Linear Search

Given an array, search for an element K.

arr[] = [2 -1 4 9 8 10]

TC : O(n)   K = "raj"

arr[] = ["yash", "raj", "ayush"]

TC: (n x len of string)   TC : O(n) x [Time taken for comparison]

Search Space & target

i) Search for an article on dhoni in the newspaper

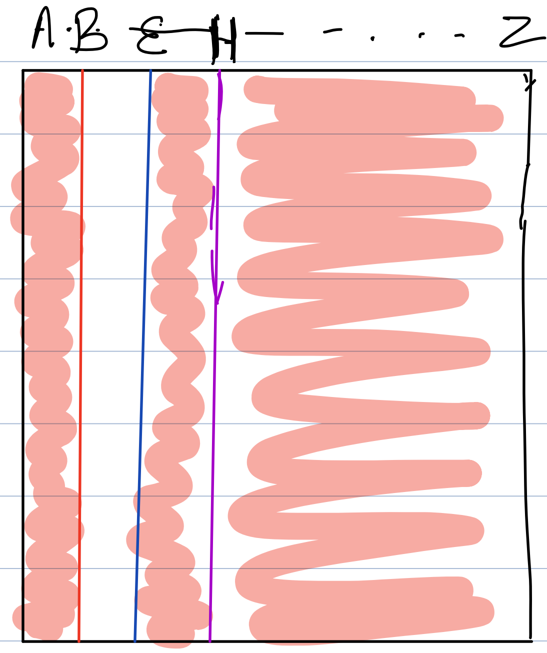
dhoni ↓ target   newspaper ↓ search space.

2) Search for K in an array

K → target   array → search space

Dictionary

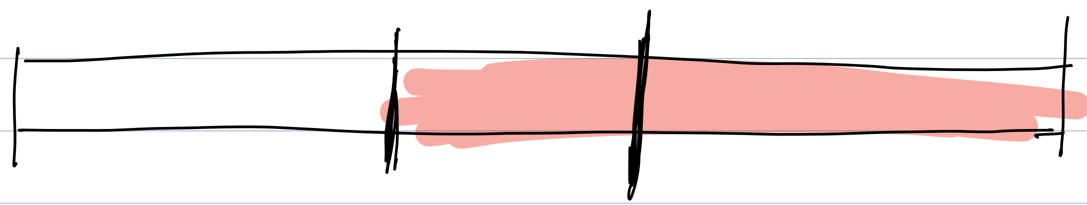
Search for a word "dog"



Why searching is easier in Dictionary ?

40%

→ Data is sorted. !



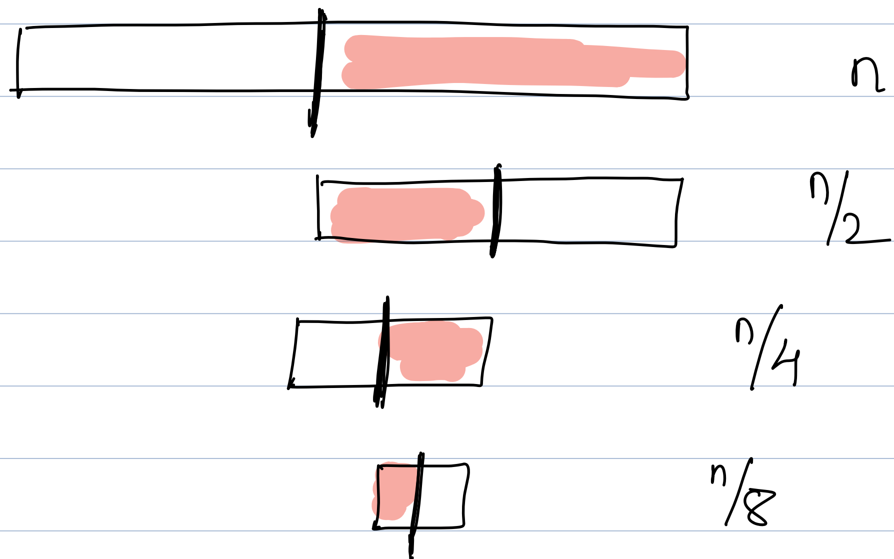
40%

$n \times \frac{4}{10} \times \frac{4}{10} \times \frac{4}{10} \times \frac{4}{10} \Rightarrow$

How to optimally divide Search space.

→ Search At middle to ensure we are reducing search space by half (50%) every time.

Binary Search : Where we reduce the search space by half every iteration using a specified logic



↳ eventually we reach to one element.

Binary Search

=> Divide Search Space by half.

Ex1 Arr = [3 6 9 12 14 19 20 23 25 27]

Q Given a sorted array with distinct elements. Search for the index of element K.

If k is not present return -1.

