最大子序列和问题

最大子序列和问题

问题描述：

输入一组整数，求出这组数字子序列和中最大值。也就是只要求出最大子序列的和，不必求出最大的那个序列。例如：

序列：-2 11 -4 13 -5 -2，则最大子序列和为20。

序列：-6 2 4 -7 5 3 2 -1 6 -9 10 -2，则最大子序列和为16。

算法一：

```
//穷举法，复杂度O(n^3)
long maxSubSum1(const vector<int>& a)
{
    long maxSum = 0;
    for (int i = 0; i < a.size(); i++)
    {
        for (int j = i; j < a.size(); j++)
        {
            long thisSum = 0;

            for (int k = i; k <= j; k++)
            {
                thisSum += a[k];
            }
            if (thisSum > maxSum)
                maxSum = thisSum;
        }
    }
    return maxSum;
}
```

这是一个O(N^3) 的算法，算法本身很容易理解，而且很直观的感觉做了很多无用操作。例如：i = 0, j = 3时，会计算a[0] + a[1] +...+ a[3]；而当i = 0, j = 4时候又会计算a[0] + a[1] +...a[4]。

算法二：

通过撤出一个for循环来避免三次时间。

```
//也是穷举法，不过减去了上面的一些不必要操作O(n^2)
long maxSubSum2(const vector<int>& a)
{
    long maxSum = 0;
    for (int i = 0; i < a.size(); i++)
    {
        long thisSum = 0;
        for (int j = i; j < a.size(); j++)
        {
            thisSum += a[j];
```

昵称：InfantSorrow
园龄：6年
粉丝：42
关注：0
+加关注

< 2009年4月 >

日	一	二	三	四	五	六
29	30	31	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	1	2
3	4	5	6	7	8	9

搜索

找找看

常用链接

我的随笔
我的评论
我的参与
最新评论
我的标签
更多链接

我的标签

Android Jni(1)
C#控件(1)
C#事件(1)

```
            if (thisSum > maxSum)
                maxSum = thisSum;
        }
    }
    return maxSum;
}
```

这是一个非常直观的穷举法（比上面的分析还有简单些），而且没有多余重复的操作，复杂度为O(N^2)。其中，thisSum表示a[i] + a[i+1] + … + a[j-1]。

算法三：

对这个问题，有一个相对复杂的O(NlogN)的解法，就是使用递归。如果要是求出序列的位置的话，这将是最好的算法了（因为我们后面还会有个O(N)的算法，但是不能求出最大子序列的位置）。该方法我们采用“分治策略”（divide-and-conquer）。

在我们例子中，最大子序列可能在三个地方出现，或者在左半部，或者在右半部，或者跨越输入数据的中部而占据左右两部分。前两种情况递归求解，第三种情况的最大和可以通过求出前半部分最大和（包含前半部分最后一个元素）以及后半部分最大和（包含后半部分的第一个元素）相加而得到。

```
//递归法，复杂度是O(nlogn)
long maxSumRec(const vector<int>& a, int left, int right)
{
    if (left == right)
    {
        if (a[left] > 0)
            return a[left];
        else
            return 0;
    }
    int center = (left + right) / 2;
    long maxLeftSum = maxSumRec(a, left, center);
    long maxRightSum = maxSumRec(a, center+1, right);

    //求出以左边对后一个数字结尾的序列最大值
    long maxLeftBorderSum = 0, leftBorderSum = 0;
    for (int i = center; i >= left; i--)
    {
        leftBorderSum += a[i];
        if (leftBorderSum > maxLeftBorderSum)
            maxLeftBorderSum = leftBorderSum;
    }

    //求出以右边对后一个数字结尾的序列最大值
    long maxRightBorderSum = 0, rightBorderSum = 0;
    for (int j = center+1; j <= right; j++)
    {
        rightBorderSum += a[j];
        if (rightBorderSum > maxRightBorderSum)
            maxRightBorderSum = rightBorderSum;
    }

    return max3(maxLeftSum, maxRightSum,
        maxLeftBorderSum + maxRightBorderSum);
}

long maxSubSum3(const vector<int>& a)
{
    return maxSumRec(a, 0, a.size()-1);
}
```

另外max3(long,long,long)表示求三个long中的最大值：

```
//求出三个long中的最大值
long max3(long a, long b, long c)
{
    if (a < b)
    {
        a = b;
```


