Binary Search

what is searching?

arr[] = { 1, 2, 5, 7, 13, 20, 24, 25, 26, 30 } →

ele = 20

find?

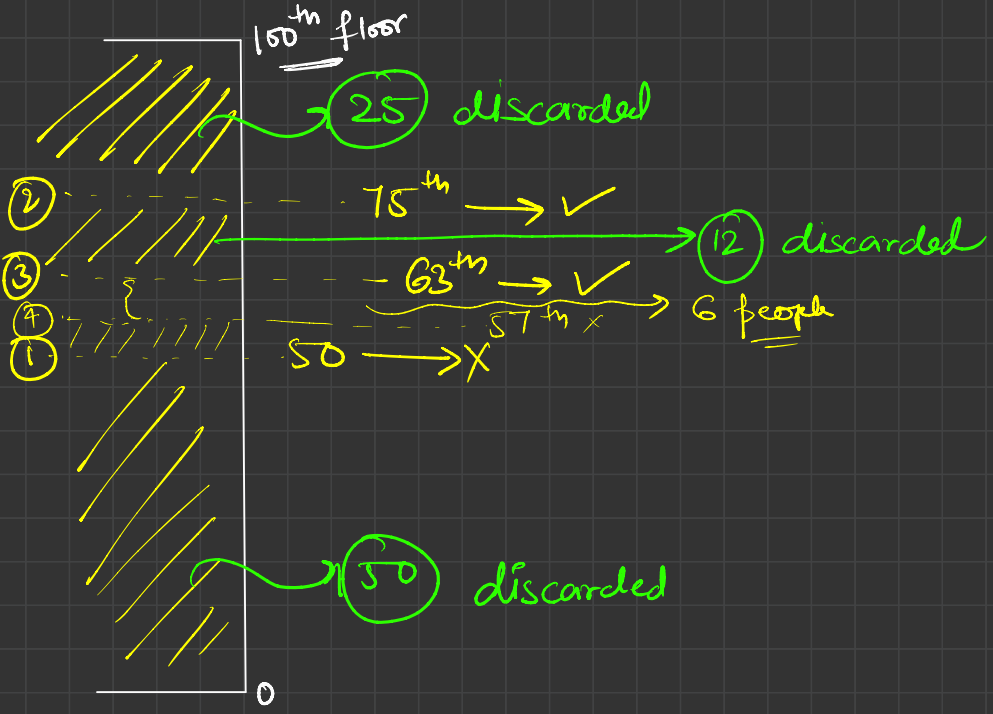
```
for (int i = 0 → n)
{ if (arr[i] == ele)
      return i;
}
```

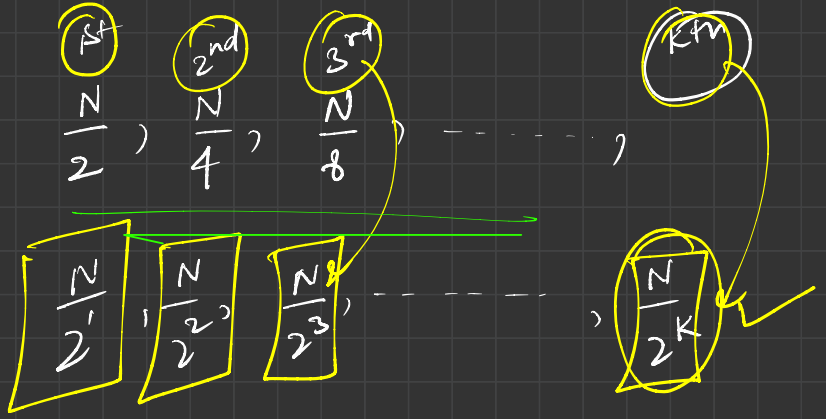
TC: O(N)

linear search!

Whenever you see sorted array,
and you have to search something,
that can be done in TC: O(log N)
using Binary Search!

Binary → Applied over sorted region.


