

# 53-最大子数组和

## 题述

### 53. 最大子数组和

难度 **简单**

4307



给你一个整数数组 `nums`，请你找出一个具有最大和的连续子数组（子数组最少包含一个元素），返回其最大和。

**子数组** 是数组中的一个连续部分。

#### 示例 1:

输入: `nums = [-2,1,-3,4,-1,2,1,-5,4]`

输出: 6

解释: 连续子数组 `[4,-1,2,1]` 的和最大，为 6。

#### 示例 2:

输入: `nums = [1]`

输出: 1

#### 示例 3:

输入: `nums = [5,4,-1,7,8]`

输出: 23

#### 提示:

- `1 <= nums.length <= 105`
- `-104 <= nums[i] <= 104`

## 思路

暴力双循环没啥好说的。

讲讲贪心:

如果 -2 1 在一起，计算起点的时候，一定是从1开始计算，因为负数只会拉低总和，这就是贪心贪的地方！

局部最优：当前“连续和”为负数的时候立刻放弃，从下一个元素重新计算“连续和”，因为负数加上下一个元素“连续和”只会越来越小。

全局最优：选取最大“连续和”

局部最优的情况下，并记录最大的“连续和”，可以推出全局最优。

从代码角度上来讲：遍历nums，从头开始用count累积，如果count一旦加上nums[i]变为负数，那么就应该从nums[i+1]开始从0累积count了，因为已经变为负数的count，只会拖累总和。

看一下代码应该就可以理解了。

## 题解

### 暴力双for

```
class Solution {
public:
    int maxSubArray(vector<int>& nums)
    {
        //暴力
        int result=INT32_MIN;
        int count=0;
        for(int i=0;i<nums.size();i++)
        {
            //设置起始位置
            count=0;
            for(int j = i;j < nums.size();j++)
            {
                // 每次从起始位置i开始遍历寻找最大值
                count += nums[j];
                result =count > result ? count : result;
            }
        }
        return result;
    }
};
```

执行结果： **超出时间限制** [显示详情](#)

[添加备注](#)

最后执行的输入：

[5356,-7142,-3862,-8237,5603,-1209,-5513,4050,637,-3649,-4230,8208,-21... [查看全部](#)

| 提交结果   | 执行用时   | 内存消耗    | 语言  | 提交时间             | 备注                |
|--------|--------|---------|-----|------------------|-------------------|
| 超出时间限制 | N/A    | N/A     | C++ | 2022/02/07 09:22 | <a href="#">P</a> |
| 通过     | 504 ms | 12.7 MB | C++ | 2021/08/27 14:37 | <a href="#">P</a> |

## 贪心

```
class Solution {
public:
    int maxSubArray(vector<int>& nums)
    {
        //贪心
        //局部最优：当前“连续和”为负数的时候立刻放弃，从下一个元素重新计算“连续和”，因为负数加上下一个元素“连续和”只会越来越小。
        //全局最优：选取最大连续和
        //局部最优的情况下，并记录最大的“连续和”，可以推出全局最优
        int result=INT32_MIN;
        int count=0;
        for(int i=0;i<nums.size();i++)
        {
            count+=nums[i];
            if(count > result)
            {
                //取区间累计的最大值（相当于不断确定最大子序终止位置）
                result = count;
            }
            if(count <= 0) count = 0;    //如果连续和变为0，那么重置count 相当于重置最大子序起始位置
        }
        return result;
    }
};
```

执行结果： **通过** [显示详情](#)

[添加备注](#)

执行用时： **84 ms**，在所有 C++ 提交中击败了 **82.15%** 的用户

内存消耗： **66.1 MB**，在所有 C++ 提交中击败了 **59.71%** 的用户

通过测试用例： **209 / 209**

炫耀一下：



[写题解，分享我的解题思路](#)

| 提交结果   | 执行用时   | 内存消耗    | 语言  | 提交时间             | 备注                |
|--------|--------|---------|-----|------------------|-------------------|
| 通过     | 84 ms  | 66.1 MB | C++ | 2022/02/07 09:39 | <a href="#">P</a> |
| 超出时间限制 | N/A    | N/A     | C++ | 2022/02/07 09:22 | <a href="#">P</a> |
| 通过     | 504 ms | 12.7 MB | C++ | 2021/08/27 14:37 | <a href="#">P</a> |

# Python

```
class Solution:
    def maxSubArray(self, nums: List[int]) -> int:
        result=-float('inf')
        count=0
        for i in range(len(nums)):
            count += nums[i]
            if count > result:
                result = count
            if count <= 0:
                count = 0
        return result
```

执行结果: **通过** [显示详情 >](#)

[添加评论](#)

执行用时: **104 ms** , 在所有 Python3 提交中击败了 **94.23%** 的用户

内存消耗: **25.5 MB** , 在所有 Python3 提交中击败了 **48.57%** 的用户

通过测试用例: **209 / 209**

炫耀一下:



[写题解，分享我的解题思路](#)

| 提交结果          | 执行用时   | 内存消耗    | 语言      | 提交时间             |
|---------------|--------|---------|---------|------------------|
| <b>通过</b>     | 104 ms | 25.5 MB | Python3 | 2022/02/07 09:42 |
| <b>通过</b>     | 84 ms  | 66.1 MB | C++     | 2022/02/07 09:39 |
| <b>超出时间限制</b> | N/A    | N/A     | C++     | 2022/02/07 09:22 |
| <b>通过</b>     | 504 ms | 12.7 MB | C++     | 2021/08/27 14:37 |

## 思考

先想好贪心策略