**北京科技大学实验报告**

学院： 计通学院 专业：计算机科学与技术 班级： 计1503

姓名： 唐誉源 学号：41503302 实验日期： 2017 年 11 月 16 日

#### **实验名称：**

算法设计与分析实验2背包问题

#### **实验目的：**

1. 利用动态规划思想求解0-1背包问题
2. 给出装入物品最大价值
3. 要求多组测试数据输入输出

#### **实验设备（环境）及要求：**

操作系统: macOS High Sierra 10.12.1;

编辑器: sublime txt 3;

编译链接: gcc.

#### **实验原理：**

利用动态规划问题求解0-1背包问题时，可以用一个二维数组Vm[i][j]表示在只有i个物品，容量为j的情况下背包问题的最优解，那么当物品种类变大为i+1时，第i+1个物品可以选择放进背包或者不放进背包，假设放进背包那么Vm[i+1][j]=Vm[i][j]-weight[i+1]+value[i+1]；如果不放进背包，那么Vm[i+1][j]=Vm[i][j]。则得到了状态方程：Vm[i+1][j]=max(Vm[i][j],Vm[i][j]-weight[i+1]+value[i+1])。

进一步优化内存使用，在计算Vm[i][j]可以看出，在计算Vm[i][j]时只使用了Vm[i-1][j]，没有使用其他子问题，因此在存储子问题的解时，可以只存储Vm[i-1]子问题的解，并通过对j的循环，只使用一个一维数组。

最终的状态转换方程：Vm[j] = max(Vm[j], Vm[j-weight[i]]+value[i])，其中Vm[j]表示在承重为j的背包中可以取得的最大价值。

#### **实验内容与步骤：**

##### *（1）实验内容*

给出背包容量，以及n个物品的价值和提及，问：在不超过背包容量的情况，装的物品最大价值是多少呢？

输入：多组测试数据。每组数据三行：

* 第一行2个整数，。表示物品个数和背包空间。
* 第二行n个整数，表示第i个物品的价值。
* 第二行n个整数，表示第i个物品的体积。

##### *（2）主要步骤*

由实验原理中的结合动态规划思想所得到的状态方程：

int getMaximumValue(int n, int w, int weight[], int value[]) {  
 int Vm[w+1];  
 **for** (int i=0; i < w+1; i++) {  
 Vm[i]=0;  
 }  
 **for** (int i = 0; i < n; i++) {  
 **for** (int j=w; j>=weight[i]; j--) {  
 Vm[j] = max(Vm[j], Vm[j-weight[i]]+value[i]);  
 }  
 }  
 **return** Vm[w];  
}

在设计main函数时，为了保证多组测试数据的输入，设置成每次依次读入n、w、weight[]和value[]，之后直接计算出其结果M记录到数组中。循环上述步骤，并在遇到结束符（在此设置为'-1 -1'）时停止读入，开始输出数组M中记录的结果。

除此之外还加入了输入数据范围的限制，如果超出范围则重复输入。

int main() {  
 int n, w, M[100];  
 int i, j;  
 int weight[1001]={0}, value[1001]={0};  
 printf("Please input n, w (0 <= n <= 1000 && 0 <= w <= 5000): \n");  
 scanf("%d %d", &n, &w);  
 **for** (i = 0; n!=-1; ++i) {  
 **if** (n<0||n>1000||w<0||w>5000) {  
 printf("Overflowing, please retry.");  
 **continue**;  
 }  
 **for** (int i = 0; i < n; i++) {  
 scanf("%d", &value[i]);  
 **if** (value[i]<0||value[i]>1000) {  
 i--;  
 }  
 }  
 **for** (int i = 0; i < n; i++) {  
 scanf("%d", &weight[i]);  
 **if** (weight[i]<0||weight[i]>w) {  
 i--;  
 }  
 }  
 M[i] = getMaximumValue(n, w, weight, value);  
 printf("Please input n, w (0 <= n <= 1000 && 0 <= w <= 5000): \n");  
 scanf("%d %d", &n, &w);  
 }  
 printf("Maximum values:\n");  
 **for** (j = 0; j < i; ++j)  
 {  
 printf("%d\n", M[j]);  
 }  
 **return** 0;  
}

