* Binary Search *

→ Linear Search → T.C = O(n)

* Binary Search :

→ Applied only on monotonic functions.

↳ value should be in increasing or decreasing functions.

Algo: start = 0; end = n-1; mid = $s + \frac{(e-s)}{2}$

while s <=

1. find mid
2. compare with key. if found then return else--
3. compare greater or less than

(mid > key) ↙ e = mid - 1, go to step 1

(mid < key) ↘ s = mid + 1, go to step 2.

Q: | 3 | 5 | 9 | 13 | 27 | key = 13.

Ans. start = 0 ; e = n-1; mid = $s + \frac{(e-s)}{2}$ = 2

(9 < 13)

arr[mid] < key

$\therefore$ $s = mid + 1$

↳ $mid = s + \frac{(e-s)}{2} = 3$

$\therefore$ $arr[mid] = key$ $(9 = 13)$ } → return the ans that key is found.

Q. | 3 | 7 | 11 | 13 | 19 | 27 | key = 27

Ans $s = 0$; $e = n-1 = 5$; $mid = s + \frac{(e-s)}{2} = 2$

→ $arr[mid] < key$.

$\therefore$ $s = mid+1$; $mid = s + \frac{(e-s)}{2} = 4$

→ $arr[mid] < key$

$s = mid+1$; $mid = 5$

→ $arr[mid] == key$. → cout << "found" << endl;

T.C => ① n size

② n/2 size

n/4

n/8

n/16

⋮

0

length = 1

$n + n/2 + n/4 + n/8 + n/16 + \cdots + \frac{n}{2^k} =$

$O(\log_2(N))$

$\therefore \frac{n}{2^k} = 1$

$n = 2^k$

$k = \log_2(N)$