Linear Search: Works on any kind of arrays.
TC: $O(n)$

Binary Search: Can search for a specific element only in sorted arrays.
TC: $O(\log n)$

0	1	2	3	4	5	6
1	2	5	8	10	15	20

find = $\boxed{15}$

low = ~~0~~ 4
high = 6

$mid = \frac{low+high}{2} = 3$

$mid = \frac{4+6}{2} = 5$

$\boxed{5}$ ✓

$[O(1)]$

$n \to \frac{n}{2} \to \frac{n}{4} \cdots\cdots \to 1 \Big] \Rightarrow 1 \to 2 \to 4 \cdots\cdots \to \frac{n}{2} \to n$

first term = 1
mult. factor = 2
$\Rightarrow$ no. of terms = K

$1 \cdot 2^{k-1} = n \Rightarrow k-1 = \log_2 n$

$\boxed{k = O(\log n)}$

~~$\log_{10} n$~~ ~~$\log_{10}$~~

$a \cdot r^{k-1}$

0	1	2	3	4	5	6	7	8
1,	3,	5,	5,	5,	5,	67,	123,	125

, x = 5

first_occ = 2
last_occ = 5

$[\frac{n}{2} \Rightarrow O(n)$

0	1	2	3	4	5	6	7	8
1,	3,	5,	5,	5,	5,	67,	123,	125

x = 5

low = ~~0~~ 2
high = ~~8~~ ~~3~~ 1

mid = ~~4~~ ~~1~~ 2

first_occ = ~~-1~~ ~~4~~ 2

Performing the normal BS but in case arr[mid] == x, we update the first_occ & discard the right half.

$mid = \frac{(low + high)}{2}$

$\left[mid = low + \frac{(high - low)}{2} \right]$

$n \le 10^5$
$n = 10^{10}$

N = 25 largest 'mid' s.t. $mid * mid \le N$

4 → 2
~~5~~ → 2
9 → 3

low = 0
high = 25/2 = 12

$mid: \frac{0+12}{2} = 6$

6 * 6 = 36

low = 0
high: 5

$mid: \frac{0+5}{2} = 2$

2 * 2 = 4

result = 2

low : 3
high = 5

$mid = \frac{3+5}{2} = 4$

4 * 4 = 16

result = 4

low : 5
high = 5

$mid: \frac{5+5}{2} = 5$

5 * 5 = 25

$\boxed{result : 5}$

low = 6
high = 5] low > high ⇒ STOP

N = 30 Find the largest 'mid' s.t. $mid^2 \le N$

low = 0
high: 15

mid: 7

7 * 7 = 49 > N ⇒ Disc. the right half

low : 0
high: 6

mid = 3

3 * 3 = 9 ≤ N ⇒ result : 3
Disc. the left half

low = 4
high = 6

mid = 5

5 * 5 : 25 ≤ N ⇒ $\boxed{result = 5}$
Disc. the left half

low : 6
high : 6

mid = 6

6 * 6 : 36 > N ⇒ Disc. the right half

low: 6
high = 5] low > high ⇒ Sol.

0 ... 6

---

n-1

[10, 20, 10, 30] M = 2

Total no. of permutations = ${}^{n-1}C_M$

$O({}^{n-1}C_n \times n)$

$\boxed{\text{We will perform B.S. on the result.}}$

[10, 20, 10, 30] M = 2

low = 30 , high = ~~60~~ 70

[Find the no. of students req. s.t. no student reads more than 'mid' no. of pages]

(n = N)
low = 30

(M = 1)
high = ~~60~~ 70

$mid: \frac{30+70}{2} = \underline{50}$

$\boxed{M}$

[~~60~~ 51, 52]

[10, 20, 10, 30]
1 1 1 2

$\boxed{result = 50}$

⇒ Disc. the right half.

low = 30 high = 49

$mid = \frac{30+49}{2} = 39$

[10, 20, 10, 30]
1 1 2 3

3 > M

⇒ Disc. the left half.

low: 40 high = 49

$mid = \frac{40+49}{2} = 44$

[10, 20, 10, 30]
1 1 1 2

$\boxed{result = 44}$

⇒ Disc. the right half

low = 40 high: 43

$mid = \frac{40+43}{2} = 41$

[10, 20, 10, 30]
1 1 1 2

$\boxed{result = 41}$

⇒ Disc. the right half

low = 40 high: 40

$mid: \frac{40+40}{2} : 40$

[10, 20, 10, 30]
1 1 1 2

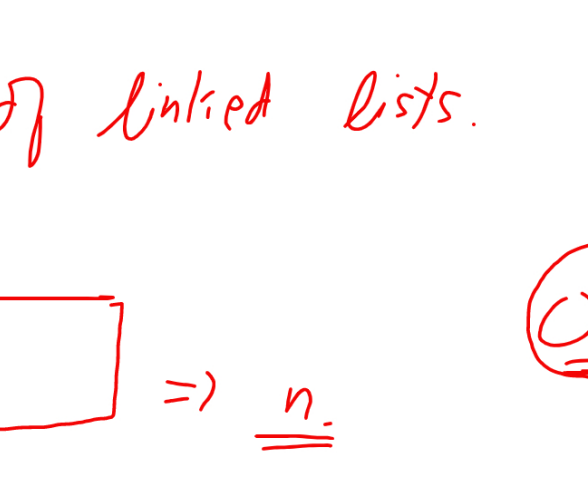
$\boxed{result = 40}$

⇒ Disc. the right half

low = 40 high = 39 ⇒ STOP

result : 40

TC: $O(\log(sum) * n)$



⇒ $O(\log(\text{no. of elements in the initial range}))$

no. of elements in the initial range: [sum − max element + 1]

