“分层图思想”及其在信息学竞赛中的应用

肖天
（天津市南开中学）

【摘要】本文通过对几道信息学竞赛题的解决，提出了一种解决问题的建模思想——分层图思想。该思想通过挖掘问题性质，将原问题抽象得出的图复制为若干层并连接形成更大的图，使本来难以用数学语言表达得图论模型变得简明严谨，为进一步解决问题打下了良好的基础。

【关键字】分层图思想　图论　数学模型　最短路　信息学竞赛

【正文】

1　引论

人们在借助计算机解决一个实际问题时，无非就是详细地告诉计算机应该怎么做，使它能通过人们给定的输入得到人们想要的输出。由于一般的计算机只能处理数字信号，所以只有把实际问题转化为数学问题，计算机才能帮助我们。这一步就是建立数学模型。

数学模型的建立在通过计算机解决问题的过程中非常重要。它把计算机无法理解的问题加以转化，使一切事物量化，最终变为只含数学过程的问题。它是人脑与计算机沟通的桥梁。不仅如此，数学模型的好坏直接影响着人与计算机之间的信息交流，影响着计算机对问题的“理解”。好的数学模型能够抓住问题的本质，表述简捷明了，易于人们找到有效的解决方法，并通过编制程序的方式将解决方法告诉计算机；相反，对于同一个问题，如果数学模型不能抓住问题本质，人们就可能无法解决问题，或者找不到有效的方法，更不用提告诉计算机如何做了。

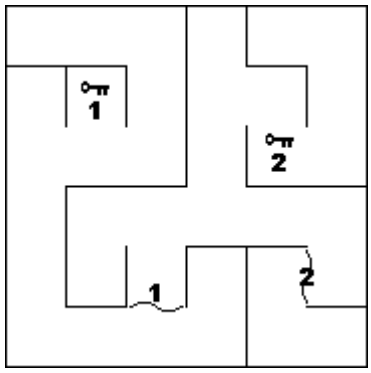
由于建立数学模型是为了解决问题，所以人们在做这项工作时往往希望把问题归结为已经很好解决的经典问题或若干这样问题的有机结合。这样，只要应用前人的研究成果就可以了。比如，排序、求图的单源最短路、网络流等等都是经典问题，前人不仅给出一般解法，而且对各种特殊情况和变形作了深入的研究。但事情并不总像人们希望的那样，有的问题即使可以归结为已有问题，在其中加入一些干扰因素后，原有性质就会发生改变，原来建立起的数学模型难以再用严谨的数学语言表达。这样问题中的部分图论问题可以用本文提出的“分层图思想”解决。

该思想注重对原问题性质的挖掘，通过对原问题数学模型的扩展，将干扰因素融入新的数学模型之中，恢复了模型的严谨性，进而与已解决问题产生联系，得到有效算法。

2　提出“分层图思想”

2.1 一个问题的解决

例题 1：拯救大兵瑞恩[1]

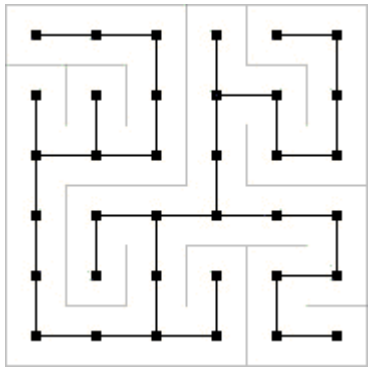


问题简述：有一个长方形的迷宫，被分成了 N 行 M 列，共 $N*M$ 个单元。两个相邻（有公共边）的单元之间可以互通，或有一扇锁着的门，或者存在一堵不可逾越的墙。迷宫中一些单元存放着钥匙，且所有的门被分为 P 类，打开同类门的钥匙相同，打开不同类门的钥匙不同。（如右图）

要求从迷宫左上角走到右下角营救大兵瑞恩，每从一个单元移动到相邻单元记为一步。只有拿到钥匙，才能打开相应的门。试求最少步数。

此题的标准解法是动态规划：以拿到的钥匙种类划分阶段，时间复杂度为 $O(2^PN^2)$。（详见[1]）

其实此算法可以用“分层图思想”做出更简明的解释。



首先忽略钥匙和门，那么问题就是在一个给定隐式图中求一条最短路，数学模型很简单：已知图 G，其中顶点与地图中的单元一一对应。当且仅当两格相邻且之间无墙时，他们对应的顶点间有一条边（如右图）。求从左上角对应顶点到右下角对应顶点最短路长度。

加入钥匙和门的因素，则所求最短路有了一个限制因素，即只有先到存在钥匙的格子，才能通过相应的门。换句话说，通过图中某些边是有条件的。所以不能再简单地求最短路了，而是要考虑何时那些边能通过，何时不能通过。这就要记录拿到了哪些钥匙。

此时，我们需对原模型进行改造：将原图 G 复制 2^P 个，记为 $G(s_1,s_2,\ldots,s_P)$，其中 s_i=0 表示未拿到第 i 类钥匙，s_i=1 表示已拿到第 i 类钥匙，i=1,2,…,P。对每个图 $G(s_1,s_2,\ldots,s_P)$，若 s_i=0，则将所有 i 类门对应的边去掉，因为没有此类钥匙不能通过此类门。再将其余所有边改为双向弧，权为 1。对所有顶点对

