Split Array Largest Sum:-

410. Split Array Largest Sum

Hard · Topics · Companies

Given an integer array `nums` and an integer `k`, split `nums` into `k` non-empty subarrays such that the largest sum of any subarray is **minimized**.

Return the *minimized largest sum of the split*.

A **subarray** is a contiguous part of the array.

Example 1:

```
Input: nums = [7,2,5,10,8], k = 2
Output: 18
Explanation: There are four ways to split nums into two subarrays.
The best way is to split it into [7,2,5] and [10,8], where the largest sum among the two subarrays is only 18.
```

Example 2:

```
Input: nums = [1,2,3,4,5], k = 2
Output: 9
Explanation: There are four ways to split nums into two subarrays.
The best way is to split it into [1,2,3] and [4,5], where the largest sum among the two subarrays is only 9.
```

Constraints:

- 1 <= nums.length <= 1000
- 0 <= nums[i] <= 10^6
- 1 <= k <= min(50, nums.length)

Typical minimum of maximum / maximum of minimum problem.

Binary Search.

eg: nums = [7, 2, 5, 10, 8]
k = 2

① [7] [2, 5, 10, 8] → 25
② [7, 2] [5, 10, 8] → 23
③ [7, 2, 5] [10, 8] → 18 ← minimum
④ [7, 2, 5, 10] [8] → 24

Let's try to build intuition:-

range:- max(nums) → sum(nums)

Let S be the maximum sum of any subarray, check if total subarrays == 2.

eg in [7, 2, 5, 10, 8]

lo = 10 hi = 7 + 2 + 5 + 10 + 8 = 32

mid = 21 18 → range (18 − 31)

21 7, 2 5 (7, 2, 5)(10, 8)

hi = mid

23 24 21 25

14, 10, 8

18 21 19 18, 19

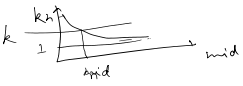
15, 2, 18, 21

18

mid ↑ k ↓

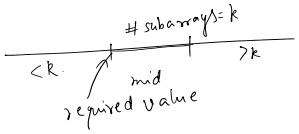
mid ↓ k ↑

k



mid:

There will be a range of mid where # subarrays = k




```
while(lo<hi){
if(test(mid)>k) hi = mid
else lo = mid+1;
}
return lo
```

// debugging

eg a = [1, 1]
k = 1

lo = 1, hi = 2.

① mid = 1

[1, 1] 1
↑
[1, 1] 1+1 => 2 1
↑ ans = 1

return ans = 2;

test > mid

lo = 2

Solution Code & Result :-