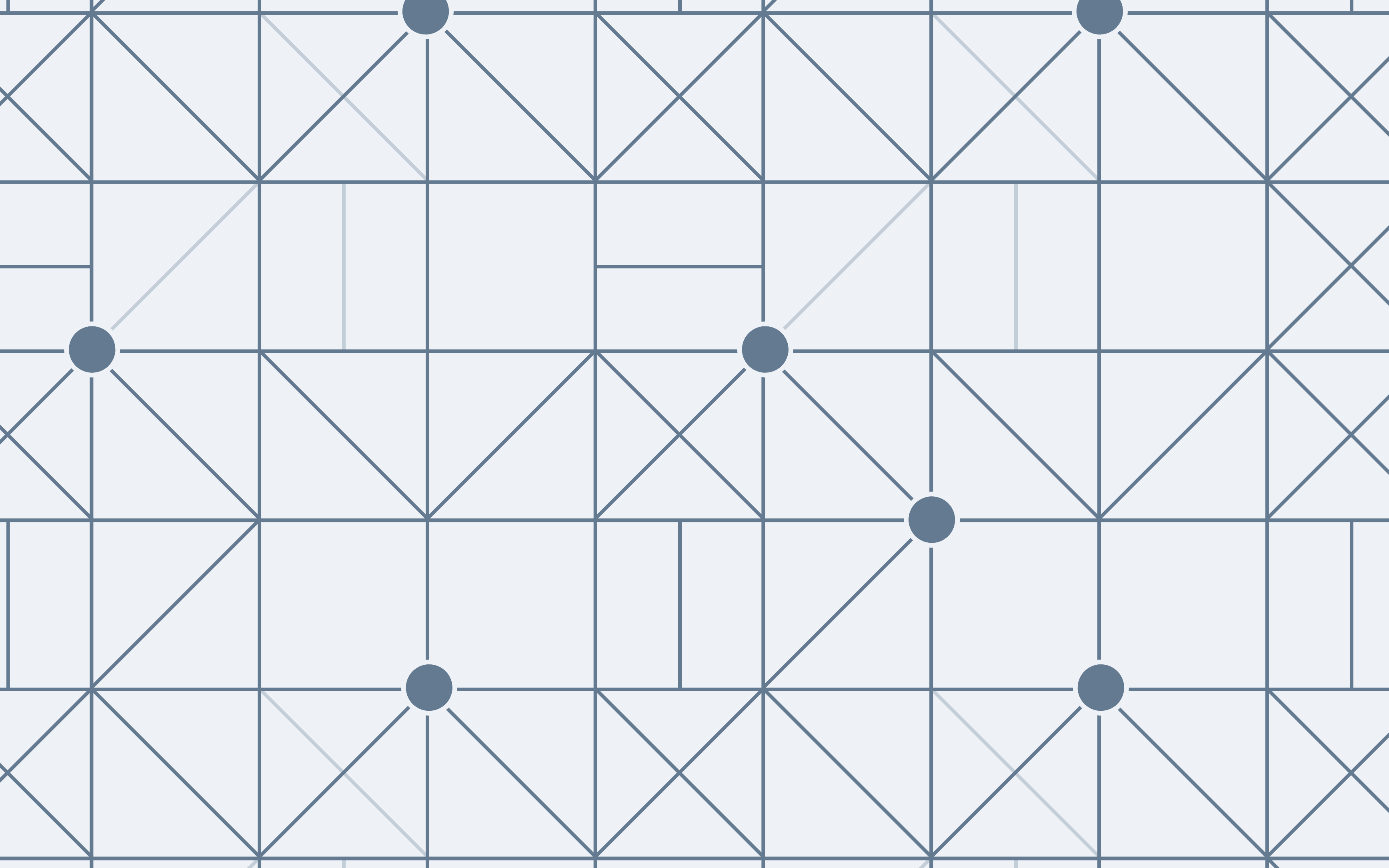


Binary Search. (Algorithm) (searching Algo)

