Binary Search (Algorithm)

$$\begin{array}{ccccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ \text{int[] arr} = \{ & 1, & 3, & 7, & 10, & 11, & 14, & 20, & 40 \} \end{array}$$

target = 14

$\xrightarrow{\hspace{6cm}}$ sorted

Brute force

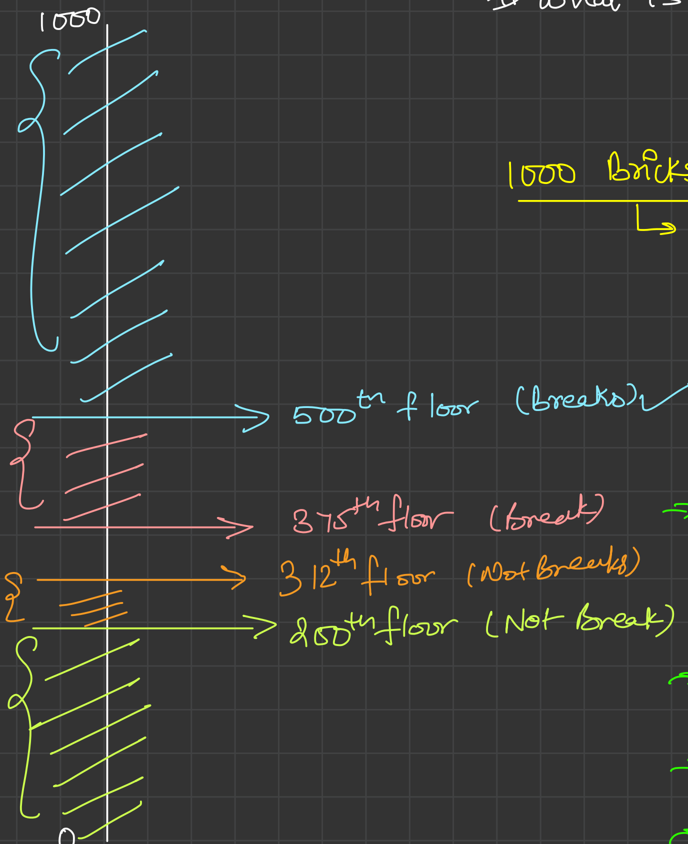
```
for (int i=0 -> n)
{
    if (arr[i] = = target)
        return i;
}
```

linear Search

$\boxed{\begin{array}{l} TC: O(N) \\ SC: O(1) \end{array}}$

what is the lowest floor from which if you throw a brick, it breaks.

1000 Bricks
↳ Brute force



→ By using only one Brick, I eleminated around 500 floors.

→ By using 2nd Brick, I eleminated 250 more floors

→ By using 3rd Brick, I eleminated 125 more floors

→ By using 4th Brick, I eleminated 62 more floors

⋮

1^{st}	2^{nd}	3^{rd}		k^{th}
$\frac{N}{2}$	$\frac{N}{4}$	$\frac{N}{8}$	- - - - - - . . . - -	1
$=$	$=$	$=$		$=$
$\frac{N}{2^1}$	$\frac{N}{2^2}$	$\frac{N}{2^3}$		$\frac{N}{2^K}$

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

$$N = 2^K$$

taking log Base 2 Both Side

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

$$\log_2 N = K \cancel{\log_2 2}^{1}$$

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

Hence, I will require

$$\log_2 (1000) \text{ Bricks}$$

$$3 \log_2 10 \quad (3.1)$$

$$\approx 9.3 \approx \boxed{10 \text{ Bricks}}$$

Binary Search

$$\begin{array}{cccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array}$$

int[] arr = { 1, 3, 7, 10, 11, 14, 20, 40 } target = 11

si mid ei

Sorted (asc)

while(si <= ei)

defined search region

{
int mid = (si+ei)/2;

if (arr[mid] == target)
    return mid;

else if (arr[mid] < target)
    si = mid+1;

else [arr[mid] > target]
    ei = mid-1;
}

TC : $O(\log_2 N)$

SC : $O(1)$

Binary Search

It is always implemented over a $\boxed{\text{sorted region}}$

→ sorted region

→ TC: $O(\log_2 N)$ (Expected)

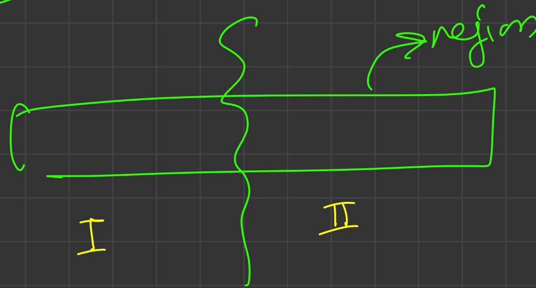
} 99% chances

→ **Binary Search!**

Binary Search

0/1

two options to search



You eliminate one part and take another.

