**算法设计与分析**

**实验报告**

|  |  |
| --- | --- |
| **专业班级** | **计算机科学与技术21-1班** |
| **学生姓名** | **何超** |
| **学 号** | **202001021107** |
| **指导老师** | **张鹏** |

**山东科技大学**

|  |  |  |  |
| --- | --- | --- | --- |
| **实验名称** | 最大子段和问题 | | |
| **实验日期** | **2022/06/03** | **实验地点** | J13-332 |
| **指导老师** | **张鹏** | **实验成绩** |  |

**一、实验目的和要求：**

掌握动态规划的设计思想。

尝试设计动态规划具体的代码实现

尝试使用不同算法实现问题需求

试比较不同算法之间性能差别

**二、实验内容：**

问题描述：

给定一组数据a[]，长度为n，求出其所有子段中加和最大的子段。

实验内容：

使用暴力法，分治法，动态规划法求解最大字段和

比较不同算法时间性能

定义测试数据，测试代码正确性

**三、算法描述及实验步骤：**

算法描述：

1.暴力法

使用两个游标a,b，使两个游标以n²的循环找出数组中所有的子段，分别计算每个子段的子段和，记录并比较是否为最长。最后输出最长子段的起始位置与结束位置。

此算法复杂度为n³，可以优化为n²，也就是并不需要每次都重新从起始位置求和加到终点位置.可以充分利用之前的计算结果，换一种穷举思路,对于起点i，可以遍历所有长度为1,2,…,n-i+1的子区间和，以求得和最大的一个。这样也遍历了所有的起点的不同长度的子区间，同时，对于相同起点的不同长度的子区间，可以利用前面的计算结果来计算后面的。比如，i为起点长度为2的子区间和就等于长度为1的子区间的和+a[i+1]即可，这样就省掉了一个循环，计算时间复杂度减少到了O(n^2)。

2.分治法

我们可以发现，所有子区间[start, end]只可能有以下三种可能性:

1.在[1, n/2]区域内

2.在[n/2+1, n]区域内

3.起点位于[1,n/2],终点位于[n/2+1,n]内

问题的解应为三种可能的最大情况，情况1和情况2符合子问题递归特性，可以使用递归求出. 然而我们需要单独处理第三种情形，因为他包括了n/2和n/2+1两个位置，我们可以利用穷举的思路求出：

1.以n/2为终点，往左移动扩张，求出和最大的一个left\_max

2.以n/2+1为起点，往右移动扩张，求出和最大的一个right\_max

3.left\_max+right\_max为第三种情况可能的最大值

3.动态规划法：

在该问题中，所求区间是连续的，则我们可以设定子问题为：如j为终点的最大子区间包含了位置j-1,则以j-1为终点的最大子区间必然包括在其中。由此推演出解决问题的逻辑大致为：

1.令b[j]表示以位置 j 为终点的所有子区间中和最大的一个

2.如果b[j-1] >0, 那么显然b[j] = b[j-1] + a[j]，用之前最大的一个加上a[j]即可，因为a[j]必须包含

3.如果b[j-1]<=0,那么b[j] = a[j] ,因为其是最大，所以前面的负数不能使其更大

实验步骤：

1. 按上述算法实现思路编写C++代码
2. 自定义数据测试各类实现方法正确性

**四、实验调试过程及输入输出结果：**

1.暴力法实现细节

为了利用每次计算之前的计算结果，我们可以将sum清空的语句置于i的每次自增而不是j，即：for(int i = 1; i <= n;i++)  int sum = 0;

