SCALER

Searching - 1

TABLE OF CONTENTS

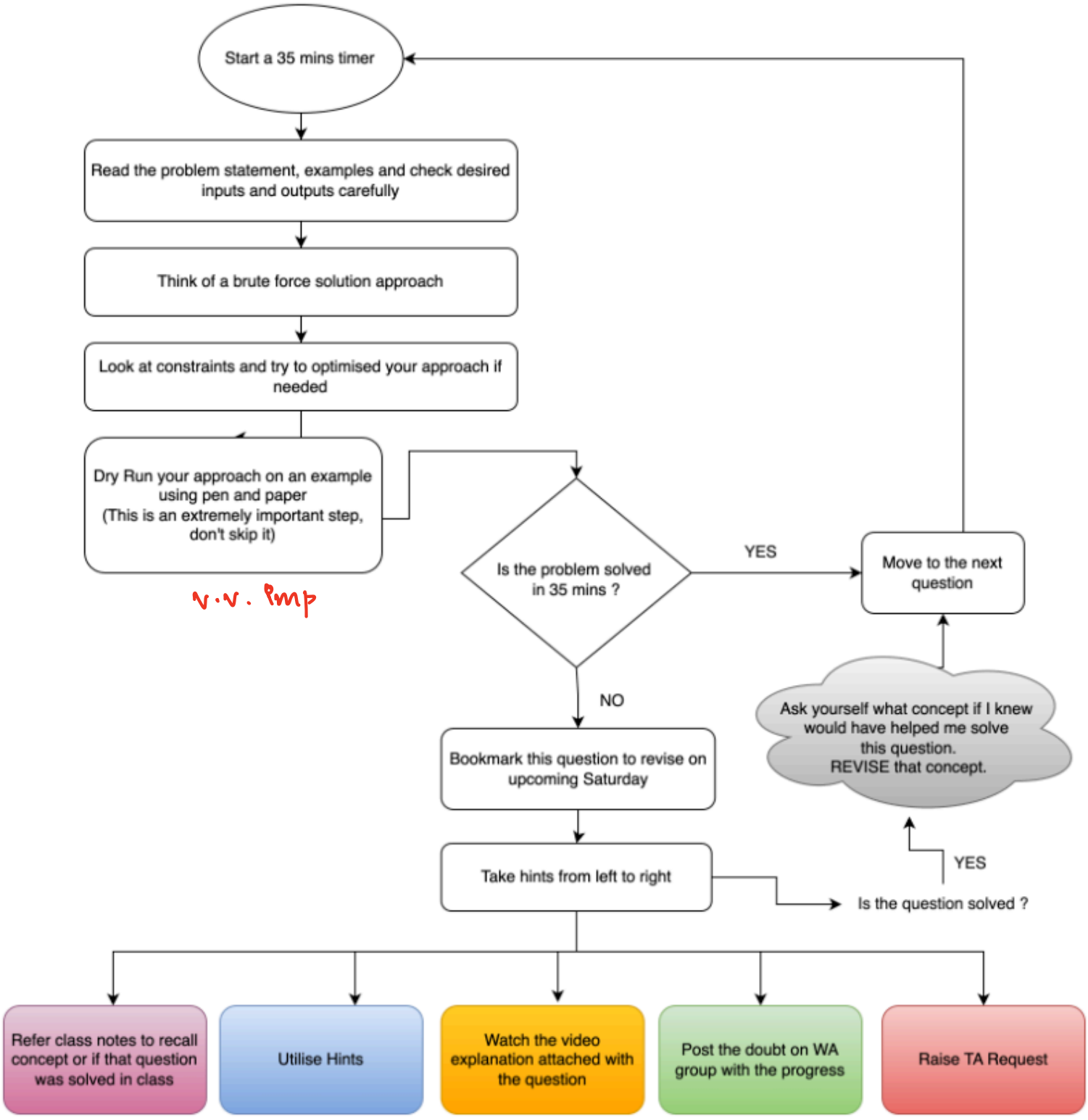
PSP - 68.6 %

Notes

Challenges are what makes life interesting. Overcoming them is what makes life meaningful.

