多维背包问题设计分析

对于这个多维背包问题，我使用了回溯法的算法进行求解，用C语言写代码。

写代码的过程是这样的，首先我导入C语言的两个标准库，一个是标准输入输出函数，一个是动态内存分配函数。接着定义一个一维数组用来存储约束条件，一维数组的定义和使用，首先要先定义一个足够大的数组，然后利用scanf的返回值来确定输入结束。scanf的返回值是实际输入数值的个数。比如输入整型数值时，scanf("%d",&a);如果输入a成功，那么返回值会是1。因为约束条件没有具体的值，所以我用到了一个给定范围的输入方式限定可以输入的结果，范围取的足够大，可以满足约束条件数量的需要。接着我定义了一个digui函数，采用回溯法的思想去设计代码，函数里面定义了很多数值变量，进行递归的判断。伪代码如下：

int digui(int d,int m,int n,int dv,int ys)//定义一个digui函数，定义递归深度d，约束数m，变量数n，当前价值量dv，约束条件ys

{

if(i>n)//递归结束

{

if(dv>max\_vaule)//判断

{

max\_value=dv;//当前价值量赋给最大价值量

}

}

else

for (int j=0;j<=1;j++)//判断第i个物品是否要放进来

{

x[i]=j;

if (dv+x[i]<=ys)//每一行的所取量小于等于每一行的约束条件

{

dv=dv+v[i]\*x[i];//价值量增加

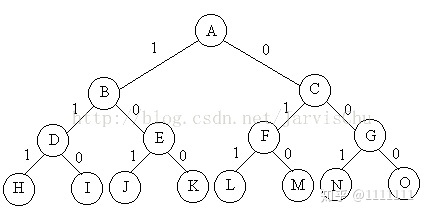
digui(i+1,dv,v[i]);//递归调用，继续循坏

dv=dv-v[i]\*x[i];//去掉没有用的物品

}

}

}



回溯法的思想就是列出一棵二叉树，比如说上面的图，1代表取这个值，0代表不取这个值，我们对每一个结点进行取或者不取，递归深度就是每一层物品的个数，当我们递归到叶子结点的时候这一条路就停止，得到一个结果，然后往前回溯，递归，比如说上面的图，当我们已经取值到了叶子结点H，得到了一个结果，我们往前回溯到D，这时候可对D不取，对E取值，这样我们又得到了一条新的路径，以此类推，对结点进行取值不取值，一直回溯，直到把所有的结点回溯完，再对每一条的路径的最终价值进行比较，选出最大的数值，就是我们要求的结果，这个是我对回溯法的理解。所以用到这个思想去设计相应的代码。在递归的过程中，把价值量不断赋给最大价值量，而多维背包问题这里还多了一个约束条件，这个也要考虑进来，每一行的价值量之和不能超过约束条件，这是一个判断条件。回溯递归的时候还要把没有用的数据去掉，代码里面用到了，不过没有写对，知道有这个想法，没有操作出来。就是这样一直回溯回溯，求出最大的价值量。

定义了函数之后开始写主函数的代码，主函数那里首先先把最大价值量初始化为0，定义约束数和变量数，打印输出语句，输入物品的价值。在这里物品的价值不适合直接输入，所以我采用了另外一个方法，定义二维指针，用二维数组来存储物品的价值，用到了动态分配内存的方式。基本代码如下：

int \*\*p; //定义二维指针。

int i,j;

scanf("%d%d",&m,&n); //输入行数和列数，即约束数和变量数

if(m<=0||n<=0) return -1; //行数或列数非法

p = (int \*\*) malloc(sizeof(int \*) \*m);//申请一组一维指针空间，动态申请内存空间

for(i=0;i<m;i++)

p[i]=(int \*)malloc(sizeof(int) \*n); //对于每个一维指针，申请一行数据的空间

for(i=0;i<m;i++)

for(j=0;j<n;j++)

scanf("%d",&p[i][j]);//输入第i行第j列的数据。

在代码里面我写了申请内存空间，进行了一个输入数据的判断，输入数据有0的情况就会报错，需要重新输入。二维数组的行数和列数我把它相应对比于约束数和变量数，这个是和算法的测试数据相对应的。二维数组的定义过程是这样的：1 定义二维指针。2 确定数组维数，即行数和列数。3 申请行数个一维指针大小空间，并赋值给二维指针。4 对于每个一维指针，申请列数个元素的空间，并赋值给对应的一维指针。5 输入或赋值数据。6 使用双重循环，逐个范围存储单元并输出。7 逐个释放一维指针上的内存。8 释放二维指针上的内存。其中的两个循坏是最重要的，按照这个思路写了代码，可以把输入的数据输出成矩阵的形式，因为用到了动态分配内存，所以后面我对内存进行了释放。

for(i=0; i<m;i++)

free(p[i]);

free(p);

return 0;

定义完了二维数组对物品的价值的存储后我还要对约束条件进行一个输入，因为约束条件不是一个，有很多个，所以我用到了一维数组对约束条件进行存储，这里的代码对应着主函数前面的代码，对输入约束条件的个数可以进行一个判断，显示出来。主函数里面我想用一个调用，这样可以返回digui函数，在把最终结果返回我们定义的最大价值量，最后打印结果，工作完成。

设计算法，写出代码运行，我发现还有很多错误，一个是递归函数那里，还有就是定义二维数组的时候，有的变量我定义了，但是运行起来的时候会报错，我不知道该怎么改。

回溯法求解多维背包问题是一个比较好的算法，跟穷举法类似，不过回溯法我们可以设置一个限界函数，减少代码的运行量，提高效率。在这里，对所写的代码的时间复杂度进行一个分析，代码里面运用到了多个循坏，还有嵌套循环，那么时间复杂度会高一点，O(n的2n次方)，对于时间复杂度我不是很会计算，希望老师指正。

以上是我对多维背包问题的算法思想以及代码的设计与分析，请老师检阅。