Binary Search

Today's content

→ Searching Basics

Problems

→ Search in sorted Array

→ First occurrence

→ Finding local minima

→ Square root of a no.

Searching story -

Bro/Sis missing → Police

- Identification (What to search) — Target
- Last seen (Where to search) — Search space

Example

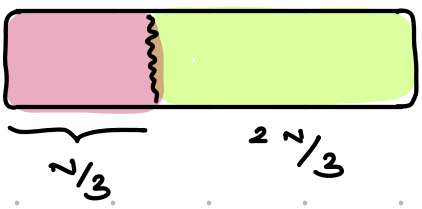
Word → {Book, Newspaper, Dictionary}

Phone no. → {Diary, Phonebook, contact list}

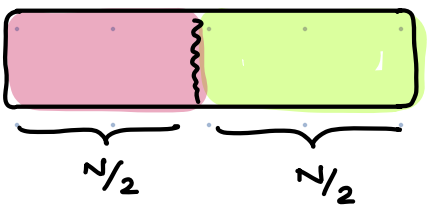
Conclusion → If search space is sorted, then our searching becomes easy.

Dog in Dictionary

Dict = {A B C D E F G . . . . Z}



Worst : N/3 elements are getting discarder : larger search space



Better

Worst : N/2 elements are getting discarded in every step

When to apply Binary Search?

→ After splitting search space in two equal parts, if we are able to discard one half of the search space with some logic then we can apply binary search.

Q Given a sorted array, search if K is present.

A[ ] = {3 6 9 12 14 19 20 23}

K = 12 ⟶ true

K = 15 ⟶ false

Idea 1 → Linear search

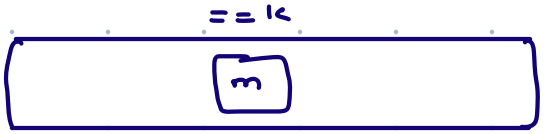
TC : O(N)
SC : O(1)

Idea 2 → Hashing concept

TC : O(N)
SC : O(N)

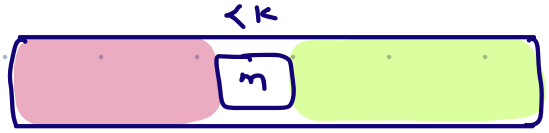
Idea 3 → Binary Search

Case 1



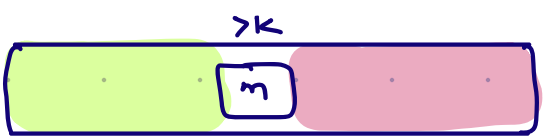
if (A[m] == k) → true

Case 2

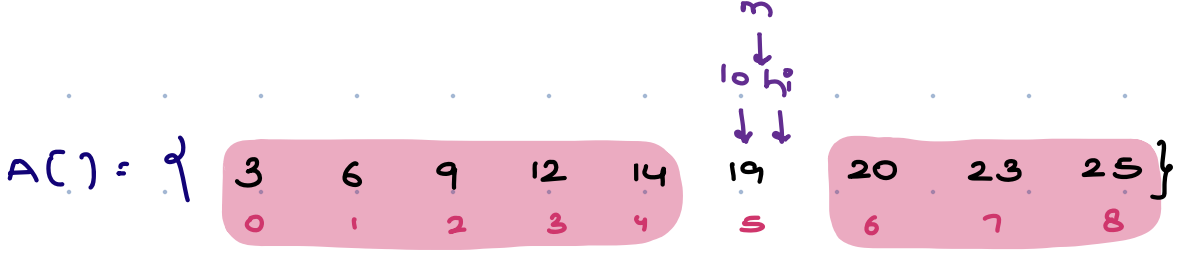


if (A[m] < k) → discard left
(move on RHS)

Case 3

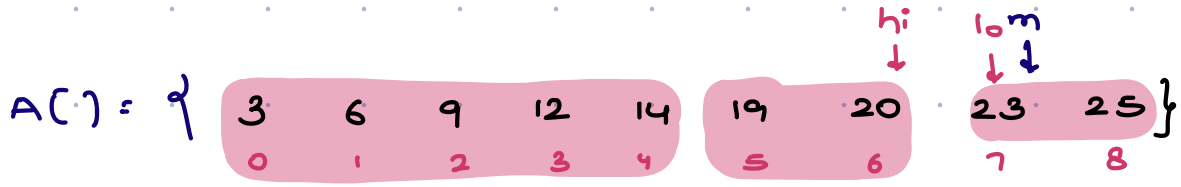


if (A[m] > k) → discard right
(move on LHS)



$$Mid = \frac{lo + hi}{2}$$

lo	hi	mid	compare A[mid] & k
0	8	4	A[4] < 19 : go to RHS lo = m+1
5	8	6	A[6] > 19 : go to LHS : hi = m-1
5	5	5	A[5] == 19 → return true.