Can we apply B.S only on sorted array? No ! :)

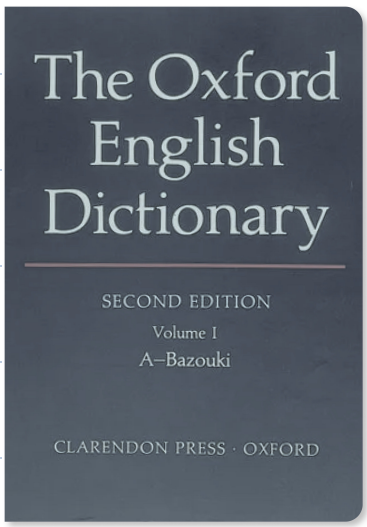
How to solve a problem ?



Searching

Searching a word in newspaper

Searching a word in dictionary



Binary Search

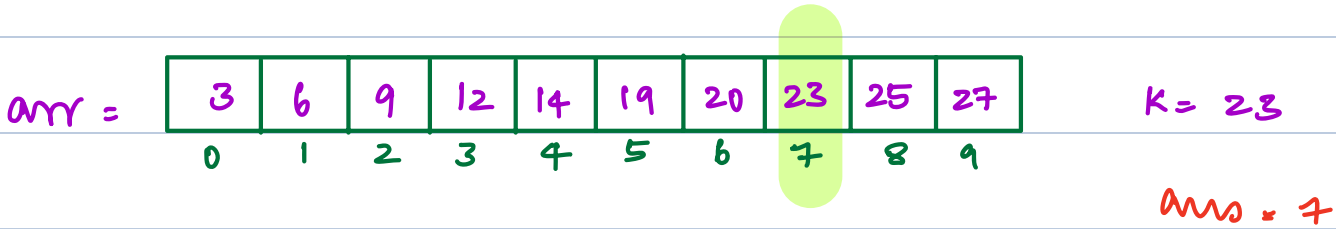
Search space : The area to know the result exist and we search there only.

Target : The Item we are looking for.

Condition : Condition to discard some part of search space repeatedly

→ we discard one half of the array

Given a sorted arr[ ]. Search an element K. If K is present return it's index otherwise return -1.



BF Approach

```
Int findK (Int arr[], Int k) {
    for (Int i=0; i< arr.length; i++) {
        if (arr[i] == k) return i;
    }
    return -1;
}
```

T.C = O(N) S.C = O(1)

Idea -2 Binary Search

Search Space: The entire array

Target : Input K

Condition :

arr =

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

R, L

K = 21

L	R	MID	Remarks
0	9	(0+9)/2 = 4	14 < 21 (go right) l = mid +1
5	9	(5+9)/2 = 7	23 > 21 (go left) r = mid -1
5	6	(5+6)/2 = 5	19 < 21 (go right)
6	6	(6+6)/2 = 6	20 < 21 (go right)
7	6		stop

l = mid+1

r = mid -1

pseudo code

```
int binary_search (int [] A, int B) {
    int L=0, R= A.length-1;
    while (L <= R) {
        int mid = (L+R)/2;   →   L + (R-L)/2
        if (A[mid] == B) return mid;
        else if (A[mid] < B) L=mid+1;
        else R = mid - 1;
    }
    return -1;
}
```