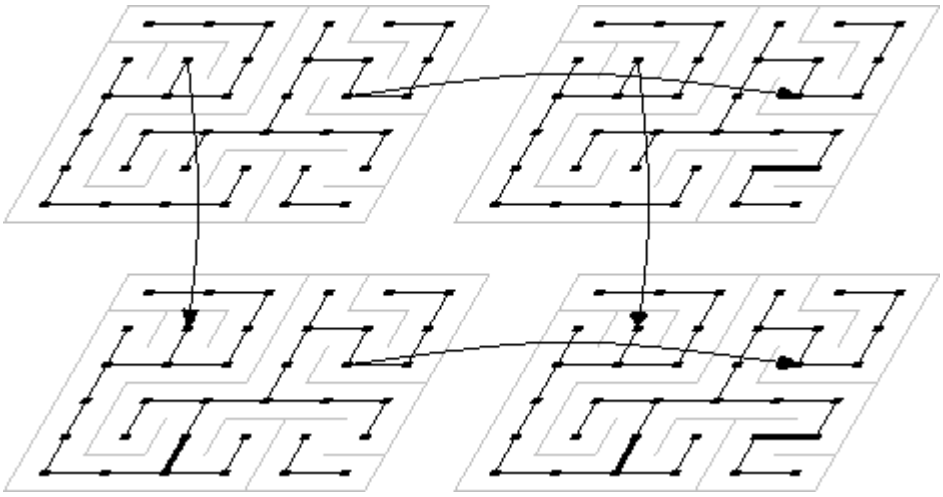
[1]选自 1999 年国际信息学奥林匹克中国国家队组队选拔赛，题目全文见参考文献[1]

(u, v)及整数 i，若它满足

- u、v 分别在 $G(s_1,s_2,\ldots,s_{i-1},0,s_{i+1},\ldots,s_P)$、$G(s_1,s_2,\ldots,s_{i-1},1,s_{i+1},\ldots,s_P)$中，且均对应$(x, y)$单元
- (x, y)格中有 i 类钥匙

则添加有向弧 uv，权为 0，表示走到(x, y)单元就可以由未拿到第 i 类钥匙转变为已拿到第 i 类钥匙，而不需要消耗额外的步数。

这样，这 2^p 个图被连成了一个 2^p“层”的有向图（如下图）。问题转化为求由该图 $G(0,0,\ldots,0)$层表示左上角的顶点到每一层表示右下角的顶点的最短路长度的最小值。注意，这里需要求的不是到 $G(1,1,\ldots,1)$ 层表示右下角的顶点的最短路长度，因为要成功营救大兵瑞恩很可能并不需要拿全所有的钥匙。



2.2 小结

可见，用此思想可以建立起更简洁、严谨的数学模型，进而很容易得到有效算法。重要的是，新建立的图有一些很好的性质：

由于层是由复制得到的，所以所有层都非常相似，以至于我们只要在逻辑上分出层的概念即可，根本不用在程序中进行新层的存储，甚至几乎不需要花时间去处理。

由于层之间的相似性，很多计算结果都是相同的。所以我们只需对这些计算进行一次，把结果存起来，而不需要反复计算。如此看来，虽然看起来图变大了，但实际上问题的规模并没有变大。

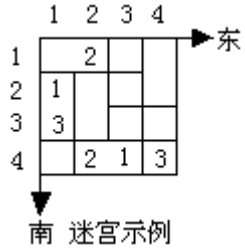
层之间是拓扑有序的。这也就意味着在层之间可以很容易实现递推等处理，为发现有效算法打下了良好的基础。

上述特点在例 1 的解决中体现得并不明显，所以本文给出的算法并不比标准算法更好。但这些特点说明这个分层图思想还是很有潜力的，尤其是各层有很多公共计算结果这一点，有可能大大消除冗余计算，进而降低算法时间复杂度。我们可以在下一个例子中看到这些。

3 分层图思想的应用

例题 2：迷宫改造[2]

题目简述：



有一个长方形迷宫，其在南北方向被划分为 N 行，在东西方向被划分为 M 列，于是整个迷宫被划分为 $N*M$ 个单元。我们用一个有序对（单元的行号，单元的列号）来表示单元位置。南北或东西方向相邻的两个单元之间可能存在一堵墙，也可能没有，墙是不可逾越的。

假定有 P 个人，他们分别从 P 个指定的起点出发，要求他们只能向南或向东移动，分别到达 P 个指定的终点。

问至少拆掉多少堵墙使得这是可行的（墙拆掉后可以任意通行），即所有的人可以从各自的起点出发依照游戏规则到达各自的终点。

参数限定：

$3 \le N$，$M \le 20$；$1 \le P \le 3$

为说明方便，下面我们只讨论 $M=N$，$P=3$ 的情况。（$P<3$ 时，可以添加起点终点重合的人而使 P 达到 3）

解法 1：仿照《方格取数》[3]一题进行动态规划。

以平行于副对角线的斜线划分阶段，令 $S(i,x_1,x_2,x_3)$表示三个人分别走到第 i 条斜线上第 x_1,x_2,x_3 行需要拆掉的最少墙数。对于每一个状态，每个人有向南和向东两种走法，共 $2^3=8$ 中决策。则有状态转移方程

---

[2] 选自 1999 年全国青少年信息学奥林匹克冬令营比赛，文中引用的是此题第二问的一部分，题目全文见附录

[3]全国青少年信息学奥林匹克分区联赛复赛高中组第 4 题，原文及解法见参考文献[2]

$$S(i,x_1,x_2,x_3)=\min\begin{cases}S(i-1,x_1-1,x_2-1,x_3-1)+?\\S(i-1,x_1-1,x_2-1,x_3\ \ )+?\\S(i-1,x_1-1,x_2\ \ ,x_3-1)+?\\S(i-1,x_1-1,x_2\ \ ,x_3\ \ )+?\\S(i-1,x_1\ \ ,x_2-1,x_3-1)+?\\S(i-1,x_1\ \ ,x_2-1,x_3\ \ )+?\\S(i-1,x_1\ \ ,x_2\ \ ,x_3-1)+?\\S(i-1,x_1\ \ ,x_2\ \ ,x_3\ \ )+?\end{cases}$$

