School of Data and Computer Science, Sun Yat-sen University

E04 Futoshiki Puzzle (Forward Checking)

16110917 Zhaoshuai Liu

September 28, 2018

Contents

1 Futoshiki

Futoshiki is a board-based puzzle game, also known under the name Unequal. It is playable on a square board having a given fixed size (4×4 for example).

The purpose of the game is to discover the digits hidden inside the board's cells; each cell is filled with a digit between 1 and the board's size. On each row and column each digit appears exactly once; therefore, when revealed, the digits of the board form a so-called Latin square.

At the beginning of the game some digits might be revealed. The board might also contain some inequalities between the board cells; these inequalities must be respected and can be used as clues in order to discover the remaining hidden digits.

Each puzzle is guaranteed to have a solution and only one.

You can play this game online: `http://www.futoshiki.org/`.

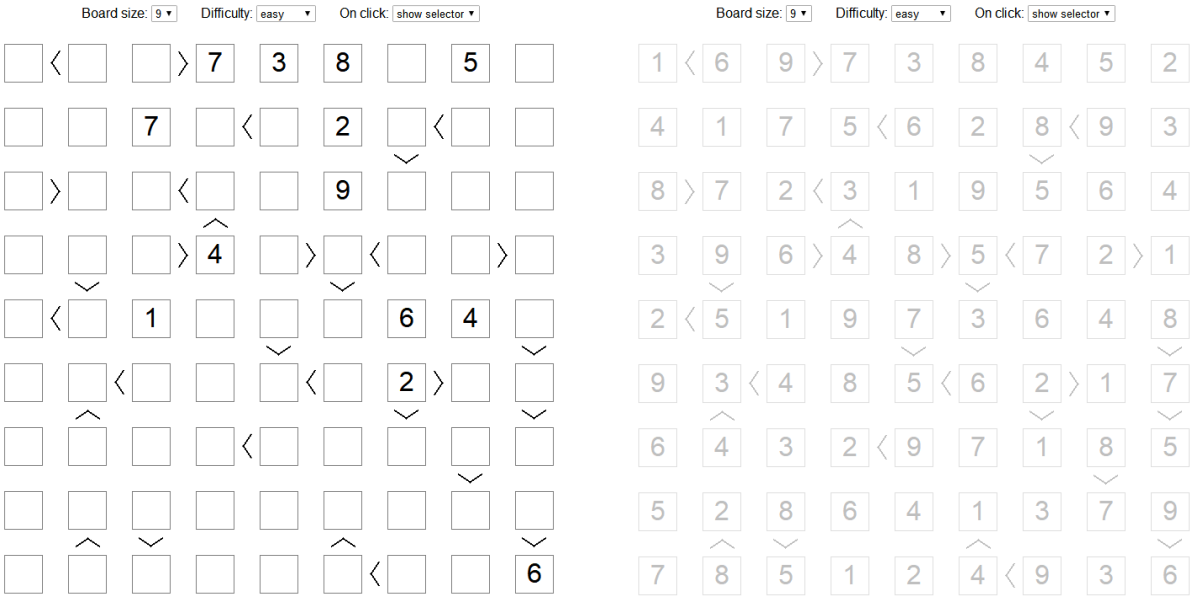

Figure 1: An Futoshiki Puzzle

2 Tasks

1. Please solve the above Futoshiki puzzle (Figure **??**) with forward checking algorithm.
2. Write the related codes and take a screenshot of the running results in the file named E04_YourNumber.pdf, and send it to ai_2018@foxmail.com.