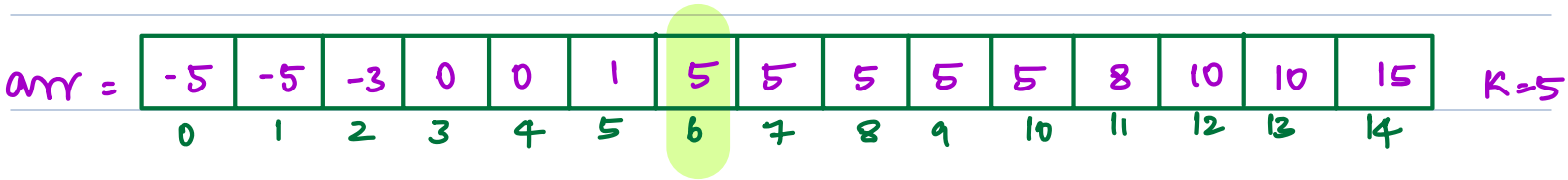
T.C = O(log N)

S.C = O(1)

Story Based Question

All emails in your mailbox are sorted chronologically can you find the first mail that you received in 2024?

Given a sorted arr[N]. Find first occurrence of K.

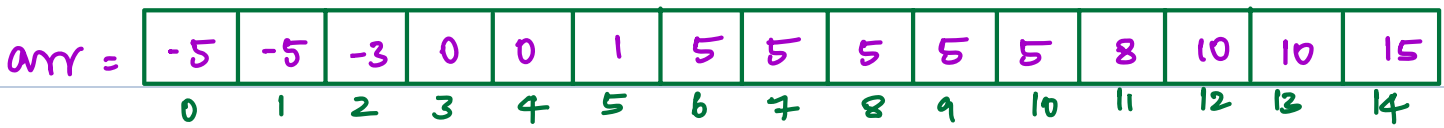


BF Approach

Linear Search

T.C = O(N) S.C = O(1)

Idea -2



ans = ~~1~~ ≠ 6

L	R	MID	Remarks
0	14	7	a[7] == 5 (go left)
0	6	3	a[3] < 5 (go right)
4	6	5	a[5] < 5 (go right)
6	6	6	a[6] == 5 (go left)
6	5		Stop

pseudo code

```
int binary_search (int [] A, int B) {
    int L=0, R= A.length-1, ans=-1;
    while (L <= R) {
        int mid = (L+R)/2;
        if (A[mid] == B) { ans = mid; R= mid-1}
        else if (A[mid] < B) L=mid+1;
        else R = mid-1;
    }
    return ans;
}
```

T.C = O(logn)

S.C = O(1)

Quiz 3

When searching for first occurrence in sorted array

if (arr[mid] == K) ⟶ store ans

proceed left

Every element occurs twice except for 1. Find that unique element.

Note : Duplicates are adjacent to each other and array isn't necessarily sorted.

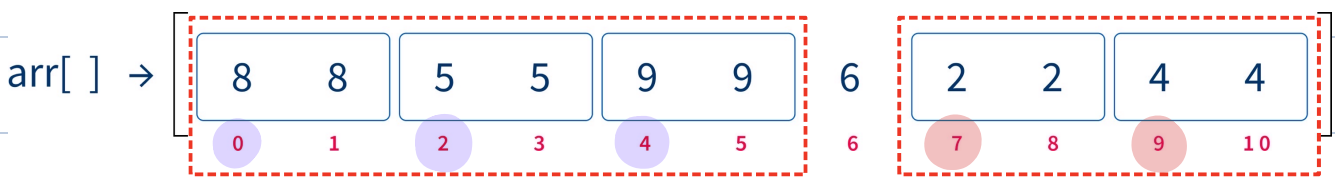
Idea -1

XOR to get unique element

T.C : O(n) S.C: O(1)

