DSA Journey ??

↳ Assignments & Add.

↓ Expert Mock Interview.

X → DSA, SQL, LLD

R2 DSA C2 ⇒ 24th Dec 11:59PM

1.5 hrs

L ⇒ Practice Mode

Searching

What to search for (Target) + Where to search for (Search Space)

Scenario 1

Capitalise ⇒ NewsPapers (Un Sorted)

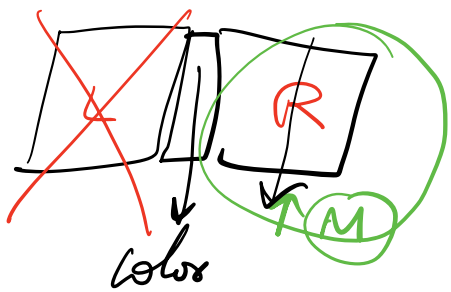
(Target) (Search space)

(Sorted)
Dictionary
(Search Space)

Sorted Data ⇒ Binary Search.

Materialise ⇒ Dictionary

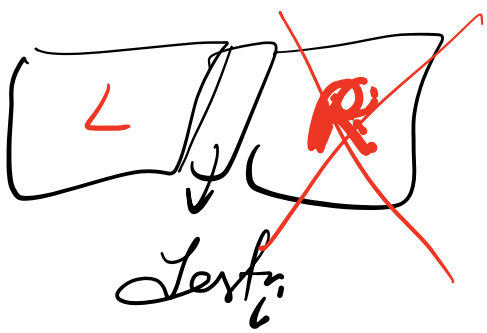
1) Open a random page ⇒ Color.



