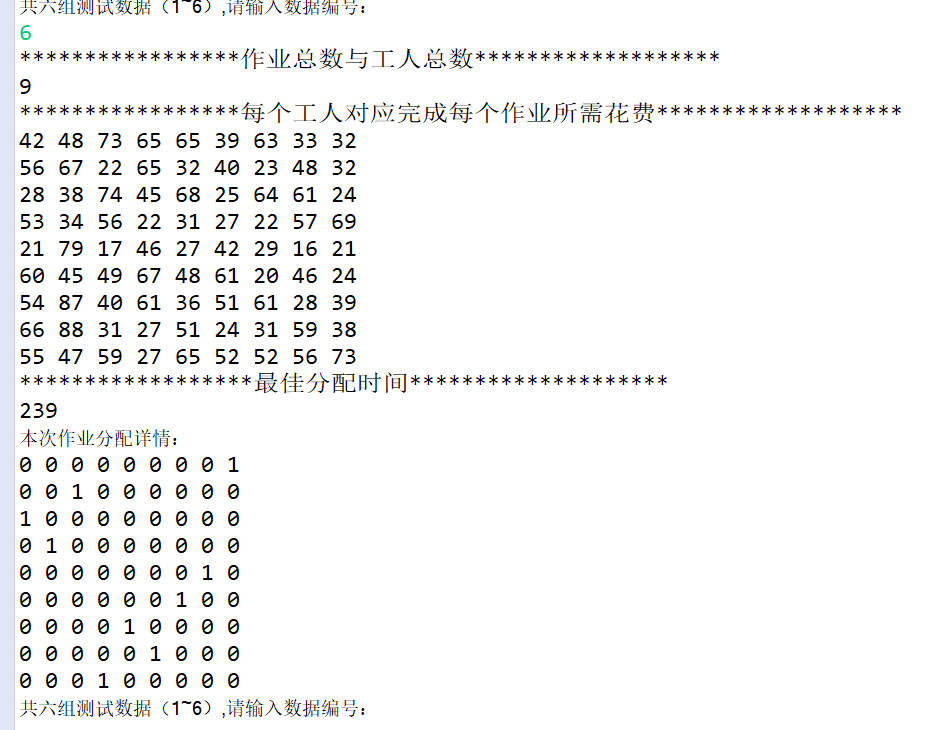
**学号**：S201961582

**姓名**：任鲁南

**联系电话**：18766861670

**运行环境**：eclipse

**运行过程说明**：根据提示输入6个测试数据文件名的任意一个，可连续输入，按回车时退出程序。实例如下：输入input\_assign04\_06.dat



**算法设计思路**：

**数据结构的设计**：

int n; // 记录工作人员数和作业数

int work\_cost[][]; // 记录第i号工作人员做第j号工作所需的时间

int res[][]; // 记录分配工作记录

int temp = 1; // 记录分配次数

int isworked[]; // 记录第j号作业是否被做了

int bestCost; //记录最佳作业分配时间

int currentCost; //记录本次作业分配时间

int assign[][]; //记录分配序列

**算法分析：**通过回溯法搜索整个作业序列，对每个人逐次读取每个作业，然后依次选取作业，作业的选取是任意的，故设置每个作业使用位，在每次读取作业时，判断作业是否被读取，如果未被读取，那么将读取这个作业，进入下一个人选取阶段,时间复杂度为：O(n^2),空间复杂度为：O(n^2)

**算法详细描述**：（伪代码）

//回溯递归算法

public void backTack(int i){

if (i >= this.n) { //说明已经遍历到叶子节点了,会产生一个当前最优解

if(this.bestCost > this.currentCost) {

bestCost = currentCost; //把当前最小花费赋值给最优花费时间

assign = new int[this.n][this.n]; //记录此次作业的分配情况

for(int m = 0; m < this.res.length; m++){

for(int n = 0; n < this.res[m].length; n++){

assign[m][n] = this.res[m][n] ;

}

}

}

}

}

//没有遍历到叶子节点

for (int j = 0; j <this.n; j++) {//在第i层时的选择，i到n都可以选择放在这一层,也就是说第i个人完成i到n个作业都有可能

if(this.isworked[j] == 0){ //判断本次作业中该项作业是否被选取

this.isworked[j] = 1; //选取本次作业，作业使用状态变为1

this.res[i][j] = 1;

currentCost += this.work\_cost[i][j]; //计算分配时间

backTack(i+1);

currentCost -= this.work\_cost[i][j];

this.isworked[j] = 0;

this.res[i][j] = 0;

}

}

}