Binary Search

17 August 2024 16:06

Binary Search

→ This is done on Sorted Arrays.
→ There will be a sorted array given & then we need to perform Binary Search.

arr = [2, 4, 9, 10, 12, 14, 18, 19] → ascending

arr = [19, 18, 14, 12, 10, 8, -3, -16] → descending

} Sorted Arrays

Q: arr = [0:2, 1:4, 2:6, 3:9, 4:11, 5:12, 6:14, 7:20, 8:36, 9:48]

Target = 36

Binary Search Algorithm:

1. Find the middle element.
2. Check if target > middle ⇒ Search in the right side of the array

   else ⇒ Search in the left side of the array
3. If target == middle ⇒ Answer Found (10 (1):

Working:

s = 0, e = 9

arr = [0:2, 1:4, 2:6, 3:9, 4:11, 5:12, 6:14, 7:20, 8:36, 9:48]

target = 36

1: mid = $\frac{0+9}{2}$ = 4 (Integer value)

(middle element is at the 4th index)

(middle element is at the 4th index)

2: Check target > middle ; is 36 > 11 ?

YES!!

∴ arr's start & end will be updated like:

s (index 5), e (index 9)

arr = [2, 4, 6, 9, 11, 12, 14, 20, 36, 48]

(indices: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9; index 7 (20) circled)

1: $mid = \frac{5+9}{2} = 7$

2: is 36 > 20 ? **YES!!**

s (index 8), e (index 9)

arr = [2, 4, 6, 9, 11, 12, 14, 20, 36, 48]

(indices: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9; index 8 (36) circled)

1: $mid = \frac{8+9}{2} = 8$

2: Here, target == middle

∴ 36 = 36 ∴ **Element Found !!**

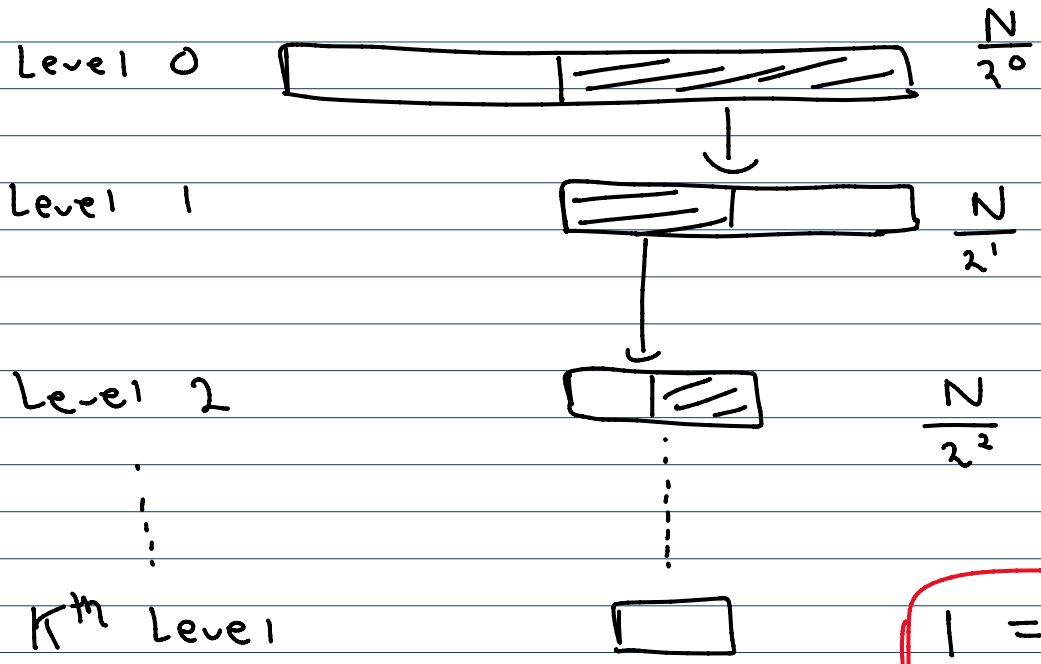
at index '8'

if 's' > 'e' : Element not found in the array.

Time Complexity

Best Case: $O(1)$ [element which we are searching is in the middle only]

Worst Case: $O(\log N)$



$$N = 2^K$$

$$\log N = \log(2^K)$$

$$\log N = K \log(2)$$

$$K = \frac{\log N}{\log 2} \Rightarrow \boxed{K = \log_2 N}$$

Total Comparisions in Worst Case → $\log N$

Suppose, Searching 1,000,000 elements (Considering worst case in both)

na a h

1,000,000 comparisions	$\log_2 1{,}000{,}000 \simeq 20$

Better way to find "mid"

$mid = \frac{start + end}{2}$ -> This may exceed the range

$\therefore mid = start + \frac{(end - start)}{2}$

$= \frac{2s + e - s}{2}$

$= \boxed{\frac{s+e}{2}}$

∴ Formula remains the same but only the way of writing is different. Thus, in the 2nd way, the range won't exceed.

