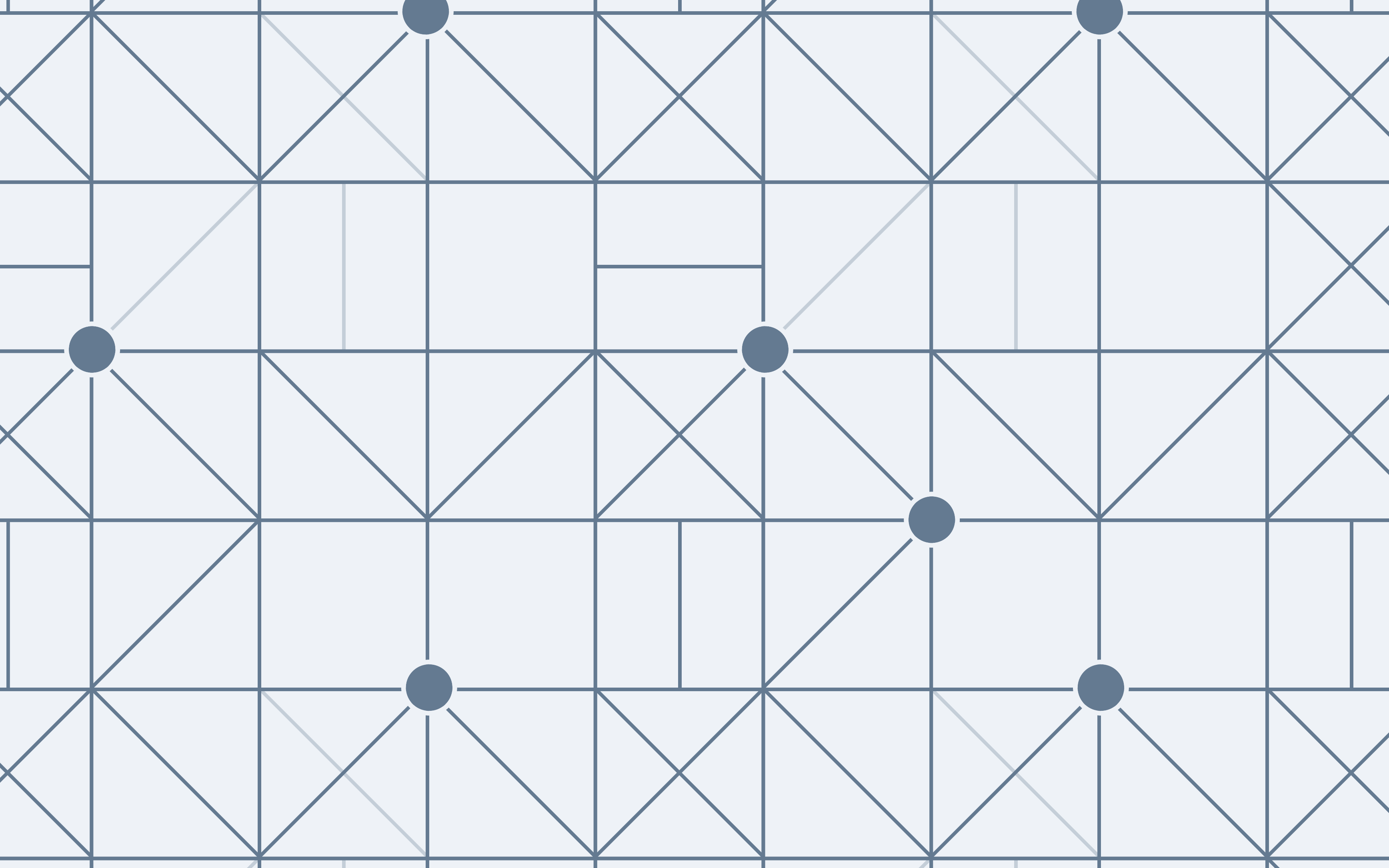


Binary Search. {Algorithm} {Searching Algorithm}

```
                0    1    2    3     4     5     6    7
int[] arr = {   1 ,  3 ,  7 ,  10 ,  11 ,  14 ,  20 , 24 }      target = 14
```

find?

Brute force

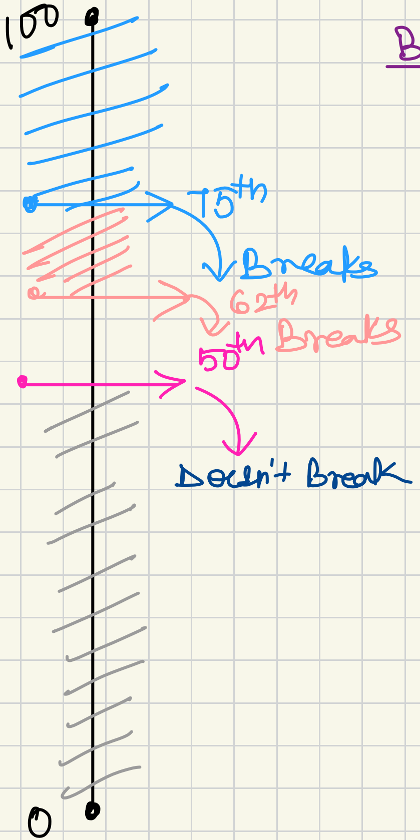
```
for (int i = 0 ——> n)
{    if (arr[i] == target)
            return i;
}
```

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

Linear Search.

