Searching :-

Key	Search space
Linear	Array / Linked list
Binary	Array / Binary Search Tree
Hashing	Hash Table
Tries	Trie / Binary Trie

'MAP'

★ Linear v/s Binary Search on Array

Linear Search

'CAT'

Valid | Invalid

A B S Z

'A' 'Z'

1st step : Look on 'S' → reduce search space

Array: 1 5 7 12 15 18 19 20 25 30

key = 15

1) Look on 5? → move right
2) Look on 7 → move right
3) Look on 12 → ✓ Cut it!

mid = $\frac{\text{high}-\text{low}}{2}$ = 3

★ Look on $\frac{mid}{2}$:

→ Each step : You reduce your search space by half. HOWEVER CAUTIOUS

★ Look on $\frac{3}{4}$ th :

→ Each step : You reduce $\frac{3}{4}$th of your search space. HOWEVER CAUTIOUS

Array Size N,

$T(N) = 1 + T(N/2)$
$= 1 + (1 + T(N/4))$
$= 2 + T(N/2^2)$
$= 3 + T(N/2^3)$
$\vdots$
$= k + T(N/2^k)$

$T(1) = 1$

$\frac{N}{2^k} = 1 \Rightarrow N = 2^k \Rightarrow k = \log_2 N$

$T(N) = 1 + T(\frac{N}{2})$
$T(N) = \log_2 N$

3 points

2 3 4 5 6 7 8 9 10 11 12 13 14 15

Division | # equations | # terms / steps | # steps

2 | 1 | $\log_2 N$ | $1 \times \log_2 N$

3 | 2 | $\log_3 N$ | $2 \times \log_3 N$

K | (K-1) | $\log_K N$ | $(K-1) \times \log_K N$

$P_1 = \frac{(low + high)}{2} = 2$

$P_2 = 2 \times \frac{(low + high)}{3} = 4$

$2 \log_3 N = \log_3 N^2$

$3 = \frac{\log N}{\log K}$

$\log_K N^2 = \frac{\log N^2}{\log K} > \log_K N$

Binary Search : Search for a value in a sorted array in $O(\log_2 N)$

```
bool BST[int ar[], int key, int low, int high]
{
    low = 0, high = n-1; int mid;
    {
        mid = (low+high)/2; // low + ((high-low)/2);
        if (ar[mid] == key) return True;
        else if (ar[mid] < key)
        {
            // move to right
            low = mid+1;
        }
        else
        {
            // move to left
            high = mid-1;
        }
    }
}
```

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

```
bool BSR(int ar[], int key, int low, int high)
{
    if (high < low) return False;
    int mid = (low + high)/2;

    if (ar[mid] == key) return True;
    if (ar[mid] < key) // move right
        BSR(ar, key, mid+1, high);
    else
        BSR(ar, key, low, mid-1);
}
```

TC : $O(\log N)$
SC : $O(\log N)$
Recursion Stack

Linear N	BS $\log N$
1	1
2	1
$1024 = 2^{10}$	10
$\vdots$ $= 2^{32}$	32

Q2 Given a sorted array and given Q queries. For each query 'q', find the index of insertion.

Array : -5 -2 0 5 7 12 15 19 20 25

0 1 2 3 4 5 6 7 8 9

q : 6 = 4
q : 10 = 5

(A1) Brute Force :

For each query 'q'

Iterate over array, first index where $ar[i] > q$. return i

TC : $O(n) \times q$
SC : $O(1)$

(B.S.) ans = -1; (q = 6)

1) Look at mid, $ar[mid] \geq q$
low, high
0 9
ans = mid = 4 // update ans
high = mid - 1 // move left for better ans

2) low, high, $ar[mid] > q$
low = mid + 1 // move to right

int BS (int ar[], int q, int low, int high)
{
low = 0, high = n-1; ans = -1;
int mid;
while (low ≤ high)
{
mid = low + $\frac{(high-low)}{2}$;

if (ar[mid] > q) // valid part
ans = mid
// move to better ans
high = mid - 1

else // invalid part
low = mid + 1;
}
}
return ans;

'q'
TC : $(\log N) \times q$
SC : $O(1)$

Q3 Given a sorted array (with duplicates) and an integer 'x'. Return count of 'x'.

Array -5 -5 -3 1 2 2 2 2 2 4 4 7 7

0 1 2 3 4 5 6 7 8 9 10 11 12

x = 2 ⇒ 5

(B1) Brute Force :
Iterate & update count variable

TC : $O(n)$

(B2) Binary Search.

TC : $O(\log N) + O(k)$
$= O(N)$

4 5 6 7 8
2 2 2 2 2

B1.2 P_1 | B1.1 | P_2 B1.2

P_1 = mid; high = mid - 1; | update & move to better ans | P_2 = mid; low = mid + 1

Code.

```
BS1 [int ar[], int x, int low, int high]
{
    low = 0, high = n-1; ans = -1;
    ans = y;
    while (low ≤ high)
    {
        mid = low + ((high-low)/2);

        if (ar[mid] < x) {
            low = mid + 1;
        }
        else if (ar[mid] > x) {
            high = mid - 1;
        }
        else { // valid part
            ans = mid;
            high = mid - 1;
            low = mid + 1;
        }
    }
    return ans;
}
```

BS2
ans = -2;
ans = y - 1;

Find mid = $\frac{(L+R)}{2}$

TC : T(mid - 1) + T(mid - 2) + T(mid - 3) ∴ $O(\log N) + O(\log N)$ + O(1)

TC : $O(\log N)$

Q4. Given a number P, find sq. root of P?

P = 4 → Ans = 2

P = 5 → Ans = 2 (Floor)