其中，？表示相应决策需要拆的墙数，显然它只能在{0,1,2,3}中取值。由于情况较多，这里不再赘述。

另外，由于每个人的起点并不一定在地图左上角，终点也不一定在地图右下角，所以递推时还要处理一些细节。详见附录中的程序。

此算法的时间复杂度为 $O(N^4)$。

解法 2：应用分层图思想建立数学模型，将问题转化为最短路问题

我们先来分析一下本题的特点。

首先，每个人移动路线的可行性是不受其他成员影响的，而在计算代价时不同人之间仅当他们经过同一堵墙时才有关系。如果忽略这种关系，那么每个人的路线应是他起点终点之间的一条最短路。换句话说，有的人可能会迁就其他人而导致其路线不是最短路，但他们不在公共路线上时每个成员的路线都是最短路。我们定义：

定义 1：**纯路段**是某一可行解的一部分，他与每个家庭成员路线的交集要么是空集，要么是它本身（即它是该成员路线的一部分）。直观的说，某一可行解的纯路段不分岔。

定义 2：**极大纯路段**是某个可行解的纯路段，但不是该解中任何其它纯路段的子路段。

这样，我们刚才得到的性质就可以表述为：最优解中的每个纯路段都是连接其两端点的最短路。所以，一个最优解可以由其中所有极大纯路段端点的集合唯一确定。于是我们就有了一个最朴素的算法：枚举这些端点。虽然这个算法的时间复杂度令人难以忍受，但他给了我们一个思路，即充分利用纯路段的最短路性质。

让我们进一步挖掘此题特点。此题与普通求最短路的问题最大的区别在于其中有多个人，他们有公共路段时，该路段的权只被计算一次。因此抓住公共路段是解决问题的关键。

经分析可以发现公共路段有以下特点：

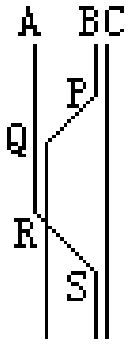
定理 1：存在一最优解，其中任两个人的公共路段不会多于一条，即任两

个人的路线不会在会合、分离之后再次会合。

证明：假设某一最优解 T 中存在两个人 A、B，他们在 P 点分离后又在 Q 点会合，设他们在 P、Q 之间的路线的权分别为 W_A、W_B。则他们在 P、Q 之间的总代价为 $W=W_A+W_B$。那么必然存在另一可行解 T'，它与 T 的唯一差别是 B 在 P、Q 之间的路线改为与 A 相同。则 T'中 A、B 在 P、Q 之间的总代价为 $W'=W_A$。由于 $W'\leq W$，所以 T'也是最优解。因此可以通过有限次像上面的变换去掉某一最优解中的所有“会合——分离——再会合”的情况而保持其最优解的性质。证毕。

定理 2：当 $P\leq 3$ 时，存在一最优解，其中所有人的路线之并集无环。（路线看作是无向路径）

证明：这实际是对定理 1 的简单推广。假设某一最优解 T 中有环，且应用定理 1 去掉两人产生的环之后仍存在环。那么该环必然由三人产生，且只有右图一种情况。类似于定理 1 的证明，C 人的 PS 段路线显然可以由 $PQRS$ 代替，而省去 PS 段的费用，新得到的解必是最优解。如此总能得到无环的最优解。证毕。



定理 3：当 $P\leq 3$ 时，对一个无环的最优解 T，其中的所有公共路段可以用一条从左上角到右下角的路线覆盖。

证明：假设 T 中有两条公共路段不能被一条从左上角到右下角的假想路线覆盖，那么显然他们不能属于同一个人的路线（否则此人的路线即满足要求）。又因每条公共路段至少属于两人，所以 $P\geq 4$。与 $P\leq 3$ 矛盾。证毕。

定义：对于一个最优解 T，如果一条从左上角到右下角的路线可以覆盖 T 中的所有公共路段，则称该路线为 ***T* 的主路线**。

这样，一个最优解可以这样描述：它是由一条主路线和一些从主路线伸向各成员起点和终点的支路线组成的。在这个最优解中，每个成员从他的起点出发，要么先通过支路进入主路线，经过一段路后再走上支路到达终点，要么不经过主路线直接走到终点。由于主路线覆盖了所有的公共路段，所以所有的支路都是纯路段，他们都有最短路的性质。应用这个性质，我们可以通过预处理求得所有支路的长度，就像对待 $P=1$ 的情况一样进行动态规划，时间复杂度为 $O(N^2)$。

接下来，只有如何确定主路线的问题我们尚未解决。容易看出，这个问题与

P=1 的情况十分类似，只不过同时要确定支路与主路线会合点和分离点。可以把支路与主路线的会合与分离看作主路线状态的改变。这样，就可以用“图的分层思想”解决这个问题了。