#### **实验数据：**

样例输入：

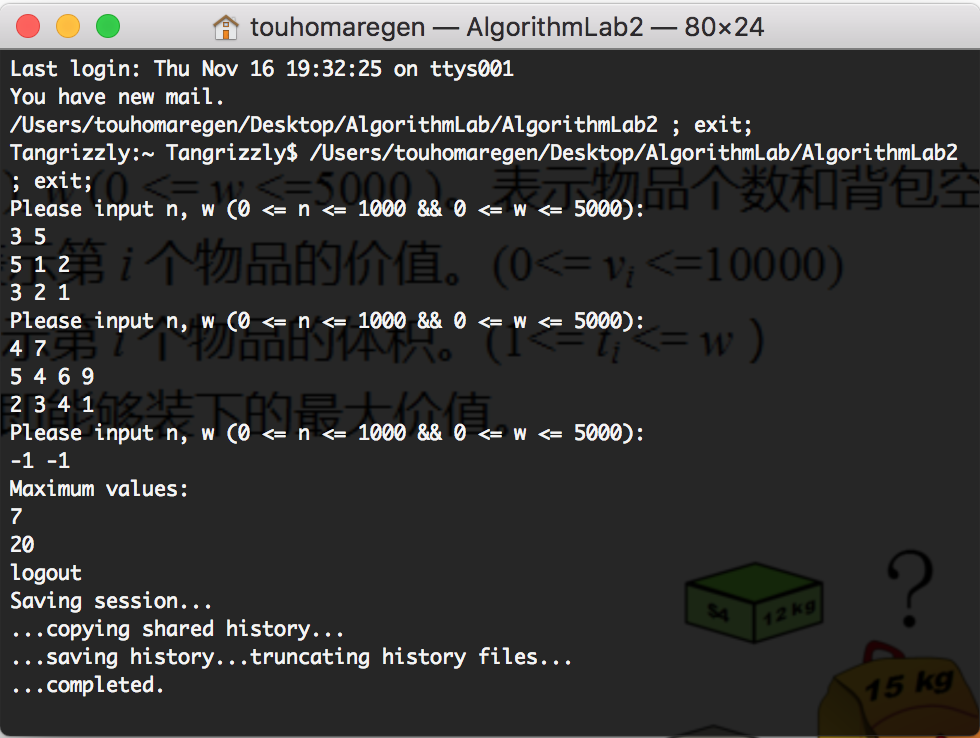
3 5  
5 1 2  
3 2 1

样例输出：

7

#### **实验结果与分析：**

运行结果截图展示：



多次连续输入测试组数据，之后统一输出结果7 20，可以看出运行结果完整而正确地显示了计算结果。利用动态规划思想解决0-1背包问题，能够得到全局最优结果。

#### **实验结论：**

##### *（1）实验结论*

利用动态规划思想求解0-1背包问题，逐步由子问题扩展到一个完整的问题，记录了每一步可能的最优结果，并在此基础上得到下一步最优的结果，最终得到物品最大价值。在main函数中利用设定结束符和数组保存结果的方法实现多组测试数据的输入输出。

##### *（2）讨论*

1. 在实验原理中，叙述了本程序状态方程从Vm[i+1][j]=Vm[i][j]-weight[i+1]+value[i+1]到Vm[j] = max(Vm[j], Vm[j-weight[i]]+value[i])的两次优化过程，最后可只用一个一维数组解决问题，优化了存储空间；
2. 每个输入数值都设置范围检测，如果不符合范围，需要重复输入，保证程序的在各种情况下的完备性；
3. 实现多组测试数据的输入输出的问题时，也可以事先提前输入测试数据的组数，以替代结束符的应用；
4. 选择输入数据之后直接调用函数计算结果记录在数组中，而不是用数组先记录输入的测试数据之后在进行函数调用输出结果的方法，减少了存储单元的使用量；
5. 测试数据组数被控制在了100以下，如果超出范围需要调整程序中记录运算结果M的数组大小；