最新评论

1. Re:最大子序列和问题
好 支持一下
--一叶1993

2. Re:最大子序列和问题
@在大地画满窗子三个标识符就可以...
--jinyihan

3. Re:图像去噪算法简介
你好，若是对图像去噪处理进行多核是实现，改用哪种算法好呢？
--孤军各方

4. Re:关于考研学校分析
考哪里了？学长
--netxiaosheng

5. Re:一种简单的图像显著性计算模型
博主 你这里似乎有几个问题1.myPhase = angle(myFFT); 这句用的算法是 angle(z) = atan2(imag(z),real(z)).而不是 angle(z) = atan.....
--Virtual Earth

6. Re:选择问题
说错了，是O(NlogN)
--Silhouette°

7. Re:选择问题
请问，算法2A中，“如果使用buildHeap，构造堆的最坏情形就是O(N)的时间”是为什么？难道不应该是O(logN)吗？O(N)不是平均时间吗？
--Silhouette°

8. Re:我的考研2010(二)

```
        }
        if (a > c)
            return a;
        else
            return c;
    }
```

对这个算法进行分析：

$$T(1) = 1$$
$$T(N) = 2T(N/2) + O(N)$$

最后得出算法的复杂度为：**O(NlogN)** 。

算法四：

下面介绍一个线性的算法，这个算法是许多聪明算法的典型：运行时间是明显的，但是正确性则很不明显（不容易理解）。

```
//线性的算法O(N)
long maxSubSum4(const vector<int>& a)
{
    long maxSum = 0, thisSum = 0;
    for (int j = 0; j < a.size(); j++)
    {
        thisSum += a[j];
        if (thisSum > maxSum)
            maxSum = thisSum;
        else if (thisSum < 0)
            thisSum = 0;
    }
    return maxSum;
}
```

很容易理解时间界**O(N)** 是正确的，但是要是弄明白为什么正确就比较费力了。其实这个是算法二的一个改进。分析的时候也是i代表当前序列的起点，j代表当前序列的终点。如果我们不需要知道最佳子序列的位置，那么i就可以优化掉。

重点的一个思想是：如果a[i]是负数那么它不可能代表最有序列的起点，因为任何包含a[i]的作为起点的子序列都可以通过用a[i+1]作为起点来改进。类似的有，任何的负的子序列不可能是最优子序列的前缀。例如说，循环中我们检测到从a[i]到a[j]的子序列是负数，那么我们就可以推进i。**关键的结论是我们不仅可以把i推进到i+1，而且我们实际可以把它一直推进到j+1。**

举例来说，令p是i+1到j之间的任何一个下标，由于前面假设了a[i]+…+a[j]是负数，则开始于下标p的任意子序列都不会大于在下标i并且包含从a[i]到a[p-1]的子序列对应的子序列（j是使得从下标i开始成为负数的第一个下标）。因此，把i推进到j+1是安全的，不会错过最优解。**注意的是：虽然，如果有以a[j]结尾的某序列和是负数就表明了这个序列中的任何一个数不可能是与a[j]后面的数形成的最大子序列的开头，但是并不表明a[j]前面的某个序列就不是最大序列，也就是说不能确定最大子序列在a[j]前还是a[j]后，即最大子序列位置不能求出。但是能确保maxSum的值是当前最大的子序列和。**这个算法还有一个有点就是，它只对数据进行一次扫描，一旦a[j]被读入处理就不需要再记忆。它是一个**联机算法。**

联机算法：在任意时刻算法都能够对它已读入的数据给出当前数据的解。

常量空间线性时间的联机算法几乎是完美的算法。

附录：

程序测试：

先通过文件读写函数产生一组随机数并且读入到一个vector<int>中：

```
//COUNT和MAX_NUM分别表示随机数个数和最大值
const long COUNT = 1000;
const int MAX_NUM = 200;

//读文件
```

膜拜

--豆芽的博客

9. Re:最大子序列和问题
都是负数失效，因为summax) { max=sum; }　　}貌似这种可以A...

--静静探索

10. Re:最大子序列和问题
nice

--阿鲁巴

```
bool readFile(vector<int>& input, string fileName)
{
    ifstream infile(fileName.c_str());
    if (!infile)
        return false;
    int s;
    while(infile>>s)
    {
        input.push_back(s);
    }
    return true;
}

//写大量随机测试数据
bool writeTestData(string fileName)
{
    ofstream outfile(fileName.c_str());
    if (!outfile)
        return false;
    srand((unsigned)time(NULL));
    for (int i = 0; i < COUNT; i++)
    {
        if (rand() % 2 == 0)
            outfile << rand() % MAX_NUM << '\n';
        else
            outfile << ~(rand() % MAX_NUM) << '\n';
    }
    return true;
}
```

