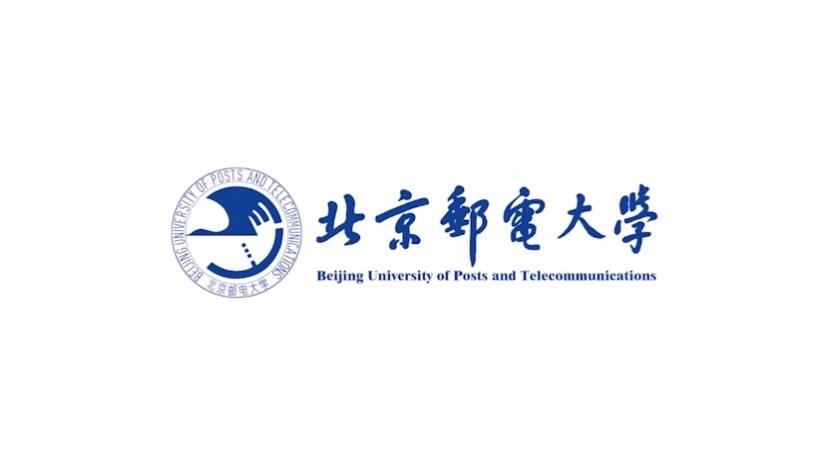
**算法设计与分析实验报告**



实验题目：最大字段和三种实现算法的时间复杂度分析

姓名： 陈俊卉

学号： 2020212256

日期： 2022.9.27

# 一、实验环境

(列出实验的操作环境，如操作系统，编程语言版本等，更多信息可视各自实际情况填写)

1. 操作系统：windows 10
2. 编程语言：c++
3. 编程工具：vscode及其组件

# 二、实验内容

具体要求请参照实验指导书中各实验的“实验内容及要求”部分。

(示例：1.描述你对实验算法的设计思路;2.给出算法关键部分的说明以及代码实现截图；3.对测试数据、测试程序(没有要求则非必须)进行说明，如测试覆盖程度，最好最坏平均三种情况等等，并给出测试结果截图等信息)

### 1.算法的设计与实现

##### **(1) 暴力枚举 [O(N^3)]**

###### **① 实验代码：**

int MaxSubsequenceSum1(const int A[],int N) {   
    /\* ((O(1)+O(j-i+1)+O(2))\*O(N-i))\*N + 1 \*/  
 int ThisSum,MaxSum,i,j,k;  
 MaxSum = 0;                             /\* O(1) \*/  
 for (i = 0; i < N; i++) {               /\* ((O(1)+O(j-i+1)+O(2))\*O(N-i))\*N \*/  
 for (j = i; j < N; j++) {           /\* (O(1)+O(j-i+1)+O(2))\*O(N-i) \*/  
 ThisSum = 0;                    /\* O(1) \*/  
 for (k = i; k <= j; k++) {      /\* O(j-i+1) \*/  
 ThisSum += A[k];  
 }  
 if(ThisSum > MaxSum)            /\* each loop: O(1)+O(1) \*/  
 MaxSum = ThisSum;           /\* each loop: O(1) \*/  
 }  
 }  
 return MaxSum;  
}  
/\* j-i+1<N,N-i<N, so ((O(1)+O(j-i+1)+O(2))\*O(N-i))\*N + 1 <= O(N)\*O(N)\*O(N) +1 = O(N^3)   \*/

###### **② 原理：**

枚举所有的子列，并将所有的子列和与MaxSum比较。

##### **(2) 暴力枚举 [O(N^2)]**

###### **① 实验代码：**

int MaxSubsequenceSum2(const int A[],int N)  /\* O(N^2) \*/  
{  
 int ThisSum,MaxSum,i,j;  
 MaxSum = 0;  
 for(i = 0;i < N; i++)                    /\* (O(1)+(O(1)+O(1)+O(1))\*(N-i))\*N \*/  
 {  
 ThisSum = 0;                         /\* O(1) \*/  
 for(j = i;j < N;j++)                 /\* (O(1)+O(1)+O(1))\*(N-i) \*/  
 {  
 ThisSum += A[j];                 /\* O(1) \*/  
 if(ThisSum > MaxSum)             /\* each loop: O(1)+O(1) \*/  
 {  
 MaxSum = ThisSum;            /\* each loop: O(1) \*/  
 }  
 }  
 }  
 return MaxSum;  
}

