力扣+最大子数组和+7.4

题目：

解题思路一：

涉及到数组，连续等关键字，第一个想法就是动态规划。我们假设下标能存储最大子序列的和，那么当前下标f[i]所能获取的值一定是建立在f[i-1]的基础之上的，而且需要连续最大，那么就说明我们需要返回的每个位置对应的最大值就可以了。

换言之就是我们需要得到的是f[i]=max(f[i]+nums[i],nums[i]),而真正的结果就是result=max(result,f[i]);由于子问题的解组成父问题的解过后就可以丢弃，所以这里我们只需要使用两个变量或者开辟一个长度为二的数组即可，假设我们开的是数组，那么有dp[0]=nums[0]，我们把结果保存在dp[0]中，由于数据的正负性质，我们设dp[1]=0，每次取dp[1]+nums[i]与nums[i]的大值与dp[0]进行比较更新就可以得到结果

```
#include<iostream>
#include<vector>
```

```
#include<string.h>
#include<stack>

using namespace std;
class Solution {
public:
    int maxSubArray(vector<int>& nums) {
        if (nums.size() == 1) return nums[0];
        int dp[2];
        dp[1] = 0; dp[0] = nums[0];
        for (int i : nums)
        {
            dp[1] = max(dp[1] + i, i);
            dp[0] = max(dp[0], dp[1]);
        }
        return dp[0];
    }
};
int main()
{
    vector<int> test = { 2,1,-3,4,-1,2,1,-5,4 };
    Solution s;
    cout<<s.maxSubArray(test);
    return 0;
}
```

解题思路二：

贪心,对于数据而言，如果当前值之和是>0的，那么加下一个数据就是有意义的，如果值是小于0的，那么无论下一个数据是什么，必须是导致数据量变小的，因此需要舍弃小于0的数据和，此时把和置为0，这种情况对应的是负数左边的值已经对右边的值没有意义了，因为无论多大，加起来都是减小，所以置为0重新计算数据和，但是原有的数据和，只要是递增的就保留，从削减开始就不保留，直到结束或者当前数据和置为0后新的数据和与原有的最大数据和进行比较

```
class Solution
{
public:
    int maxSubArray(vector<int> &nums)
    {
        //类似寻找最大最小值的题目，初始值一定要定义成理论上的最小最大值
        int result = INT_MIN;
        int numsSize = int(nums.size());
        int sum = 0;
        for (int i = 0; i < numsSize; i++)
        {
            sum += nums[i];
            result = max(result, sum);
            //如果sum < 0，重新开始找子序串
            if (sum < 0)
            {
                sum = 0;
            }
        }

        return result;
    }
```

```
};
```