**Approach : Binary Search**

1. Left = max(arr) -> largest sum that can be possible ranges from largest value in the array till the sum of array
2. Right = sum(arr)
3. While left <= right:
   1. Mid = left + (right-left) // 2
   2. Check if mid is a valid value, that is check if we can form exactly k subarrays by allowing sum of each subarray to not exceed ‘mid’
   3. If possible, save the value and try to minimize the value: left = mid-1
   4. Otherwise, maximize the value so a valid subarray list is formed.
4. Return res

**CODE:**

def isValid(self,nums,midMin,k):

curSum = 0

noOfSubarr = 1

for num in nums:

curSum += num

if curSum > midMin:

# reset curSum

curSum = num

noOfSubarr += 1

if noOfSubarr > k:

return False

return True

def splitArray(self, nums: List[int], k: int) -> int:

left = max(nums)

right = sum(nums)-1

res = sum(nums)

while left <= right:

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

if self.isValid(nums,mid,k):

# add to res

res = min(res, mid)

# reduce/minimize value

right = mid -1

else:

#increase capacity sum of each subarray

left = mid+1

return res