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

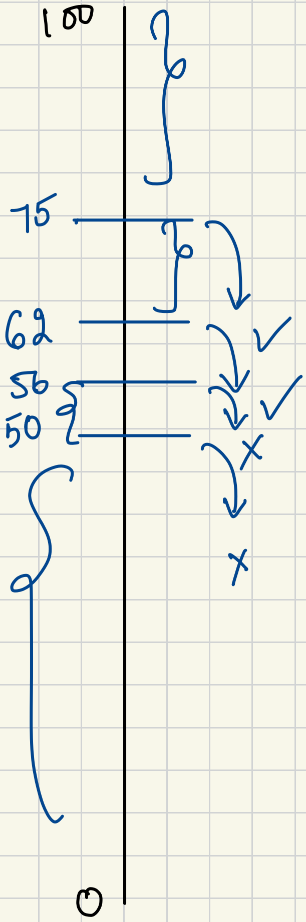
find?

Brute force.

linear search

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

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



what is the lowest floor from which you will throw a brick and it will break;

Brute force = <u>100 Bricks</u>

using 1st Brick I eliminated 50 floors

using 2nd Brick I eliminated 25 floors

using 3rd Brick I eliminated 13 floors

using 4th Brick, I eliminated 6 floors

after 1^{st} 2^{nd} 3^{rd} - - - - - - - - K^{th} ✓

$$\frac{N}{2} \to \frac{N}{2^1} \qquad \frac{N}{4} \to \frac{N}{2^2} \qquad \frac{N}{8} \to \frac{N}{2^3} \qquad \cdots \qquad 1 \to \frac{N}{2^K}$$

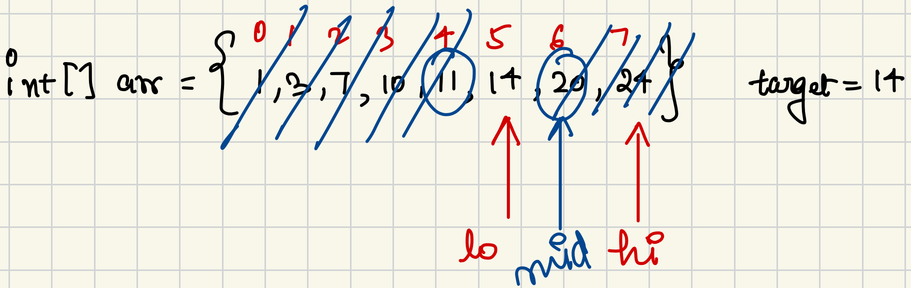
$$\log_2 100 \to 2 \times 3.1 \approx \boxed{6 \text{ to } 7}$$ ✓

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

$$2^K = N$$

$$\boxed{K = \log_2 N}$$ ✓ → Brick required!

{ i needed $\log_2 N$ Bricks in total



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

① Define Search Range

② try to eliminate half of the range and search in other half

$$\frac{\text{target} > \text{mid}}{\text{target} < \text{mid}}$$

Binary Search Algorithm :

① Step 1 : define range of search

② Step2 : divide range into 2 half's

③ step3 : try eliminating one half

④ Step4 : update range to other half

⑤ step5 : go back to step2 until ans is found!

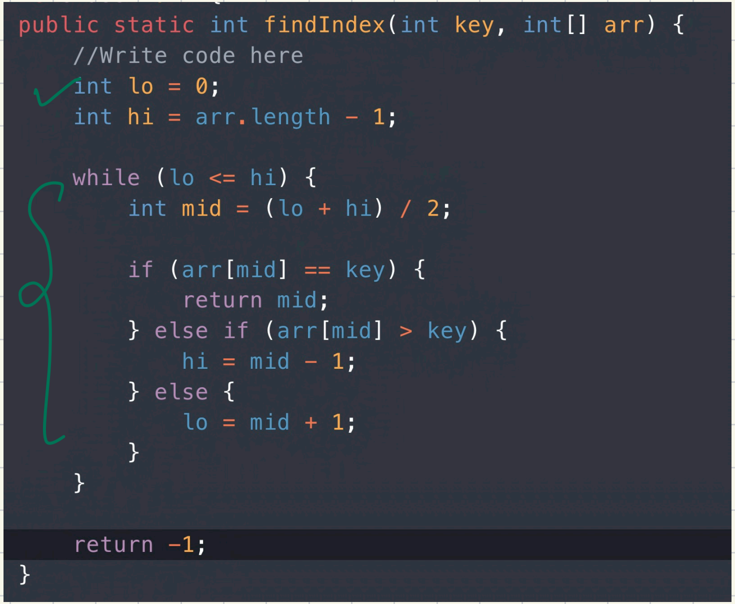
$$\boxed{TC : O(\log_2 N) \qquad SC : O(1)}$$