Ex1

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

Arr = [3 6 9 12 14 19 20 23 25 27]

$K \Rightarrow 9$

$\Rightarrow lo \Rightarrow 0$

Find middle

$\Rightarrow hi \Rightarrow 9$

1) $mid \Rightarrow \left(\frac{lo + hi}{2}\right) = (int + int)$

2) $\boxed{mid \Rightarrow lo + \frac{(hi - lo)}{2}}$

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

K => 9

Steps	lo	hi	mid	element
1	0	9	4	14, hi=3
2	0	3	1	6, lo=>2
3	2	3	2	9

Case 1 : arr [mid] > K

hi => mid - 1

Case 2 : arr [mid] < K

lo => mid + 1

Case 3 : arr [mid] == K

return mid

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

K => 24

Steps	lo	hi	mid	element
1	0	9	4	14
2	5	9	7	23
3	8	9	8	25
	8	7		

```
int binarySearch (int arr[], int k) {

    int lo => 0;                    Tc: O(logn)
    int hi => arr.size() -1;        Sc: O(1);

    while ( lo <= hi) {
        int mid => ((lo+hi)/2);
        if (arr[mid] == k) return mid;
        else if (arr[mid] > k) hi => mid-1;
        else lo => mid+1
    }
    return -1;
}
```

Q_2 Given a sorted array! Find the floor of element K.

floor $\downarrow$ greatest element $\leq$ K.

Ex: Arr =

$$\begin{array}{cccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ \end{array}$$
$$[\,3 \quad 6 \quad 9 \quad 12 \quad 14 \quad 19 \quad 20 \quad 23 \quad 25 \quad 27\,]$$

1) floor (9) = 9
2) floor (10) = 9
3) floor (24) ⇒ 23

4) floor (2) ⇒ Integer.min

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

floor (24) ⇒ 23

Steps	lo	hi	mid	element
1	0	9	4	14
2	5	9	7	23
3	8	9	8	25
	8	7		

Case 1 : arr [mid] > K

hi ⇒ mid - 1

Case 2 : arr [mid] < K , ans ⇒ arr[mid];

lo ⇒ mid + 1

Case 3 : arr [mid] == K

return arr[mid];

Q_3 Given a sorted array ! Find the 1^{st} occurrence of element K.

↳ you need to return index!

Ex1

$$\begin{array}{cccccccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ \text{Arr} = & [-1 & -1 & 0 & 2 & 2 & 5 & 5 & 5 & 7 & 8 & 8] \end{array}$$

$K \Rightarrow 5 \Rightarrow ans = 5$

Case 1 : $arr[mid] > K$

$hi \Rightarrow mid - 1$

Case 2 : $arr[mid] < K$

$lo \Rightarrow mid + 1$

Case 3 : $arr[mid] == K$

$ans \Rightarrow mid$

$hi \Rightarrow mid - 1;$

Variations of the above Questions

1) Find the last occurrence of K.

2) lower_bound (K) => first element such that $\geq K$.

returns index

	0	1	2	3	4
arr =	2	5	10	16	20

1) lower_bound (5) => 1
2) lower_bound (9) => 2
3) lower_bound (-1) => 0
4) lower_bound (50) => 5 = size of array

3) upper_bound (K) => first element such that $> K$.

returns index

	0	1	2	3	4
arr =	2	5	10	16	20

1) upper_bound (5) => 2
2) upper_bound (9) => 2
3) upper_bound (-1) => 0
4) upper_bound (50) => 5 = size of array

Q Given a sorted Array! Find the number of occurence of K.

$$\begin{array}{c|ccccccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ \text{Ex1 Arr} = & [-1 & -1 & 0 & 2 & 2 & 5 & 5 & 5 & 7 & 8 & 8] \end{array}$$

Ex1 K ⇒ 5 ans ⇒ 3

⇒ Liner Search Tc : O(n)

1) Lower_bound (5) ⇒ 5 ⇒ l

2) Upper_bound (5) ⇒ 8 ⇒ u

$\boxed{\text{ans} \Rightarrow u - l}$

$$\begin{array}{c|ccccccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ \text{Ex1 Arr} = & [-1 & -1 & 0 & 2 & 2 & 5 & 5 & 5 & 7 & 8 & 8] \end{array}$$

Ex1 K ⇒ 1

1) lower_bound ⇒ 3
2) upper_bound ⇒ 3

Q Given a 2D sorted matrix. Search for K.

$$A = \begin{bmatrix} 2 & 6 & 8 & 10 \\ 12 & 18 & 20 & 30 \\ 33 & 40 & 42 & 46 \\ 50 & 55 & 60 & 66 \end{bmatrix}$$

rows = n

colum ⇒ m

$\boxed{12 \Rightarrow 40}$

1) Approach 1 : Flatten into an array.
Do binary search
Tc: O(mn + log mn).
SC: O(m × n);

2) Approach2 : Do binary search on every row.
Tc: (n × log m)
SC: O(1)

3) Approach3 : Do binary search on every colum
Tc: (m × log n)
SC: O(1)

4) Approach 4 : Lowerbound on last colum.
the Binary search on the row.
Tc: O(log m + log n)
SC: O(1).

[illegible]