3 Codes

```
#include <iostream>
#include <cstdio>
#include <queue>
#include <cstring>
#include <vector>
#include <ctime>
#include <cstdlib>
#include <algorithm>

using namespace std;

// board
static int board[9][9] = { { 0, 0, 0, 7, 3, 8, 0, 5, 0 },
{ 0, 0, 7, 0, 0, 2, 0, 0, 0 },
{ 0, 0, 0, 0, 0, 9, 0, 0, 0 },
{ 0, 0, 0, 4, 0, 0, 0, 0, 0 },
{ 0, 0, 1, 0, 0, 0, 6, 4, 0 },
{ 0, 0, 0, 0, 0, 0, 2, 0, 0 },
{ 0, 0, 0, 0, 0, 0, 0, 0, 0 },
{ 0, 0, 0, 0, 0, 0, 0, 0, 0 },
{ 0, 0, 0, 0, 0, 0, 0, 0, 6 } };

// The state of a position
struct Node {
        int row, col;
        // able_put_down[i] = 1
        // means board[row][col] = i + 1 is valid
        // if able_put_down[i] <= 0, you can't "put down"
        // i + 1 on board[row][col]
        int able_put_down[9];
        // states_num means how many numbers
```

```
        // can be put down on board[row][col]
        int states_num;
        Node(int row = 0, int col = 0) :row(row), col(col),
                states_num(9) {
                for (int i = 0; i < 9; ++i) {
                        able_put_down[i] = 1;
                }
        }
}node_board[9][9];

// Compare struct
struct myCmp {
        bool operator()(Node *a, Node *b) {
                if (a->states_num == b->states_num) {
                        if (a->row > b->row)return true;
                        else if (a->row == b->row) {
                                return a->col > b->col;
                        }
                }
                // sorted by states_num
                return a->states_num > b->states_num;
        }
}obj;

// state[i * 9 + j][x * 9 + y] = -1 means board[i][j]
// must be smaller than board[x][y];
// state[i * 9 + j][x * 9 + y] = -1 means board[i][j]
// must be bigger than board[x][y];
static int state_compare[81][81];

// update the able_put_down and states_num in the row and col
// is_cancel = 0 means "put down" num on board[row][col],
// is_cancel = 1 means "pick up" num on board[row][col]
```

```
void updateCanPutDown(int num, int row, int col, bool is_cancel
    = 0) {
        int row_steps[4] = { 0, -1, 0, 1 },
                col_steps[4] = { -1, 0, 1, 0 };
        // update the row
        for (int i = 0; i < 9; ++i) {
                if (is_cancel) {
                        node_board[row][i].able_put_down[num -
                            1]++;
                        if (node_board[row][i].able_put_down[num
                             - 1] > 0)
                        {
                                node_board[row][i].states_num++;
                        }
                }
                else {
                        if (node_board[row][i].able_put_down[num
                             - 1] > 0)
                        {
                                node_board[row][i].states_num--;
                        }
                        node_board[row][i].able_put_down[num -
                            1]--;
                }
        }
        // update the col
        for (int i = 0; i < 9; ++i) {
                if (is_cancel) {
                        node_board[i][col].able_put_down[num -
                            1]++;
                        if (node_board[i][col].able_put_down[num
                             - 1] > 0)
                        {
```

```
                        node_board[i][col].states_num++;
                }
        }
        else {
                if (node_board[i][col].able_put_down[num
                    - 1] > 0)
                {
                        node_board[i][col].states_num--;
                }
                node_board[i][col].able_put_down[num -
                   1]--;
        }
}
// update the neighbour(left, up, right, down)
int now_pos = row * 9 + col, next_pos = 0, next_row = 0,
    next_col = 0;
for (int i = 0; i < 4; ++i) {
        next_row = row + row_steps[i];
        next_col = col + col_steps[i];
        if (next_row < 0 || next_row >= 9 || next_col <
           0
                || next_col >= 9) {
                continue;
        }
        next_pos = next_row * 9 + next_col;
        if (state_compare[now_pos][next_pos] < 0) {
                for (int i = 0; i < num - 1; ++i) {
                        if (is_cancel) {
                                node_board[next_row][
                                   next_col].
                                   able_put_down[i]++;
                                if (node_board[next_row
                                   ][next_col].
```

```
                                    able_put_down[i] > 0)
                                    {
                                        node_board[
                                            next_row][
                                            next_col].
                                            states_num++;
                                    }
                                }
                                else {
                                    if (node_board[next_row
                                        ][next_col].
                                        able_put_down[i] > 0)
                                        {
                                        node_board[
                                            next_row][
                                            next_col].
                                            states_num--;
                                    }
                                    node_board[next_row][
                                        next_col].
                                        able_put_down[i]--;
                                }
                            }
                        }
                        else if (state_compare[now_pos][next_pos] > 0) {
                            for (int i = num; i < 9; ++i) {
                                if (is_cancel) {
                                    node_board[next_row][
                                        next_col].
                                        able_put_down[i]++;
                                    if (node_board[next_row
                                        ][next_col].
                                        able_put_down[i] > 0)
```

```
                                        {
                                            node_board[
                                                next_row][
                                                next_col].
