37. Sudoku Solver

题目描述：https://leetcode.com/problems/sudoku-solver/

> 解数独～～

解题思路：

递归

代码：

```
class Solution {
public:
    bool check(vector<vector<char> >& board, vector<vector<int> > &row, vector<vector<int> > &col, vector<vector<int> > &block, int i, int j, int num) {
        int k = i/3*3+j/3;
        if(row[i][num-1]||col[j][num-1]||block[k][num-1])
            return false;
        return true;
    }
    bool findSolve(vector<vector<char>>& board, vector<vector<int> > &row, vector<vector<int> > &col, vector<vector<int> > &block, int i, int j) {
        int n = board.size();
        if(i >= n) {
            return true;
        }
        if(j >= n) {
            return findSolve(board, row, col, block, i+1, 0);
        }
        if(board[i][j] != '.')
            return findSolve(board, row, col, block, i, j+1);
        for(int l = 1; l <= n; l++) {
            if(check(board, row, col, block, i, j, l)) {
                board[i][j] = l+'0';
                row[i][l-1] = 1;
                col[j][l-1] = 1;
                block[i/3*3+j/3][l-1] = 1;
                if(findSolve(board, row, col, block, i, j+1)) {
                    return true;
                }
                else {
                    board[i][j] = '.';
                    row[i][l-1] = 0;
                    col[j][l-1] = 0;
                    block[i/3*3+j/3][l-1] = 0;
                }
            }
        }
        return false;
    }
    void solveSudoku(vector<vector<char>>& board) {
        int n = board.size();
        vector<vector<int> > row(n, vector<int>(n, 0));
        vector<vector<int> > col(n, vector<int>(n, 0));
        vector<vector<int> > block(n, vector<int>(n, 0));
        for(int i = 0; i < n; i++) {
            for(int j = 0; j < n; j++) {
                int k = i/3*3+j/3;
                if(board[i][j] == '.') {
                    continue;
                }
                row[i][board[i][j]-'1'] = 1;
                col[j][board[i][j]-'1'] = 1;
                block[k][board[i][j]-'1'] = 1;
            }
        }
        findSolve(board, row, col, block, 0, 0);
        return ;
    }
};
```