《算法设计与分析》第3次作业

姓名：叶宏庭　　　　学号：71118415

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：

类似于矩阵连乘问题，采用动态规划法，自底向上,定义A[i][j]为从i到j的最优解，由此可得到以下的递推方程：

$$A[i][j] = \max\left(A[i][k]*A[k+1][j], A[i][k]+A[k+1][j]\right) \quad i \le k \le j$$

从递推方程可以看出，当计算A[i][j]时，我们用k($i \le k \le j$)，来寻找最优子结构，从而得到最优解，最优值。

```
for (int i = 1; i <= N; i++)
 A[i][i] = nums[i];
for (int i = 1; i <= N; i++) {
 A[i][i + 1] = max(nums[i] * nums[i + 1], nums[i] + nums[i + 1]);
}
for (int ord = 2; ord <= N-1; ord++) {
 for (int i = 1; i + ord <= N; i++) {
  int maxc = 0;
  for (int j = i; j < i + ord; j++) {
   int m = max(A[i][j] * A[j+1][i + ord], A[i][j] + A[j+1][i + ord]);
   if (m > maxc) maxc = m;
  }
  A[i][i + ord] = maxc;
 }
}
```

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：

思路：设A[i]为前i个字的最大的所有词汇的质量和，对于A[i]，第i个字可以与其前面的k个字组成词汇，因此我们需要对k的全部取值进行一个遍历，便可得到最大的质量和。

递推方程：

$$A[i] = \max\left(A[i-k] + quality(x_{i-k+1}...x_i)\right) \quad 0 \le k \le i$$

伪代码：

```
for (int i = 0; i < n; i++) {
 int maxq = 0;
 for (int k = 0; k < n; k++) {
  if (A[i - k] + quanlity(x[i - k + 1:i]) > maxq)
   maxq = A[i - k] + quanlity(x[i - k + 1:i]);
 }
 A[i] = maxq;
}
```

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。

(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：

(1)　采用两个for循环嵌套，计算出任意两天之间的股票差值，就可得到最大利润。

```
int maxpro = 0;
for (int i = 0; i < n; i++) {
 for (int j = i + 1; j < n; j++) {
  if (A[j] - A[i] > maxpro)
   maxpro = A[j] - A[i];
 }
}
```

(2) 从前往后遍历，每一次都比较当前值与最小值，更新最小值，并且计算当前值与最小值的差，更新最大利润。

```
int maxpro = 0;
int minprice = INT_MAX;
for (int i = 0; i < n; i++) {
 if (A[i] < minprice) 更新最小价格
  minprice = A[i];
 if (A[i] - minprice > maxpro) 更新最大利润
  maxpro = A[i] - minprice;
}
```

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。

答：

首先，假定我们确定了最大子矩阵的首行与末行，我们设b[j]为第j列的和，则这时问题转化为求b这个数组的最大子段和(以下假设A[j]为b的前j个元素的最大子段和)，因此我们只需采用两个for循环嵌套，就可以遍历每一种首行与末行的情况，在每一种情况中求最大子段和，最后就可以得到最大子矩阵，易得时间复杂度为$O(n^3)$。

最优子结构：A[j]表示b[0:j]**的最大子段和。**

递推公式：

$$A[j] = \begin{cases} b[j] & A[j-1] < 0, \\ A[j-1] + b[j] & A[j-1] > 0. \end{cases}$$

用一个5*5的二维数组实例说明解题过程：

$$\text{实例：}\begin{matrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 & 0 \end{matrix}$$

首先采用两个for循环嵌套，确定首行与末行((1,1)，(1,2)，(1,3)...(5,5)),(1,1)时，求出最大子段和为3，(1,2)时，首先求出每一列的和b=(1,0,0,1,1)，再对b求出最大子段和为3，依次计算，直到(5,5)计算完毕，得出最大子矩阵和为5

结果截图：