lo	hi	mid	compare A[mid] & k
0	8	4	A[4] < 21 : go to RHS lo = m+1
5	8	6	A[6] < 21 : go to RHS lo = m+1
7	8	7	A[7] > 21 : go to LHS hi = m-1
7	6	→	search space exhausted

```
boolean search ( int [] A, int K )
{
    lo = 0
    hi = N-1
    while ( lo ≤ hi )
    {
        m = (lo + hi) / 2
        if ( A[m] == K )  return true
        else if (A[m] < K)  lo = m+1
        else  hi = m-1
    }
    return false;
}
```

TC : O(log N)
SC : O(1)

$N \longrightarrow \frac{N}{2} \longrightarrow \frac{N}{4} \longrightarrow \frac{N}{8} \longrightarrow \frac{N}{16} \cdots 1$ } TC : O(log N)

Q Given a sorted array & we need to find first index of ele

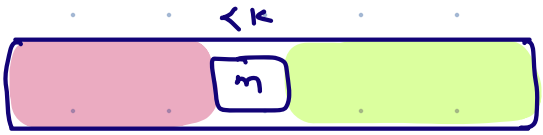
A[] = { 0 0 0 0 2 2 2 3 4 4 }
indices: 0 1 2 3 4 5 6 7 8 9

K = 2 ⟶ 4
K = 0 ⟶ 0

Case 1


```
if ( A[m] == K )
{
    ans = m
    hi = m-1
}
```

Case 2

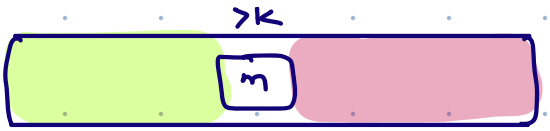


if (A[m] < K) :

go to RHS

lo = m+1

Case 3



if (A[m] > K) :

go to LHS

hi = m-1

A[ ] = { 0 0 0 0 2 2 2 3 4 4 }

0 1 2 3 4 5 6 7 8 9

lo m hi

```
boolean search ( int [] A, int K )
{
    lo = 0          ans = -1
    hi = N-1
    while ( lo ≤ hi )
    {
        m = (lo + hi) / 2
        if ( A[m] == K ) { ans = m ;  hi = m-1 }
        else if (A[m] < K)   lo = m+1
        else    hi = m-1
    }
    return ans;
}
```

TC : O(log N)

SC : O(1)

	0	1	2	3	4	5	6	7	8	9	10
A[ ] = {	0	2	2	2	2	2	2	3	4	4	5 }

k = 2

hi, lo, m

lo	hi	mid	compare A[m] & k
0	10	5	A[5] == 2 ans = 5, hi = m-1
0	4	2	A[2] == 2 ans = 2, hi = m-1
0	1	0	A[0] < 2, lo = m+1
1	1	1	A[1] == 2 . ans = 1, hi = m-1
1	0	⟶	Stop

Ans = 1

Last occurrence of an element.

8:04 → 8:15 AM

Local Minima

Given an unsorted array of distinct elements, return any local minima.

An element is a local minima, if it's less than its adjacent ele.

Local minima ⟶ A[i-1] > A[i] < A[i+1]

A[0] < A[1]

A[N-1] < A[N-2]

A[] = {9 8 7 3 6 4 1 5 2}

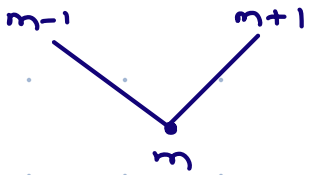
Idea 1 → Iterate & check if A[k] < A[k-1] &

A[k] < A[k+1]

TC : O(N)
SC : O(1)

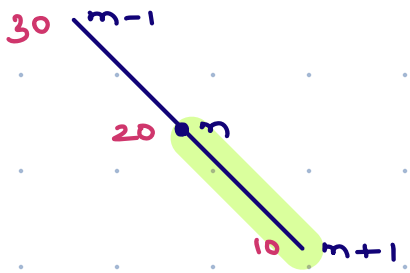
Idea 2 → Binary Search

Case 1



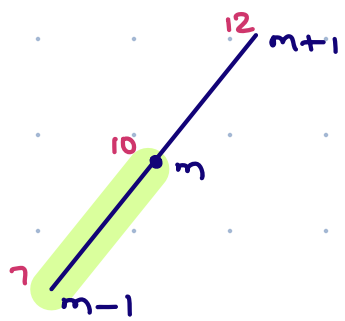
→ return A[m]

