Binary Search

- Searching Algorithm

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

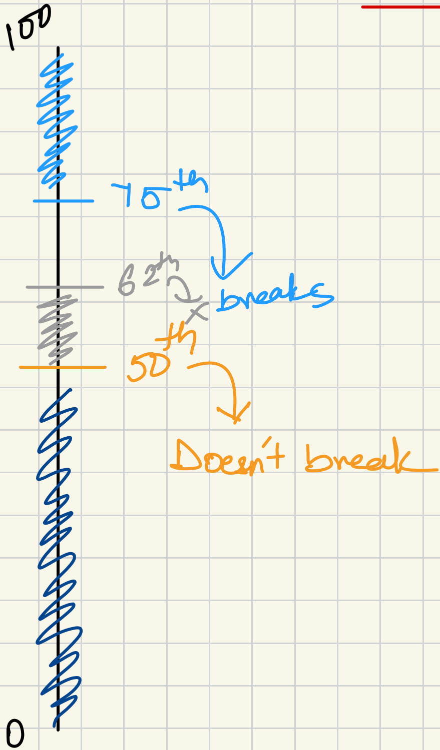
- ## Linear Search

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

TC: $O(n)$

SC: $O(1)$

Puzzle



min. floor from which brick will break

NOTE: single brick can't be used again,
use min no. of 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.

Suppose, $N \rightarrow$ floors

throw	1^{st}	2^{nd}	3^{rd}	. . . . .	K^{th}
eliminated	$N/2$	$N/4$	$N/8$		1
	$\frac{N}{2^1}$	$\frac{N}{2^2}$	$\frac{N}{2^3}$		$\frac{N}{2^K}$

$$\log_2(100) = 2\,\cancel{\log_2 10}^{\,3.1} = \underline{\underline{7}}$$

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

$$N = 2^k$$

taking $\log_2$ Both Sides

$$\log_2(N) = \log_2(2^k)$$

$$\log_2(N) = k\,\cancel{\log_2 2}^{\,1}$$

$$\boxed{k = \log_2(N)}$$

Hence,

No. of throws made $\log_2(N)$

0 1 2 3 4 5 6 7

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

lo   mid   hi

←   →

$x < mid$ $x > mid$

0 ~~1~~ 2 3 ~~4~~ 5 6 7

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

lo mid hi

0 1 2 3 4 5 6 7

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

lo hi

mid

found

Time complexity = $O(\log_2 N)$

Space Complexity = $O(1)$

Binary Search :

- define region of search
- Calc mid , and divide region into 2 half
- try to eliminate one half , and repeat all the steps until answer is found

Binary Search

region is sorted

expected

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

99% of chances BS Ques.

Search insert Position / Ceil value / find just greater

0	1	2	3	4	5	6	7

int[] arr = { 1 , 3 , 7 , 10 , 11 , 12 , 15 , 19 }

Sorted Array

Key = 2

Brute force

Linear Search : TC : O(N)

SC : O(1)

$$
\begin{array}{ccccccccc}
 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
arr = \{ & 1, & \cancel{3}, & \cancel{7}, & \cancel{10}, & \cancel{11}, & \cancel{12}, & \cancel{15}, & \cancel{19} \;\}
\end{array}
$$

key = 2

lo

hi

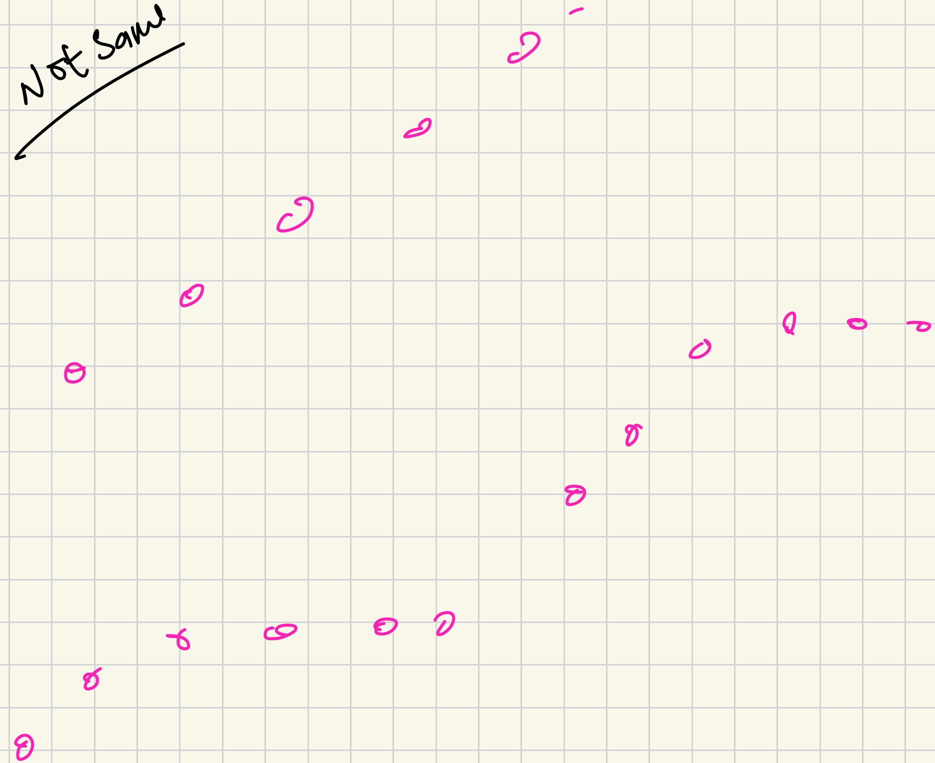
mid

pans = $\cancel{3}$ 1 ✓

① inc. array

② non dec. array

} Not Same



find first and last occ. of an Element

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

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

ei

si

mid

first Occ.

↓

~~2~~

1

Square Root.

→ int x

↓

sqrt(x)

Case : perfect sqare → find sqrt $\sqrt{x}$

Case : not perfect square = floor($\sqrt{x}$)

eg

$x = 36$

$sqrt = 6$

$x = 40$

$sqrt = \underline{6.67}$

$\downarrow$

6

$x = 110$

$sqrt = 10$

$x = 81$

$sqrt = 9$

Brute force.

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

TC: $O(x)$
SC: $O(1)$

```
for(int i=1; i*i <= x; i++)
{
    pans = i;
}
return i;
```

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

SC: $O(1)$

Better TC: $O(x) > O(\sqrt{x}) > O(\log_2 x)$

$n < 10$

0 ———— 1 ———— 10

~~1~~ ~~2~~ 4

~~5~~ ~~2~~ ~~3~~ ~~4~~

$x = \cancel{10}$ ~~4~~ 3

$\boxed{\text{ans} = \cancel{2}\ 3}$

TC: $O(\log_2 x)$

$\sqrt{x}$

$\log_2 x$

Suppose $x = 10000$

$\log_2(10000)$

$\sqrt{10000} = \boxed{100}$ ✓

$\log_2 10^4 = \cancel{4\log_2 10} \rightarrow 13.1$

≈ 13