《算法分析与设计》

课后作业

作业编号 作业4

学 号

姓 名

专 业 软件工程

学 院 计算机与人工智能学院

二0二二年四月

第5章 动态规划算法

总体要求：

1. 采用动态规划算法求解下面的问题。
2. 写出算法的求解思路。
3. 采用C/C++语言编程实现题目要求。
4. 设计具体实例，分析算法的求解过程，并验证算法的正确性。
5. 分析算法的时间复杂度。
6. 程序以源代码方式提交，包括源程序及工程文件；第（2）（4）（5）部分以Word文档方式提交。将上述两个部分压缩后以.rar文件提交。

题目1　一个序列由N个元素组成，现希望从该序列中挑选出**至少**F个连续的元素，使这些数的均值（挑选出的连续数之和/数的个数）最大。

输入要求：输入第1行包含两个整数N和F，其后的1行包含N个整数，表示该序列元素的个数。

输出要求：输出占1行，为对应的最大均值，精确到小数点后三位。

输入样例：

10 6

6 4 2 10 3 8 5 9 4 1

输出样例：

6.500

int arr[10], sum[10] = {0}, f[10] = {0}, num[10] = {0};

Arr[]为输入数组，sum[i]为前i个数字和，f[i]为当前最大和，num[i]为当前最大和的数字个数

计算各变量表达式如下：

float m = f[F] / F; *// m为当前最大均值*

int tmp1 = f[i - 1] + arr[i]; *//大于F个数的数字和*

int tmp2 = sum[i] - sum[i - F]; *// F个数字和*

if(tmp1\*n2 > tmp2\*n1)*//比较均值大小*

if(f[i]/num[i]>m)

                m = f[i] / num[i];

首先读入前F个数字，并求前F个数字和，接着从第F+1个数字开始，将前F+1个数字求和，记为tmp1，tmp1即当前最大和再加上一个数字，再将F个数右移，即下一组F个数求和，记为temp2，比较两者各自的均值，最终得出最大的均值。

（4）1）

输入样例：

10 6

6 4 2 10 3 8 5 9 4 1

输出样例：

6.500

前6个数字和sum[6]=33,类推可得,sum[7]=38,sum[8]=47,sum[9]=48；

Tmp1即为sum；Tmp2为不同的F个数字和；

比较过程如下：

Tmp1=sum[6]=33,tmp2=32,交换，当前最大和为33，数字个数num为6；

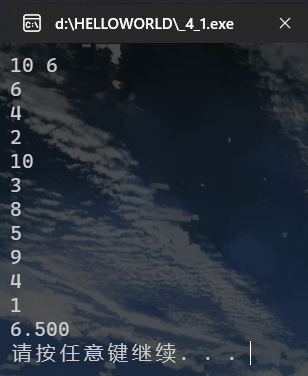
Tmp1=sum[7]=38,tmp2=37,交换，当前最大和为37，数字个数num为6；(38/7 < 37/6)

Tmp1=sum[8]=47,tmp2=10+3+8+5+9+4=39,交换，当前最大和为39，数字个数num为6；（47/8<39/6)

Tmp1=sum[9]=48,tmp2=30,不交换；

所以最终找到最大连续数字和为39，数字个数为6，最终应输出6.5，与程序运行结果相同，算法正确

运行结果如图所示：



2）

输入：9 4

-2,1,-3**,4,-1,2,1**,-5,4

输出：1.500

前4个数字和sum[4]=0,类推可得sum[5]=-1,sum[6]=1,sum[7]=2,sum[8]=-3,sum[9]=1；

Tmp1即为sum；Tmp2为不同的F个数字和；

比较过程如下：

Tmp1=sum[5]=-1,tmp2=1-3+4-1=1,交换，当前最大和为1，数字个数num为4；

Tmp1=sum[6]=1,tmp2=-3+4-1+2=2,交换，当前最大和为2，数字个数num为4；

Tmp1=sum[7]=2,tmp2=4-1+2+1=6,交换，当前最大和为6，数字个数num为4；

Tmp1=sum[8]=-3,tmp2=-1+2+1-5=-3,不交换；

Tmp1=sum[9]=1,tmp2=2+1-5+4=2,不交换；

所以最终找到最大连续数字和为6，数字个数为4，最终输出1.5，与程序运行结果相同，算法正确。

运行结果如图所示：



(5)分析算法可知，主体循环只与输入的数组大小N有关，被拆分为两次输入，一共还需循环N次，所以算法时间复杂度T（n）=F+N+F=N.所以时间复杂度为O（N）

题目2：商店为了促销往往将若干个商品捆绑在一起以低于商品原来的价格进行销售，这样可以鼓励顾客购买更多的商品。请帮助顾客计算他最少能以多少的价格才能购买到他所需的商品。注意，他不希望购买的商品与他的要求不符，也就是他购买的商品既不能多也不能少。