由于每个成员由起点到终点的过程可以分为三部分：支路——主路线——支路，所以每个成员对主路线有三种状态：未进入——已进入——已离开。于是，对于每个成员，我们把地图复制为三层。一共复制为 3^P 层，用 $G(s_1, s_2, s_3)$表示三个人状态为(s_1, s_2, s_3)的层。未进入、已进入、已离开三种状态分别用 0、1、2 表示。针对成员 1，我们从 $G(0, s_2, s_3)$中每个点向 $G(1, s_2, s_3)$中相应点引一条有向弧，权为该成员起点到该点的最短距离（即已经在预处理中求得的支路长度），如果无法从该成员起点到达该点，则权为∞。同样，从 $G(1, s_2, s_3)$中每个点向 $G(2, s_2, s_3)$中相应点引一条有向弧，权为该点到该成员终点的最短距离，如果无法从该点到达该成员终点，则权为∞。然后，我们求从 $G(0, 0, 0)$左上角的顶点到其他点的单源最短路径。这里需要注意一点，答案并不一定是从 $G(0, 0, 0)$左上角的顶点到 $G(2, 2, 2)$右下角的顶点最短路长度。因为此时我们忽略了一点，即前面提过的有的成员可能“不经过主路线直接走到终点”。所以我们应该考虑每个成员是否进入主路线，共 8 种情况。

至此，我们已经圆满解决了这个问题。解法 2 中预处理时间复杂度为 $O(N^2)$，之后求最短路的时间复杂度为 $O(N^2)$，所以总时间复杂度是 $O(N^2)$，这大大低于解法 1。

但是，解法 1 可以将 P 推广到更大的数，而解法 2 只能解决 P≤ 3 的问题。实际上，解法 1 中提到的《方格取数》一题应用网络流模型可以有更好的算法，可是作者依照这个思路并没有建立起此题的网络流模型。但这仍是一个很有价值的问题，有待解决。

4 总结

挖掘问题性质是该思想的关键，针对我们不能用严谨的数学语言表达的因素，进行图的分层，即相当于分类讨论，将未知变为已知。这样做实际是强化了原有问题的性质。通过这样的处理，新得到的图论模型特点更突出，更易于找到解决方法。

【参考文献】

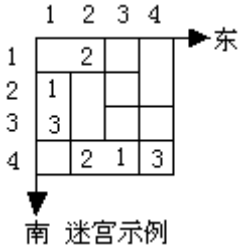
[1]江涛，王颖寒，骆静辰编，信息学奥林匹克竞赛题解精编，南京大学出版社，2000.6
[2]章维铣编著，全国青少年信息学（计算机）奥林匹克分区联赛试题解析（中学），南京大学出版社，2001.6

【附录】

1. 《迷宫改造》题目原文：

问题描述：

XX 娱乐公司最近获得了一些古希腊迷宫的拥有权，为了使这些古典式的迷宫能够吸引更多的游客，XX 公司对这些迷宫进行了合理的改造。你的任务是根据所给的一个迷宫，针对公司的要求，在原有的迷宫的基础上设计出一个最佳的新式迷宫。



迷宫的外型是一个长方形，其在南北方向被划分为 N 行，在东西方向被划分为 M 列，于是整个迷宫被划分为 $N*M$ 个单元。我们用一个有序对（单元的行号，单元的列号）来表示单元位置。南北或东西方向相邻的两个单元之间存在一堵墙或者一扇门，墙是不可逾越的，门是双向的且可以任意通过。出于保护文物的目的，XX 公司决定只适当地将墙改置为门，而不进行其他改造，并且要求新迷宫是最佳的，即新置的门的总数要最少。

公司首先需要你设计出一个最佳迷宫，使游客可以在改造后的新式迷宫中的任一单元出发，到达其他任何单元。这样的最佳迷宫称为第一类最佳迷宫。（占每个测试点分值的 30%）

另外，公司计划推出一个面向家庭的迷宫游戏，游戏规则如下：

假定有 P 个家庭成员，他们分别从 P 个指定的起点出发，要求他们只能向南或向东移动，分别到达 P 个指定的终点。

公司需要你针对上述游戏规则，设计一个最佳的迷宫，使得这样的游戏是可行的，即所有的家庭成员可以从各自的起点出发依照游戏规则到达各自的终点。这样的最佳迷宫称为第二类最佳迷宫。（占每个测试点分值的 70%）

输入输出要求

输入要求

输入文件

第一行是两个整数，依次表示 N，M 的值；

第二行是一个整数 K，表示原迷宫中门的总个数；

第 i+2 行（1≤ i≤ K〉，有四个整数，依次为 X_{i1},Y_{i1},X_{i2},Y_{i2},表示第 X_{i1} 行第 Y_{i1} 列的单元与第 X_{i2} 行 Y_{i2} 列的单元之间有一扇门。（其中$|X_{i1}-X_{i2}|+|Y_{i1}-Y_{i2}|=1$）

第 K+3 行是一个整数，表示 P 的值；

第 K+3+j 行（1≤j≤P），有四个整数：X_{j1},Y_{j1},X_{j2},Y_{j2},分别表示第 j 个家庭成员出发的起点位置（X_{j1},Y_{j1}）和他要到达的终点位置（X_{j2},Y_{j2}）。（其中 $X_{j1}\le X_{j2}, Y_{j1}\le Y_{j2}, (X_{j1},Y_{j1})\neq(X_{j2},Y_{j2})$）

注意：输入数据同一行各相邻整数之间用一个空格分隔。

输出要求

输出文件

共 P+2 行

第一行是一个整数，表示你所设计的第一类最佳迷宫中，新置的门的个数；

第二行是一个整数，表示你所设计的第二类最佳迷宫总的起点到终点的一条可行路径，其中

第 i+2 行（1≤i≤P）表示第 i 个家庭成员的一条可行路径（包括起点），该行只有一个由大写字母 S 和大写字母 E 组成的字符串，第 j 位字符表示他在路径上的第 j 个单元的移动方向：大写字母 S 表示向南，大写字母 E 表示向东。