###### **② 原理：**

与(1)类似，枚举所有的子列，并将所有的子列和与MaxSum比较。但优化在：子列和其实可以从第二个循环里直接累加而来，而不需要像①一样每一次都重新进行遍历相加。

##### **(3) 分治 [O(NlogN)]**

###### **① 实验代码：**

int MaxSubsequenceSum3(const int A[],int left,int right) // T(n) = T(n/2) + n =>   O(nlogn)  
{  
 int leftmax;  
 int rightmax;   
 int search\_left\_max;  
 int search\_left = 0;  
 int search\_right\_max;  
 int search\_right = 0;  
 int midmax;  
 int mid = (left + right) /2;  
 int ThisSum;  
 // finished condition  
 if (left == right)  
 {  
 return A[left];  
 }  
 else  
 {  
 // maxsum through mid  
 // search left  
 for(int i = mid;i >= left; i--)  
 {   
 search\_left += A[i];  
 if(i == mid)  
 {  
 search\_left\_max = search\_left;  
 }  
 else{  
 if(search\_left > search\_left\_max)  
 {  
 search\_left\_max = search\_left;  
 }  
 }  
 }  
​  
 // search right  
 for(int i = mid + 1; i<=right; i++)  
 {  
 search\_right += A[i];  
 if(i == mid + 1)  
 {  
 search\_right\_max = search\_right;  
 }  
 else{  
 if(search\_right > search\_right\_max)  
 {  
 search\_right\_max = search\_right;  
 }  
 }  
 }  
​  
 midmax = search\_left\_max + search\_right\_max;  
​  
 // maxsum for left  
 leftmax = MaxSubsequenceSum3(A,left,mid);  
 // maxsum for right  
 rightmax = MaxSubsequenceSum3(A,mid+1,right);  
​  
 return fmax(fmax(leftmax,rightmax),midmax);  
​  
 }  
}

###### **② 原理：**

使用分治思想：但对于每一段分治子列，其子列有三种可能：左侧子列，右侧子列与跨越mid的子列。其中左侧子列和右侧子列的maxsum可以通过分治递归计算得出，而跨越mid的子列需要在递归返回时使用for循环从中间开始向左向右延伸求得此种子列最大值。于是我们可以得出时间复杂度的递推公式：T(N) = T(N/2) + N. 经过计算得出复杂度为O(NlogN).

##### **(4) 动态规划 O(N)**

###### **①实验代码：**

int MaxSubsequenceSum4(const int A[],int N) { /\* O(1)+O(N)\*(O(1)+O(1)+O(1)) \*/  
 int ThisSum,MaxSum,j;  
 ThisSum = MaxSum = 0;            /\* O(1) \*/  
 for (j = 0; j < N; j++) {        /\* total: O(N)\*(O(1)+O(1)+O(1)) \*/  
 ThisSum += A[j];             /\* each loop: O(1)     ; total: O(N)\*O(1) \*/  
​  
 /\* if-else part total for each loop:   O(1)+O(1)     ; total: O(N)\*(O(1)+O(1))   \*/  
​  
 if(ThisSum > MaxSum) {       /\* each loop: O(1)+O(1) ; total: O(N)\*(O(1)+O(1)) \*/  
 MaxSum = ThisSum ;       /\* each loop: O(1)     ; total: O(N)\*O(1) \*/  
 } else if (ThisSum < 0) {    /\* each loop: O(1)+O(1) ; total: O(N)\*(O(1)+O(1)) \*/  
 ThisSum = 0;             /\* each loop: O(1)     ; total: O(N)\*O(1) \*/  
 }  
 }  
 return MaxSum;  
}  
/\* O(1)+O(N)\*(O(1)+O(1)+O(1)) = O(N) \*/

###### **② 原理：**

只考虑累加得到正数。若累加后得到负数，则直接舍弃(ThisSum = 0)，而MaxSum会被保留。

**但这是以长度为0也作为子列作为条件的。即若所有的数都是负数，其子列长度取0，最大和取0。同样地，ppt所提供的暴力枚举也存在这个问题。其原因是MaxSum初始化应该为最小负数，而不应该是0。**

若考虑子列长度至少为1，则代码应该如下（同样为O(N)）：

int MaxSubsequenceSum5(const int A[],int N) {  
 int dp[N];  
 int MaxSum = A[0];  
 dp[0] = A[0];  
 for (int i = 1;i < N; i++) {  
 dp[i] = fmax(dp[i-1] + A[i] , A[i]);  
 MaxSum = fmax(MaxSum, dp[i]);  
 }  
 return MaxSum;  
}

/\* O(N) \*/

dp数组内表示**以该下标为结尾的子列和的最大值**，事实上并不需要这个dp数组，只需要ThisSum表示即可：

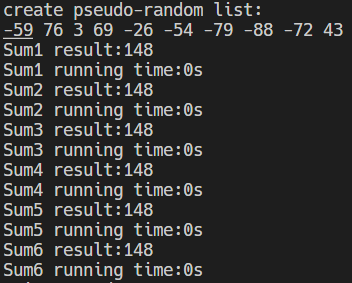
int MaxSubsequenceSum6(const int A[],int N) {  
 int MaxSum = A[0];  
 int ThisSum = A[0];  
 for (int i = 1;i < N; i++) {  
 ThisSum = fmax(ThisSum + A[i] , A[i]);  
 MaxSum = fmax(MaxSum, ThisSum);  
 }  
 return MaxSum;  
}