9:55 → 10:05

Idea -2



Search Space : The entire array

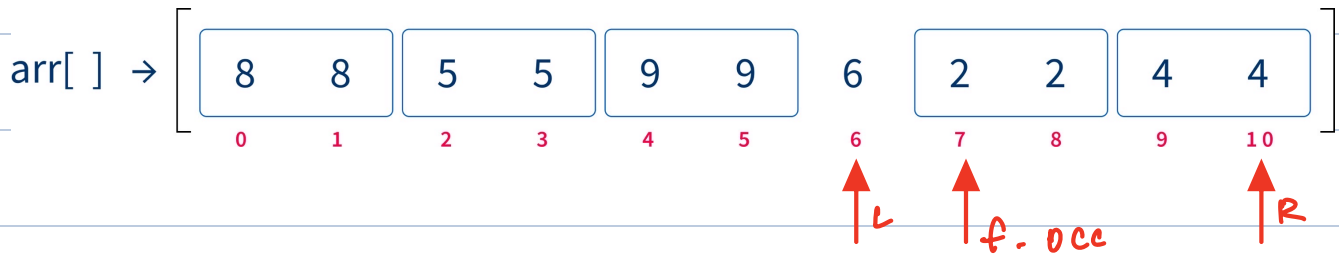
Target : unique element

Condition :

First occurrence Index

odd (move left) even (move right)

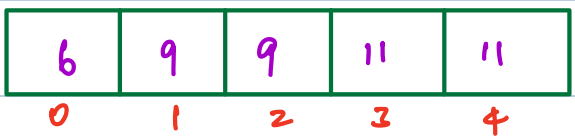
Here the concept is that the first_occ and my unique element will be on even index. When the first_occ began starting with odd index, my unique element would be on the left



L	R	MID	1st Occ.	Remark
0	10	5	4	4 % 2 == 0 f_occ + 2
6	10	8	7	7 % 2 == 1 f_occ - 1
6	6	6	———	a[6] ✓

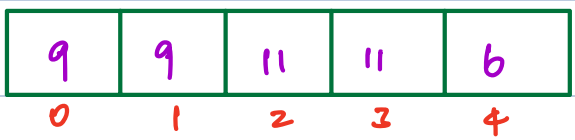
Edge Cases

ans at 0th index



mid == 0 || a[mid-1] != a[mid]

ans at n-1th index



mid == n-1 || a[mid+1] != a[mid]

combine to get unique element

Here the concept is that the first_occ and my unique element will be on even index. When the first_occ began starting with odd index, my unique element would be on the left

pseudo code

```
Int uniqueElement (Int [] A) {
    Int L=0; R= A.length-1; N = A.length;
    while (L <= R) {                          // Handle edge conditions
        Int mid = (L+R)/2;
        // correct ans conditions
        if ( (mid ==0 || (a[mid-1] != a[mid]))
            && (mid == N-1 || a[mid+1] != a[mid]))
                return a[mid];
        // Get the f_occ
        Int f_occ = mid;
        if (A[mid-1] == A[mid]) {
            f_occ = mid-1;
        }
        // Basic conditions
        if (f_occ %2 ==0) { L= f_occ+2; }
        else { R = f_occ-1; }
    }
}
```

T.C = O(logn) S.C = O(1)

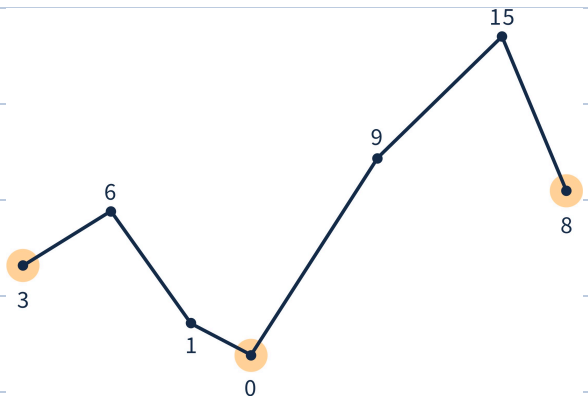
Given arr[N]. Find any one local minima.

Local Minima : An element which is less than or equals to it's adjacent elements.

