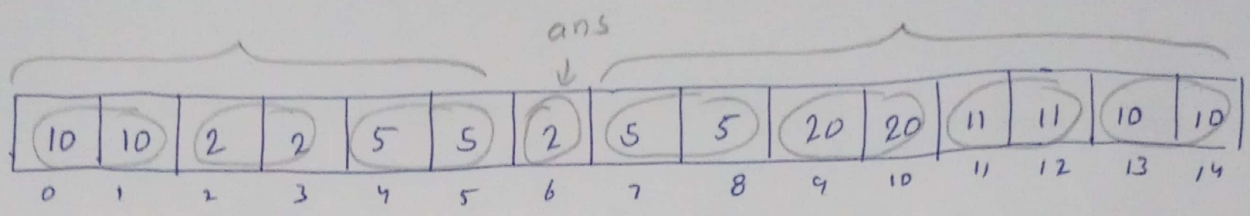
↓

ans

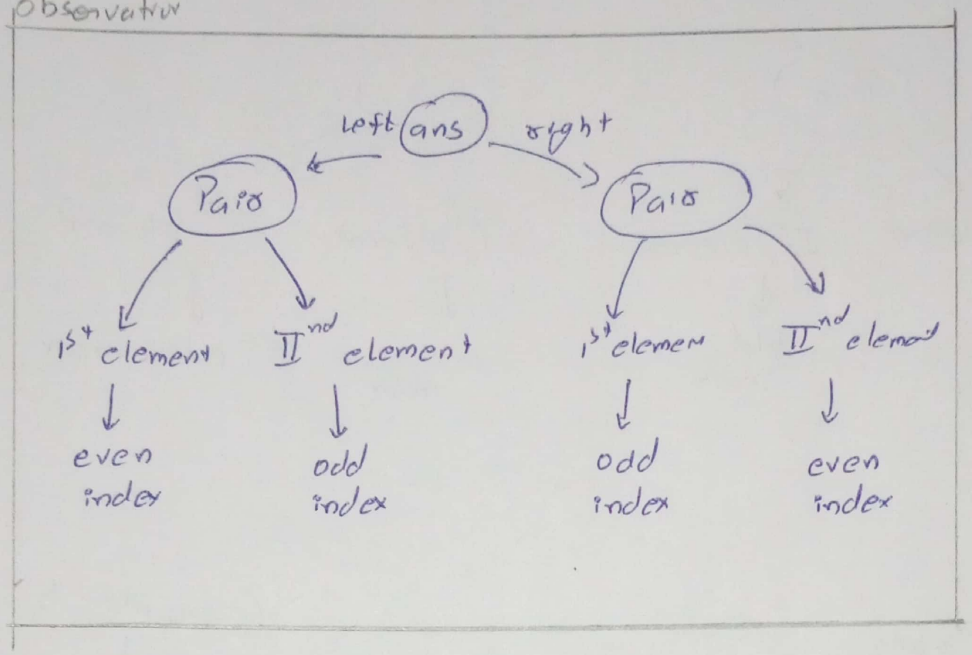
↓

even Index

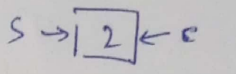
if (mid & 1) → true → odd number



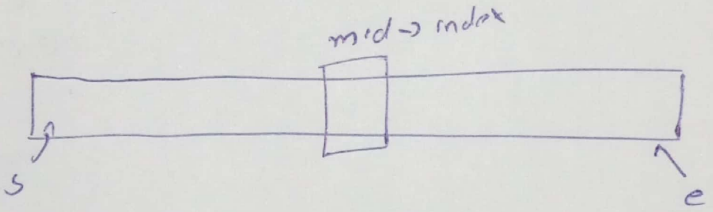

observation




single elem




if (s == e)
return s




if (mid % 2 == 0) ⇒ even

↳ if (arr[mid] == arr[mid+1]

↳ right → s = mid + 2

else

↳ right / ans → catch

e = mid → catch

why

Peax in array

if (mid % 2 == 1)

ans right me hai

↳ if (arr[mid] == arr[mid-1])

↳ right → s = mid+1

else

↳ Left → e = mid-1

Pseudocode

```
int s=0, e=n-1, mid = s+(e-s)/2

while (s <= e) {

    if (s == e)
        return s;

    if (mid ∧ 1)
        ↳ if (arr[mid] == arr[mid-1] → s = mid+1
          else e = mid-1

    else
        ↳ if (arr[mid] == arr[mid+1]) → s = mid+2

          else → e = mid
```