样本数据

样本输入文件

```
4 4
5
1 1 1 2
1 4 2 4
2 1 3 1
2 2 3 2
4 2 4 3
3
2 1 4 3
1 2 4 2
3 1 4 4
```

样本输出文件

```
10
4
SESE
SSS
ESEE
```

评分标准

每个测试点规定时间限制，如果在规定时间限制内，选手程序不能够正常

退出，即判超时，该测试点计 0 分，而不考虑输出文件的正误。如在规定时间限制内，选手程序能够正常退出，则考虑输出文件：

第 1 行独立评分，即若第一行的输出正确，得测试点分值的 30%；

第 2~2+*P* 行联合评分，即：

若第 2 行的输出正确，而第 3~2+*P* 行的输出错误，得测试点分值的 40%

若第 2~2+*P* 行的输出都正确，得测试点分值的 70%

参数限定

$3 \le N$，$M \le 20$；$1 \le P \le 3$

备注

源程序的文件名：Maze.pas/Maze.c/Maze.cpp/Maze.bas

可执行文件的文件名：Maze.exe

输入文件的文件名键盘输入
输出文件的文件名键盘输入

2. 《迷宫改造》解法 1 的程序：

```
{written by Xiao Tian}
{Algorithm : dynamic programming}
Program Maze;
Const
    Ofs:Array[0..3,1..2] Of Integer{^<>v}
    =((-1,0),(0,-1),(0,1),(1,0));
    cUp=0; cLeft=1; cRight=2; cDown=3;
    InName='Maze.in0';
    OutName='Maze.out';
    MM=60;
Type
    qType=Array[0..8] Of LongInt;
Var
    InFile,OutFile:Text;
    Map:Array[1..MM,1..MM,0..3] Of ShortInt;
    N,M,DoorC,ManC:LongInt;
    Data:Array[1..3,1..4] Of Integer;
    i,j,k,Int1,Int2,Int3,Int4:LongInt;
    x:Array[1..3] Of Integer;
    Top,Bottom:Array[0..3] Of LongInt;
    S:Array[1..MM*2-1,0..MM,0..MM,0..MM] Of Integer;
```

```
    IsInWay:Array[1..3,1..MM,1..MM] Of ShortInt;
    {-1=not in his way; 0=just at the start point; 1=in his way}
    IsCanMoveUp,IsCanMoveLeft:Array[1..3,1..MM,1..MM] Of Boolean;
    wUp,wLeft:Array[1..3] Of Integer;
    q,qInit:qType;
Function MinOfArr(q:qType;MaxIndex:Integer):Integer;
{find the smallest element in q[]}
Var
    i,P:Integer;
Begin
    q[0]:=MaxInt;
    P:=0;
    For i:=1 To MaxIndex Do
     If q[i]<q[P] Then
        P:=i;
    MinOfArr:=q[P];
End;
Function Max(Int1,Int2:Integer):Integer;
Begin
    If Int1>Int2 Then Max:=Int1 Else Max:=Int2;
End;
Function Min(Int1,Int2:Integer):Integer;
Begin
    If Int1<Int2 Then Min:=Int1 Else Min:=Int2;
End;
Begin
    Assign(InFile,InName);
    Reset(InFile);
    Assign(OutFile,OutName);
    Rewrite(OutFile);

    ReadLn(InFile,N,M);
    ReadLn(InFile,DoorC);
    FillChar(Map,SizeOf(Map),1);
    For i:=1 To DoorC Do
    Begin
        ReadLn(InFile,Int1,Int2,Int3,Int4);
        For j:=0 To 3 Do
         If (Int1+Ofs[j,1]=Int3) And (Int2+Ofs[j,2]=Int4) Then
        Begin
            Map[Int1,Int2,j]:=0;
            Map[Int3,Int4,3-j]:=0;
            Break;
        End;
```

```
End;
ReadLn(InFile,ManC);
For i:=1 To ManC Do
    ReadLn(InFile,Data[i,1],Data[i,2],Data[i,3],Data[i,4]);
Close(InFile);

For i:=ManC+1 To 3 Do
Begin
    Data[i,1]:=N;
    Data[i,2]:=M;
    Data[i,3]:=N;
    Data[i,4]:=M;
End;

{Problem 2}
{preprocessing}
For i:=1 To 8 Do qInit[i]:=MaxInt;
FillChar(IsCanMoveUp,SizeOf(IsCanMoveUp),False);
FillChar(IsCanMoveLeft,SizeOf(IsCanMoveLeft),False);
For i:=1 To 3 Do
Begin
    For j:=Data[i,1]+1 To Data[i,3] Do
     For k:=Data[i,2] To Data[i,4] Do
        IsCanMoveUp[i,j,k]:=True;
    For j:=Data[i,1] To Data[i,3] Do
     For k:=Data[i,2]+1 To Data[i,4] Do
        IsCanMoveLeft[i][j][k]:=True;
End;

{dp}
FillWord(S,SizeOf(S) Div 2,9999);
S[1,1,1,1]:=0;
For i:=2 To N+M-1 Do
Begin
    Top[0]:=Max(1,i-M+1);
    Bottom[0]:=Min(i,N);
    For j:=1 To 3 Do
        If Data[j,1]+Data[j,2]-1=i Then
        Begin Top[j]:=Data[j,1]; Bottom[j]:=Data[j,1] End
        Else If Data[j,3]+Data[j,4]-1=i Then
