[报告] F - Find the Winning Move

[Source]

http://acm.zju.edu.cn/onlinejudge/showProblem.do?problemCode=1703

[Description]

在 4×4 棋盘上的 tic-tac-toe。定义 forced win：在 x 的回合，存在一步 x 的下法，在这之后无论 o 怎么走都有 x 必胜。现给出一种局面，此时是 x 的回合，按照从上到下、从左到右的顺序找出第一个能够形成 forced win 的 x 的下法。

[Solution]

我的思路就是先在题目给出的局面上枚举 x 下一步的位置。再以此为基础，dfs 回溯枚举接下来 o 和 x 可能的下法。要想有 x 必胜，须(1)在 o 的回合，对于所有 o 的下法都有 x 必胜；(2)在 x 的回合，存在 x 的下法使 x 必胜。

用一个字符数组 board 保存当前局面。定义 int r_x[4], r_o[4], c_x[4], c_o[4], d_x[2], d_o[2];记录各行、列、对角线上 x、o 出现的次数，用来辅助判断游戏胜负情况。具体程序见代码和注释吧（略渣见谅）。

另外在网上找到两组感觉挺管用的数据：

Input

?
.ooo
.x..
xx..
x...
?
.o..
.xx.
xo.o
.ox.
$

Output

(0,0)
(0,0)

[Code]

```
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<queue>
using namespace std;
char T[5], board[4][10];
int r_x[4], r_o[4], c_x[4], c_o[4], d_x[2], d_o[2];
void add_x(int i, int j) //在(i,j)下 x
{
    board[i][j] = 'x';
```

```
    if (i == j)
        d_x[0]++;
    if (i+j == 3)
        d_x[1]++;
    r_x[i]++;
    c_x[j]++;
}
void del_x(int i, int j) //撤销在(i,j)下 x
{
    board[i][j] = '.';
    if (i == j)
        d_x[0]--;
    if (i+j == 3)
        d_x[1]--;
    r_x[i]--;
    c_x[j]--;
}
void add_o(int i, int j) //在(i,j)下 o
{
    board[i][j] = 'o';
    if (i == j)
        d_o[0]++;
    if (i+j == 3)
        d_o[1]++;
    r_o[i]++;
    c_o[j]++;
}
void del_o(int i, int j) //撤销在(i,j)下 o
{
    board[i][j] = '.';
    if (i == j)
        d_o[0]--;
    if (i+j == 3)
        d_o[1]--;
    r_o[i]--;
    c_o[j]--;
}
bool Search(int x) //Search(1)表示当前局面 x 走下一步，x 是否必胜；Search(0)表示当前局面 o 走下一步，x 是否必胜
{
    if (x)
    {
        int cnt = 0; //此时为 x 的回合，若存在某行 x 冲 3，则 x 必胜
        for (int k = 0; k < 4; k++)
```

```
            if (r_x[k] == 3 && r_o[k] == 0)
                cnt++;
        for (int k = 0; k < 4; k++)
            if (c_x[k] == 3 && c_o[k] == 0)
                cnt++;
        for (int k = 0; k < 2; k++)
            if (d_x[k] == 3 && d_o[k] == 0)
                cnt++;
        if (cnt) return 1;
    }
    bool flag_o = 1; //在 o 的回合，存在一步 o 使 x 不能必胜时就为 false，对于所有 o 的下法都有 x 必胜时，
才 true
    bool flag_x = 0; //在 x 的回合， 只要存在 x 的下法使 x 必胜， 就 true
    int cnt_ = 0; //枚举过的合法下法的次数
    for (int i = 0; i < 4; i++)
        for (int j = 0; j < 4; j++)
            if (board[i][j] == '.' && flag_o && !flag_x)
            {
                cnt_++;
                if (x)
                {
                    add_x(i, j);
                    if (Search(0)) flag_x = 1;
                    del_x(i, j);
                }
                else
                {
                    add_o(i, j);
                    if (!Search(1)) flag_o = 0;
                    del_o(i, j);
                }
            }
    if (!cnt_) return 0; //cnt_为零则棋盘满了游戏结束， 此时必为 o 的回合
    return x ? flag_x : flag_o;
}
int main()
{
    while (fgets(T, 5, stdin))
    {
        if (T[0] == '$') break;
        for (int i = 0; i < 4; i++)
            fgets(board[i], 10, stdin);
        memset(r_x, 0, sizeof(r_x));
        memset(r_o, 0, sizeof(r_o));
```

```
        memset(c_x, 0, sizeof(c_x));
        memset(c_o, 0, sizeof(c_o));
        memset(d_x, 0, sizeof(d_x));
        memset(d_o, 0, sizeof(d_o));
        for (int i = 0; i < 4; i++)
            for (int j = 0; j < 4; j++)
            {
                if (board[i][j] == 'x')
                    add_x(i, j);
                if (board[i][j] == 'o')
                    add_o(i, j);
            }
        bool flag = 0; //为 true 则存在 forced win 下法
        for (int i = 0; i < 4; i++)
            for (int j = 0; j < 4; j++)
                if (board[i][j] == '.' && !flag)
                {
                    add_x(i, j);
                    int cnt = 0; //记录 4 个 x 连成一条线的个数
                    for (int k = 0; k < 4; k++)
                        if (r_x[k] == 4) cnt++;
                    for (int k = 0; k < 4; k++)
                        if (c_x[k] == 4) cnt++;
                    for (int k = 0; k < 2; k++)
                        if (d_x[k] == 4) cnt++;
                    if (cnt || Search(0)) //cnt>0 说明下完这一步后 x 就已经赢了，不需要再 dfs
                        flag = 1, printf("(%d,%d)\n", i, j);
                    del_x(i, j);
                }
        if (!flag) printf("#####\n");
    }
    return 0;
}
```