[1, 2, 3, 5, 6, 7, 8, 9, 10]

[2, 3, 5, 6, 7, 8, 9, 10, 1]

[3, 5, 6, 7, 8, 9, 10, 1, 2]

[5, 6, 7, 8, 9, 10, 1, 2, 3] ✓

$O(n\log n + \log n) = O(n\log n)$

$\boxed{O(n)}$

0 1 2 3 4 5 6 7 8

[5, 6, 7, 8, 9, 10, 1, 2, 3]   key = 10

0 1 2 3 4

X [5, 1, 2, 3, 4]   Key = 5

0 1 2 3 4 5 6 7 8

⇒ [5, 6, 7, 8, 9, 10, 1, 2, 3]

[0, 0] ✓	[0, 6] X
[0, 1] ✓	[0, 7] X
[0, 2] ✓	[0, 8] X
[0, 3] ✓	
[0, 4] ✓	
[0, 5] ✓	

0 1 2 3 4 5 6 7 8

key = 7 [5, 6, 7, 8, 9, $\boxed{10}$, 1, 2, 3] $\boxed{\text{Key} = 10}$

(9 > arr[0]) ⇒ left half is sorted

Key is greater than the max^m of sorted half

⇒ Key does not occur in the left half.

⇒ move to the right half.

low = 5, high = 8

mid = $\frac{5+8}{2}$ = 6

(1 < arr[0]) ⇒ left half is not sorted

⇒ Right half is sorted.

But key does not lies in the range of sorted half

⇒ Discard the sorted half (right half)

⇒ low = 5, high = 5

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

low = 0	high = 6 ⇒ mid = 3
low = 0	high = 2 ⇒ mid = 1
low = 0	high = 0

A = [10, 20, 10, 30]

$\boxed{M = 2}$

For M = 4   [10] [20] [10] [30] ⇒ (30)

For M = 1   [10, 20, 10, 30] ⇒ (70)

lower-limit of ans = 30

upper-limit of ans = 70

⇒ Now, since we have a range in which the ans could lie, we will apply the Binary Search on the output (ans)

result = ~~50~~ ~~44~~ ~~41~~ 40

① low = 30   mid = $\frac{30+70}{2}$ = 50
high = 70

Find the no. of students such that no student reads more than 'mid' no. of pages.

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

⇒ We need 2 students such that none of them reads more than 50 pages.

⇒ Since (2 ≤ M) ⇒ any value greater than or equal to 50 is a valid probable output.

(51, 52 ... ~~100~~ 60)

⇒ Discard the right half.

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

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

⇒ Since $\boxed{3 > M}$, anything less than or equal to 39 is not a valid probable output.

⇒ Discard the left half.

③ low = 40   mid = $\frac{40+49}{2}$ = 44
high = 49

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

⇒ Since (2 ≤ M), anything greater than or equal to 44 are valid probable outputs

⇒ Discard the right half.

④ low = 40   mid = $\frac{40+43}{2}$ = 41
high = 43

[10, 20, 10, 30] ✓
  1   1   1   2

⇒ Anything ≥ 41 is a valid prob output.

⇒ Discard the right half.

⑤ low = 40   mid = 40
high = 40

[10, 20, 10, 30] ✓
  1   1   1   2

⇒ Anything ≥ 40 is a valid prob o/p
⇒ Discard the right half.

⑥ low = 40   ⇒ Range exhausted
high = 39   ⇒ return the $\boxed{\text{result} = 40}$

$\boxed{\text{TRY: Painter's Partition Problem}}$

[10, 20, 10, 30]   M = 2   mid = 50
  1   1   1   2

student_count = ~~1~~ 2

pages_count = ~~10~~ ~~30~~ ~~40~~ 30

~~low~~ ~~high~~

high = sum

(sum - low)

$\boxed{O(\log(\text{sum}))}$

No. of B.S. iterations = $O(\log(\text{sum}))$

work done in each B.S iteration = $O(n)$

⇒ TC = $O(n * \log(\text{sum}))$

Map
{Key → value}
(Key has to be unique)

Set
{values}
(Each value has to be unique)

If keys are in range [0, n-1]

0	1	2			n-1
-1	-1	-1	— — - - - —		-1

← keys
← values

([0, 10^9])

key	value
{ 10^5,	6 }
{ 10^7,	7 }
{ 7,	8 }
⋮	

10 pairs

→ (10^9+1)

(10^9)   (10^3+1)

Key → $\boxed{f(\text{key})}$ ⟶ new key in range [0, 1000]
↳ Hashing function

↓
(10^5, 7)
(10^5, 8)

0 → {} → {} → {} → {}
1 → {} → {} → {} → {} → {}
→ 2 → {} → {} → {} → {}
3 →
4 →
⋮
1001 →

(Linked lists have a searching time of O(len))

For internal implementation of Hash tables, we assume that the hash-function will be good enough and will divide the (key-value) pairs almost evenly throughout the hash-table.

Also, the internal implementation is self-adaptive based upon the (key-value) pairs it gets

---

[4, 2, -3, 1, 6]   [4, 2, -6]

([i, j])

```
for (int i=0; i<n; i++) {
        cur_sum = 0;
        for (int j = i; j<n; j++) {
                cur_sum += arr[j];
                if (cur_sum ==0) return true;
        }
}
return false;
```

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

0 1 2 3 4

arr: [4, 2, -3, 1, 6]

pref_sum: [4, 6, 3, 4, 10]

S_1 S_2

pref_sum [i] = arr[0] + arr[1] ... + arr[i]

[1, 3] == 0.

$S_1 = S_2$

$\Rightarrow \boxed{S_2 - S_1 = 0}$

arr: [1]
PS: [1]

{int → Boolean}

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

pref_sum: [1, 0, 2, 5, 9]

[0, 1, 0, 2, 5, 9]

0 1 2 3 4

[1, 2, -2, 2, -2]

pref_sum: [1 3 1 3 1]

result = ~~0~~ ~~1~~ ~~2~~ 3

(1, 2) ✓
(2, 3) ✓
(3, 4) ✓
(1, 4) ✓

0 1 2 3 4

[1, 2, -2, 2, -2]

pref_sum: 1, 3, 1, 3, 1

freq:

key	value
1	→ ~~1~~ ~~2~~ 3
3	→ ~~1~~ 2

$\boxed{\text{result}: ~~0~~ ~~1~~ ~~2~~ 4}$

0 1 2

[0, 0, 0]

(0, 0)   (1, 1)
(0, 1)   (1, 2)
(0, 2)   (2, 2)

0 1 2 3 4 5 6

[1, 0, 0, 1, 0, 1, 1]

↳ Replace all 0's with -1

[1, -1, -1, 1, -1, 1, 1]

↓

Count the subarrays with sum = 0

(0, 1)   (4, 5)
(2, 3)   (2, 5)
(0, 3)   (0, 5)
(3, 4)   (1, 6)