        Begin Top[j]:=Data[j,3]; Bottom[j]:=Data[j,3] End
        Else Begin Top[j]:=Top[0]; Bottom[j]:=Bottom[0] End;
    For x[1]:=Top[1] To Bottom[1] Do
     For x[2]:=Top[2] To Bottom[2] Do
```

```
          For x[3]:=Top[3] To Bottom[3] Do
        Begin
            q:=qInit;
            For j:=1 To 3 Do
            Begin
                If IsCanMoveUp[j,x[j],i-x[j]+1] Then
                    wUp[j]:=Map[x[j],i-x[j]+1,cUp]
                Else wUp[j]:=0;
                If IsCanMoveLeft[j,x[j],i-x[j]+1] Then
                    wLeft[j]:=Map[x[j],i-x[j]+1,cLeft]
                Else wLeft[j]:=0;
            End;
            If (x[1]>1) And (x[2]>1) And (x[3]>1) Then
            Begin
                q[1]:=
                   S[i-1,x[1]-1,x[2]-1,x[3]-1]+wUp[1]+wUp[2]+wUp[3];
                If x[1]=x[2] Then Dec(q[1],wUp[1] And wUp[2]);
                If x[2]=x[3] Then Dec(q[1],wUp[2] And wUp[3]);
                If x[3]=x[1] Then Dec(q[1],wUp[3] And wUp[1]);
                 If (x[1]=x[2]) And (x[2]=x[3]) Then Inc(q[1],wUp[1]
And wUp[2] And wUp[3]);
            End;
            If (x[1]<i) And (x[2]>1) And (x[3]>1) Then
            Begin
                q[2]:=
                   S[i-1,x[1],x[2]-1,x[3]-1]+wLeft[1]+wUp[2]+wUp[3];
                If x[2]=x[3] Then Dec(q[2],wUp[2] And wUp[3]);
            End;
            If (x[1]>1) And (x[2]<i) And (x[3]>1) Then
            Begin
                q[3]:=
                   S[i-1,x[1]-1,x[2],x[3]-1]+wUp[1]+wLeft[2]+wUp[3];
                If x[3]=x[1] Then Dec(q[3],wUp[3] And wUp[1]);
            End;
            If (x[1]<i) And (x[2]<i) And (x[3]>1) Then
            Begin
                q[4]:=
                   S[i-1,x[1],x[2],x[3]-1]+wLeft[1]+wLeft[2]+wUp[3];
                If x[1]=x[2] Then Dec(q[4],wLeft[1] And wLeft[2]);
            End;
            If (x[1]>1) And (x[2]>1) And (x[3]<i) Then
            Begin
                q[5]:=
                   S[i-1,x[1]-1,x[2]-1,x[3]]+wUp[1]+wUp[2]+wLeft[3];
```

```
                If x[1]=x[2] Then Dec(q[5],wUp[1] And wUp[2]);
            End;
            If (x[1]<i) And (x[2]>1) And (x[3]<i) Then
            Begin
                q[6]:=
                   S[i-1,x[1],x[2]-1,x[3]]+wLeft[1]+wUp[2]+wLeft[3];
                If x[1]=x[3] Then Dec(q[6],wLeft[1] And wLeft[3]);
            End;
            If (x[1]>1) And (x[2]<i) And (x[3]<i) Then
            Begin
                q[7]:=
                   S[i-1,x[1]-1,x[2],x[3]]+wUp[1]+wLeft[2]+wLeft[3];
                If x[2]=x[3] Then Dec(q[7],wLeft[2] And wLeft[3]);
            End;
            If (x[1]<i) And (x[2]<i) And (x[3]<i) Then
            Begin
                q[8]:=
                   S[i-1,x[1],x[2],x[3]]+wLeft[1]+wLeft[2]+wLeft[3];
                If x[1]=x[2] Then Dec(q[8],wLeft[1] And wLeft[2]);
                If x[2]=x[3] Then Dec(q[8],wLeft[2] And wLeft[3]);
                If x[3]=x[1] Then Dec(q[8],wLeft[3] And wLeft[1]);
                              If (x[1]=x[2]) And (x[2]=x[3]) Then
Inc(q[8],wLeft[1] And wLeft[2] And wLeft[3]);
            End;
            S[i,x[1],x[2],x[3]]:=MinOfArr(q,8);
        End;
    End;

    WriteLn(OutFile,S[M+N-1,N,N,N]);
    Close(OutFile);
End.
```

