# 算法思路分析

先对当前空白处进行尝试性填充，如果填充有效，则继续；如果无效，则重置为空白；不断递归，直到找到解或者处于没有解的情况[题目中表明一定存在一个解，所以最后返回时一定找到了解]。

步骤：

1. corner case：数组为空，数盘不是9x9；直接返回；

2. 使用回溯法进行问题求解；从左上角0,0开始

2.1 如果当前单元格为空，用1-9进行逐个尝试性填充，

2.2 然后使用isValid方法进行有效性验证，确保所在行、列、3x3小方格内没有重复数字出现；如果出现，返回false，进行回退，将单元格重置为空；如果没有出现，进行递归，继续进行回溯法判断，知道找到最终解，返回。

class Solution {

public:

void solveSudoku(vector<vector<char>>& board) {

if (board.empty() || board.size() != 9 || board[0].size() != 9)

return;

dfs(board, 0, 0);

}

bool isValid(vector<vector<char>>& board, int i, int j){

for (int t=0; t< 9; t++){

if (t != j && board[i][t] == board[i][j]) return false;

if (t != i && board[t][j] == board[i][j]) return false;

}

int row = i / 3 \* 3, col = j / 3 \* 3;

for (int m=row; m< row+3; m++){

for (int n=col; n< col+3; n++){

if (m!=i && n!=j && board[m][n] == board[i][j])

return false;

}

}

return true;

}

bool dfs(vector<vector<char>>& board, int i, int j){

if (i >= 9) return true;

if (j >= 9) return dfs(board, i+1, 0);

if (board[i][j] == '.'){

for (char t='1'; t<= '9'; t++){

board[i][j] = t;

if (isValid(board, i, j)){

if (dfs(board, i, j+1)) return true;

}

board[i][j] = '.';

}

}

else

return dfs(board, i, j+1);

return false;

}

};