                                                states_num++;
                                        }
                                    }
                                    else {
                                        if (node_board[next_row
                                            ][next_col].
                                            able_put_down[i] > 0)
                                            {
                                            node_board[
                                                next_row][
                                                next_col].
                                                states_num--;
                                        }
                                        node_board[next_row][
                                            next_col].
                                            able_put_down[i]--;
                                    }
                                }
                            }
                        }
                }
        }

        void init() {
                // init the relation between the state
                state_compare[0][1] = -1;
                state_compare[1][0] = 1;
                state_compare[2][3] = 1;
                state_compare[3][2] = -1;
                state_compare[12][13] = -1;
```

```
state_compare[13][12] = 1;
state_compare[15][16] = -1;
state_compare[16][15] = 1;
state_compare[15][24] = 1;
state_compare[24][15] = -1;
state_compare[18][19] = 1;
state_compare[19][18] = -1;
state_compare[20][21] = -1;
state_compare[21][20] = 1;
state_compare[21][30] = -1;
state_compare[30][21] = 1;
state_compare[28][37] = 1;
state_compare[37][28] = -1;
state_compare[29][30] = 1;
state_compare[30][29] = -1;
state_compare[31][32] = 1;
state_compare[32][31] = -1;
state_compare[32][41] = 1;
state_compare[41][32] = -1;
state_compare[32][33] = -1;
state_compare[33][32] = 1;
state_compare[34][35] = 1;
state_compare[35][34] = -1;
state_compare[36][37] = -1;
state_compare[37][36] = 1;
state_compare[40][49] = 1;
state_compare[49][40] = -1;
state_compare[44][53] = 1;
state_compare[53][44] = -1;
state_compare[46][55] = -1;
state_compare[55][46] = 1;
state_compare[46][47] = -1;
state_compare[47][46] = 1;
```

```
state_compare[49][50] = -1;
state_compare[50][49] = 1;
state_compare[51][52] = 1;
state_compare[52][51] = -1;
state_compare[51][60] = 1;
state_compare[60][51] = -1;
state_compare[53][62] = 1;
state_compare[62][53] = -1;
state_compare[57][58] = -1;
state_compare[58][57] = 1;
state_compare[61][70] = 1;
state_compare[70][61] = -1;
state_compare[64][73] = -1;
state_compare[73][64] = 1;
state_compare[65][74] = 1;
state_compare[74][65] = -1;
state_compare[68][77] = -1;
state_compare[77][68] = 1;
state_compare[71][80] = 1;
state_compare[80][71] = -1;
state_compare[77][78] = -1;
state_compare[78][77] = 1;
// init the board
for (int i = 0; i < 9; ++i) {
        for (int j = 0; j < 9; ++j) {
                node_board[i][j].row = i;
                node_board[i][j].col = j;
                if (board[i][j] > 0) {
                        updateCanPutDown(board[i][j], i,
                            j);
                }
                for (int x = 0; x < 9; ++x) {
                        for (int y = 0; y < 9; ++y) {
```

```
                                        if (abs(state_compare[i
                                            * 9 + j][x * 9 + y])
                                            != 1) {
                                                state_compare[i
                                                    * 9 + j][x *
                                                    9 + y] = 0;
                                        }
                                    }
                                }
                            }
                        }
                    }

// solve the question
bool run(int board[9][9], vector<Node*> &search_queue, int depth
    ) {
        if (search_queue.size() == 0) {
                return true;
        }
        Node *now_node = search_queue.back();
        if (now_node->states_num <= 0) {
                return false;
        }
        int now_row = now_node->row, now_col = now_node->col;
        bool flag = false;
        for (int num = 1; num <= 9; ++num) {
                if (node_board[now_row][now_col].able_put_down[
                    num - 1] > 0) {
                        // "put down" num on board[now_row][
                            now_col]
                        board[now_row][now_col] = num;
                        updateCanPutDown(num, now_row, now_col);
                        search_queue.pop_back();
```

```
                        sort(search_queue.begin(), search_queue.
                            end(), obj);
                        flag = run(board, search_queue, depth +
                            1);
                        if (flag)break;
                        // "pick up" num on board[now_row][
                            now_col]
                        updateCanPutDown(num, now_row, now_col,
                            1);
                        search_queue.push_back(now_node);
                        sort(search_queue.begin(), search_queue.
                            end(), obj);
                        board[now_row][now_col] = 0;
                }
        }
        return flag;
}

int main() {
        init();
        vector<Node*> search_queue;
        double t = clock();
        for (int i = 0; i < 9; ++i) {
                for (int j = 0; j < 9; ++j) {
                        if (board[i][j] == 0) {
                                search_queue.push_back(&
                                    node_board[i][j]);
                        }
                }
        }
        sort(search_queue.begin(), search_queue.end(), obj);
        if (run(board, search_queue, 0)) {
                for (int i = 0; i < 9; ++i) {
```

```
                        for (int j = 0; j < 9; ++j) {
                                printf("%d ", board[i][j]);
                        }
                        printf("\n");
                }
        }
        else {
                printf("False\n");
        }
        printf("Time: %lfs\n", (clock() - t) / CLOCKS_PER_SEC);
        system("pause");
        return 0;
}
```

4 Results

- The all code is in E04.cpp.
- You can run the E04.exe to see the results below.
- It's recommended to compile the code in VS2017