

**算法**

**设计思路及分析**

班级： 计算机171班

姓名： 甘惠欢

学号： 201706401109

|  |
| --- |
| **问题简述** |
| 运用学习过的算法设计与分析方法：设计MKP问题的一种算法（多维背包问题） |
| **问题描述** |
| 下面通过对应的第一组数据来分析问题：  6（背包量） 10（属性） 3800（最优解）  100 600 1200 2400 500 2000（每个背包的价值）  8 12 13 64 22 41  8 12 13 75 22 41  3 6 4 18 6 4  5 10 8 32 6 12  5 13 8 42 6 20  5 13 8 48 6 20 质量  0 0 0 0 8 0  3 0 4 0 8 0  3 2 4 0 8 4  3 2 4 8 8 4  80 96 20 36 44 48 10 18 22 24（每组的限制条件）  选取物品装入背包内不超过限制条件，第一组质量不超80，第二组96，以此类推，于此同时还要从整体出发，即每一组选取后剩下的也要相同 |
| **实现方法** |
| 用Java语言实现回溯法解决多维背包问题 |
| **算法思路** |
| 即先判断一下是否所有的背包都符合条件，如果符合则装入，递归下一物品，不符合直接递归。 |
| **复杂度分析** |
| 时间复杂度：N乘以2N次方  空间复杂度：N的平方 |
| **设计步骤** |
| **1.设置一个abc方法**  void abc(double now,int w[],int num,int n,int k,int[] W,int[][] Weights,double Values[]){// Weights[] []质量数组的行下标 ；W[]的第几个元素  if(k>n){  if(now>ans){  ans=now; // 如果现在的价值大于之前的价值则赋值  for(int i=0;i<ls;++i)  x[i]=a[i];  lx=ls;  //System.out.println(ans);  }  return ;  }  int i=1; //从第一个物品开始计算  while (i<num&&w[i]+Weights[i][k]<=W[i]) {i++;} // 背包的物品相加要满足在约束条件内  if(i==num){  change(k,num,w,Weights,1);  //for(int j=1;j<=num;++j)  //w[j]+=Weights[j][k]  a[ls]=k; // 用于记录最优解的数组  ls++;  abc(now+Values[k],w,num, n,k+1, W, Weights, Values); // 加上价值，加上物品后继续进行循环  change(k,num,w,Weights,-1); // 若不满足条件则不取这个物品，回溯到上一个物品或上一步  ls--; // 返回上一个  }  abc(now,w,num, n,k+1, W, Weights, Values); // 继续调用函数进行下一步操作  }  **2.在这个方法中运用了if()函数**  if(k>n){  if(now>ans){  ans=now; // 如果现在的价值大于之前的价值则赋值  for(int i=0;i<ls;++i)  x[i]=a[i];  lx=ls;  //System.out.println(ans);  }  return ;  }  **3.while（）方法，跳出while（）循环执行abc（）方法**  int i=1;//从第一个物品开始计算  while (i<num&&w[i]+Weights[i][k]<=W[i]) {i++;} // 背包的物品相加要满足在约束条件内  if(i==num){  change(k,num,w,Weights,1);  //for(int j=1;j<=num;++j)  //w[j]+=Weights[j][k]  a[ls]=k; // 用于记录最优解的数组  ls++;  abc(now+Values[k],w,num, n,k+1, W, Weights, Values); // 加上价值，加上物品后继续进行循环  change(k,num,w,Weights,-1); // 若不满足条件则不取这个物品，回溯到上一个物品或上一步  ls--; // 返回上一个  }  abc(now,w,num, n,k+1, W, Weights, Values); // 继续调用函数进行下一步操作  **4.初始化**  void change(int k,int num,int b[],int a[][],int val) //初始化操作  {  for(int i=1;i<=num;++i)  b[i]+=val\*a[i][k]; // 满足约束条件的情况下将价值赋予b的数组中  }  void print()  {  System.out.println("选择的物品为");  for(int i=0;i<lx;++i)  System.out.print(x[i]+" ");  System.out.println();  }  **5.定义**  class sf{  int n ; //定义物品个数  int c;//定义背包数量  int W[]; //定义总容量  int Weights[] []; //定义表示重量的数组  double Values[] ; //定义表示价值的数组  double maxValue = 0; //定义最大价值  **6.给定范围，键盘输入n,c, Values[i], Weights[i][j], W[j]**  public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  int[] W = new int[3800];//定义总价值  int Weights[][] = new int[100][100];  double Values[] = new double[100];  // 将空格看作是两个字符串的间隔    int n = sc.nextInt();// 物品的个数  int c = sc.nextInt();//背包的个数  //下标从0开始，表示第1个物品，从键盘输入相应的值 ，输入部分  for(int i = 1; i <= n; i++) { //当行=0，行<=物品个数6，行+1  Values[i] = sc.nextDouble(); //从键盘输入价值  }  for(int i = 1; i <=c; i++) { //当行=0，行<=背包数量10，行+1  for(int j = 1; j <= n; j++) { //当列=0，列<物品个数6，列+1  Weights[i][j]=sc.nextInt(); //从键盘输入重量  }  }  for(int j = 1;j <=c; j++) {  //当列=0，列<背包数量10，列+1  W[j] = sc.nextInt();//得到列的总价值  }  /\*  //检验数据是否输入错误  System.out.println(n+" "+c);  for(int i = 0; i < n; i++) {  System.out.print(Values[i]+" ");  }  System.out.println();  for(int i = 0; i < c; i++) {  for(int j = 0; j < n; j++) {  System.out.print(Weights[i][j]+" ");  }  System.out.println();  }  for(int j = 0; j < c; j++) {  System.out.print(W[j]+" ");  }  System.out.println();\*/  **7.调用abc（）方法，并打印出最优解ans**  int[] w=new int[51];  Bag bag = new Bag();  bag.abc(0,w,c, n,1, W, Weights, Values);//调用abc函数  bag.print();  System.out.println("总价值为"+bag.ans);//打印出最优的价值 |
| **设计结果** |
| 代码测试过程看github链接：https://github.com/ganhuihuan/gan3.0/tree/master |
| **反思及总结** |
| 在此次算法设计当中，我认为最大的问题是，要从数据的整体出发，刚开始我所设计的算法能解决行的数据，但无论如何也没有办法得到老师给出的值，后面经讨论明白了多维背包问题要看整体，继而转变思路，不但看每一行，还要行列结合 |