🔴 → hi
🔵 → lo

🟡 → mid

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

key = 13



```
public static int findIndex(int key, int[] arr) {
    //Write code here
    int lo = 0;
    int hi = arr.length - 1;

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

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

    return -1;
}
```

TC: $O(\log_2 N)$

SC: $O(1)$

Binary Search.

→ region should be sorted! X

✓ { where you can take decision to eliminate one half and take another

→ Expected TC: $O(\log_2 N)$ } 99% time think

Binary Search

Search insert position / ceil value / find just greater person

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

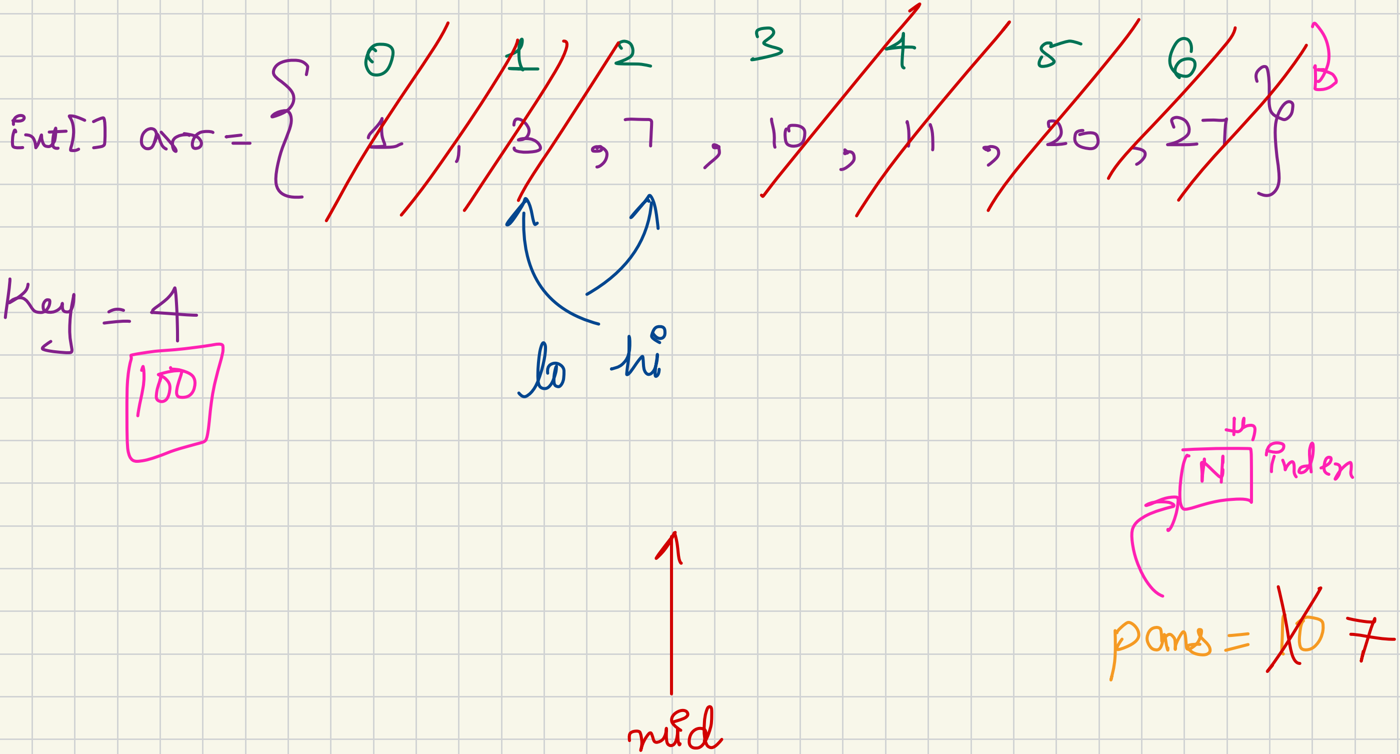
0	1	2	3	4	5	6
1	3	7	10	11	20	27

Key = 2

Brute force

Linear Search: whenever someone greater found
return that index

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



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

Key = ~~4~~ 100

ans = ~~10~~ 7

find first and last Position of an Element

1. inc. array
2. non dec. array

given a non-decreasing array

```
                0  1  2  3  4  5  6  7  8  9  10  11  12
int[] arr = {   1, 2, 2, 2, 2, 2, 3, 4, 4, 4, 10, 20, 20 }
```

Key = 2

first pos   last pos

Brute force

→ linear Search TC : O(N) SC : O(1)