输入要求：

输入的第一行包含一个整数n（0<=n<=5），表示该顾客需要购买的商品的数量。其后的n行，每一行有三个整数c (1 <= c <= 999),k(1 <= k <= 5),p (1 <= p <= 999)，分别表示商品的**编号c，购买该商品的数量k 以及该商品的单价p**；之后的一行是一个**整数m (0 <= m <= 99)，表示可以选择的优惠类型**；其后的m行，每一行表示一种优惠类型。每一行的第一个数是一个整数**s (1 <= s <= 5)，表示该种优惠包含多少个可以捆绑在一起的优惠商品**，每一个捆绑的商品都包含该商品的**编号c**以及对应的**数量h**(1 <= h <= 5), 最后一个是该种捆绑优惠的商品的**总价格t**(1 <= t <= 9999)。比如某种捆绑优惠包含2种不同的商品，第一种商品的编号为7，数量为1，第二种商品的编号为8，数量为2，总的价格为10，则该种捆绑优惠可表示为：

2 7 1 8 2 10

输出要求：输出一个整数，占一行，表示购买所需商品的最低价格。

输入样例：

所需商品数量n 2

编号c 数量k 单价p

7 3 2

8 2 5

优惠类型M 2

捆绑商品种类S 编号c 数量h 总价t

1 7 3 **5**

2 7 1

8 2 **10**

输出样例：

14

1. 输入：

所需商品数量n

编号c 数量k 单价p

优惠类型m

捆绑商品种类s 编号c 数量h 总价t

输出：最低价格dp

变量表达式如下：

int a = i - Disproducts[s][ID[1]];

    int b = j - Disproducts[s][ID[2]];

    int c = k - Disproducts[s][ID[3]];

    int d = l - Disproducts[s][ID[4]];

    int e = r - Disproducts[s][ID[5]];

题目要求购买商品不超过5个，采用一个五维数组dp，dp[a,b,c,d,e]来表示。首先计算直接购买，之后寻找在当前需求量下的值，当满足优惠条件时，把所需物品数量减去捆绑销售方案中物品数量，如果有多个优惠方案，比较最小值。如果捆绑销售不如直接购买，则直接计算。最终可以找到所要购买的物品数量对应的最小值。

状态转移方程：minx = min(dp[a][b][c][d][e] + Disprice[q], minx)。

(4)1）输入样例：

2

7 3 2

8 2 5

2

1 7 3 5

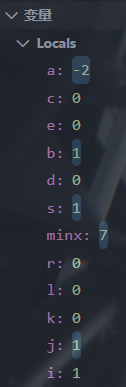
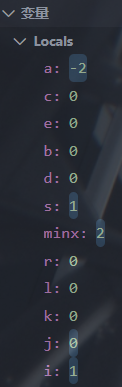
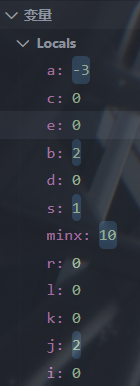
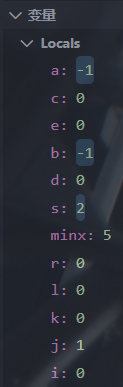
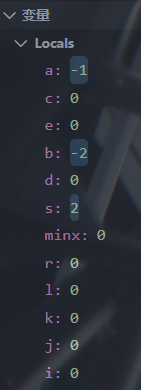
2 7 1 8 2 10

中间变量如图表所示:

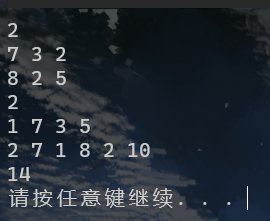
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| （i，j） | minx | （i，j） | minx | （i，j） | minx | （i，j） | minx |
| (0,0) | 0 | (0,1) | 5 | (0,2) | 10 | (1,0) | 2 |
| (1,1) | 7 | (1,2) | 12 | (2,0) | 4 | (2,1) | 9 |
| (2,2) | 12 | (3,0) | 5 | (3,1) | 10 | (3,2) | 14 |

由表可知最终最小花费为14

部分中间变量截图：



运行结果如图所示：



2）

2

7 3 4

8 2 10

2

1 7 3 10

2 7 1 8 2 20

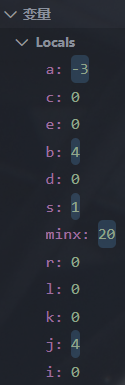
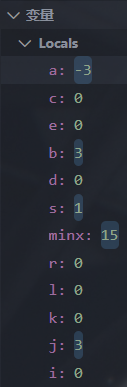
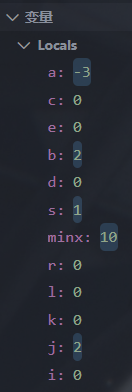
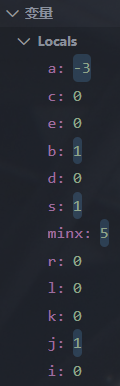
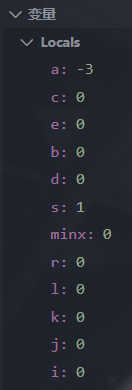
中间变量如图表所示:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| （i，j） | minx | （i，j） | minx | （i，j） | minx | （i，j） | minx |
| (0,0) | 0 | (0,1) | 10 | (0,2) | 20 | (1,0) | 4 |
| (1,1) | 14 | (1,2) | 24 | (2,0) | 8 | (2,1) | 18 |
| (2,2) | 24 | (3,0) | 10 | (3,1) | 20 | (3,2) | 28 |

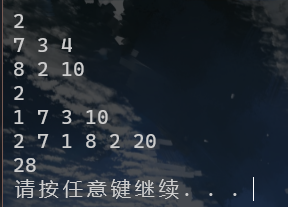
所以能看出购买完所需商品，最小的花费为28元

由表可知最终最小花费为28

部分中间变量截图：



运行结果如图所示：



(5)算法主体由六个for循环，外面五层控制商品数量，最内层循环由捆绑销售的组合数m决定，当每个循环都被执行最大次数时，有时间复杂度为O(（m\*n）^5)

题目3： 定义树中某结点的平衡为删除该结点后所形成的森林中各树结点数的最大值。如下图左边的树，如果删除其中的一个结点3，则该树变成右边的森林，也就是多棵树，这些树的结点数分别为6，1，1，1，其最大值为6，因此结点3的平衡为6。



在给定一棵树的情况下，求该树中所有结点的最小平衡数。

输入要求：

输入的第一行是一个整数n（(1 <= n<= 20)，表示测试数据的组数。每一组测试数据的第一行是一个整数m (1 <= m <= 20,000)，表示树中的一对结点，其后的N-1行，每一行包含两个整数，表示树中相连的两个结点。树中结点的编号从1开始编号。

输出要求：

对于每一组测试数据输出一行，表示该树中所有结点的平衡数的最小值。

输入样例：

1

9

1 2

1 3

1 4

2 5

3 6

3 7

3 8

4 9

4 10

输出样例：

4

(2)

输入: udg[]

输出变量：min

中间变量：sign[]，标记节点,sum[]，节点总数,DP[], 最大子树的节点数

求每个节点的子树中节点数量最多的子树，然后选节点最小的子树.算法的主要部分是求每个节点的子树中节点数量最多的子树，存在DP数组中，然后再选择最小的DP值，即求子树中节点最少的子树节点数，使用递归的DP，选择任意一个结点为整棵树的根，前序遍历，sum数组来标记i为根的子树的节点总数，写入DP数组

其求解公式为：DP[i]=max(m-sum[i],max(sum[j]))