Order Agnostic Binary Search

-> This approach is used when we don't know if the array is sorted in ascending order or descending order

	0	1	2	3 (M)	4	5	6	7 (e)
arr =	90,	75,	18,	12,	6,	4,	3,	1

target = 75

IF target > middle => search in left OR else right

IF start > end => Descending

IF Start > end => Descending

IF end > Start => Ascending

Searching in Matrices

	0	1	2
0	18	9	12
1	36	-4	91
2	44	36	16

Ans = [1, 2]

Target = 91

```
For r=0; r<n, r++
    For c=0; c<n; c++;
        if arr[r][c] = tgt
            Ans
else return -1;
```

$N * N = N^2 = O(N^2)$

$O(N * m)$

IF N rows & m cols

Q. Matrix is sorted in row wise & col wise manner

	0	1	2	3
0	10	20	30	40
1	15	25	35	45
2	28	29	37	49
3	33	34	38	50

Lowest bound

upper bound

Target = 37

Case 1: if element == target

=> ans. is found

Case 2: If element < target

row ++

Case 3: If element > target

col --

For lowest bound, we consider the 1st element of 0th row (Here 10)

For upper bound, we consider the entire last column

For upper bound, we consider the entire last column.

Col --

Explanation for Case 3:

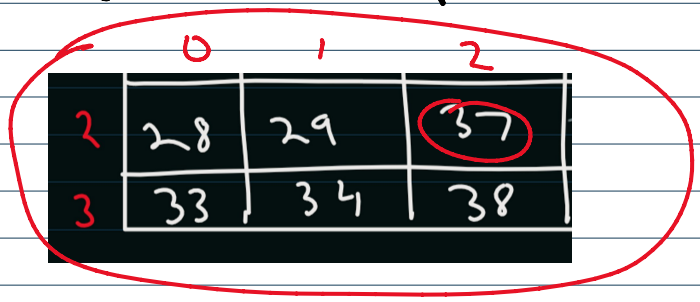
1] For column, we start with last column & take its 1st element (40). We observe that $40 > 37$. ∴ All elements in that column will be > target. (∵ the matrix is sorted). Thus we ignore the column. Thus, we do {Col --}

2] Then we come to the 2nd column & find out that $30 < 37$. That leads us to Case 2

Explanation for Case 2:

1] We start with element 30 (arr[0][2]) bcoz srch space is now reduced. We see the 30 < target (37). Thus all elements in that row will be < target. Hence, we ignore the row. Thus we do {rowtt}

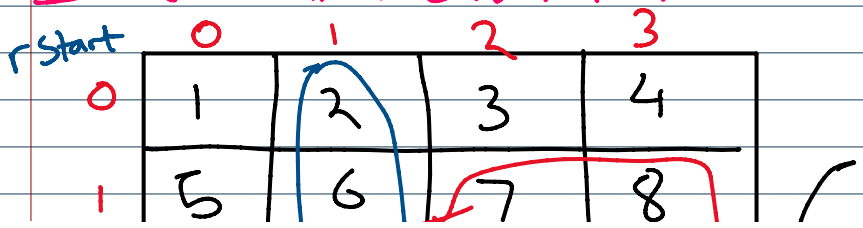
2] We also ignore the 2nd row for the same reason. Thus our srch space now will be:



Now we come in arr[2][2]. Here the element == target

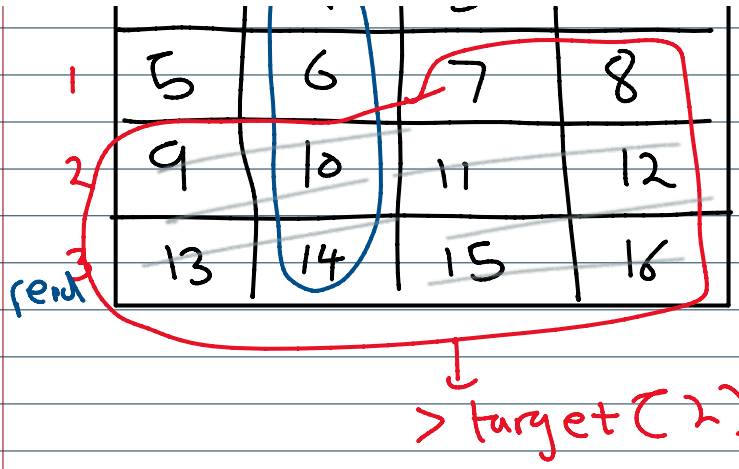
Thus, answer found

Q Search in a Sorted Matrix



Target = 2

Take middle column & perform BS on it



find middle column & perform BS on it

Upper & lower bounds are given (rstart, rend)

middle element = 6.
we see that 6 > 2.
Then obviously all the rows below it will be ignored

① IF element == target

// Ans found

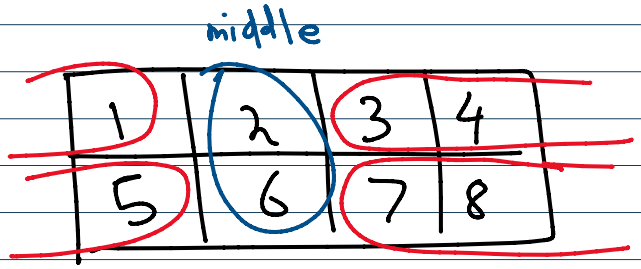
② IF element > target

// Ignore rows after it

③ IF element < target

// Ignore rows above it

In the end, only 2 rows will be left (consider above example)



If target = 3

① Check whether the mid column you are at contains the answer or not

② If not, then consider 4 parts.

③ Srch in all 4 parts till you get answer

Time Complexity: T.C = log(N) (BS har rahe hai)

We considered N rows, & we are taking M columns.

↓

Searching

↓
Searching
across 4 columns

(3) Search in all 4 parts till you get answer

4 log(m)
↳ can be ignored

$$T.C = O(\log(N) + \log(m))$$