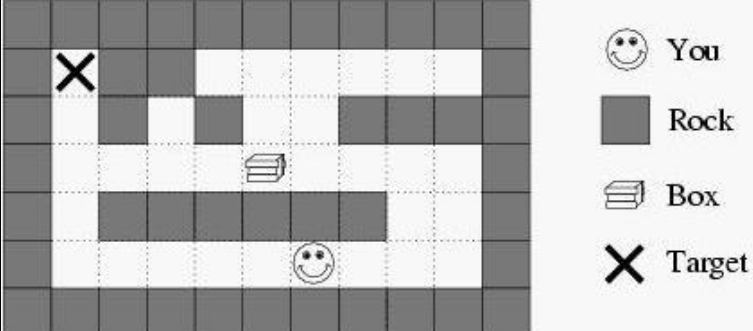
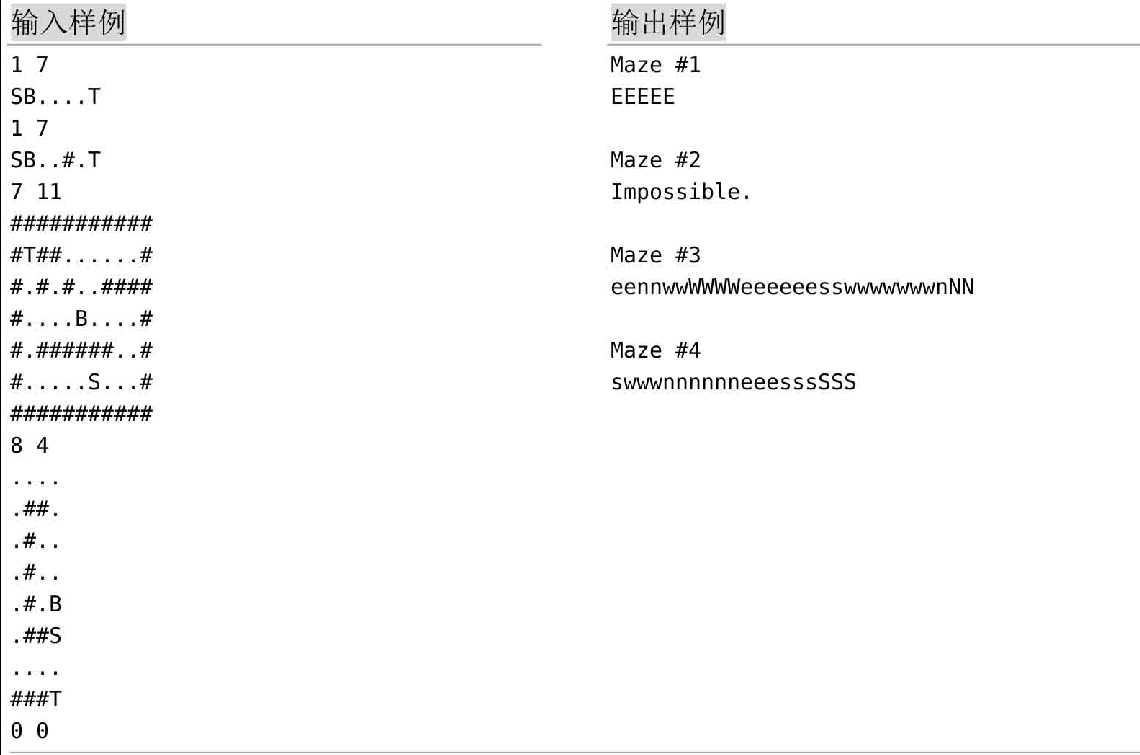
**POJ1475**

**题目描述（POJ1475）：**想象一下，你站在一个由方格组成的二维迷宫里，这些格子可能被填满岩石，也可能没被填满岩石。你可以一步一个格子地往北、往南、往东或往西移动。这样的动作叫作“走”。其中一个空单元格包含一个箱子，你可以站在箱子旁边，推动箱子到相邻的自由单元格。这样的动作叫作“推”。箱子除了用推的方式，不能移动，如果你把它推到角落里，就再也不能把它从角落里拿出来了。将其中一个空单元格标记为目标单元格。你的工作是通过一系列走和推把箱子带到目标格子里。由于箱子很重，所以要尽量减少推的次数。编写程序，计算最好的移动（走和推）顺序。



**输入：**输入包含多个测试用例。每个测试用例的第1行都包含两个整数r和c（均小于或等于20），表示迷宫的行数和列数。接下来是r行，每行都包含c个字符，每个字符都描述迷宫中的一个格子，对被填满岩石的格子用“#”表示，对空格用“.”表示。对起始位置用“S”表示，对箱子的起始位置用“B”表示，对目标单元格用“T”表示。输入端以两个0终止。

**输出：**对于输入中的每个迷宫，都首先输出迷宫的编号。如果无法将箱子带到目标单元格里，则输出“Impossible.”，否则输出一个最小推送次数的序列。如果有多个这样的序列，则请选择一个最小总移动（走和推）次数的序列。如果仍然有多个这样的序列，则任何一个都可被接受。将序列输出为由字符N、S、E、W、n、s、e和w组成的字符串，大写字母表示推，小写字母表示走，字母分别表示北、南、东和西这4个方向。在每个测试用例之后都输出一个空行。



**题解：**本题为推箱子问题，要求先保证推箱子的次数最少，在此基础上再让人走的总步数最少。推箱子时，人只有站在箱子反方向的前一个位置，才可以将箱子推向下一个位置，如下图所示。很明显，图中的箱子无法向上移动，因为人无法到达箱子的下面位置。因此在移动箱子时，不仅需要判断新位置有没有岩石，还需要判断人是否可以到达反方向的前一个位置，在两者均有效时，才会让人移动。

**先求解箱子到目标位置的最短路径（BFS1）**，在推箱子的过程中，每推一步，都根据推的方向和箱子的位置得到箱子的前一个位置，**再求解人到达这个位置的最短路径（BFS2）**。在**BFS1里面嵌套了BFS2，属于嵌套广度优先搜索**。

**1. 算法设计**

（1）定义一个标识数组vis[][]并将其初始化为0，标识所有位置都未被访问。

（2）创建一个队列q维护箱子的状态，将人的初始位置(sx, sy)、箱子的初始位置(bx, by)和初始路径("")入队，标记箱子的位置vis[bx][by]=1。

（3）如果队列不空，则队头now出队，否则返回false。

（4）从箱子的当前位置开始，向北、南、东和西这4个方向扩展。

• 得到箱子的新位置：nbx=now.bx+dir[i][0]; nby=now.by+dir[i][1]。

• 得到箱子的前一个位置：tx=now.bx-dir[i][0]; ty=now.by-dir[i][1]。

• 如果这两个位置有效，则执行BFS2搜索人到达箱子的前一个位置(tx, ty)的最短路径，并记录路径path。如果BFS2搜索成功，则判断是否达到目标，如果是，则返回答案ans=now.path+path+dpathB[i]；否则标记箱子的新位置被访问vis[nbx][nby]=1，将人的新位置(now.bx,now.by)、箱子的新位置(nbx,nby)和已走过的路径(now.path+path+ dpathB[i])入队。

（5）转向步骤3。

**2. 算法实现**

