BTH004-N皇后问题

姓名：赵水 学号：201806150329

1 问题：

对于给定的n个皇后，求将其摆放入N * N方格内且互不冲突的所有可行方案。

2 算法：

采用回溯算法。对于该问题，考虑到每行最多只能摆放一个皇后，所以以行数做为最大递归的次数。当在第k行摆放第k枚皇后后：遍历k+1行，如果k+1行存在一个位置可供第k+1枚皇后摆放，则进行一次递归，继续寻找k+2行的可行位置。当递归结束后，进行该层（行）的回溯，及回收一开始摆放的第k个皇后，移动其位置以进行下一轮迭代。当递归到最大层数时，即说明从1到N个皇后按照该方案摆放可行，将该方案添加到结果集中。

3 代码：

```
def NQueen(row):
    tabulistcl = []
    tabulistincl1 = []
    tabulistincl2 = []
    solutions = []
    currentSolution = []

 solveNQ(row,0,solutions,currentSolution,tabulistcl,tabulistincl1,tabulistincl2)

    return solutions,len(solutions)

def
solveNQ(row,n,solutions,currentSolution,tabulistcl,tabulistincl1,tabulistincl2):
    if n == row:
        solutions.append(currentSolution.copy())
    else:
        for i in range(row):
            currentSolution.append(i)
            if
isSafe(currentSolution[n],n,tabulistcl,tabulistincl1,tabulistincl2):
                tabulistcl.append(currentSolution[n])
                tabulistincl1.append(currentSolution[n] - n)
                tabulistincl2.append(currentSolution[n] + n)

 solveNQ(row,n+1,solutions,currentSolution,tabulistcl,tabulistincl1,tabulistincl
2)
                tabulistcl.pop(n)
                tabulistincl1.pop(n)
                tabulistincl2.pop(n)
            currentSolution.pop(n)
```

```
#判断摆放第row行时，摆在在第col个位置是否可行
def isSafe(col,row,tabulistcl,tabulistincl1,tabulistincl2):
    if col in tabulistcl or col - row in tabulistincl1 or col + row in
tabulistincl2:
        return False
    return True
```

