Leetcode 410 – Split Array Largest Sum

# Problem Title:

410. Split Array Largest ≠Sum

🔗 [Leetcode Link](https://leetcode.com/problems/split-array-largest-sum/)

### Problem Statement:

Given an integer array nums and an integer k, split nums into k non-empty **continuous subarrays** such that the **largest sum among these subarrays is minimized**.

Return the minimum possible value of this **largest sum**.

### Input:

* nums = [7,2,5,10,8]
* k = 2

### Output:

* 18

### Key Insight:

We can **binary search** on the **answer** (i.e., the maximum subarray sum).

* **Lower bound (left)** = max element in nums → must be at least this big
* **Upper bound (right)** = sum of all elements → worst case: only 1 subarray

We binary search for the minimum possible **maximum subarray sum** such that we can split the array into ≤ k parts.

### Java Code:

class Solution {

public int splitArray(int[] nums, int k) {

int left = getMax(nums); // minimum possible max sum

int right = getSum(nums); // maximum possible max sum

while (left < right) {

int mid = left + (right - left) / 2;

if (canSplit(nums, k, mid)) {

right = mid;

} else {

left = mid + 1;

}

}

return left;

}

// Check if we can split into ≤ k subarrays with each subarray sum ≤ maxSum

private boolean canSplit(int[] nums, int k, int maxSum) {

int count = 1;

int currentSum = 0;

for (int num : nums) {

if (currentSum + num > maxSum) {

count++;

currentSum = num;

if (count > k) return false;

} else {

currentSum += num;

}

}

return true;

}

private int getMax(int[] nums) {

int max = nums[0];

for (int num : nums) {

max = Math.max(max, num);

}

return max;

}

private int getSum(int[] nums) {

int sum = 0;

for (int num : nums) sum += num;

return sum;

}

}

### Dry Run (with nums = [7,2,5,10,8], k = 2)

|  |  |  |  |
| --- | --- | --- | --- |
| Iteration | mid | canSplit(nums, 2, mid) | Action |
| 1 | 21 | ✅ yes (split into [7,2,5] and [10,8]) | right = 21 |
| 2 | 17 | ❌ no | left = 18 |
| 3 | 19 | ✅ yes (split into [7,2,5], [10,8]) | right = 19 |
| 4 | 18 | ✅ yes (same split) | right = 18 |
| 5 | 18 | Exit loop (left == right) | ✅ Return 18 |

### Final Split:

[7,2,5] → sum = 14  
[10,8] → sum = 18

Maximum subarray sum = 18 → **minimum possible** maximum

### Time Complexity:

* O(N \* log(S))  
  Where:
  + N = length of array
  + S = sum of all elements - max of array