Why we don't prefer recursive approach of BS?

Q Search insert position.

$$arr[] = \begin{array}{ccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ \{1, & 3, & 7, & 10, & 11, & 20, & 40\} \end{array}$$

key = 2

Brute force.

```
{ for(int i=0 ---> n)
{        if (arr[i] >= key) return i;
  return N;
```

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

Actually we are finding just greater value than key

ceil value of key

0 1 2 3 4 5 6

arr[] = { 1 , 3 , 7 , 10 , 11 , 20 , 40 }

key = 2

si

ceil = ?

ei

pceil = ~~10~~ 3

if (arr[mid] == key)
    return mid;

else if (arr[mid] > key)
    pceil = arr[mid];
    ei = mid-1;

else
    si = mid+1;

$\boxed{\text{default value of ceil pos} = N}$

find first and last Pos. of a element

{
inc array

non-dec. array

→ we have duplicates

0	1	2	3	4	5	6	7	8	9	10	11	12
1	2	2	2	2	2	3	4	4	10	20	30	30

int[] arr = { 1, 2, 2, 2, 2, 2, 3, 4, 4, 10, 20, 30, 30 }

ele = 2

first Occ = 1   Last Occ = 5

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, 10, 20, 30, 30 };   ele = 2

first Occ

ei

si

mid

Case   arr[mid] == ele
fo = mid;
ei = mid - 1;

Case   arr[mid] > ele
ei = mid - 1;

Case   arr[mid] < ele
si = mid + 1;

Square Root

int N

↳ sqrt?

Brute force

(1) for(i = 1 ⟶ n − 1)

↳ if i * i == N

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

(2) for(int i = 1 ; i * i <= n ; i++)

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

Square root N

$1 \; - - - - - - - - - \; N$ } → search region

si ↑ … ↑ ei

Case:

mid * mid == N

↳ return mid

$\boxed{20}$

↳ 4

Case:

mid * mid > N

↳ ei = mid-1;

Case:

mid * mid < N → store pfloor

↳ si = mid+1;

$\log_2 N$

$\log_2 N^2$

$\rightarrow 2\log_2 N$

$\log_2 2N$

$\rightarrow \cancel{\log_2 2}^{\,1} + \log_2 N$

$\log_2 N/2$

$\rightarrow \log_2 N - \cancel{\log_2 2}^{\,1}$

$=$

Search in a sorted 2D Matrix

int[][] arr = $\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 \end{bmatrix}$

target = 12

4 X 5

Bruteforce

1. iterate over each ele, of the 2D Matrix

TC: $O(N^2)$ SC: $O(1)$

int[][] arr = $\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 \end{bmatrix}$ → find ceil

target = 12

$\frac{4 \times 5}{N \times M}$

TC: $O(\log_2 N \times M)$

SC: $O(1)$

target = 12

$$\text{int}[\,][\,]\ arr = \begin{array}{c|ccccc} & 0 & 1 & 2 & 3 & 4 \\ \hline 0 & {}^{0}1 & {}^{1}2 & {}^{2}3 & {}^{3}4 & {}^{4}5 \\ 1 & {}^{5}6 & {}^{6}7 & {}^{7}8 & {}^{8}9 & {}^{9}10 \\ 2 & {}^{10}11 & {}^{11}12 & {}^{12}13 & {}^{13}14 & {}^{14}15 \\ 3 & {}^{15}16 & {}^{16}17 & {}^{17}18 & {}^{18}19 & {}^{19}20 \end{array}$$

4×5

$si = 0 \qquad ei = N \times M - 1$

int $\underline{\underline{mid}}$

$$\left.\begin{aligned} &\text{int } r = mid / M \\ &\text{int } c = mid \,\%\, M \end{aligned}\right\} O$$