3. 《迷宫改造》解法 2 的程序（打印路径部分尚有一些 bug）：

```
{written by Xiao Tian}
Program Maze;
Const
    Ofs:Array[0..3,1..2] Of Integer{^<>v}
    =((-1,0),(0,-1),(0,1),(1,0));
    InName='Maze.in0';
    OutName='Maze.out';
    MM=400;
Type
    qType=Array[0..8] Of LongInt;
Var
```

```
    InFile,OutFile:Text;
    Map:Array[1..MM,1..MM,0..3] Of ShortInt;
    N,M,DoorC,ManC:LongInt;
    Data:Array[1..3,1..4] Of Integer;
    Flag:Array[1..MM,1..MM] Of Boolean;
    i,j,k,Int1,Int2,Int3,Int4,Ans:LongInt;
    st1,st2,st3,Delta,X,Y:Integer;
    q,qInit:qType;
    SP:Array[1..3,1..2,1..MM,1..MM] Of Integer;
    SP_Path:Array[1..3,1..2,1..MM,1..MM] Of Char;
    {len of Shortest Path}
    {index : which man, which end, x, y}
    Directly:Array[1..3] Of Integer;
    S:Array[1..MM,1..MM,0..2,0..2,0..2] Of Integer;
    S_Path:Array[1..MM,1..MM,0..2,0..2,0..2] Of ShortInt;
    {index : x, y, states of the 3 men(0=not joined yet, 1=together,
2=left already)}
    AnsMode:ShortInt;
    Path:Array[1..3] Of String;
    CommonStep:Char;
Procedure DFS(X,Y:Integer);
Var
    i:Integer;
Begin
    Flag[X,Y]:=True;
    For i:=0 To 3 Do
     If Map[X,Y,i]=0 Then
      If Not Flag[X+Ofs[i,1],Y+Ofs[i,2]] Then
         DFS(X+Ofs[i,1],Y+Ofs[i,2]);
End;
Function MinOfArr(q:qType;MaxIndex:Integer;Var P:ShortInt):Integer;
{find the smallest element in q[], P is the index of it}
Var
    i:Integer;
Begin
    q[0]:=MaxInt;
    P:=0;
    For i:=1 To MaxIndex Do
     If q[i]<q[P] Then
         P:=i;
    MinOfArr:=q[P];
End;
Function BuildPath(ManID,EndID,X,Y:Integer):String;
Var
```

```
    CurStr:String;
Begin
    CurStr:='';
    While SP_Path[ManID,EndID,X,Y]<>'x' Do
    Begin
        If SP_Path[ManID,EndID,X,Y]='S' Then
        Begin
            If EndID=1 Then
            Begin
                CurStr:='S'+CurStr;
                Dec(X);
            End
            Else Begin
                CurStr:=CurStr+'S';
                Inc(X);
            End;
        End
        Else Begin
            If EndID=1 Then
            Begin
                CurStr:='E'+CurStr;
                Dec(Y);
            End
            Else Begin
                CurStr:=CurStr+'E';
                Inc(Y);
            End;
        End;
    End;
    BuildPath:=CurStr;
End;
Begin
    Assign(InFile,InName);
    Reset(InFile);
    Assign(OutFile,OutName);
    Rewrite(OutFile);

    ReadLn(InFile,N,M);
    ReadLn(InFile,DoorC);
    FillChar(Map,SizeOf(Map),1);
    For i:=1 To DoorC Do
    Begin
        ReadLn(InFile,Int1,Int2,Int3,Int4);
        For j:=0 To 3 Do
```

```
         If (Int1+Ofs[j,1]=Int3) And (Int2+Ofs[j,2]=Int4) Then
        Begin
            Map[Int1,Int2,j]:=0;
            Map[Int3,Int4,3-j]:=0;
            Break;
        End;
    End;
    ReadLn(InFile,ManC);
    For i:=1 To ManC Do
        ReadLn(InFile,Data[i,1],Data[i,2],Data[i,3],Data[i,4]);
    Close(InFile);

    For i:=ManC+1 To 3 Do
    Begin
        Data[i,1]:=N;
        Data[i,2]:=M;
        Data[i,3]:=N;
        Data[i,4]:=M;
    End;
    For i:=1 To 8 Do qInit[i]:=MaxInt;

    {Problem 2}
     {calculate the shortest paths from(to) the 6 ends of the 3 men
to(from) everywhere}
    FillWord(SP,SizeOf(SP) Div 2,MaxInt);
    For i:=1 To 3 Do
    Begin
        {about beginning point}
        SP[i,1,Data[i,1],Data[i,2]]:=0;
        SP_Path[i,1,Data[i,1],Data[i,2]]:='x';
        For j:=Data[i,1]+1 To N Do
        Begin
            SP[i,1,j,Data[i,2]]:=
                 SP[i,1,j-1,Data[i,2]]+Map[j,Data[i,2],0];
            SP_Path[i,1,j,Data[i,2]]:='S';
        End;
        For k:=Data[i,2]+1 To M Do
        Begin
            SP[i,1,Data[i,1],k]:=
                 SP[i,1,Data[i,1],k-1]+Map[Data[i,1],k,1];
            SP_Path[i,1,Data[i,1],k]:='E';
        End;
        For j:=Data[i,1]+1 To N Do
         For k:=Data[i,2]+1 To M Do
```

```
    Begin
        If SP[i,1,j,k]>SP[i,1,j-1,k]+Map[j,k,0] Then
        Begin
            SP[i,1,j,k]:=SP[i,1,j-1,k]+Map[j,k,0];
            SP_Path[i,1,j,k]:='S';
        End;
        If SP[i,1,j,k]>SP[i,1,j,k-1]+Map[j,k,1] Then
        Begin
