算法笔记

1. 双指针算法
2. 滑动窗口算法
3. 动态规划算法(DP)
4. 深度优先搜搜算法(DFS)
5. 广度优先搜索算法(BFS)
6. 二叉树(BT)和二叉搜索树(BST)
7. 二分算法
8. 排序算法(快排和归并排序)
9. 单调栈和单调队列

10、回溯算法

(1).N皇后问题

#inlcude <vector>

#inlcude <algorithm>

using namespace std;

vector<vector<string>> res;

vector<vector<string>> solveNQueens(int n)

{

vector<string> board(n, string(n, ‘.’));

backtrack(board, 0);

return res;

}

Void backtrack(vector<string>& biard, int row)

{

If(row == board.size())

{

res.push\_back(board);

Return;

}

int n = board[row].size();

for(int col = 0; col < n; col++)

{

if(!isValid(board, row, col))

continue;

board[row][col] = ‘Q’;

backtrack(board, row + 1);

board[row][col] = ‘.’;

}

}

Bool isValid(vector<string>& board, int row, int col)

{

int n = board.size();

for(int i = 0; i < n; i++)

{

if(board[i][col] == ‘Q’)

return false;

}

for(int i = row – 1, j = col + 1; i >=0, j < n; i--, j++)

{

if (board[i][j] == ‘Q’)

return false;

}

for (int i = row – 1, j = col – 1; i >= 0, j >= 0; i--, j--)

{

if (board[i][j] == ‘Q’)

return false;

}

return true;

}

数独问题

全排列问题

组合问题

子集问题

11、位运算

12、并查集

13、贪心算法

14、KMP算法