《算法设计与分析》第3次作业答案

姓名：XXX　　　　　学号：XXXXXXXX

算法分析题

题目1：给出N个$1-9$的数字$(v_1, v_2, ..., v_N)$，不改变它们的相对位置，在中间加入K个乘号和$N-K-1$个加号，（括号随便加）使最终结果尽量大。因为乘号和加号一共就是$N-1$个了，所以恰好每两个相邻数字之间都有一个符号。请给出算法思路、递推方程及其解释，并用伪代码描述算法。

例如：$N = 5, K = 2$，5个数字分别为1、2、3、4、5，可以加成：

$1 * 2 * (3 + 4 + 5) = 24$

$1 * (2 + 3) * (4 + 5) = 45$

$(1 * 2 + 3) * (4 + 5) = 45$

答：

算法思路：对于 n 个 1-9 的数字序列$(v_1, v_2, ..., v_n)$，设插入 k 个乘号后的最终结果为 $f(n, k)$,其中 n 表示有 n 个数，k 表示其中插入了 k 个乘号。最终结果的最大值取决于乘号的个数以及乘号所在的位置。我们假设在原序列中的第 m 个位置（第 m 个数和第 $m+1$ 个数中间）插入了最后一个乘号，则在序列$(v_1, v_2, ..., v_m)$中已经插入了 $k-1$ 个乘号，根据以上定义，我们可以定义动态规划的递推方程如下所示：

$$f(n, k) = max_{k \leq m < n}\{f(m, k-1) \times \sum_{i=m+1}^{n} v_i\}$$

上面的算法思路已经对递推方程进行了解释。

伪代码：

Algorithm 1 动态规划算法作业第一题伪代码

Input: 序列元素个数 N; 需要插入的乘号的个数 K; 序列 $Array$;

Output: 插入乘号之后可以得到的最大结果;

```
function MaxValue(N, K, Array)
    初始化二维数组dp, N行K列
    初始化sum(i) ← 根据Array计算出的序列前i个数字之和
    for i ← 1 to N do
        dp(i, 0) ← sum(i)
    end for
    for j ← 0 to K do
        dp(1, j) ← Array(0)
    end for
```

```
        for i ← 2 to N do
            for j ← 1 to K do
                for m ← j to i do
                    dp(i,j) ← max{dp(i,j), dp(m,j−1) × (sum(i) − sum(m))}
                end for
            end for
        end for
        return dp[N][K];
    end function
```

题目2：在自然语言处理中一个重要的问题是分词，例如句子“他说的确实在理”中“的确”“确实”“实在”“在理”都是常见的词汇，但是计算机必须为给定的句子准确判断出正确分词方法。一个简化的分词问题如下：给定一个长字符串$y = y_1y_2...y_n$，分词是把y切分成若干连续部分，每部分都单独成为词汇。我们用函数$quality(x)$判断切分后的某词汇$x = x_1x_2...x_k$的质量，函数值越高表示该词汇的正确性越高。分词的好坏用所有词汇的质量的和来表示。例如对句子“确实在理”分词，$quality$(确实) + $quality$(在理) > $quality$(确)+$quality$(实在)+$quality$(理)。请设计一个动态规划算法对字符串y分词，要求最大化所有词汇的质量和。（假定你可以调用$quality(x)$函数在一步内得到任何长度的词汇的质量）,请给出算法思路、递推方程及其解释，并用伪代码描述算法。

答：

算法思路：假设长字符串划分后的所有词汇的最大质量和为 $S(n)$，另外 $S(i)$代表的是$y_1y_2...y_i$ 经过划分后的最大质量和，其中$1 \leq i \leq n$。另外，第i个字 y_i 可以和他前面的 k 个字组成词汇,其中 $0 \leq k < i$。因此对于文字 i 对结果带来的影响，我们只需要对 k 的所有可能取值进行遍历，并找到一个可以得到最大结果的 k，得到最大的$S(i)$。根据以上定义，我们可以定义动态规划的递推方程如下所示：

$$S(i) = max_{0 \leq k < i}\{S(i-k) + quality(y_{i-k+1}...y_i\}$$

其中$quality(y_{i-k+1}...y_i)$表示词汇$y_{i-k+1}...y_i$的质量。

伪代码：

Algorithm 2 动态规划算法作业第二题伪代码

Input: 长字符串 y;

Output: 所有词汇的最大质量和;

```
function MaxQuality(y)
    初始化数组dp ← 0
    dp(1) ← quality(y_1)
    for i ← 1 to n do
        maxValue ← 0
        for k ← 0 to i do
            if dp(i − k) + quality(y_{i−k+1}...y_i) > maxValue then
                maxValue ← dp(i − k) + quality(y_{i−k+1}...y_i)
            end if
        end for
```

```
        dp(i) ← maxValue
    end for
    return dp[N];
end function
```

题目3：买卖股票的最佳时机简单版：给定一个数组，它的第 i 个元素是一支给定股票第 i 天的价格。如果你最多只允许完成一笔交易（即买入和卖出一支股票一次），设计一个算法来计算你所能获取的最大利润。注意：你不能在买入股票前卖出股票。示例如下：

输入: [7,1,5,3,6,4]

输出: 5

解释: 在第 2 天（股票价格 = 1）的时候买入，在第 5 天（股票价格 = 6）的时候卖出，最大利润 = 6-1 = 5 。注意利润不能是 7-1 = 6, 因为卖出价格需要大于买入价格。

