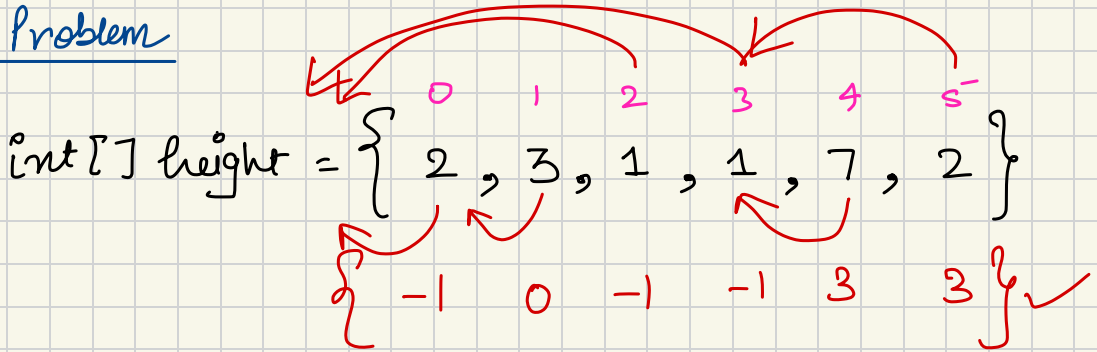
Height Problem



✓ next smaller on left

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

Agenda

- Binary search Basics
- Search insert position
- Find first and last Position
- Square root of a number
- Search in a 2D Matrix

Binary Search. {Algorithm} searching purposes

int[] arr = {1, 3, 7, 10, 11, 14, 20, 26} key = 14

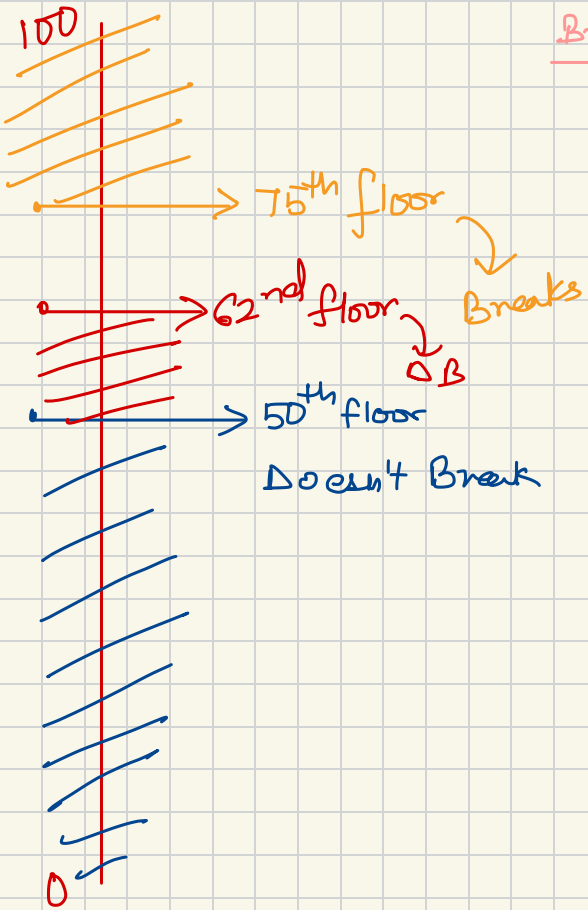
Brute force

✓ Linear search

```
for (int i = 0; i < n; i++)
{
    if (arr[i] == key)
        return i;
}
```

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

Puzzle . min floor from which if you throw the brick it breaks.



