**问题:给定n个整数（可能为负数）组成的序列a[1],a[2],a[3],…,a[n],求该序列如a[i]+a[i+1]+…+a[j]的子段和的最大值。当所给的整均为负数时定义子段和为0，依此定义，所求的最优值为:Max{0,a[i]+a[i+1]+…+a[j]},1<=i<=j<=n例如，当（a1,a2,a3,a4,a4,a6）=(-2,11,-4,13,-5,-2)时，最大子段和为20。**

注意区分sum=0的情形

代码(mine):

|  |
| --- |
| class Solution {  public:      int FindGreatestSumOfSubArray(vector<int> array) {          int size = array.size();          int sum = -RAND\_MAX\*RAND\_MAX , b = 0;          if (size== 0)return 0;          for (int i = 0; i < size; i++){              if (b>0)b += array[i];              else b = array[i];              if (b > sum)sum = b;          }          return sum;      }  };  代码(2): |

class Solution {

public:

    int FindGreatestSumOfSubArray(vector<int> array) {

        if(array.size()==0) return 0;

        int curSum = array[0], maxSum = array[0];

        for(int i = 1; i < array.size(); i++){

            curSum += array[i];

            if(curSum < array[i]){

                curSum = array[i];

            }

            if(curSum > maxSum){

                maxSum = curSum;

            }

        }

        return maxSum;

    }

};

代码(3)：

int MaxSum(int \*v,int n,int \*besti, int \*bestj)  
{  
    int sum=0;  
    int i,j;  
    for (i=1;i<=n;i++)  
    {  
        int thissum=0;  
        for (j=i;j<=n;j++)  
        {  
            thissum+=v[j];  
            if (thissum>sum)  
            {  
                sum=thissum;  
                \*besti=i;  
                \*bestj=j;  
            }  
        }  
    }  
    return sum;  
}

**分析：动态规划算法的应用.**