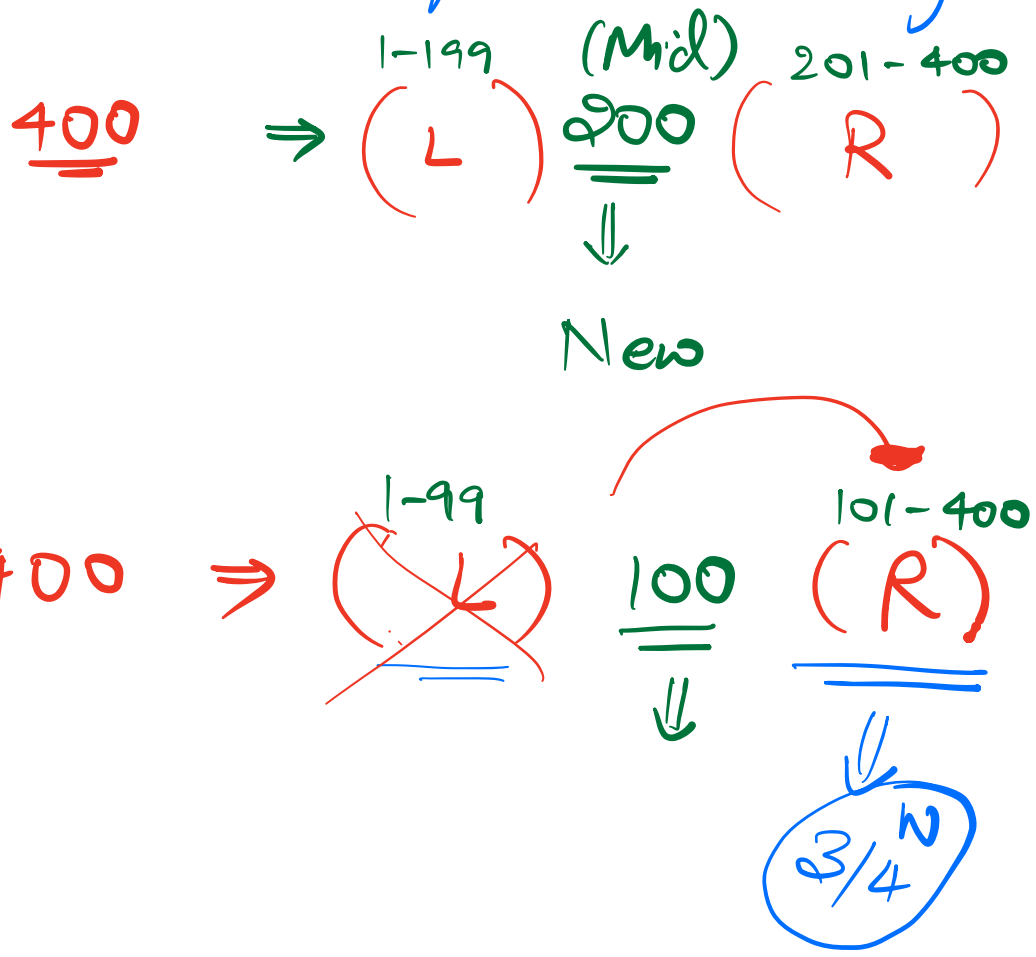
M > C

⇒ Sorted lexographically

Open a random page in R ⇒ Tasty



N pages ⇒ WC ⇒ Max size of search space we can reject.



Given a sorted array with distinct elements.
Search if a given element K is present or not. (Return true/false)

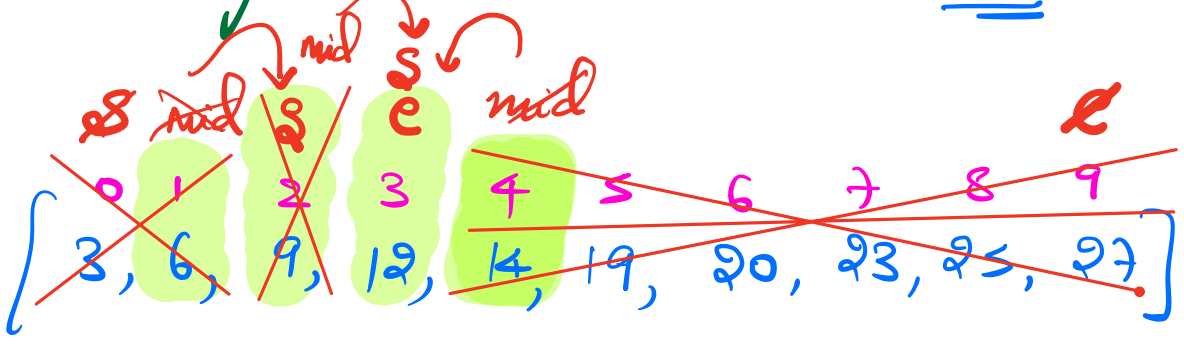
	0	1	2	3	4	5	6	7	8	9
A =	3,	6,	9,	12,	14,	19,	20,	23,	25,	27

K = 12 ⇒ true
K = 26 ⇒ false

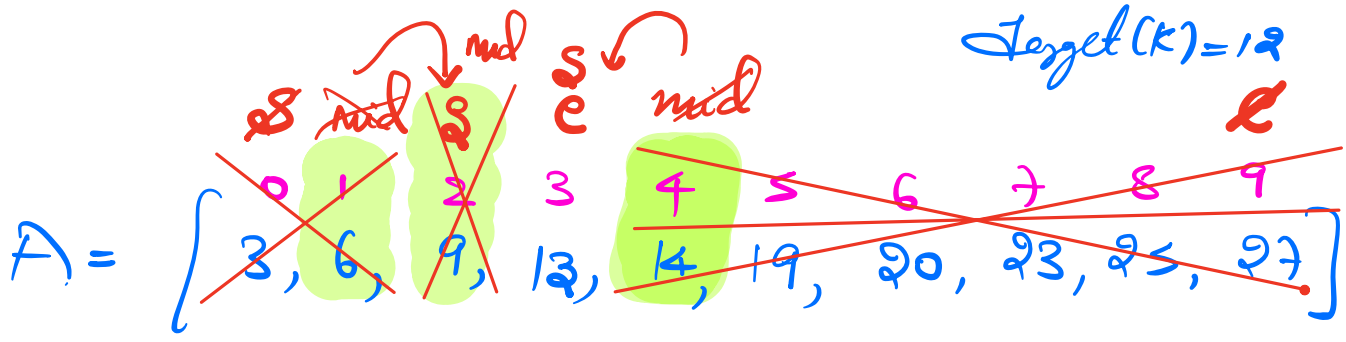
Soln 1) Brute force ⇒ Linear Search
T.C. = O(N)

2) Binary Search

K = 12 (Target)