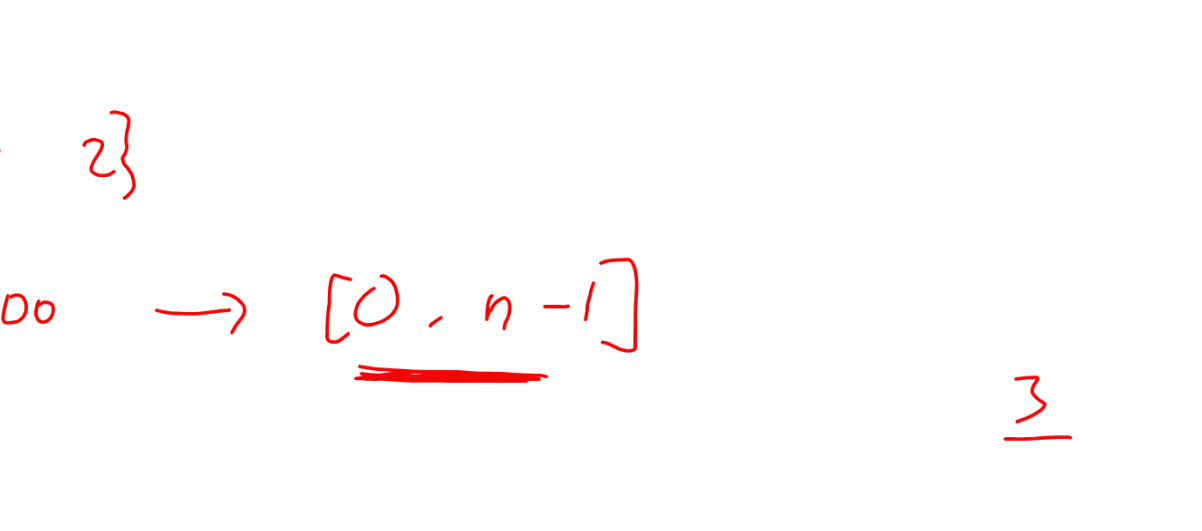
Hash Tables:

keys → 0 1 2 3
values → | 7 | 5 | 1 | 3 | } Arrays

$\boxed{\Rightarrow \{key, value\}}$

{1, 2, 3, −1, 3 - - }

↳ Internally is an array of linked lists.



$\textcircled{O(1)}$

⇒ Take a key → Map the key to an index in the array → add the {key, value} at that index.

Mapping is done by a Hash functions.

#1: For good implementations of hash functions we assume that the mapping of an integer takes $O(1)$ time.

{100, 2}
key = 100 → [0, n − 1]

Insert : $O(1)$

3

~~#3~~: [No. of keys are infinite but the size of the array is always finite. Therefore, we maintain an array of linked lists.]

#2: For a good hash function, no. of collisions is minimal.

Based on assumption #2, we can say that:
Search : $O(1)$ time.
Delete: $O(1)$ time (as deletion from any pos. ~~index~~ of a linked list takes $O(1)$ time)

8 0 8 16 24 ....

"abcd" (0 1 2 3) ⇒ $[97 * 37^0 + 98 * 37^1 + 99 * 37^2 + 100 * 37^3] \% -$

O(len) O(1).

[4, 2, −3, 1, 6] → True

[1, 2, 0, 3] → True

[−1, −2, 7, 8] → False.

```
for (i : 0 → n-1) {
  sum = 0
  for (j : i → n-1){
     sum += arr[j]
     if(sum == 0) return True;
  }
}
```

$[O(n^2)]$

{i, j}

0	1	2	3	4
4,	2,	−3,	1,	6

i = ~~0~~ 1
j = ~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ ~~2~~ 3

sum = ~~0~~ ~~4~~ ~~6~~ ~~3~~ ~~4~~ ~~10~~ ~~0~~ ~~2~~ ~~-1~~ 0 ⇒ True.

TC: $O(n^2)$
Aux. Space: $O(1)$

(sum till here)

	0	1	2	3	4
	4,	2,	−3,	1,	6
sth :	4	6	3	4	10

4 → 0
6 → 1
3 → 2

{i+1, j} → 0

{int → Bool}
[4 → True]
6 → True
3 → True

	0	1	2	3	4
	1,	−1,	3,	7,	8
sth :	1	0	3	10	18

⇒ [1, 0, 0, 1, 0, 1, 1]

↳ [1, −1, −1, 1, −1, 1, 1] ⇒ $\boxed{\text{no. of subarrays with zero sum.}}$