(4)测试样例1：

1

9

1 2

1 3

1 4

2 5

3 6

3 7

3 8

4 9

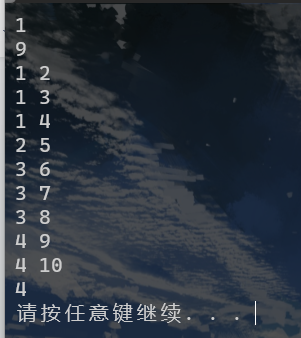
4 10

用前序遍历顺序的来描述算法的运行过程：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 子树的节点数 | | | | | | |
| 节点编号i | SUM[i] | DP[i] | m-sum[i] | sum[j1] | sum[j2] | sum[j3] |
| 5 | 1 | 9 | 9 |  |  |  |
| 2 | 2 | 8 | 8 | 1 |  |  |
| 6 | 1 | 9 | 9 |  |  |  |
| 7 | 1 | 9 | 9 |  |  |  |
| 8 | 1 | 9 | 9 |  |  |  |
| 3 | 4 | 6 | 6 | 1 | 1 | 1 |
| 9 | 1 | 9 | 9 |  |  |  |
| 10 | 1 | 9 | 9 |  |  |  |
| 4 | 3 | 7 | 7 | 1 | 1 |  |
| 1 | 10 | 4 | 0 | 2 | 4 | 3 |

灰色即为最大子树，再从DP中选择最小节点数为4

运行结果如图所示：



测试样例2：

1

10

1 2

1 3

3 4

3 5

3 6

4 7

4 8

5 9

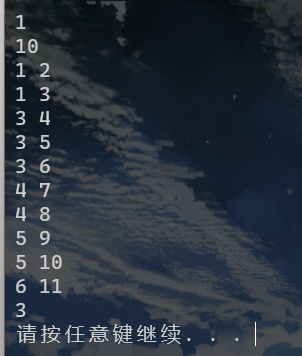
5 10

6 11

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 子树的节点数 | | | | | | |
| 节点编号i | SUM[i] | DP[i] | m-sum[i] | sum[j1] | sum[j2] | sum[j3] |
| 2 | 1 | 10 | 10 |  |  |  |
| 7 | 1 | 10 | 10 |  |  |  |
| 8 | 1 | 10 | 10 |  |  |  |
| 4 | 3 | 8 | 8 | 1 | 1 |  |
| 9 | 1 | 10 | 10 |  |  |  |
| 10 | 1 | 10 | 10 |  |  |  |
| 5 | 3 | 8 | 8 | 1 | 1 |  |
| 11 | 1 | 10 | 10 |  |  |  |
| 6 | 2 | 9 | 9 | 1 |  | 3 |
| 3 | 9 | 3 | 3 | 3 | 3 |  |

灰色即为最大子树，再从DP中选择最小节点数为3

运行结果如图所示：



(5)输入节点数量m，操作的执行次数只与输入规模m有关，由于树的存储方式为邻接矩阵，C(m) =m\*m，所以，其算法的时间复杂度为O(m^2)

评分标准：（总分100分，每题按100分评分）

1. 采用动态规划算法实现。（10分）
2. 算法的求解思路描述要求：（20分）
3. 有正确的输入输出变量定义。（5分）
4. 有明确的计算方法，且计算表达式由所定义的变量构成。（5分）
5. 计算过程准确无误，符合题目要求。（10分）
6. C/C++程序要求（30分）
7. 程序编译、运行正确无误。（5分）
8. 程序书写规范，变量、函数定义符合规范。（5分）
9. 程序与算法求解过程一致，且符合题目要求（20分）
10. 设计具体实例，分析算法执行过程并进行实验验证（20分）
11. 设计符合题目要求的测试数据，并能考虑各种输入情况。要求不少于2组测试数据。（5分）
12. 正确分析每一组测试数据输入时程序的执行过程，包括程序执行过程中中间变量的变化情况以及程序的执行结果。（10分）
13. 正确调试程序，并给出程序执行过程中中间变量的值以及程序的运行结果。（5分）
14. 算法复杂度分析要求（20分）
15. 算法复杂度分析方法及过程正确。（15分）
16. 算法复杂度分析结果正确。（5分）

提交截止时间：2022年5月7日20：00，每延后一天扣20分。

提交方式：电子稿，命名规则：学号\_姓名\_作业4.rar，其中的Word文档命名为：学号\_姓名\_作业4.doc(docx)。

邮件发送给:hyhuang@home.swjtu.edu.cn