Graphical user interface, text, application, email

Description automatically generated

**Solution 1:**

class Solution:

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

        maxVal = self.maxSubArraySum(nums,0,len(nums)-1)

        return maxVal

    def maxSubArraySum(self, num,l,h):

        if(l>h):

            return -10000

        if(l==h):

            return num[l]

        m = (l+h)//2

        return max(self.maxSubArraySum(num,l,m-1),self.maxSubArraySum(num,m+1,h),self.maxCrossingSum(num,l,m,h))

    def maxCrossingSum(self, num,l,m,h):

        sm=0

        left\_sum=-10000

        for i in range(m,l-1,-1):

            sm = sm+num[i]

            if(sm>left\_sum):

                left\_sum = sm

        sm=0

        right\_sum=-10000

        for i in range(m,h+1):

            sm = sm+num[i]

            if(sm>right\_sum):

                right\_sum = sm

        return max(left\_sum + right\_sum - num[m], left\_sum, right\_sum)

demo = Solution()

demo.maxSubArray([-2,1,-3,4,-1,2,1,-5,4])

**Output:**

6

**Time Complexity:** ?

**Space Complexity:** ?

**Solution 2:**

class Solution:

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

        maxSub = nums[0]

        sumSubNum = 0

        for n in nums:

            if sumSubNum < 0:

                sumSubNum = 0

            sumSubNum += n

            maxSub = max(maxSub, sumSubNum)

        return maxSub

demo = Solution()

demo.maxSubArray([-2,1,-3,4,-1,2,1,-5,4])

**Output:**

6

**Time Complexity:** O(n)

**Space Complexity:** O(n)