150th floor
25
discarded
2
75th
V
12
discarded
3
63th
V
57th
6 people
4
1
50
X
50
discarded
0

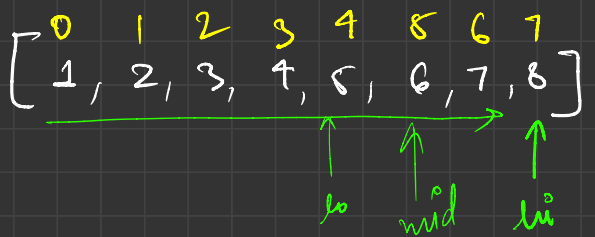


$$\frac{N}{2^K} = 1$$

$$N = 2^K$$

$$\log_2 N = \log_2 2^K \Rightarrow \boxed{\log_2 N = K}$$

Binary Search



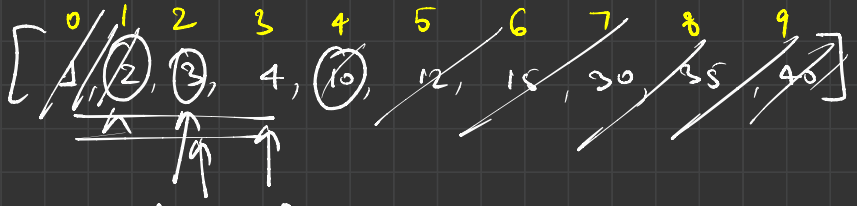
ele = 6

$\frac{4+7}{2} = 5$

if (arr[mid] < ele) → lo = mid+1;
{ discard left side }

TC: O(logN)

$\frac{N}{2}, \frac{N}{4}$ - - - - - -



ele = 3

0+9 / 2 = 4

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

2

Sorted array → find

Binary search

TC : O(log N)

```java
public static int BinarySearch(int n, int[] arr, int ele) {
  int lo = 0;
  int hi = n - 1;

  while (lo <= hi) {
    int mid = (lo + hi) / 2;

    if (arr[mid] == ele) {
      return mid;
    } else if (arr[mid] > ele) {
      hi = mid - 1;
    } else {
      lo = mid + 1;
    }
  }

  return -1;
}
```

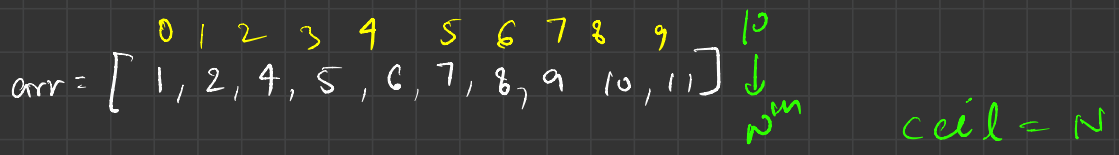
TC: O(logN)

SC: O(1)

0 1 2 3 4 5 6 7 8 9

arr[] = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

Insert position in a sorted Array



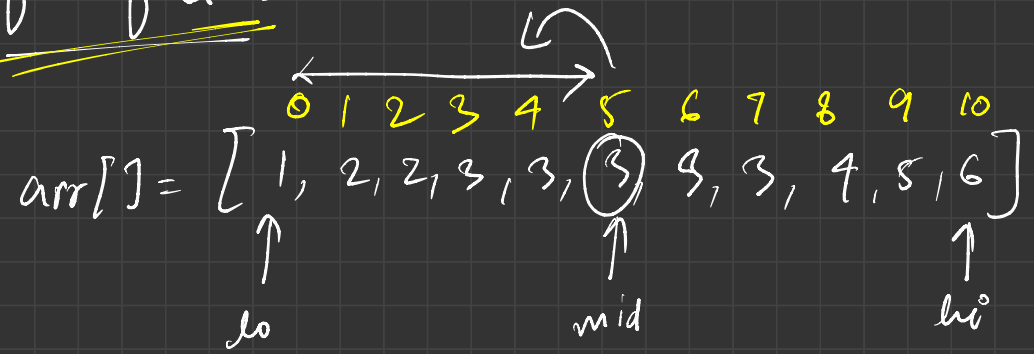

```java
public static int searchInsert(int[] a, int b) {
  // Write code here
  int ceil = a.length;

  int lo = 0;
  int hi = a.length - 1;

  while (lo <= hi) {
    int mid = (lo + hi) / 2;
    if (a[mid] == b) {
      return mid;
    } else if (a[mid] > b) {
      ceil = mid;
      hi = mid - 1;
    } else {
      lo = mid + 1;
    }
  }

  return ceil;
}
```

find first Occ.



ele = 3

potential firstOcc = 5

mid

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

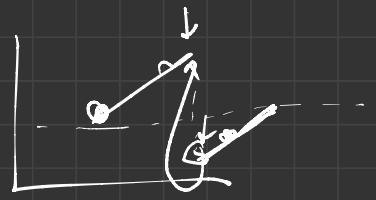
lo

hi

ele = 1

first occ = ~~2~~ 0

arr = [7, 8, 1, 2, 3, 4, 5]