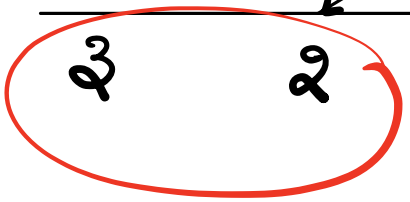
start	end	mid	
0	9	4	⇒ A[mid] > target (14 > 12) Go to left e = mid - 1 (update ss)
0	3	1	A[mid] < target Go to right s = mid + 1
2	3	2	A[mid] < target Go to right s = mid + 1
3	3	3	A[mid] == target return True.



start	end	mid	
0	9	4	⇒ A[mid] > target (14 > 12)

Go to left
e = mid - 1 (update ss)

0	3	1	A[mid] < target Go to right s = mid + 1
2	3	2	A[mid] < target Go to right s = mid + 1
3	3	3	A[mid] > target Go to left e = mid - 1



A[3, 2] ?? Valid ??

NO

Code

```
bool isPresent(A, K) {

    s = 0;
    e = A.length - 1;

    while (s <= e) {
        int mid = s + (e-s)/2;

        if (A[mid] == K) {
            return true;
```

```
        }
        else if (A[mid] > target) {
            e = mid - 1;
        }
        else {
            s = mid + 1;
        }
    }
    return false;
}
```

$T.C. = O(\log_2 N)$

$S.C. = O(1)$

$N \rightarrow N/2 \rightarrow N/4 \rightarrow N/8 \cdots\cdots 0$

Total no. of steps

Given a sorted array of N elements.

Given a target, find the index of first occurance of the target.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
A =	-5,	-5,	-3,	0,	0,	1,	1,	5,	5,	5,	5,	5,	5,	5,	8,	10,	10,	15,	15

→

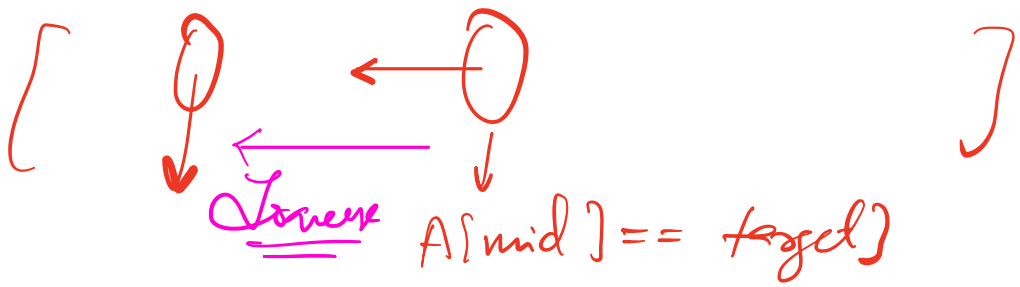
$K = 5 \Rightarrow 7$

$K = 6 \Rightarrow -1$ (Not present)

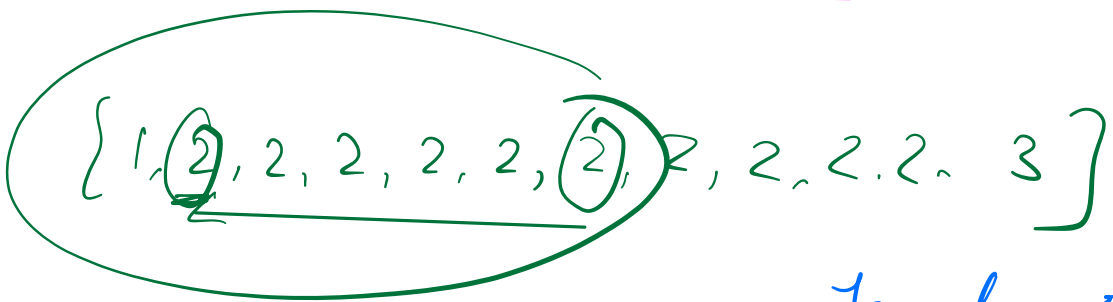
Sol^n 1) Brute force $\Rightarrow$ Linear Search

T.C. = O(N)