Puzzle.

min th floor from which if you throw a fresh brick it will break?



Brute force

→ 100 fresh Bricks.

using 1st brick I eliminated 50 floors

using 2nd brick I eliminated 25 floors,

using 3rd brick I eliminated 12 floors.

using kth brick, I eliminated 1 floor.

No. of throws = $\log_2 100 = 2 \times \log_2 10 = \sim 7$ throws

N - floors

throw No.	1	2	3	4	. . .	K
eliminated.	$\frac{N}{2}$	$\frac{N}{4}$	$\frac{N}{8}$	$\frac{N}{16}$		1
	$\frac{N}{2^1}$	$\frac{N}{2^2}$	$\frac{N}{2^3}$	$\frac{N}{2^4}$		$\frac{N}{2^K}$

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

$$2^K = N$$

taking log Both Sides $\log_2$

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

$$K \cancel{\log_2 2} = \log_2 N$$

$$\boxed{K = \log_2 N}$$

$$\boxed{\text{Hence No. of throws} = \log_2 N}$$

$$\begin{array}{cccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ \cancel{1} & \cancel{3} & \cancel{7} & \boxed{\cancel{10}} & 11 & \boxed{\cancel{14}} & \cancel{20} & \cancel{24} \end{array}$$

int[] arr = { ~~1~~, ~~3~~, ~~7~~, ~~10~~, 11, ~~14~~, ~~20~~, ~~24~~ }; target = 11

Sorted array ;

lo

hi

mid

$\log_2(8) = \log_2 2^3$

$= 3\cancel{\log_2 2}^{1}$

$= 3$ steps

TC: $O(\log_2 N)$

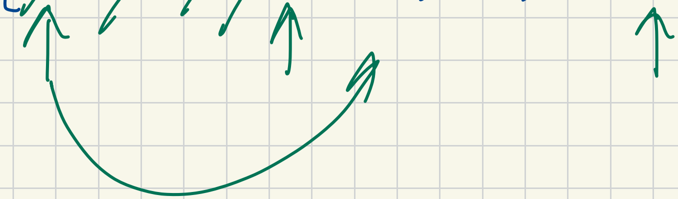
SC: $O(1)$;

① define range of search

② get the mid point

0 1 2 3 4 5 6 7

int[] arr = {~~1~~, ~~3~~, ~~7~~, ~~10~~, 11, 14, 20, 24}   target = 14



Case 1: arr[mid] == target

↳ got the ans

Case2: arr[mid] < target

↳ lo = mid+1;

Case3. arr[mid] > target

↳ hi = mid-1;

Binary Search

↳ $\boxed{0/1}$

① Step 1 : Define range your ans can be present.

② Step 2 ; Divide range into 2 equal halves.

③ Step 3 : try eliminating one half and take another

④ Step4 : update your search range, and start again with step 2.

$\boxed{TC: O(\log_2 N) \quad SC: O(1)}$

Binary Search

→ search region should be sorted X

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

→ 99% of time BS Ques.

Search insert position / ceil value / find just greater person

```
              0    1    2    3    4    5    6
int[] arr = { 1 ,  3 ,  7 , 10 , 11 , 20 , 27 }        key = 2
```

Brute force

↳ linear search, { return first person's index greater than your value }

✓ TC: O(N)
SC: O(1)

0 1 2 3 4 5 6

int[] arr = {~~1~~, ~~3~~, ~~7~~, ~~10~~, ~~11~~, ~~20~~, ~~27~~}   key = 2

lo

ans = ~~10~~ 3

Case 1   arr[mid] == key

hi

return mid

Case 2   arr[mid] > key

mid

hi = mid - 1;

ans = arr[mid]

