Binary Search

Monday, 8 July 2024 8:53 PM

↳ searching Algorithms

arr →

0	1	2	3	4	5	6	7
1	3	4	6	8	9	11	15

target: 8

dbt ✓

clr ✓

rpt ✓

Is 8? present in arr?

↳ 8 → 4th index

1) Linear Search

↳ iterating over all the indexes one by one & comparing.

↳ TC: $O(n)$ → N comparisons required

↳ Will work only on sorted array? → NO

↳ Both sorted & unsorted arrays.

Search Space

↳ Indexes where you will find the target

Target will be between 0 to 7

0	1	2	3	4	5	6	7
1	3	4	6	8	9	11	15

→ Sorted

target → 8

Binary Search can be used for sorted sequences.

↳ Bi → 2

↳ sorted array → Will work for repeated

↳ * trying to reduce the search space

mid, lo, hi, mid, hi

0	1	2	3	4	5	6	7
1	3	4	6	8	9	11	15

target = 8

SS

lo → 0

hi → 7

mid = $\frac{0+7}{2}$ = 3.5 → 3

Reduce the search space : SS 12

6 < 8

arr[mid] < target

→ search on right side

lo → 4

hi → 7

mid = $\frac{4+7}{2}$ = $\frac{11}{2}$ = 5.5 → 5

9 > 8

arr[mid] > target

→ search on left side

lo → 4

hi → 4

mid = $\frac{4+4}{2}$ = $\frac{8}{2}$ = 4

8 == 8

Answer ⇒ 4

1) $lo < \cancel{=} hi$

2) $lo \cancel{==} hi$

3) $\cancel{mid} == target$

4) $\boxed{lo <= hi}$ ✓

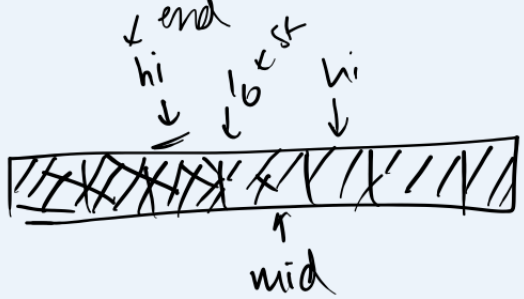
5) $lo \cancel{!=} hi$

↳ element is not present

$\underline{1}$

$\overset{lo}{\downarrow}$ $\overset{hi}{\downarrow}$

$\underset{0}{1}, 2, 3$



$\underline{x} > target$

↳ $hi = mid - 1$

-1.

$\underline{10}$

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

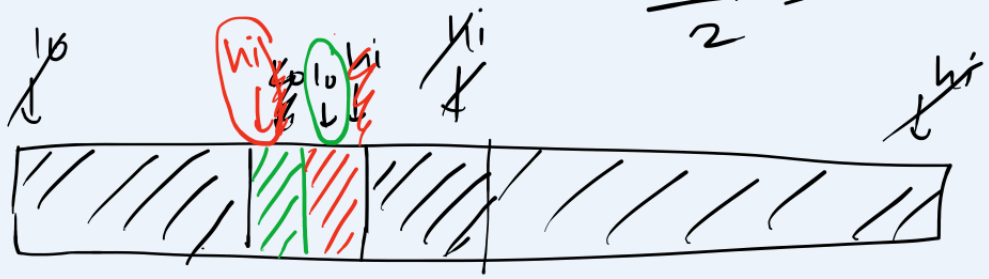
$$- \quad = \frac{(9 \times 10^9)}{2} \rightarrow \underset{\downarrow \; \approx 10^9}{\underline{int}} \quad \underset{\downarrow}{\approx 10^{10}}$$

Integer Overflow

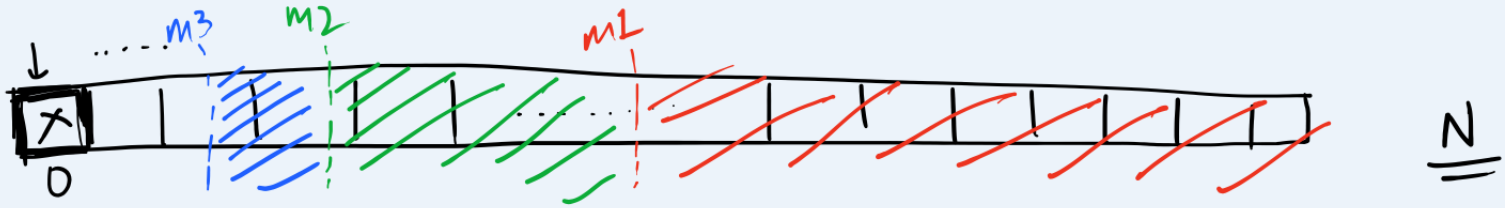
$$\boxed{lo + \frac{(hi - lo)}{2}} = \frac{2lo + (hi - lo)}{2}$$

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

10:18



* In every iteration, the search space should reduce else we encounter an infinite loop.