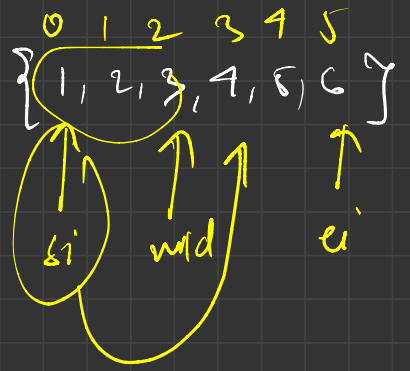


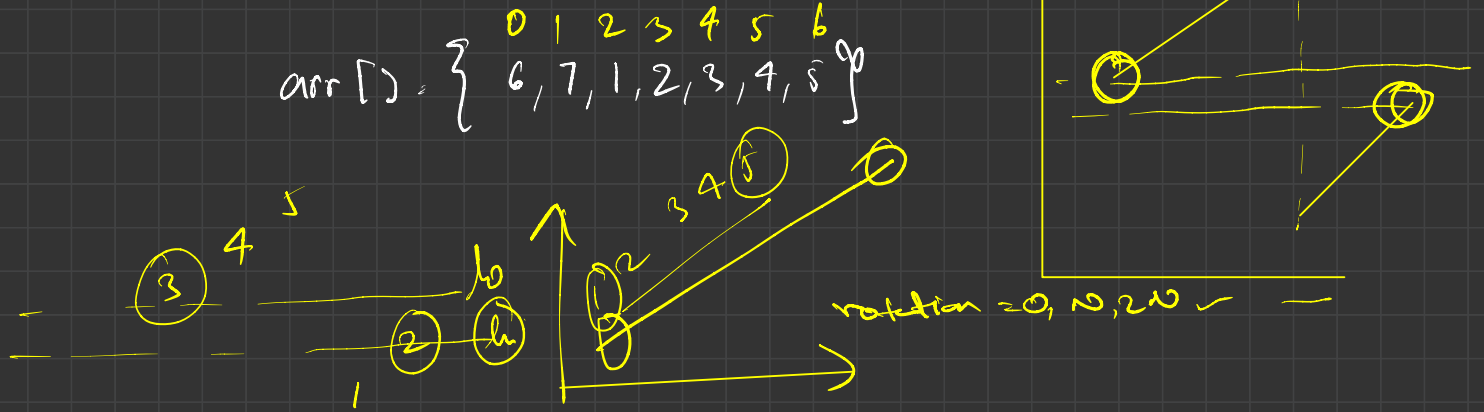
```
if (arr[mid] > arr[mid+1])
    return mid+1;
else if (arr[mid-1] > arr[mid])
    return mid;
else if (arr[lo] <= arr[mid])
    lo = mid+1;
else
    hi = mid-1;
```

```java
static int findMin(int arr[], int lo, int hi) {
  //Write your code here
  while (lo <= hi) {
    int mid = (lo + hi) / 2;

    if (mid - 1 >= 0 && arr[mid] < arr[mid - 1]) {
      return arr[mid];
    } else if (mid + 1 < arr.length - 1 && arr[mid] > arr[mid + 1]) {
      return arr[mid + 1];
    } else if (arr[lo] <= arr[mid]) {
      lo = mid + 1;
    } else {
      hi = mid - 1;
    }
  }

  return -1;
}
```





Square Root

```
potential Ans = 0;
for(i -> 0 -> n)
{
    if (i*i <= n)
        pAns = i;
}
```

T : O(n)

n = 10

```
int i=0;
for( ; i*i<=n; i++)
{ }
return i*i==n ? i : i-1;
```

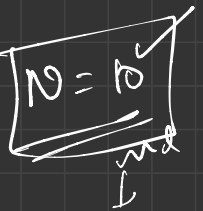
i = 0 ✗ ✗ ✗ 4

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

16

0, 1, 2, 3, 4

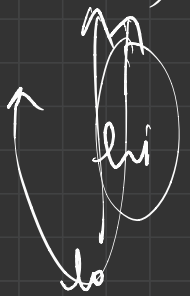
Binary Search



$3 \times 3 = 9 < 10$

(T6) 4 ✓

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]



BA = 3

$O(\log N)$

$$O(1) < O(\log N) < O(\sqrt{N}) < O(N) < O(N^m)$$

$\sqrt{10} = 3.33$

$\log(N)$