Case 2



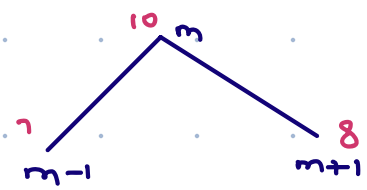
{30 20 10}
↑
m

Case 3



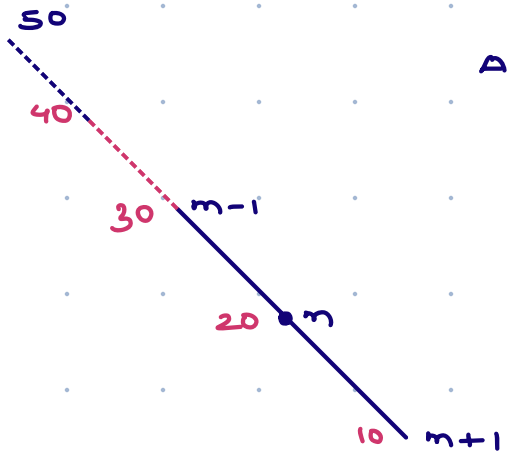
{7 10 12}
↑

Case 4



{7 10 8}
↑

* Analyse second scenario



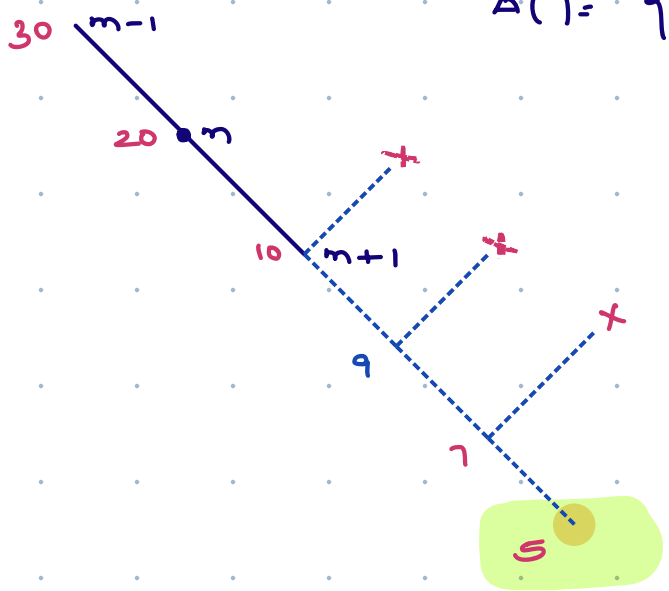
A[]= { 50 40 30 20 10 }
↑
m

For LHS

Worst case → We found no dip & no local minima

For RHS → There will always be one local minima

A[]= { 30 20 10 }
↑




```
if (A.length == 1) return A[0]
if (A[0] < A[1]) return A[0]
if (A[N-1] < A[N-2]) return A[N-1]
```

```
lo = 1
hi = N-2
while (lo ≤ hi)
    m = (lo + hi)/2
    if ( A[m] < A[m-1] && A[m] < A[m+1] ) return A[m]
    else if ( A[m] < A[m-1] && A[m] > A[m+1] ) lo = m+1
    else    hi = m-1
3
```

	0	1	2	3	4	5	6	7
A[ ] = {	1	0	5	4	3	6	8	9 }

hi, m, lo

lo	hi	m	comparison	
0	7	3	A[3] < A[2] && A[3] > A[4]	lo = m+1
4	7	5	A[5] > A[4] && A[5] < A[6]	hi = m-1
4	4	4	A[4] < A[3] && A[4] < A[5]	→ return A[4]

mid = (lo + hi) / 2

int range = 100

lo = 98

hi = 99

m = (98 + 99) / 2 ✗

mid = lo + (hi - lo) / 2

m = 98 + (99 - 98) / 2

= 98 + 0

= 98

Q Given a no. N, find sqrt(N)

sqrt(25) ⟶ 5

sqrt(18) ⟶ 4

sqrt(48) ⟶ 6

Idea → Start iterating from 1 & keep updating ans of $ans^2 \leq N$

TC : $O(\sqrt{N})$

N = 40

i	i*i ≤ N	ans
1	1*1 ≤ 40	1
2	2*2 ≤ 40	2
3	3*3 ≤ 40	3
4	4*4 ≤ 40	4
5	5*5 ≤ 40	5
6	6*6 ≤ 40	6
7	7*7 ≤ 40 → No	

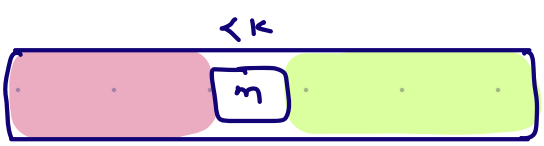
Idea 2 → Bineary Search

Target = $\sqrt{N}$ Search space = 1 to N

Case 1

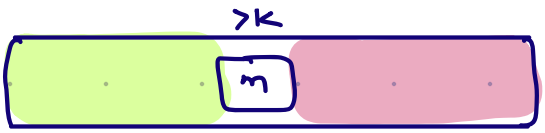
if (m*m == N) return m

Case 2



if (m*m < N)
ans = m
lo = m+1

Case 3



if (m*m > N) hi = m-1

N = 40

lo	hi	M	compare M*M with N
1	40	20	20*20 > 40 : hi = m-1
1	19	10	10*10 > 40 : hi = m-1
1	9	5	5*5 < 40 ans = 5, lo = m+1
6	9	7	7*7 > 49 : hi = m-1
6	6	6	6*6 < 40 ans = 6, lo = m+1
7	6	⟶	stop

```
ans = -1;
lo = 1      hi = N
while (lo ≤ hi) {
    int m = (lo+hi)/2;

    if (m*m == n) return m;
    else if (m*m > n) {
        hi = m-1;
    }
    else {
        ans = m;
        lo = m+1;
    }
}
return ans;
```

TC : O(log N)
SC : O(1)