测试可得：

当COUNT = 1000的时候maxSubSum1()要等10s，后三个很快。

当COUNT = 10000的时候maxSubSum2()要等8s，后两个很快。

当COUNT = 1000000的时候maxSubSum3()要等10s，maxSubSum4()要等4s。

其实当COUNT = 1000000这个时候但是作文件读写就要很耗时了，光是in.txt就达到了4.7MB了。

而COUNT = 10000000的时候光是文件读写就要半分钟，in.txt达到了47.2MB，这时候再做maxSubSum3()和maxSubSum4()的比较，maxSubSum4()需要56s，而maxSubSum3()这时候需要85s（包括了读文件的时间）。可见数据量比较大的情况下O(NlogN)的递归算法也是可行的，并不比O(N)低很多。尤其在要求出最大子序列位置的情况下，分治递归算法体现了强大的威力。

程序源码：

```
#include <iostream>
#include <string>
#include <vector>
#include <fstream>
#include <cstdlib>
#include <ctime>

using namespace std;

//COUNT和MAX_NUM分别表示随机数个数和最大值
const long COUNT = 10000;
const int MAX_NUM = 200;

//读文件
bool readFile(vector<int>& input, string fileName)
{
    ifstream infile(fileName.c_str());
```

```
    if (!infile)
        return false;
    int s;
    while(infile>>s)
    {
        input.push_back(s);
    }
    return true;
}

//写大量随机测试数据
bool writeTestData(string fileName)
{
    ofstream outfile(fileName.c_str());
    if (!outfile)
        return false;
    srand((unsigned)time(NULL));
    for (int i = 0; i < COUNT; i++)
    {
        if (rand() % 2 == 0)
            outfile << rand() % MAX_NUM << '\n';
        else
            outfile << ~(rand() % MAX_NUM) << '\n';
    }
    return true;
}

//穷举法
long maxSubSum1(const vector<int>& a)
{
    long maxSum = 0;
    for (int i = 0; i < a.size(); i++)
    {
        for (int j = i; j < a.size(); j++)
        {
            long thisSum = 0;

            for (int k = i; k <= j; k++)
            {
                thisSum += a[k];
            }
            if (thisSum > maxSum)
            maxSum = thisSum;
        }
    }
    return maxSum;
}
```

```
//也是穷举法，不过减去了上面的一些不必要操作O(n^2)
long maxSubSum2(const vector<int>& a)
{
    long maxSum = 0;
    for (int i = 0; i < a.size(); i++)
    {
        long thisSum = 0;
        for (int j = i; j < a.size(); j++)
        {
            thisSum += a[j];
            if (thisSum > maxSum)
                maxSum = thisSum;
        }
    }
    return maxSum;
}

//递归法，复杂度是O(nlogn)
long max3(long a, long b, long c)
{
    if (a < b)
    {
        a = b;
    }
    if (a > c)
        return a;
    else
    return c;
}

long maxSumRec(const vector<int>& a, int left, int right)
{
    if (left == right)
    {
        if (a[left] > 0)
            return a[left];
        else
            return 0;
    }
    int center = (left + right) / 2;
    long maxLeftSum = maxSumRec(a, left, center);
    long maxRightSum = maxSumRec(a, center+1, right);

    //求出以左边对后一个数字结尾的序列最大值
    long maxLeftBorderSum = 0, leftBorderSum = 0;
    for (int i = center; i >= left; i--)
    {
        leftBorderSum += a[i];
```

```
        if (leftBorderSum > maxLeftBorderSum)
            maxLeftBorderSum = leftBorderSum;
    }

    //求出以右边对后一个数字结尾的序列最大值
    long maxRightBorderSum = 0, rightBorderSum = 0;
    for (int j = center+1; j <= right; j++)
    {
        rightBorderSum += a[j];
        if (rightBorderSum > maxRightBorderSum)
            maxRightBorderSum = rightBorderSum;
    }

    return max3(maxLeftSum, maxRightSum,
    maxLeftBorderSum + maxRightBorderSum);
}

long maxSubSum3(const vector<int>& a)
{
    return maxSumRec(a, 0, a.size()-1);
}

//线性的算法O(N)
long maxSubSum4(const vector<int>& a)
{
    long maxSum = 0, thisSum = 0;
    for (int j = 0; j < a.size(); j++)
    {
        thisSum += a[j];
        if (thisSum > maxSum)
            maxSum = thisSum;
        else if (thisSum < 0)
            thisSum = 0;
    }
    return maxSum;
}

int main ()
{
    vector<int> input;
    /**
    if (!writeTestData("in.txt"))
    {
        cout << "写文件错误" << endl;
    }
    */

    if (readFile(input, "in.txt"))
```

```
    {
        //cout << maxSubSum1(input) << endl;
        //cout << maxSubSum2(input) << endl;
        cout << maxSubSum3(input) << endl;
        cout << maxSubSum4(input) << endl;
    }

    return 0;
}
```