$0^{th} = N \rightarrow N/2^0$

$1st = N/2 \rightarrow N/2^1$

$2nd = (N/2)/2 = N/4 \rightarrow N/2^2$

$lo = 0$

$hi = N-1$

$3rd = (N/4)/2 = N/8 \rightarrow N/2^3$

$\vdots$

$\underline{K^{th}} = \boxed{1 = N/2^K}$

$\boxed{\log(a^b) = b \cdot \log a}$

$N = 2^K$

$\log_2 N = \log_2 2^K$

$\log a^a = 1$

$\log_2 N = K \cdot \cancel{\log_2 2}^{\,1}$

$\boxed{K = \log_2 N}$

$\hookrightarrow \log_2 N$

Time complexity

$\hookrightarrow \underline{O(\log N)}$

Space complexity

$\underline{O(1)}$

Array + $\underline{\log N}$ → Binary Search

If array is not sorted?

↳ sort + B.S

$O(N^2)$ — $O(\log N)$

$\underline{O(N \log N)}$

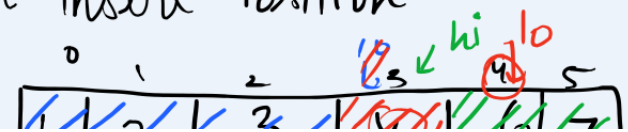
$O(N \log N) + O(\log N)$

$\underline{O(N \log N)}$

Linear

↳ $\underline{O(N)}$ ✓

1) Search Insert Position



lo = 0
hi = 5

mid = 2

lo = 3
hi = 5

mid = 4 → hi = mid - 1

lo = 3
hi = 3

mid = 3

4 < 5 → lo = mid + 1

$\boxed{lo = 4}$

Index where 5 will be inserted

2) Find First & Last Posn

0	1	2	3	4	5	6	7	8	9
1	2	3	3	3	3	5	6	6	9

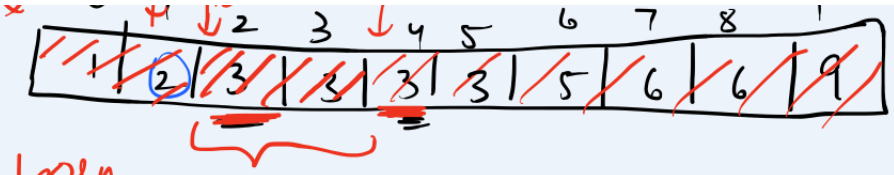
target ⇒ 3 ×

2, 5

lo ⟷ hi

hi ⟷ hi

hi



First posn

lo = 0
hi = 9

mid = 4

arr[mid] == 3 ✗ ✓

first = ~~4~~ 2

lo = 0
hi = 3

mid = 1

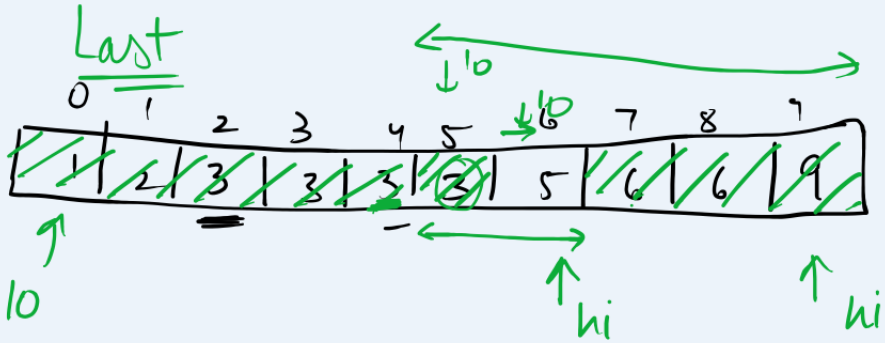
2 < 3

lo = 2
hi = 3

mid = 2

3 == 3

Last



target = 3

last = ~~4~~ 5

lo = 0
hi = 9

mid = 4

3 == 3

$hi = 7$

$lo = 5$ mid = 7 6 > 3

$hi = 9$ ↳ left

$lo = 5$ mid = 5 3 == 3

$hi = 6$

$lo = 6$ mid = 6 5 > 3

$hi = 6$ ↳ left

$lo = 6$ ✗

$hi = 5$

3) Sq root of a number

1) 1, - - - → 10^4

$1 \times 1 = 100$ ✗

$X \times X = A$

$\sqrt{10^7} \approx 10^{7/2}$

100

$1 \to 10^{7/2}$

$= 10^{3.5}$

$2 \times 2 = 4 \neq 100$ ✗

$3 \times 3 = 9 \neq 100$

$4 \times 4 = 16 \neq 100$

⋮

$10 \times 10 = 100$ ✓

Ans

① Use a for loop ✓

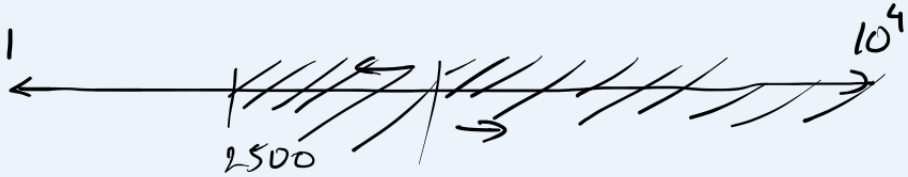
$\approx 10^4$

✗

$\sqrt{N}$ ✗

$\log N$

② Using B.S



sqrt(N)

$lo = 1$

$hi = 10{,}000$

mid = 5000

100

ans = 1

$5000 \times 5000 == 100$ ✗

$lo = 1$

$n = 1111$

$$mid = \frac{1+9999}{2} = \frac{5000}{2}$$

$$= 2500$$

$$2500 \times 2500 \neq = 100$$

$$\boxed{x^2 = A}$$

$$x = \sqrt{A}$$

11×11

121

101

$mid \Rightarrow 9$

$9 \times 9 = 81 < 101$

search on left side values < 11

9

10

10

HW

Count 1 in sorted binary array
Floor in a Sorted Array
Rotated Sorted Array Search
Peak Index in a Mountain Array

Sorted Insert Position

```
import java.io.*;
import java.util.*;

public class Main {

    public static int searchInsert(int[] a, int b) {
        int lo = 0;
        int hi = a.length-1;
        while (lo<=hi){
            int mid = lo + (hi-lo)/2;
            if(a[mid]==b){
                return mid;
