Computer Science 1001.py

Lecture 16.5: More Recursion: The Eight Queens Problem

Instructors: Amiram Yehudai, Amir Rubinstein
Teaching Assistants: Yael Baran, Michal Kleinbort

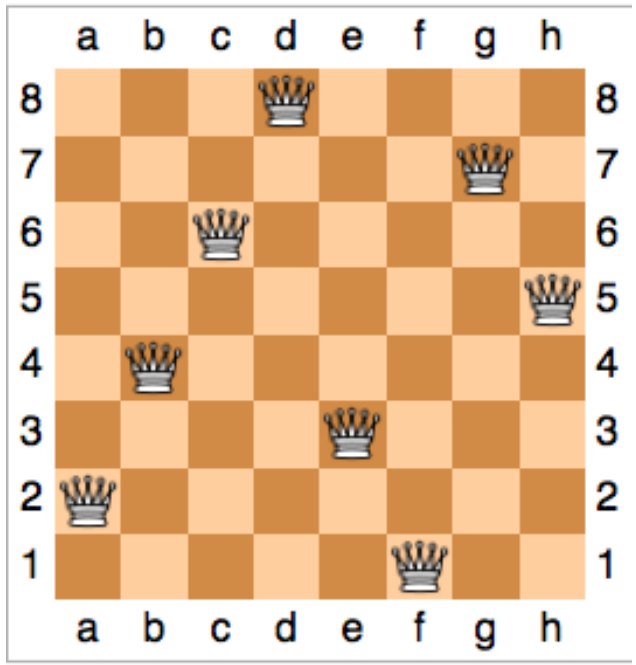
Founding Teaching Assistant (and Python Guru): Rani Hod

School of Computer Science
Tel-Aviv University, Fall Semester, 2016
http://tau-cs1001-py.wikidot.com

The N Queens Problem

The well known 8 queens problem is to determine how many possibilities are there to legally place 8 queens on an 8-by-8 chess board. Legally means no queen threatens another queen. Related questions are finding such placement, if one exists, and/or exploring the question for different number of queens and different size boards.

We will explore a possible path to the solution, employing several high tech means (*e.g* whiteboard, waving hands, etc.).



(figure from Wikipedia)

Ideas of Recursive Solution: the n Queens Problem

- We build the solution incrementally, column by column.
- We maintain a partial solution (implemented as a list).
- The partial solution is initially empty.
- We try to extend partial solutions recursively by placing a queen in all possible rows in the next column.
- We check if adding a queen to a given partial solution is legal. If it is, the partial solution is extended (and number of remaining columns decreased by 1).
- Once all columns are exhausted, we have a solution (contributing a 1 to the overall number of solutions).

Whatever we propose here (or elsewhere :-) is not the only possible approach. We do try, however, to propose a simple solution to the problem.

Functions and Signatures Used in the n Queens Problem

```
def queens(n,show=True):
    ''' how many ways to place n queens on an nXn board? '''
    partial = []    # list representing partial placement of queens
    return queens_rec(n, partial,show)

def queens_rec(n, partial,show):
    ''' Given a list representing partial placement of queens,
        can we legally extend it ? '''
    if len(partial)==n: #all n queens are placed legally
        if show:              # show the complete solution
            print(partial)
        return 1
    else:
        cnt=0
        for i in range(n):
            #try to place a queen in row i of the next column
            if legal(partial,i):
                cnt += queens_rec(n, partial+[i],show)
        return cnt
```

Functions and Signatures Used in the n Queens Problem

```
def legal(partial, i):
    ''' Can we place a queen in the next column in row i ? '''
    left = [j for j in partial if j==i]
                #any queens in the same row to the left?
    diag_up = [j for j in partial
           if j-partial.index(j) == i-len(partial)]
                                        #diagonal up-left
    diag_down = [j for j in partial
          if j+partial.index(j) == i+len(partial)]
                                        #diagonal down-left
    res = (left == diag_up == diag_down == [])
    # print("partial=",partial,"can add queen at row", i ,"?",res)
    return res
```

Example Executions

```
>>> queens(1)
[0]
1
>>> queens(2)
0
>>> queens(3)
0
>>> queens(4)
[1, 3, 0, 2]
[2, 0, 3, 1]
2
>>> queens(5)
[0, 2, 4, 1, 3]
[0, 3, 1, 4, 2]
[1, 3, 0, 2, 4]
[1, 4, 2, 0, 3]
[2, 0, 3, 1, 4]
[2, 4, 1, 3, 0]
[3, 0, 2, 4, 1]
[3, 1, 4, 2, 0]
[4, 1, 3, 0, 2]
[4, 2, 0, 3, 1]
10
```

Example Executions, cont.

Note that for all $n > 1$ the number of solutions is even, due to a simple mirror symmetry.

```
>>> queens(6)
[1, 3, 5, 0, 2, 4]
[2, 5, 1, 4, 0, 3]
[3, 0, 4, 1, 5, 2]
[4, 2, 0, 5, 3, 1]
4
```

So the number of solutions to `queens(n)` does not increase monotonically with n, as `queens(6) = 4 < queens(5) = 10`.

```
>>> for n in range(7,14):
            print(n, queens(n,show=False))

7  40
8  92
9  352
10 724
11 2680
12 14200
13 73712
14 365596
15 2279184 # ran overnight
```

Extensions to the N Queens Problem

Once we understand the solution, fairly simple modifications will yield

- Not just number of solutions, but a list specifying all solutions.
- Placing k queens on an n-by-n board, $k \leq n$.
- Placing k queens on an n-by-m board, $k \leq n \leq m$.
- Placing n rooks on an n-by-n board.
- Placing n bishops on an n-by-n board.

- Mixing queens, bishops, rooks on an n-by-n board (not so simple, but not that bad either).

Last words about recursion

- Recursion is not an easy topic for beginners.
- We hope the 8 queen example will help you a little with HW4 question 2 (especially section a, where recursion calls occur inside a loop, much like in the 8 queen problem).
- You may find this blog about recursion interesting and helpful: `http://www.gadial.net/2015/11/03/recusrion`.
- Another good source is `http://tau-cs1001-py.wdfiles.com/local--files/lecture-presentations-2016a/lec16_5.pdf`...