最小重量机器设计问题

1. 实验目的
2. 理解回溯法的深度优先搜索策略
3. 掌握用回溯法解题的算法框架
4. 问题描述

设某一机器由n个部件组成，每种部件都可以从m个不同的供应商处购得。设wij是从供应商j处购得的部件i的重量，wij是相应的价格。试设计一个算法，给出总价格不超过c的最小重量机器设计。

1. 输入

标准键盘输入。第一行有三个正整数n、m和d。接下来的2n行，每行n个数。前n行是c，后n行是w。

1. 输出

输出计算的最小重量及每个部件的供应商。

1. 数据结构

二维数组存储每个部件在不同供应商处的价格和重量，price[i][j]表示商品i在供应商j处的价格，weight[i][j]表示商品i在供应商j处的重量。

1. 算法设计

该问题属于全排列型回溯法，n个部件的m个供应商选择其中一个，共有mn种结果，选择其中价格小等于d，总重量最小的为最优解。

回溯法的剪枝函数即为当前总重量加待选取的重量小于已经得到的重量最优值。代码如下：

#include <iostream>

#include <vector>

using namespace std;

constexpr int MAX\_SIZE = 1001;

int n, m, d; //总价格不超过d

int price[MAX\_SIZE][MAX\_SIZE];//价格

int weight[MAX\_SIZE][MAX\_SIZE];//重量，初始为均使用第一个供应商

int ret;//总价格不超过d的最小重量

vector<int> ve;

void dfs(int id, int sum\_price, int sum\_weight, vector<int> &ve) {

if(id > n) { //到了尽头

if(sum\_weight < ret) { //可更新

ret = sum\_weight;

::ve = ve;

}

return;

}

for(int i = 1; i <= m; i++) {

//价格小于等于d，总重量小于等于当前最优解

if(sum\_price + price[id][i] <= d && sum\_weight + weight[id][i] <= ret) {

ve.push\_back(i);

dfs(id + 1, sum\_price + price[id][i], sum\_weight + weight[id][i], ve);

ve.pop\_back();

}

}

}

int main() {

//3 3 4 1 2 3 3 2 1 2 2 2 1 2 3 3 2 1 2 2 2

cin >> n >> m >> d;

for(int i = 1; i <= n; i++) {

for(int j = 1; j <= m; j++) {

cin >> price[i][j];

}

}

for(int i = 1; i <= n; i++) {

for(int j = 1; j <= m; j++) {

cin >> weight[i][j];

}

ret += weight[i][1];

}

vector<int> t;

dfs(1, 0, 0, t);

cout << ret << endl;

for(int it : ve) {

cout << it << " ";

}

cout << endl;

return 0;

}

1. 实验结果：

若输入为：

3 3 4

1 2 3

3 2 1

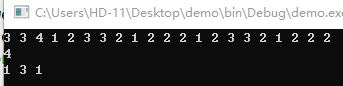
2 2 2

1 2 3

3 2 1

2 2 2

则运行结果如下：



此时最小重量为4，供应商依次选择1、3、1，总价格为4，满足条件。

1. 实验心得：

回溯算法可以解决大部分问题，得到一组向量形式的解，剪枝函数的正确设计可以有效提高回溯算法的效率。递归回溯较迭代回溯更容易理解。