算法设计与分析

分支限界法作业

姓名

班级

学号

电话

Email

日期

一、题目重述

用分支限界法解 4 皇后问题（要求：指明搜索范围，解空间结构，搜索时用的剪枝函数，并画出搜索树）

二、问题解答

搜索范围：

搜索范围是所有可能的棋盘布局，对于 4 皇后问题，即一个 4x4 的棋盘上所有不同的皇后摆放方式。

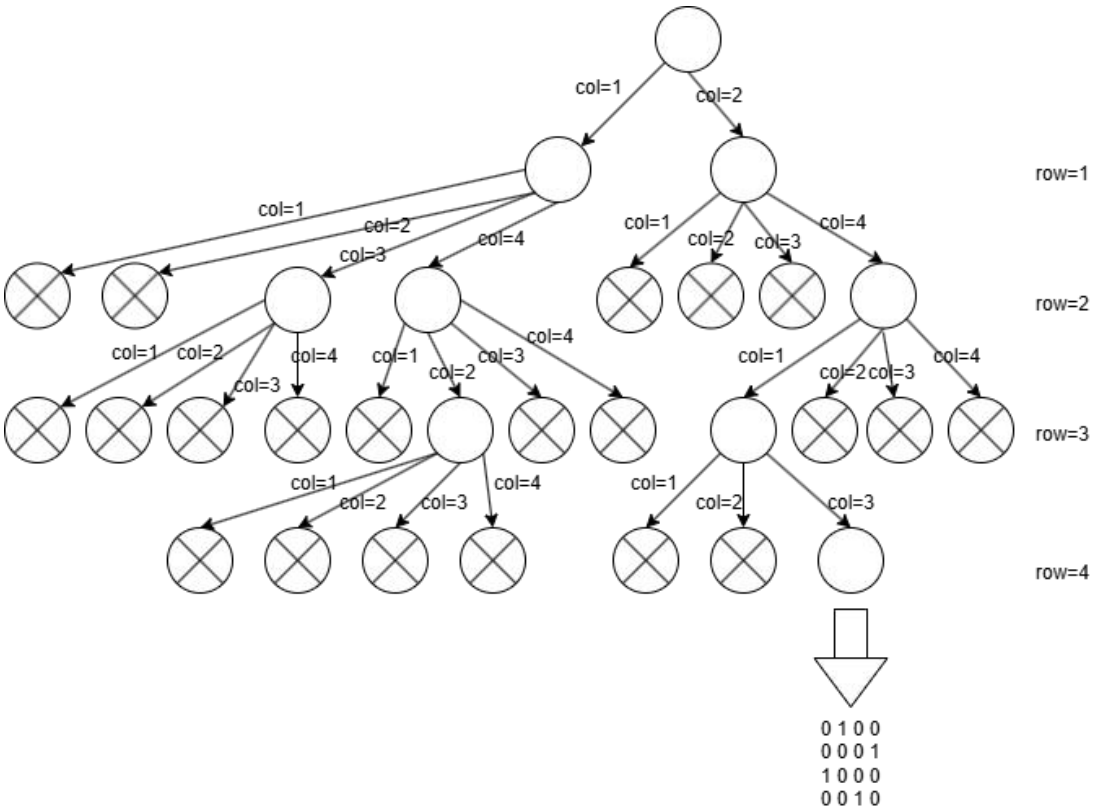
解空间结构：

解空间可以表示为一棵树，树的每一层代表棋盘上的每一行，每个节点代表在该行的某一列放置一个皇后。整棵树包含了所有可能的放置方式。

剪枝函数：

```
def is_safe(board, row, col):
    # 检查列是否有冲突
    for i in range(row):
        if board[i][col] == 1:
            return False
    # 检查左上对角线是否有冲突
    for i, j in zip(range(row, -1, -1), range(col, -1, -1)):
        if board[i][j] == 1:
            return False
    # 检查右上对角线是否有冲突
    for i, j in zip(range(row, -1, -1), range(col, len(board))):
        if board[i][j] == 1:
            return False
    return True
```

搜索树（由于左右对称，所以搜索树只画了左半部分）：



三、具体代码

```
def print_solution(board):
    # 打印棋盘布局
    for row in board:
        print(" ".join(['Q' if x else '.' for x in row]))
    print()

def is_safe(board, row, col):
    # 检查列和两个对角线上是否有冲突
    for i in range(row):
        if board[i][col] == 1:
            return False
    for i, j in zip(range(row, -1, -1), range(col, -1, -1)):
        if board[i][j] == 1:
            return False
    for i, j in zip(range(row, -1, -1), range(col, len(board))):
        if board[i][j] == 1:
            return False
    return True

def solveNQUtil(board, row):
    # 递归解决 N 皇后问题，找到所有解决方案
    if row == len(board):
```

```
            print_solution(board)
            return False
        for col in range(len(board)):
            if is_safe(board, row, col):
                board[row][col] = 1
                if not solveNQUtil(board, row + 1):
                    # 回溯
                    board[row][col] = 0
                else:
                    return True
        return False

def solveNQ():
    # 主函数
    board = [[0 for i in range(4)] for j in range(4)]
    solveNQUtil(board, 0)

solveNQ()
```

四、运行结果

运行上述代码，控制台输出如下：

```
. Q . .
. . . Q
Q . . .
. . Q .

. . Q .
Q . . .
. . . Q
. Q . .
```