4 测试

4.1 代码

```
row = 8
print(NQueen(row))
```

4.2 输出

```
([[0, 4, 7, 5, 2, 6, 1, 3], [0, 5, 7, 2, 6, 3, 1, 4], [0, 6, 3, 5, 7, 1, 4, 2],
[0, 6, 4, 7, 1, 3, 5, 2], [1, 3, 5, 7, 2, 0, 6, 4], [1, 4, 6, 0, 2, 7, 5, 3],
[1, 4, 6, 3, 0, 7, 5, 2], [1, 5, 0, 6, 3, 7, 2, 4], [1, 5, 7, 2, 0, 3, 6, 4],
[1, 6, 2, 5, 7, 4, 0, 3], [1, 6, 4, 7, 0, 3, 5, 2], [1, 7, 5, 0, 2, 4, 6, 3],
[2, 0, 6, 4, 7, 1, 3, 5], [2, 4, 1, 7, 0, 6, 3, 5], [2, 4, 1, 7, 5, 3, 6, 0],
[2, 4, 6, 0, 3, 1, 7, 5], [2, 4, 7, 3, 0, 6, 1, 5], [2, 5, 1, 4, 7, 0, 6, 3],
[2, 5, 1, 6, 0, 3, 7, 4], [2, 5, 1, 6, 4, 0, 7, 3], [2, 5, 3, 0, 7, 4, 6, 1],
[2, 5, 3, 1, 7, 4, 6, 0], [2, 5, 7, 0, 3, 6, 4, 1], [2, 5, 7, 0, 4, 6, 1, 3],
[2, 5, 7, 1, 3, 0, 6, 4], [2, 6, 1, 7, 4, 0, 3, 5], [2, 6, 1, 7, 5, 3, 0, 4],
[2, 7, 3, 6, 0, 5, 1, 4], [3, 0, 4, 7, 1, 6, 2, 5], [3, 0, 4, 7, 5, 2, 6, 1],
[3, 1, 4, 7, 5, 0, 2, 6], [3, 1, 6, 2, 5, 7, 0, 4], [3, 1, 6, 2, 5, 7, 4, 0],
[3, 1, 6, 4, 0, 7, 5, 2], [3, 1, 7, 4, 6, 0, 2, 5], [3, 1, 7, 5, 0, 2, 4, 6],
[3, 5, 0, 4, 1, 7, 2, 6], [3, 5, 7, 1, 6, 0, 2, 4], [3, 5, 7, 2, 0, 6, 4, 1],
[3, 6, 0, 7, 4, 1, 5, 2], [3, 6, 2, 7, 1, 4, 0, 5], [3, 6, 4, 1, 5, 0, 2, 7],
[3, 6, 4, 2, 0, 5, 7, 1], [3, 7, 0, 2, 5, 1, 6, 4], [3, 7, 0, 4, 6, 1, 5, 2],
[3, 7, 4, 2, 0, 6, 1, 5], [4, 0, 3, 5, 7, 1, 6, 2], [4, 0, 7, 3, 1, 6, 2, 5],
[4, 0, 7, 5, 2, 6, 1, 3], [4, 1, 3, 5, 7, 2, 0, 6], [4, 1, 3, 6, 2, 7, 5, 0],
[4, 1, 5, 0, 6, 3, 7, 2], [4, 1, 7, 0, 3, 6, 2, 5], [4, 2, 0, 5, 7, 1, 3, 6],
[4, 2, 0, 6, 1, 7, 5, 3], [4, 2, 7, 3, 6, 0, 5, 1], [4, 6, 0, 2, 7, 5, 3, 1],
[4, 6, 0, 3, 1, 7, 5, 2], [4, 6, 1, 3, 7, 0, 2, 5], [4, 6, 1, 5, 2, 0, 3, 7],
[4, 6, 1, 5, 2, 0, 7, 3], [4, 6, 3, 0, 2, 7, 5, 1], [4, 7, 3, 0, 2, 5, 1, 6],
[4, 7, 3, 0, 6, 1, 5, 2], [5, 0, 4, 1, 7, 2, 6, 3], [5, 1, 6, 0, 2, 4, 7, 3],
[5, 1, 6, 0, 3, 7, 4, 2], [5, 2, 0, 6, 4, 7, 1, 3], [5, 2, 0, 7, 3, 1, 6, 4],
[5, 2, 0, 7, 4, 1, 3, 6], [5, 2, 4, 6, 0, 3, 1, 7], [5, 2, 4, 7, 0, 3, 1, 6],
[5, 2, 6, 1, 3, 7, 0, 4], [5, 2, 6, 1, 7, 4, 0, 3], [5, 2, 6, 3, 0, 7, 1, 4],
[5, 3, 0, 4, 7, 1, 6, 2], [5, 3, 1, 7, 4, 6, 0, 2], [5, 3, 6, 0, 2, 4, 1, 7],
[5, 3, 6, 0, 7, 1, 4, 2], [5, 7, 1, 3, 0, 6, 4, 2], [6, 0, 2, 7, 5, 3, 1, 4],
[6, 1, 3, 0, 7, 4, 2, 5], [6, 1, 5, 2, 0, 3, 7, 4], [6, 2, 0, 5, 7, 4, 1, 3],
[6, 2, 7, 1, 4, 0, 5, 3], [6, 3, 1, 4, 7, 0, 2, 5], [6, 3, 1, 7, 5, 0, 2, 4],
[6, 4, 2, 0, 5, 7, 1, 3], [7, 1, 3, 0, 6, 4, 2, 5], [7, 1, 4, 2, 0, 6, 3, 5],
[7, 2, 0, 5, 1, 4, 6, 3], [7, 3, 0, 2, 5, 1, 6, 4]], 92)
```