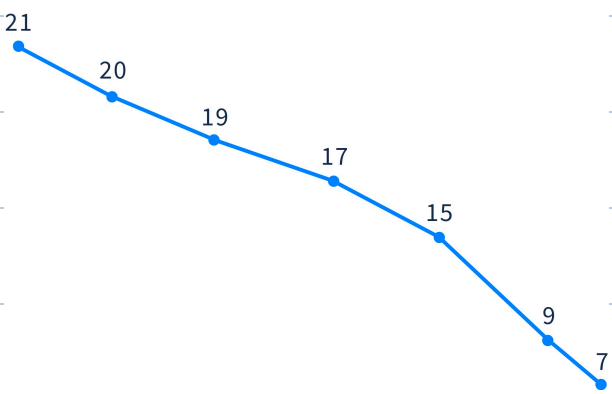
examples

arr = | 3 | 6 | 1 | 0 | 9 | 15 | 8 |
0 1 2 3 4 5 6

→ return {3, 0, 8}

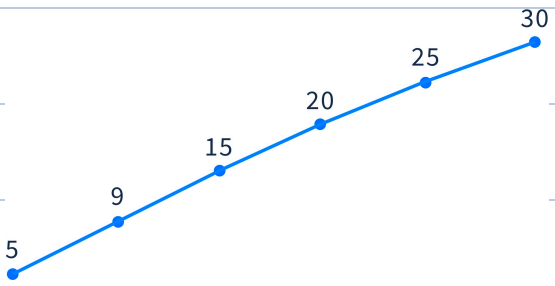


arr = | 21 | 20 | 19 | 17 | 15 | 9 | 7 |
0 1 2 3 4 5 6



arr =

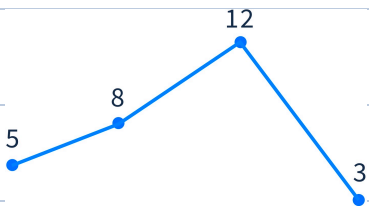
5	9	15	20	25	30
0	1	2	3	4	5



arr =

5	8	12	3
0	1	2	3

any one of {5, 3}



BF Idea

Linear Search → T.C = O(n) S.C = O(1)

Idea

Search Space: The entire array

Target: local minima

Condition: If (a[mid] <= a[mid-1] && a[mid] <= a[mid+1])

// True condition

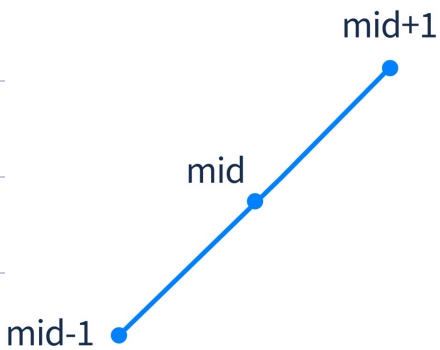
a[mid] <= a[mid-1] &&

a[mid] <= a[mid+1]

// go Right

a[mid] <= a[mid-1] &&

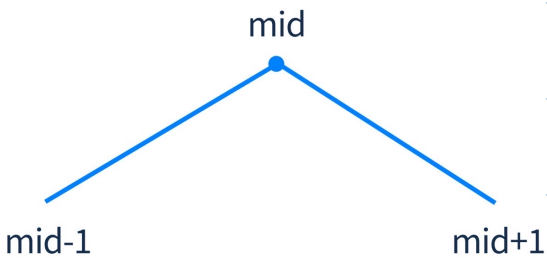
a[mid] > a[mid+1]



// go left

a[mid] > a[mid-1] &&

a[mid] <= a[mid+1]



Go either way

</ > Code

```
int localMinima(int[] A) {
    // only one element present
    int n = A.length;
    if (n == 1) return A[0];
    // local minima at 0th index
    if (A[0] <= A[1]) return A[0];
    // local minima at n-1th index
    if (A[n-1] <= A[n-2]) return A[n-1];
    int l = 0; r = n-1;
    while (l <= r) {
        int mid = (l+r)/2;
        if (A[mid] <= A[mid-1] &&
            A[mid] <= A[mid+1]) {
            return A[mid];
        }
        else if (A[mid] > A[mid+1]) l = mid+1;
        else r = mid-1;
    }
}
```

T.C = O(logn) S.C = O(1)