(1) 请设计一个时间复杂度为$O(n^2)$的算法。

答：

算法思路： 很显然，要设计一个时间复杂度为$O(n^2)$的算法，只需要利用两层的嵌套循环来计算任意两条之间的股票差值即可。

伪代码：

Algorithm 3 买卖股票$O(n^2)$复杂度算法伪代码

Input: 股票值的数组 $Array$;

Output: 买卖股票获得的最大利润;

```
function MaxProfit(Array)
    maxPro = 0
    n ← Array.length
    for i ← 1 to n do
        for k ← i + 1 to n do
            if Array[j] − Array[i] > maxPro then
                maxPro ← Array[j] − Array[i]
            end if
        end for
    end for
    return maxPro;
end function
```

(2) 请设计一个时间复杂度为$O(n)$的算法。

注意：若使用动态规划，请给出算法思路、递推方程及其解释，并用伪代码描述算法；若不是使用动态规划，请给出算法思路、并用伪代码描述算法。

答：

算法思路： 很显然，要想设计一个时间复杂度为$O(n)$的算法，只能对数组做一次遍历。可以用动态规划的方法进行解决，假设前i天的最大收益为$P(i)$，则$P(i)$可能与前 $i-1$ 天的最大收益$P(i-1)$相等，也可能是第 i 天的价格减去前 $i-1$ 天中的最小价格,因此，我们也需要记录前 $i-1$ 天的最小股票价值min。根据以上分析，可以得出动态规划的递推方程如下所示：

$$P(i) = max\{P(i-1), Array(i) - min\}$$

伪代码：

Algorithm 4 买卖股票$O(n)$复杂度算法伪代码

Input: 股票值的数组 $Array$;

Output: 买卖股票获得的最大利润;

function MAXPROFIT($Array$)

  初始化数组$dp \leftarrow 0$

  $min \leftarrow Array[0]$

  $n \leftarrow Array.length$

  for $i \leftarrow 1$ to n **do**

    if $Array[i] < min$ **then**

      $min = Array[i]$

    end if

    $dp[i] \leftarrow max\{dp[i-1], Array[i] - min\}$

  end for

  return $dp[n]$;

end function

上面只是提供一种思路，这道题比较简单，还有其他写法可以实现$O(n)$的复杂度。

算法实现题

题目1：给定一个拥有正整数和负整数的二维数组，子矩形是位于整个数组中的任何大小为1*1或更大的连续子数组。矩形的总和是该矩形中所有元素的总和。在这个问题中，具有最大和的子矩形称为最大子矩形。请求出二维数组中的最大子矩阵之和。

题目细节及提交地址：`https://vjudge.net/contest/363101`；源码使用在线提交方式，提交密码：seu711184；用户名使用学号-姓名格式。

算法思想：对于这种问题，最直接的想法就是暴力求解法，即求出所有子矩阵的元素和，然后找到最大的一个。显然这种方法复杂度很高，在提交平台上不一定能通过。考虑到本题与在一维数组中求最大字段和类似，因此我们可以将这道题的二维的矩阵做降维处理，将其转变成在一维数组中求最大字段和的问题。具体方法如下所示：

我们首先来看一维数组中求最大字段和的动态规划算法。给定一个序列$a[0], a[1]...a[n]$，要求出连续的一段，使其总和最大。我们用$a[i]$表示第 i 个元素，$dp[i]$表示以$a[i]$结尾的最大子段和。则其递推方程如下所示：

$$dp[i] = max\{a[i], dp[i-1] + a[i]\}$$

如果$dp[i-1] > 0$，则 $dp[i] = dp[i-1] + a[i]$, 如果$dp[i-1] < 0$，则 $dp[i] = a[i]$。在一维数组中用动态规划求最大字段和的时间复杂度为$O(n)$。

对于二维矩阵的情况，我们首先确定子矩阵的首行和尾行，在一个二维矩阵中，找到所有的首行和尾行组合需要两层的嵌套循环，时间复杂度为$O(n^2)$。对于每一个确定了首行和尾行的矩阵，我们先将其压缩到一行，即将每行对应的元素相加，这样就得到了一个一维数组，然后再利用动态规划算法求这个一维数组的最大字段和，时间复杂度为$O(n)$。因此本算法的时间复杂度为$O(n^3)$。

最优子结构：由于动态规划算法是针对一维数组的情况进行实现的，所以此处用一维的情

况进行阐述。$dp[i]$所包含的问题$dp[i-1]+a[i]$也是最优的。假设$dp[i-1]$不是最优的，则存在一个$dp[i-k]$使得$dp[i-k]+a[i-k+1]+...+a[i]>dp[i-1]+a[i]=dp[i]$,这与$dp[i]$的值是最大的相矛盾。因此该问题存在最优子结构，$dp[i]$ 的最优解包含子问题$dp[i-1]$ 的最优解。

递推公式：递推公式也采用一维数组下的递推公式

$$dp[i]=max\{a[i],dp[i-1]+a[i]\}$$

解释：如果$dp[i-1]>0$，则 $dp[i]=dp[i-1]+a[i]$, 如果$dp[i-1]<0$，则 $dp[i]=a[i]$。

用一个5*5的二维数组实例说明解题过程：略