Brute force

↳ 100 fresh bricks

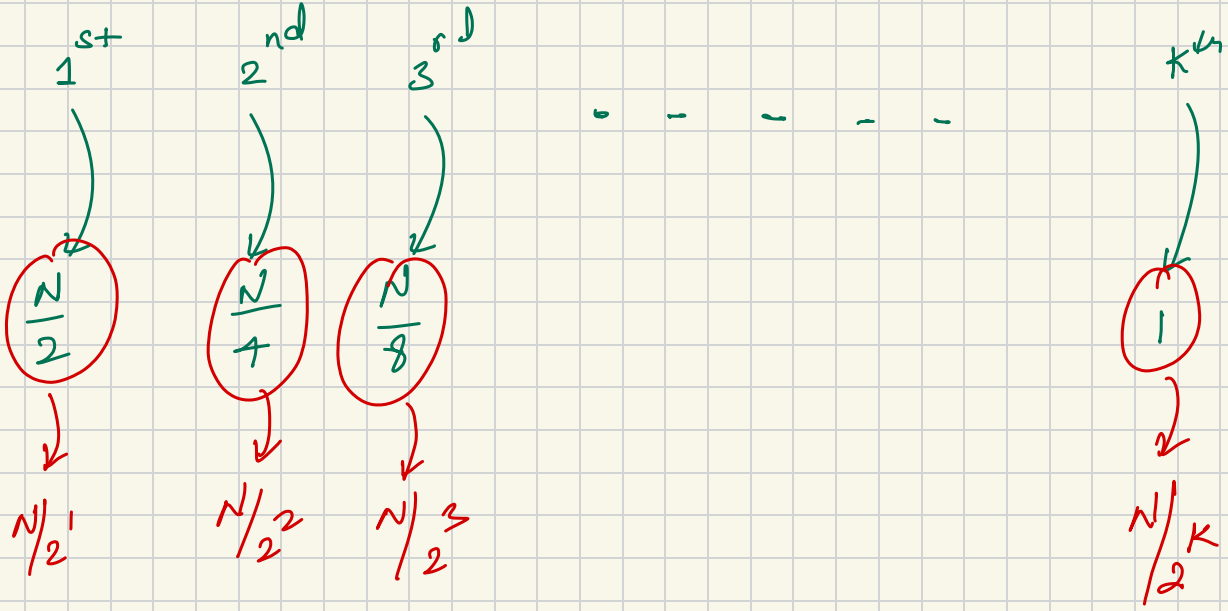
using 1st brick ↓ eliminated 50 floors

using 2nd brick ↓ eliminated 25 floors

using 3rd brick ↓ eliminated 12 floors

using Kth brick, ↓ eliminated 1 floor.

N - floors



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

$$2^K = N$$

taking $\log_2$ Both Sides

$$\log_2 2^K = \log_2 N$$

$$K \log_2 2 = \log_2 N \quad (\log_2 2 = 1)$$

$$K = \log_2 N$$

throws required,

$$\log_2(100)$$

$$2 \times \log_2 10 \approx 3.1$$

$\approx$ 7 Bricks

int[] arr = { 1, 3, 7, 10, 11, 14, 20, 26 } Key = 11

① define region of search

② divide region into two parts.

TC: $O((\log_2 N))$ } Binary search

SC: $O(1)$

case 1 key == arr[mid] : return mid

case 2 key > arr[mid] : eliminate left side

case 3 key < arr[mid] : eliminate right side

Binary Search

↳ array should be sorted { search region should be sorted }

TC: $O(\log_2 N)$ SC: $O(1)$

↳ BS 99% of time.

Search insert position / ceil value / find just greater person.

```
                 0  1  2  3  4  5  6
int[] arr = { 1, 3, 5, 10, 11, 20, 27 }      key = 2
```

Brute Force

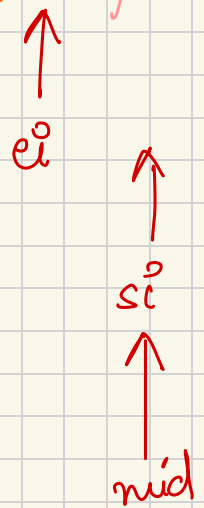
↳ Linear Search, { return first ele greater than you }