            }else if(a[mid]>b){
                hi = mid-1;
            }else{
                lo = mid+1;
            }
        }
        return lo;
    }

    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        int N = sc.nextInt();
        int[] A = new int[N];
        for(int i=0;i<N;i++){
            A[i] = sc.nextInt();
        }
        int B = sc.nextInt();
        System.out.println(searchInsert(A,B));
    }
}
```

Find First and Last Position of Element in Sorted Array

```
import java.util.*;

public class Main {
    public static void findPosition(int a[], int n,int k)
    {
        int first = first(a,n,k);
        int last = last(a,n,k);
        System.out.println(first+" "+last);
    }
    public static int first(int[] a, int n, int target){
        int lo = 0;
        int hi = n-1;
        int first = -1;
        while(lo<=hi){
            int mid = lo + (hi-lo)/2;
            if(a[mid]==target){
```

```
                first = mid;
                hi = mid - 1; // better ans on left
            } else if(a[mid]>target){
                hi = mid - 1;
            } else {
                lo = mid + 1;
            }
        }
        return first;
    }

    public static int last(int[] a, int n, int target){
        int lo = 0;
        int hi = n-1;
        int last = -1;
        while(lo<=hi){
            int mid = lo + (hi-lo)/2;
            if(a[mid]==target){
                last = mid;
                lo = mid + 1; // better ans on right
            } else if(a[mid]>target){
                hi = mid - 1;
            } else {
                lo = mid + 1;
            }
        }
        return last;
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n= sc.nextInt();
        int k= sc.nextInt();
        int array[] = new int[n];

        for(int i=0; i<n; i++){
            array[i]= sc.nextInt();
        }
        findPosition(array,n,k);
    }
}
```

Square root of a number

```
import java.util.*;
import java.lang.*;
import java.io.*;

class Main {
    public static int sqrt(int A) {
        int lo = 1;
        int hi = 10000;
        int ans = 1;
        while(lo<=hi){
            int mid = lo+(hi-lo)/2;
            int sq = mid*mid;
            if( sq == A){
                return mid;
            } else if(sq<A){
                ans = mid;
                lo = mid+1;
            } else {
                hi = mid-1;
            }
        }
        return ans;
```

```
  }

 public static void main (String[] args)
        {
  Scanner sc = new Scanner(System.in);

          int A = sc.nextInt();

      int ans = sqrt(A);
      System.out.println(ans);

        }
}
```