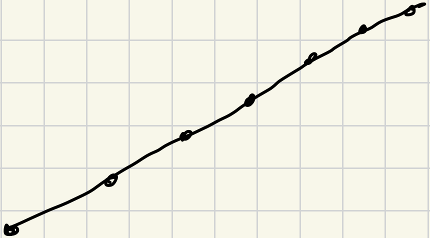
Case3   arr[mid] < key

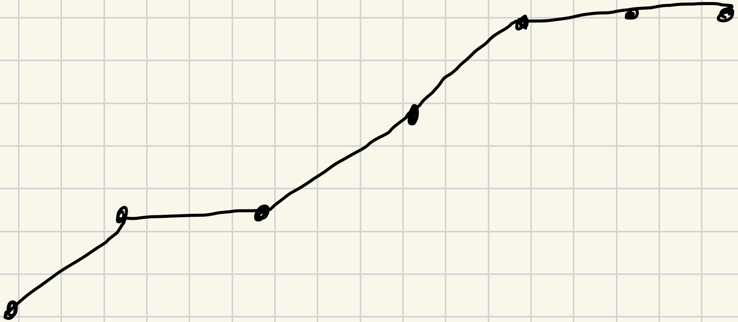
lo = mid + 1;

✓ ① inc. array

✓ ② non. dec array

} DIFFERENT





find first and last Position of an Element.

$$\begin{array}{cccccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\ 1, & 2, & 2, & 2, & 2, & 2, & 2, & 3, & 3, & 5, & 10, & 10 \\ & \uparrow & & & & & \uparrow & & & & & \\ & \text{first} & & & & & \text{last} & & & & & \end{array}$$

int [] arr = { 1, 2, 2, 2, 2, 2, 2, 3, 3, 5, 10, 10 }   ele = 2

Brute force

↳ Linear Search

TC: O(N) SC: O(1) ✓

find first position

	0	1	2	3	4	5	6	7	8	9	10	11
int[] arr = {	1,	2,	2,	2,	3,	3,	3,	3,	3,	5,	10,	10 }

ele = 2

hi

lo

pans , ~~2~~ 1

mid

Case 1 arr[mid] == ele

pans = mid;

hi = mid - 1;

Case2 arr[mid] > ele

hi = mid-1;

case3 arr[mid] < ele

lo = mid+1;

Square Root .

$$\left.\begin{array}{l} N = 36 \\ \text{Sqrt}(N) = 6 \end{array}\right\}$$

$$\left.\begin{array}{l} N = 25 \\ \text{sqrt}(N) = 5 \end{array}\right\}$$

$$\left.\begin{array}{l} N = 10 \\ \text{sqrt}(N) = 3 \end{array}\right\}$$

$$\left.\begin{array}{l} N = 27 \\ \text{Sqrt}(N) = 5 \end{array}\right\}$$

$$\left.\begin{array}{l} N = 110 \\ \text{Sqrt}(N) = 10 \end{array}\right\}$$

Brute force

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

}

return ans ;
```

TC : O(N)
S : O(1)

N = 5

i = 1
i = 2
i = 3
i = 4
i = 5

$ans = \cancel{1}\ 2$

$sqrt(5) = \boxed{2}$ ✓

Optimize.

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

TC: $O(\sqrt[2]{N})$

SC: $O(1)$

N = 6

i = 1

i = 2

~~i = 3~~

$\boxed{ans = \cancel{1}\ 2}$

Optimize.

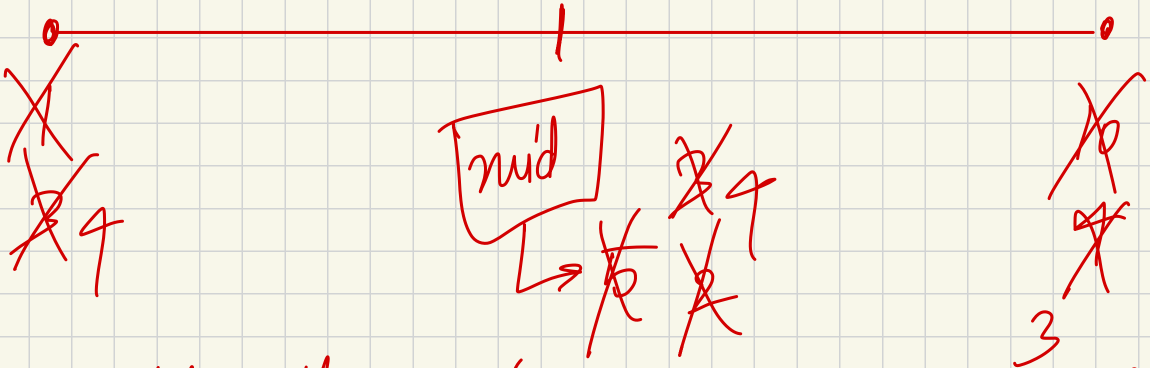
$N = 10$

ans = ~~1~~ 3

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

Range of search

$Sqrt(10) = 3$ ✓



Case $mid * mid == N$

Case2 $mid * mid > N$

$hi = mid - 1$

Case3 $mid * mid < N$

$lo = mid + 1$

$ans = mid$

$$\sqrt[2]{10^6} = 10^3$$

$$\log_2 10^6 = 6 * \log_2 10 \longrightarrow 3.1$$

$$\underline{\approx 18 \text{ or } 19}$$