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

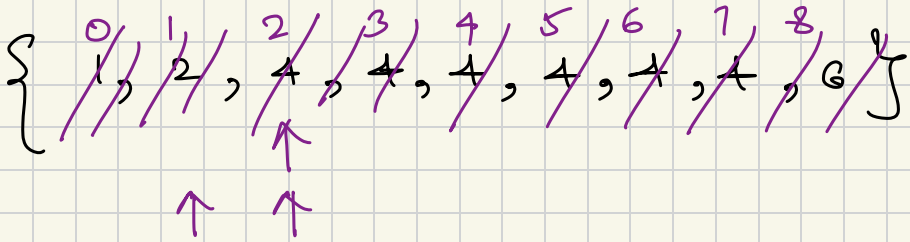
int[] arr = { 1, 3, 5, 10, 11, 20, 27 }

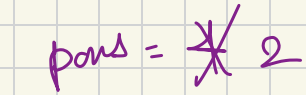
0 1 2 3 4 5 6

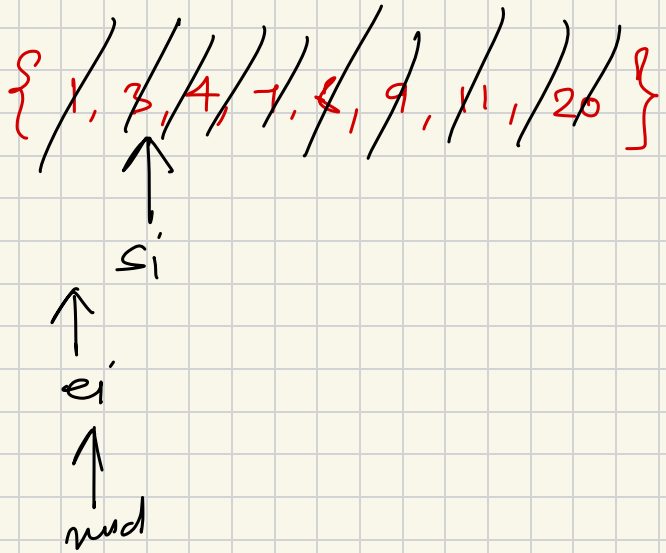
key = 4



pans = ~~10~~ 5


0 1 2 3 4 5 6 7 8
1, 2, 4, 4, 4, 4, 4, 4, 6
key = 3

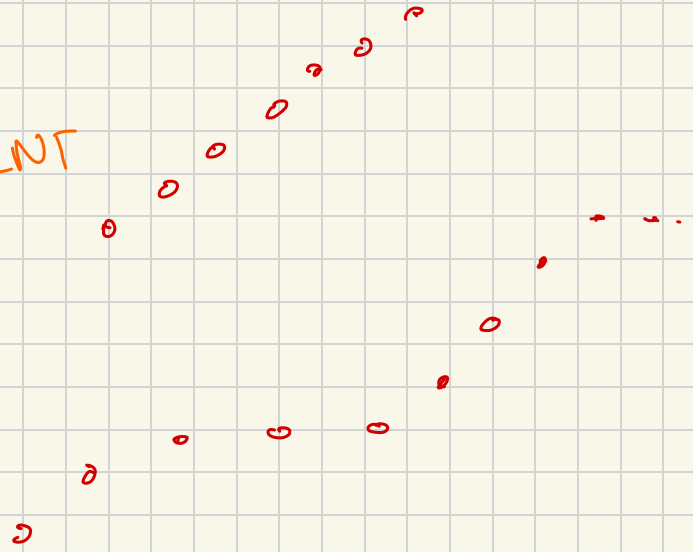

pans = 2


1, 3, 4, 7, 8, 9, 11, 20
key = 9
pans = 3
si
ei
mid

inc. array

non-dec. array

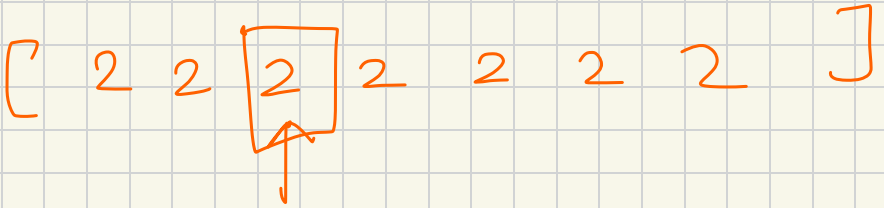
} DIFFERENT

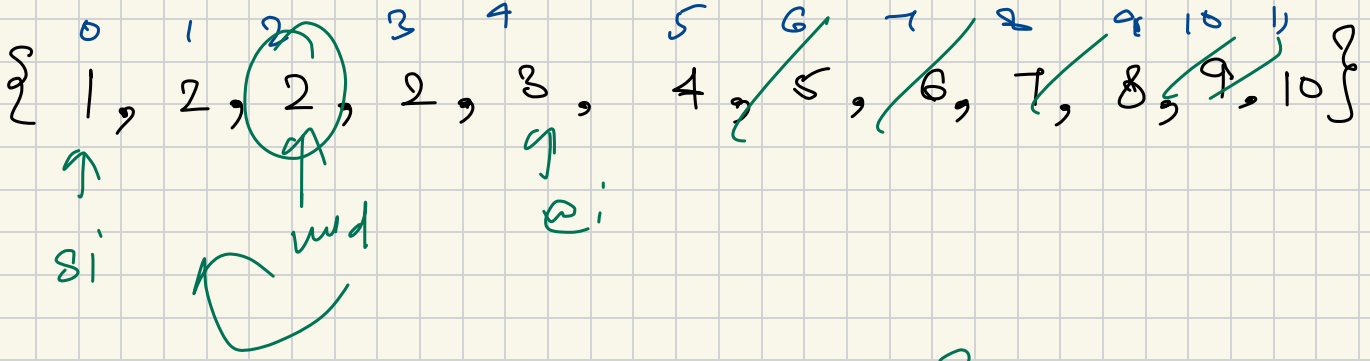


find first and last pos. of an ele.

```
int[] arr = { 1, 2, 2, 2, 3, 4, 5, 6, 7 }   key = 2
            0  1  2  3  4  5  6  7  8
```

first position





pos = 2

Square Root

N = 36

sqrt(N) = 6

N = 110

sqrt(N) = 10 ✓

Brute force

```
ans = -1;
for(int i = 1 ; i <= N ; i++)
{
    if(i * i <= N)
        ans = i
}
return ans;
```

T.C : O(N)
S.C : O(1)

Optimized

```
int ans = 0;
for(int i = 1; i*i <= N; i++)
{
    ans = i;
}
return ans;
```

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

SC : $O(1)$

Optimize

$\sqrt{20*20} == 40$

N = 40

pans = 6

{TC : O($\log_2 N$)}

✓ N = 20

pass = ✗ ✗ 4

si = ✗ ✗ ✗ 5

ei = 20 ✗ 4

mid = ✗ ✗ ✗ ✗ 4

TC: $O(\log_2 N)$

SC: $O(1)$

$\sqrt{N}$

$\sqrt{10000}$

$= 100$

$\log_2 N$

$\log_2 10^4$

$4 \times \log_2 10$

$= 13$ ✓

H.W

1) Search in a 2D Matrix