分类: Algorithms And Data Structures

标签: 最大子序列和, 算法

绿色通道: 好文要顶 关注我 收藏该文 与我联系

InfantSorrow
关注 - 0
粉丝 - 42
+加关注

8 0

(请您对文章做出评价)

« 上一篇: [转]作为大哥哥要像这样教小朋友~~~
» 下一篇: 关于考研学校分析

posted @ 2009-04-25 14:49 InfantSorrow 阅读(13805) 评论(13) 编辑 收藏

评论

#1楼 2009-04-25 14:53 | 蚊不叮[未注册用户]

这里真的不错.....昏！我也要在这里弄一个！

#2楼 2011-10-02 11:13 | xunil

楼主，请问算法三怎么求取序列的位置呢？？求解

支持(0) 反对(0)

#3楼 2012-06-05 00:02 | Griselda.

楼主，请问算法三怎么求取序列的位置呢？？求解 +1

支持(0) 反对(0)

#4楼 2012-08-22 15:20 | 白白的墙壁

要求序列的位置o（n）也可以的

支持(1) 反对(0)

#5楼 2012-11-28 23:55 | Paul_Cheung

飘过~~~

支持(0) 反对(0)

#6楼 2013-03-19 14:21 | 在大地画满窗子

飘！

支持(0) 反对(0)

#7楼 2013-03-19 14:30 | 在大地画满窗子

如果序列都是负数，目测第四个算法失效。

第四个算法，也是可以求出最大子学列的啊；只要有四个标志位：maxLeft，maxRight，sumLeft，sumRight

支持(1) 反对(0)

#8楼 2013-09-06 21:10 | 庸男勿扰

@在大地画满窗子
是的，如果序列都是负数，应该作为特殊情况单独拿出来判断

支持(0) 反对(0)

#9楼 2014-02-13 14:22 | 创世之后

O(n)没有说清楚，不全对，要标志所有的负数序列

支持(0) 反对(0)

#10楼 2014-07-03 11:48 | 阿鲁巴

nice

支持(0) 反对(0)

#11楼 2014-09-04 16:13 | 静静探索

都是负数失效，因为sum<0，不代表最大整数和必须大于等于0.

```
for(long int i=0;i<n;i++)
{
if(sum<0)
{
sum = a[i];
}
else
{
sum+=a[i];
}
if(sum>max)
{
max=sum;
}
}
```

貌似这种可以A

支持(2) 反对(0)

#12楼 2015-03-06 10:16 | jinyihan

@在大地画满窗子
三个标识符就可以

支持(0) 反对(0)

#13楼 2015-03-11 18:33 | 一叶1993

好 支持一下

支持(0) 反对(0)

刷新评论 刷新页面 返回顶部

注册用户登录后才能发表评论，请 登录 或 注册，访问网站首页。

最新**IT**新闻：

· 360公布智能家居战略 定位成安全互联网公司
· 美团王兴年会讲话：2015年是O2O的决战年
· 有例为证！苹果果然混进了时尚奢侈圈
· 帮传统行业转型 微信发布行业解决方案
· 上汽与阿里成立10亿元互联网汽车基金 打造生态圈
» 更多新闻...

最新知识库文章：

· 打造不可动摇的企业级移动化策略所需的七个步骤
· 帮设计师与工程师更好沟通的实用技巧
· 图片服务架构演进
· 软件架构师是一个角色，不是一项工作
· 给公司部门设计的SOA架构
» 更多知识库文章...