410. Split Array Largest Sum

QuestionEditorial Solution

My Submissions

- Total Accepted: **2201**
- Total Submissions: **8028**
- Difficulty: **Hard**
- Contributors: **Admin**

Given an array which consists of non-negative integers and an integer *m*, you can split the array into *m* non-empty continuous subarrays. Write an algorithm to minimize the largest sum among these *m* subarrays.

Note:

Given *m* satisfies the following constraint: 1 ≤ m ≤ length(nums) ≤ 14,000.

```
Input:
nums = [7,2,5,10,8]
m = 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.
```

```
int splitArray1(vector<int>& nums, int m, int bound)
{
    int ret = INT_MIN;
    int now = 0, cnt = 0;
    for (int i=0;i<nums.size();i++)
    {
      int x = nums[i];
       if (x > bound) return INT_MAX;

       if (now + x > bound)
       {
           ret = max(ret, now);
           now = x;
           cnt++;
           if (cnt >= m) return INT_MAX;
       }
       else
       {
           now += x;
           if (now <= bound) ret = max(ret, now);
       }
    }
```

```
    return ret;
}

int splitArray1(vector<int>& nums, int m) {
    long long sum = 0;
    for(int i=0;i<nums.size();i++)
        {
            sum += nums[i];
        }

    long long l = 0, r = sum;
    while (l < r)
    {
        int mid = l + (r - l) / 2;
        int t = splitArrayInBound(nums, m, mid);

        if (t > mid)
        {
            l = mid + 1;
        }
        else
        {
            r = mid;
        }
    }

    return l;
}

int main(int argc,char *argv[])
{
       vector<int> arr;
       int nums[] = {7,2,5,10,8};
       for(int i=0;i<5;++i) arr.push_back(nums[i]);
       int ans = splitArray1(arr,2);
       cout << ans << endl;
       return 0;
}
```