ARRAY

↳ linear data structure that stores elements in a contiguous manner.

0	1	2	3	4	5
1	3	2	5	10	11
x	x+4	x+8	x+12	...	

If 0th element is at memory location 'x', then any i^{th} element will be at $(x + 4i)$

⇒ If we want to access any random element in the array, it takes O(1) time to do that.

In arrays: Random access takes O(1) time.

Hashmap-based Soln

⇒ [0, -10, 1, 3, -20]

0 → true
-10 → true
1 → true ✓
3 → true
-20 → true

i = ~~2~~ 2

TC: O(n) ✓
Aux space: O(n)

[0, -10, 1, 3, -20]

max_elt = 3

Sorting-based: [-20, -10, 0, 1, 3]

find = ~~x~~ 2 ⇒ $\boxed{\text{return } 2}$;

[TC: O(n log n)
Aux. space: O(1)]

[GOAL: TC: O(n)
Aux. space: O(1)]

⇒ ~~x~~ [0, -10, 1, 3, -20]

For any array of size-n, answer lies in range [1, n+1]

0 1 2 3 4

n = 5, arr: [-3, -1, -5, 7, -2]

1✓ 2✓ 3✓ × 4 ×

4-1 = 3

⇒ [We will use the index-i as a BUCKET to store the info that whether element = (i+1) occurs in the array or not.]

i+1 = 3 i = 2

[The info that whether arr[i] is present or not will be stored at INDEX = (arr[i]-1)]

0 1 2 3 4

[~~0~~, ~~-10~~, 1, 3, ~~-20~~] n=5 [1, 6]

7 7 7

[7, 7, 1, 3, 7]

[7, 7, -6, 3, 7]

$\boxed{3}$ [1, 2, 3] [1, 2, 4] [1, 3]

1 ×

$\boxed{n=5}$ [1, 5]

[1, n] n+1

[~~700~~, 900, 500]

1✓ 2✓ 3✓ 4✓ 5✓

0 1 2 3 4

[-1, -2, -3, -4, -5]

⇒ return (n+1)

[1, n] (n+1)

[+1, 1]

[3 3 ...]

[~~-1~~, 1]

element = 1
bucket_index = 0

1✓ 2✓ 3×

[-1, -1, 2]

$$\begin{bmatrix} 1 & \boxed{0} & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \Rightarrow \begin{bmatrix} 0 & 0 & \boxed{0} \\ 1 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & \boxed{0} & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} \Rightarrow n \begin{bmatrix} 2 & 2 & 2 \\ 1 & 2 & 1 \\ 1 & \boxed{0} & 1 \end{bmatrix} \Rightarrow \begin{bmatrix} 2 & 2 & 2 \\ 1 & 2 & 1 \\ 2 & 2 & 2 \end{bmatrix}$$

⇓

ans ⇒ $\begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$

TC: O(nm (n+m))

$$\begin{array}{c|ccc} & 0 & 1 & 2 \\ 0 & 1 & \boxed{0} & 1 \\ 1 & 1 & 1 & 1 \\ 2 & 1 & \boxed{0} & 1 \end{array}$$

[rows: [T~~F~~, F, T~~F~~]
cols: [F, T~~F~~, F]] ×

$$\begin{array}{c|ccc} & 0 & 1 & 2 \\ 0 & \cancel{1}0 & 0 & \cancel{0}1 \\ 1 & 1 & \cancel{1}0 & 1 \\ 2 & \cancel{1}0 & 0 & \cancel{1}0 \end{array} \Rightarrow \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

TC: O(nm)
Aux. space: O(n+m)

$$\begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 2 & 1 & 1 & 1 & 1 \\ 3 & 1 & 1 & 1 & 0 \end{array}$$

$$\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ 0 & 1 & \cancel{1}0 & 1 & \cancel{1}0 \\ 1 & \cancel{1}0 & 0 & \cancel{1}0 & \cancel{1}0 \\ 2 & 1 & \cancel{1}0 & 1 & \cancel{1}0 \\ 3 & \cancel{1}0 & \cancel{1}0 & \cancel{1}0 & 0 \end{array} \Rightarrow$$

$$\begin{bmatrix} \cancel{1} & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 0 & 1 & \cancel{1}0 \\ 1 & \cancel{1}0 & 1 & 0\cancel{1} \\ 0\cancel{1} & \cancel{1}0 & \cancel{1}0 & 0 \end{bmatrix}$$

⇓

$$\begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

⇓

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$ ✓

[zeroth_row = True
zeroth_col = False]

KADANE'S ALGO

0 1 2 3 4 5

arr: [-5, 1, 3, -7, 5, 2]

subarrays: [-5], [-5, 1], [-5, 1, 3], [-5, 1, 3, -7] ...

[3, -7] [3, -3, 5, 2] ...

```
max_sum = -∞
for (int i=0; i< n; i++) {
    int cur_sum = 0
    for (int j=i; j<n; j++) {
        cur_sum += arr[j];
        if (cur_sum > max_sum) {
            max_sum = cur_sum;
        }
    }
}
```

[i ... j]

TC: O(n^2)
Aux. space: O(1)

[1, 7, -6, 3, 2, 1, 1]

result

	0	1	2	3	4	5
arr:	-5	1	3	-7	5	2
max_ending here: meh	-5	1	4	-3	5	7

max of all = $\boxed{7}$ ✓

① meh[i] = meh[i-1] + arr[i] if meh[i-1] > 0 ✓

② meh[i] = arr[i] if meh[i-1] < 0 ✓

	0	①	2	3	4	5
arr:	-5	1	3	-7	5	2
meh:	-5	1	4	-3	$\boxed{5}$	⑦

BINARY SEARCH

0	1	2	3	4	5	6
1	3	5	6	10	12	15

find 10 = k

low = 0
high = 6 ⇒ mid = $\frac{0+6}{2}$ = 3

→ arr[mid] < k ⇒ low = mid+1 = 4

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

→ arr[mid] > k ⇒ high = mid-1 = 4

low = 4
high = 4 ⇒ mid = $\frac{4+4}{2}$ = 4

→ arr[mid] == k ⇒ return mid

O(1)

1, 2, 4, 8 ......, n

(n → $\frac{n}{2}$ → $\frac{n}{4}$ → $\frac{n}{8}$ ---- → 1)

(1, 2, 4, 8 ---- $\frac{n}{4}$, $\frac{n}{2}$, n)

no. of terms = m

ar^{m-1}

GP: a=1, r=2

m^{th} term = ar^{m-1}

1st term = ar^{1-1} = a = 1
2nd term = ar^{2-1} = $a.r^1$ = 2
3rd " = ar^{3-1} = ar^2 = 4

ar^{m-1} = n

$1 \cdot \frac{2^m}{2}$ = n ⇒ 2^m = 2n

m = O($\log_2(n)$)

⇒ TC of BS = O(log(n))

0 1 2 3 4 5 6 7 8

[1, 3, 5, 5, 5, 5, 67, 123, 125], k=5

⇒ first = ~~4~~ ~~4~~ 2
last = -1

low = 0
high = 8 ⇒ mid = 4

arr[mid] == 5
⇒ first = mid = 4
⇒ high = mid-1 = 3

low = 0
high = 3 ⇒ mid: $\frac{0+3}{2}$ = 1

arr[mid] < 5
low = mid+1 = 2

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

arr[mid] == 5
⇒ $\boxed{\text{first} = \text{mid} = 2}$
high = mid-1 = 1

low = 2
high = 1 ⇒ STOP!