            SP[i,1,j,k]:=SP[i,1,j,k-1]+Map[j,k,1];
            SP_Path[i,1,j,k]:='E';
        End;
    End;

    {about finishing point}
    SP[i,2,Data[i,3],Data[i,4]]:=0;
    SP_Path[i,2,Data[i,3],Data[i,4]]:='x';
    For j:=Data[i,3]-1 DownTo 1 Do
    Begin
        SP[i,2,j,Data[i,4]]:=
             SP[i,2,j+1,Data[i,4]]+Map[j+1,Data[i,4],0];
        SP_Path[i,2,j,Data[i,4]]:='S';
    End;
    For k:=Data[i,4]-1 DownTo 1 Do
    Begin
        SP[i,2,Data[i,3],k]:=
             SP[i,2,Data[i,3],k+1]+Map[Data[i,3],k+1,1];
        SP_Path[i,2,Data[i,3],k]:='E';
    End;
    For j:=Data[i,3]-1 DownTo 1 Do
     For k:=Data[i,4]-1 DownTo 1 Do
    Begin
        If SP[i,2,j,k]>SP[i,2,j+1,k]+Map[j+1,k,0] Then
        Begin
            SP[i,2,j,k]:=SP[i,2,j+1,k]+Map[j+1,k,0];
            SP_Path[i,2,j,k]:='S';
        End;
        If SP[i,2,j,k]>SP[i,2,j,k+1]+Map[j,k+1,1] Then
        Begin
            SP[i,2,j,k]:=SP[i,2,j,k+1]+Map[j,k+1,1];
            SP_Path[i,2,j,k]:='E';
        End;
    End;
End;
For i:=1 To 3 Do
```

```
        Directly[i]:=SP[i,1,Data[i,3],Data[i,4]];

    {dynamic}
    FillWord(S,SizeOf(S) Div 2,MaxInt);
    S[1,1,0,0,0]:=0;
    S_Path[1,1,0,0,0]:=-1;
    For i:=1 To N Do
     For j:=1 To M Do
      For st1:=0 To 2 Do
       For st2:=0 To 2 Do
        For st3:=0 To 2 Do
    Begin
         If (i=1) And (j=1) And (st1=0) And (st2=0) And (st3=0) Then
Continue;
        q:=qInit;
        If i>1 Then
        Begin
            If (st1<>1) And (st2<>1) And (st3<>1) Then
                Delta:=0
            Else Delta:=Map[i,j,0];
            q[1]:=S[i-1,j,st1,st2,st3]+Delta;
        End;
        If j>1 Then
        Begin
            If (st1<>1) And (st2<>1) And (st3<>1) Then
                Delta:=0
            Else Delta:=Map[i,j,1];
            q[2]:=S[i,j-1,st1,st2,st3]+Delta;
        End;
        If st1=1 Then q[3]:=S[i,j,0,st2,st3]+SP[1,1,i,j];
        If st1=2 Then q[4]:=S[i,j,1,st2,st3]+SP[1,2,i,j];
        If st2=1 Then q[5]:=S[i,j,st1,0,st3]+SP[2,1,i,j];
        If st2=2 Then q[6]:=S[i,j,st1,1,st3]+SP[2,2,i,j];
        If st3=1 Then q[7]:=S[i,j,st1,st2,0]+SP[3,1,i,j];
        If st3=2 Then q[8]:=S[i,j,st1,st2,1]+SP[3,2,i,j];
        S[i,j,st1,st2,st3]:=MinOfArr(q,8,S_Path[i,j,st1,st2,st3]);
    End;

    {get the answer}
    q[1]:=S[N,M,2,2,2];
    q[2]:=S[N,M,0,2,2]+Directly[1];
    q[3]:=S[N,M,2,0,2]+Directly[2];
    q[4]:=S[N,M,2,2,0]+Directly[3];
    q[5]:=S[N,M,0,0,2]+Directly[1]+Directly[2];
```

```
    q[6]:=S[N,M,2,0,0]+Directly[2]+Directly[3];
    q[7]:=S[N,M,0,2,0]+Directly[3]+Directly[1];
    q[8]:=S[N,M,0,0,0]+Directly[1]+Directly[2]+Directly[3];
    Ans:=MinOfArr(q,8,AnsMode);

    WriteLn(OutFile,Ans);
(* there is some bug in this period of code
    {build the path of the 3}
    For i:=1 To 3 Do Path[i]:='';
    X:=N; Y:=M;
    Case AnsMode Of
    1:Begin st1:=2; st2:=2; st3:=2; End;
    2:Begin st1:=0; st2:=2; st3:=2; End;
    3:Begin st1:=2; st2:=0; st3:=2; End;
    4:Begin st1:=2; st2:=2; st3:=0; End;
    5:Begin st1:=0; st2:=0; st3:=2; End;
    6:Begin st1:=2; st2:=0; st3:=0; End;
    7:Begin st1:=0; st2:=2; st3:=0; End;
    8:Begin st1:=0; st2:=0; st3:=0; End;
    End;
    Repeat
        CommonStep:='x';
        Case S_Path[X,Y,st1,st2,st3] Of
        1:Begin CommonStep:='S'; Dec(X); End;
        2:Begin CommonStep:='E'; Dec(Y); End;
        3:Begin Path[1]:=BuildPath(1,1,X,Y)+Path[1]; Dec(st1); End;
        4:Begin Path[1]:=BuildPath(1,2,X,Y)+Path[1]; Dec(st1); End;
        5:Begin Path[2]:=BuildPath(2,1,X,Y)+Path[2]; Dec(st2); End;
        6:Begin Path[2]:=BuildPath(2,2,X,Y)+Path[2]; Dec(st2); End;
        7:Begin Path[3]:=BuildPath(3,1,X,Y)+Path[3]; Dec(st3); End;
        8:Begin Path[3]:=BuildPath(3,2,X,Y)+Path[3]; Dec(st3); End;
        -1:Break;
        End;
        If CommonStep<>'x' Then
        Begin
            If st1=1 Then Path[1]:=CommonStep+Path[1];
            If st2=1 Then Path[2]:=CommonStep+Path[2];
            If st3=1 Then Path[3]:=CommonStep+Path[3];
        End;
    Until False;

    For i:=1 To ManC Do WriteLn(OutFile,Path[i]);*)
    Close(OutFile);
End.
```