/\* O(N) \*/

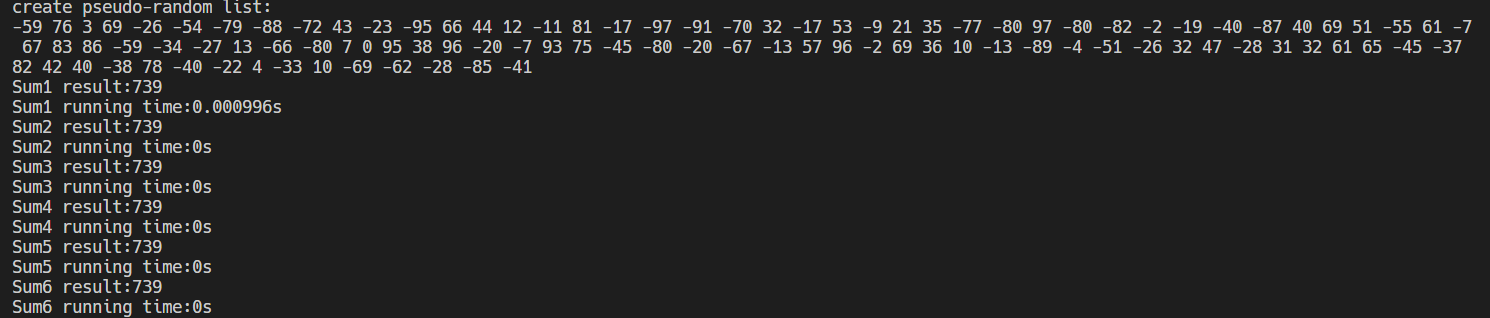
### 测试

测试结果如下图所示。

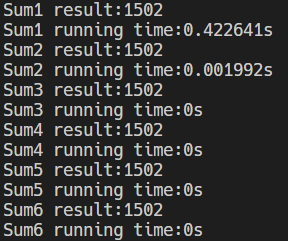
**N = 10：**



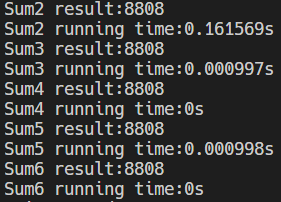
**N = 100：**



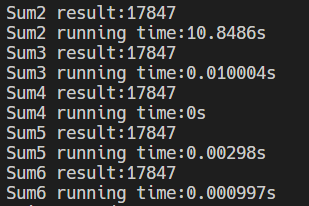
**N = 1000：**



**N = 10000：（算法一不参与测试）**



**N = 100000：(算法一不参与测试)**



**制表如下（以本机测试为准）：**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Algorithm | | 1 | 2 | 3 | 4 | 5 | 6 |
| Time | | O(N^3) | O(N^2) | O(NlogN) | O(N) | O(N) | O(N) |
| Input  size | N=10 | 0 | 0 | 0 | 0 | 0 | 0 |
| N=100 | 0.000996 | 0 | 0 | 0 | 0 | 0 |
| N=1000 | 0.422641 | 0.001992 | 0 | 0 | 0 | 0 |
| N=10000 | NA | 0.161569 | 0.000997 | 0 | 0.000998 | 0 |
| N=100000 | NA | 10.8486 | 0.0010004 | 0 | 0.00298 | 0.000997 |

# 三、出现问题及解决

(列出你在实验中遇到了哪些问题以及是如何解决的)

**问题一：使用clock精度不够，最多只精确到毫秒，难以看出算法区别。**

**解决方案**：使用c++的chrono库解决。Chrono库提供了microsecond级别的时间准确度，从而能够更加精确地比较算法之间的区别。

**问题二：Sum5中O(N)比其他O(N)的代码慢。**

原因可能是数组的写入写出需要的时间比变量的写入时间长。

**问题三：ppt所提供的算法代码：**

ppt算法代码是以长度为0也作为子列作为条件的。即若所有的数都是负数，按照算法，其最大子列长度取0，最大和取0。而笔者认为子列的最小长度应该为1才合理。这也在笔者所写的O(NlogN)代码和修改后的O(N)代码有所体现。而若答案为正数，二者输出是一样的。

**若仍采用ppt的代码，应该将MaxSum初始化为最小负数，而不应该是0**。

# 四、总结

(对所实现算法的总结评价，如时间复杂度，空间复杂度，是否有能够进一步提升的空间，不同实现之间的比较，不同情况下的效率，通过实验对此算法的认识与理解等等)

通过本次实验，我了解了最大字段和的暴力枚举、递归和动态规划算法，并且对他们进行了时间复杂度分析，进一步通过测试得到了实验数据，并对实验数据与理论不合理的地方进行了探讨并得出结论。

此外，还对ppt上的算法进行分析并提出了理解的不同之处，并加以修改得到新的代码。

此次实验令我对算法以及算法的测试有了具体的认知，这对我今后的算法分析学习有着莫大的帮助。