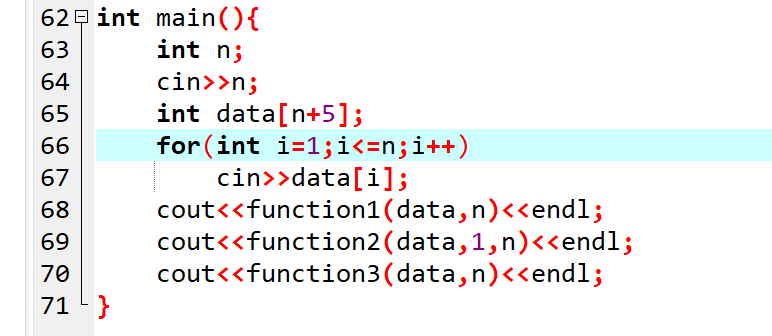
2.分治法调试：

我们应抓住分治法的情况3其特点，即其中间有两个定点，分别向两个方向扩张，来遍历所有属于第三种情形的子区间，求出最大的一个。

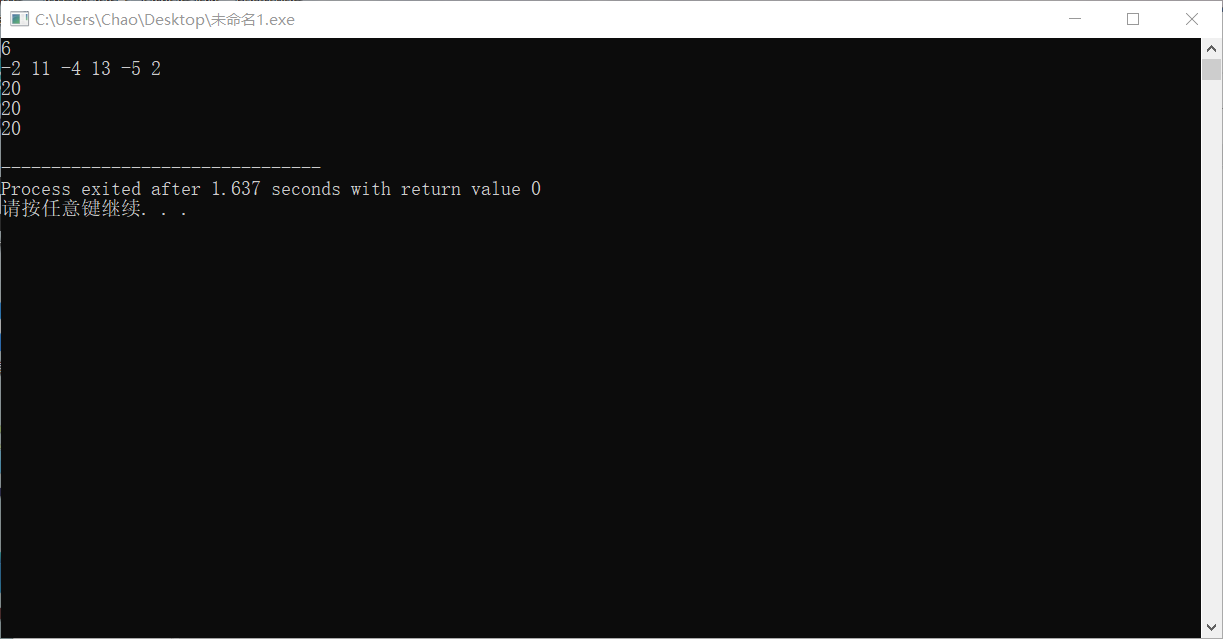
1. 设计main函数调用三种解决方法

我们将三种解决方法封装为函数形式，分别命名为function1，2，3；

使用打印语句调用三个函数，如下：



输入输出结果：



其中：第一行为暴力法求得结果

第二行为分治法求得结果

第三行为动态规划法求的结果

**五、实验总结：**

在本次实验中，我们学习了如何使用多种解法对问题的求解进行多角度的分析，虽然问题的要求是一致的，但是由于每个不同的问题解法思路不同，所以在切换新的问题解法时需要重新思考，而不被之前的做法所感染。在动态规划法解题的过程当中，应当注意问题的子问题结构与最优化问题的证明，来保证使用动态规划解决该问题的正确性。