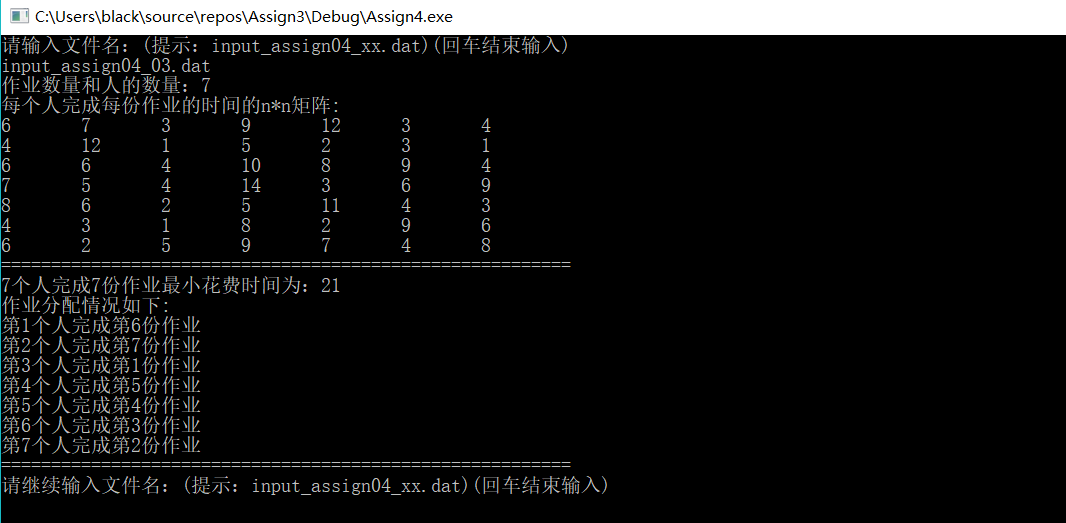
**学号**：S201861409 **姓名**：刘露蔚 **联系电话**：18811044209

**运行环境**：Visual Studio c++

**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\_03.dat



**算法设计思路**：

**算法思路**：要使得n个人完成n份作业时间最短，使用回溯法解决此问题，其解空间为一棵排列树，排列树的每一层为第i个人可能要完成的作业，回溯法从根节点出发，按照深度优先的策略遍历解空间树，在搜索至某一节点时，相应的剪枝条件是当前花费时间是否小于最小花费时间，若满足则继续扩展节点，若不满足则可以放弃扩展此节点，这样可以避免无效搜索，提高效率。当此节点不能再扩展时，回溯到此节点的根节点，并将所有条件都要置回其根节点的状态。本题采用回溯法对规模为n!的解空间进行大规模剪枝，从而使得实际搜索空间远远小于问题的解空间。

**数据结构的设计**：

int n： n用来表示作业数量

int c[i][j]： 二维数组c[i][j]用来表示第i个人完成第j份作业的时间

int bestCost： bestCost用来表示n个人完成n份作业最小花费时间

int cost： cost用来表示当前的花费时间

int bestAssign[i]：一维数组bestAssign用来表示当前最优分配序列，也就是第i个人在最优情况下完成的作业对象

int assign[i]： 一维数组assign用来表示当前分配序列，也就是第i个人在当前完成的作业对象，不一定是最优的

**算法分析：**具体来说，当扩展到某一个节点时，如果为叶子节点那必然会产生一个解，并且这个解是当前最优解，此时可以将当前作业分配情况也就是全排列的其中一个赋值给最优作业分配情况。如果不是叶子节点，则通过剪枝条件判断是否要继续遍历这棵子树，剪枝条件就是当前花费时间是否小于最优花费时间，如果满足，将节点进行交换，相当于是全排列中的固定位置放元素，元素通过交换来改变，并且继续向下一层遍历，下一层采用相同的遍历策略所以使用递归。注意，当从下一层返回来时，要将交换过的元素再交换回来，否则会出错。不管是否满足剪枝条件，都要进行回溯，回溯时要将所有条件的状态都置为上一次的状态。此算法的时间复杂度最坏为o（n！）。

**算法详细描述**：（伪代码）

//回溯递归算法

void backtrack(int i) {

if (i > n) {//说明已经遍历到叶子节点了,会产生一个当前最优解

bestCost = cost;

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

bestAssign[k] = assign[k];//将当前作业分配情况赋值给最优分配情况

}

}

else {//没有遍历到叶子节点

for (int j = i; j <= 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 ((cost+=c[i][assign[j]])<bestCost) {//目标条件：当前花费时间小于最优花费时间

swap(&assign[i], &assign[j]);//交换用于产生不同的序列

backtrack(i + 1);//递归，对于i+1到n采用同样的策略

swap(&assign[i], &assign[j]);//交换完之后要交换回来，否则会出错

}

cost -= c[i][assign[j]];//回溯，回到上一个节点

}

}

}