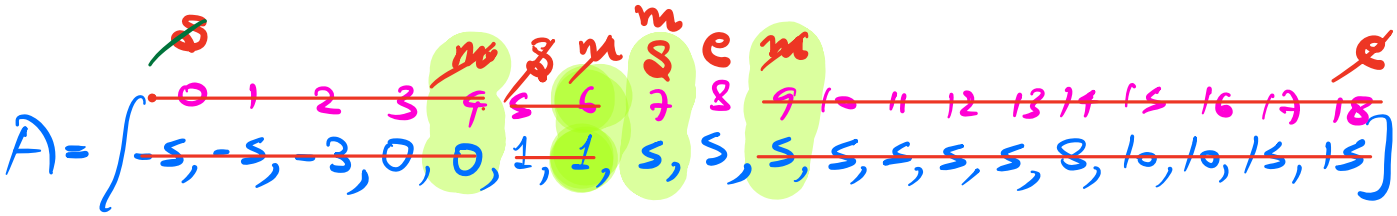
2) Binary Search



Worst T.C. = O(N)



Target = 5



start end mid

0	18	9	A[mid] == Target A[mid] == A[mid-1] (Not the 1st Occ.) Go to L (e = mid - 1)
0	8	4	A[mid] < target, Go to R s = mid + 1
5	8	6	A[mid] < target, Go to R s = mid + 1
7	8	7	A[mid] == target

A[mid] != A[mid-1]

↳ first Occurrence of mid

return 7;

Code

```
int firstOccurence (A, K) {

    s = 0;
    e = A.length - 1;

    while ( s <= e) {
        int mid = s + (e-s)/2;
```

```
if ( A[mid] == K) {
    if ( mid == 0 || A[mid-1] != A[mid] ){
        return mid;
    }
    else {
        e = mid - 1;   (G to L)
    }
}
else if (A[mid] > target) {
    e = mid - 1;
}
else {
    s = mid + 1;
}
}
return -1;
}
```

$$T.C. = O(\log_2 N)$$

$$S.C. = O(1)$$

Can we use a similar approach to find last occurence of an element in a sorted array ??

```
int lastOccurence (A, K) {

    s = 0;
    e = A.length - 1;

    while ( s <= e) {
        int mid =  s + (e-s)/2;

        if ( A[mid] == K) {
            if ( mid == N-1 || A[mid+1] != A[mid] ){
                return mid;
            }
            else {
                s = mid + 1;      (Go to R)
            }
        }
        else if (A[mid] > target) {
            e = mid - 1;
        }
        else {
            s = mid + 1;
        }
    }
    return -1;
}
```

Q) Given an integer array of size N where every element occurs twice except one which occurs only once. find the unique element.

NOTE: Duplicate elements are placed adjacent to each other.

$$A = [\underset{0}{8}, \underset{1}{8}, \underset{2}{5}, \underset{3}{5}, \underset{4}{6}, \underset{5}{9}, \underset{6}{9}]$$

m

Ans = 6

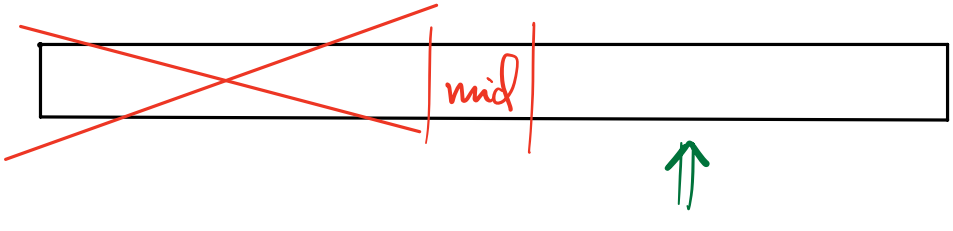
Sol^n 1) XOR of all elements

T.C. = O(N)