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

int[] arr = { ~~1~~ , ~~2~~ , ~~2~~ , ~~2~~ , ~~2~~ , 2 , ~~3~~ , ~~4~~ , ~~4~~ , ~~4~~ , ~~10~~ , ~~20~~ , ~~20~~ }

lo hi

Key = 2

pans = ~~2~~ ~~4~~ 5

Case 1 arr[mid] == key

pans = mid

Eliminate leftside lo = mid+1

Case 2 arr[mid] > key

hi = mid-1

Case 3 arr[mid] < key

Square Root

↳ Put $\boxed{x}$

↓

find sq. root

if x is a perfect square → $\sqrt{x}$

else x is not a perfect square → $\text{floor}(\sqrt{x})$

$x = 9$

ans $= 3$

$x = 100$

ans $= 10$

$x = 90$

ans $= 9$

$x = 30$

ans $= 5$

clear?

Square Root

Brute force

$\{ TC: O(x) \quad SC: O(1) \}$

```
for(int i=1; i<=x; i++)
{
        if( i*i <=x)
            ans = i;
        else
            break;
}
```

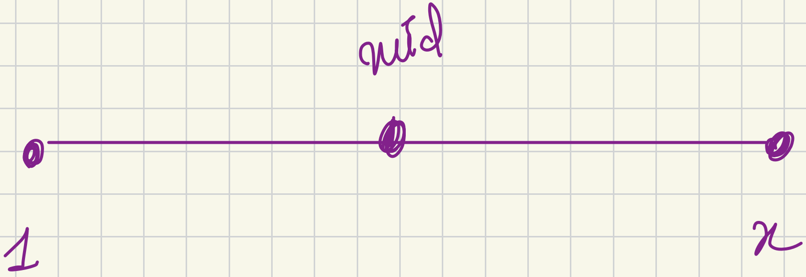
Better:

TC: $O(\sqrt{x})$ SC: $O(1)$ }

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

}
```

Better? Yes



Binary Search
↳ sq root

```
lo = 1
hi = x
if ( mid*mid == x)
        return mid;
else if (mid*mid > x)
    hi = mid-1;
else
    pass = mid; lo = mid+1;
```

TC: $O(\log_2 x)$

SC: $O(1)$

$x = 10$

~~1~~ ~~3~~ 4

~~10~~

~~4~~ 3

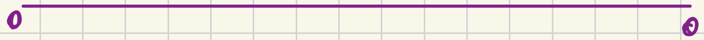
mid = ~~5~~ ~~2~~

~~3~~

4

pans = ~~2~~ 3 ✓

$x = 6$



~~1~~

~~2~~

3

~~6~~

2

mid = ~~3~~ ~~1~~ 2

pans = ~~0~~ ~~1~~ 2 ✓