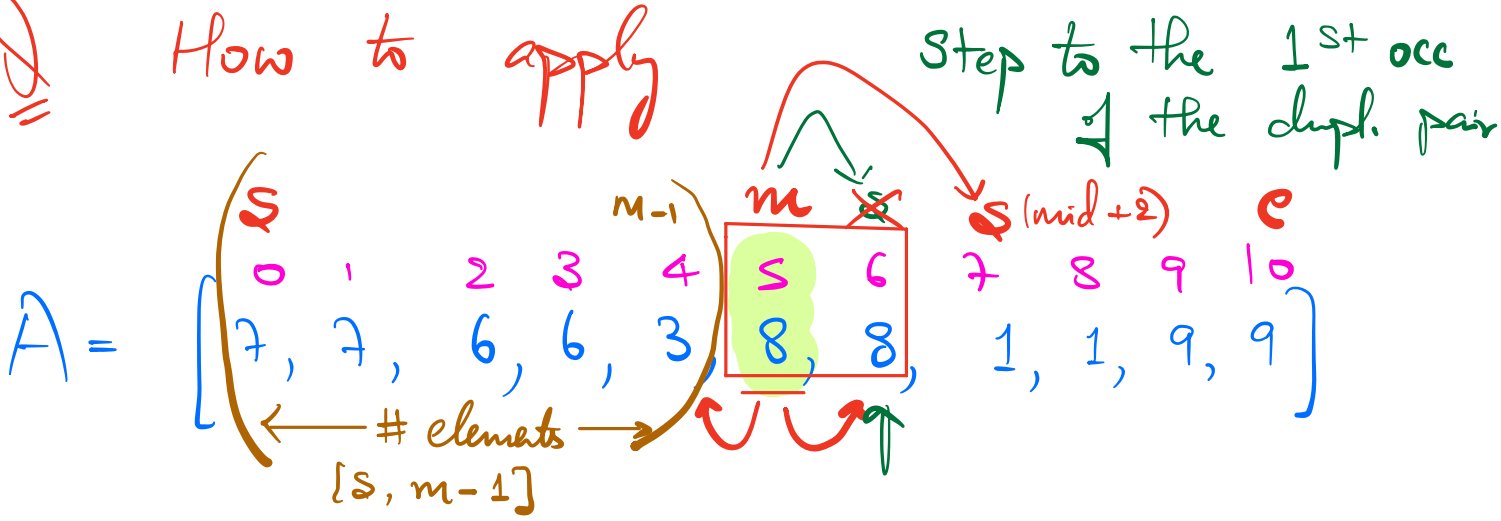
2) Optimise ??

⇓

Binary Search



How to apply



Odd ⇒ Left
Even ⇒ Right

---

s	e	mid
0	10	5

Code

```
int findUnique(A){

    s = 0;
    e = A.length - 1;

    while (s <= e){

        int mid = s + (e-s)/2;

                                    Left
        if ( (mid==0 || A[mid] != A[mid-1]) &&
             (mid== N-1 || A[mid] != A[mid+1]) ){
                                    Right
```

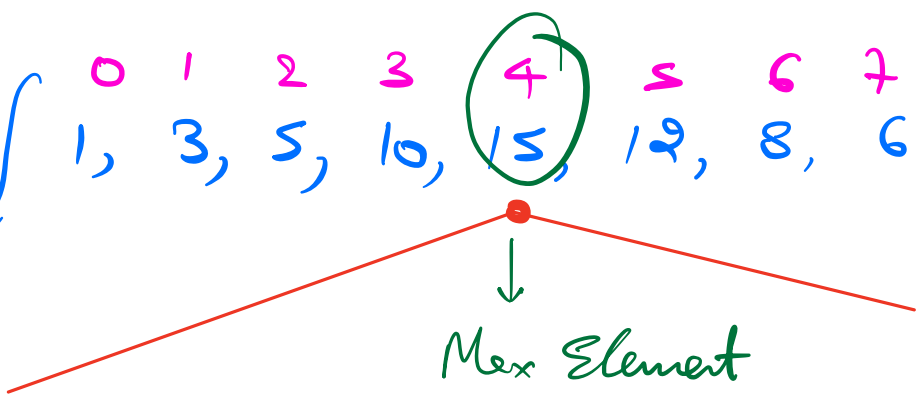
```
        return mid;
    }
    if (mid==0 || A[mid-1] == A[mid]) {
        mid = mid - 1; // Go to 1st occ.
    }

    // [s, mid-1] ⇒ (mid-1)-s+1 = (mid-s)
    if ((mid-s)%2 == 0) { // Even ⇒ R
        s = mid+2;
    }
    else {
        e = mid - 1;
    }
}
```

$T.C. = O(\log n)$

Given an increasing decreasing array find the max ele.

$A = [1, 3, 5, 10, 15, 12, 8, 6]$ (indices 0 1 2 3 4 5 6 7)



(mid==0 || A[mid] > A[mid-1]) &&
(mid==N-1 || A[mid] > A[mid+1])