28/4/23

Krishnaraj P.T.
Div. (A1) PA-20

Q. 5 - Queen's Problem - Backtracking

Algorithm.

```
NQueen (k, n)
{
    for i = 1 to n do
    { if Place (k, i) then {
            x [k] = i ;
            if (k==n) : then write (x [1 : n]);
            else : NQueen (k+1, n);
        }
    }
}

place (k, i)
{
    for j =1 to k-1 do
    { if (x [j ]= i ) or (Abs (x [j]-i) ==
                            Abs (j-k)))
        { return False }
    return true ;
    }
}
```

Output :

The expected output is an array x [n] that shows where to place a queen on the i^{th} index of x [n]

So (2, 4, 1, 3, 5) could be a sample where queens would be on 2^{nd} index of row 1, 4^{th} index of row 2 and so on.

Approach : We try to put a queen on each row, then try to put other queens on subsequent rows while checking for conflicts. If we find a conflict, we backtrack to a previous case, and try with queen on another column of the same row.

place (K, i) acts as our bounding function in NQueens() k is no. of current queen, i is column.

eg. walk through:

say $n = 5$; so $k = 1$ initially,

$x = [0, 0, 0, 0, 0]$ // index start from 1.

$\underbrace{\qquad\qquad\qquad}_{\text{say } \infty}$

so we start checking from ~~row 0, col 0~~, row 1, col 1, queen 1.

place (1, 1) is called. It iterates from $j = 1$ to $k-1$ so it will iterate through all previous queen placed rows. If a conflict is found on row, col or diagonal (using abs) then it returns False. As this is the first round, place returns True.

so $x[1] = 1$. Then $k+1$ is passed to the same function recursively

so NQueen (2, 5) is called.

Now NQueen (2, 5) tries to place (2, 1) so just below previously placed Queen. Obviously then place sees conflict as $x[1] = 1$; so returns False.

So NQueen (3, 5) tries other cols.

This is how it would go.

k=1
i=1

N Queens (1, 5)
calls Place (1, 1) → True
x[1] = 1
calls { NQueen (2, 5)
calls Place (2, 1) → False
calls Place (2, 2) → False
calls Place (2, 3) → True
x[2] = 3
calls NQueen (3, 5)
calls Place (3, 1)
Place (3, 2)
Place (3, 3)
Place (3, 4)
} = False
Place (3, 5) → True
x[3] = 5
calls NQueen (~~3~~ 4, 5)
calls Place (4, 1) → False
calls Place (4, 2) → True
x[4] = 2
calls NQueen (5, 5)
calls Place (5, 1)
Place (5, 2)
Place (5, 3)
} False
Place (5, 4) → True
x[5] = 4

So x = {1, 3, 5, 2, 4}

Now value of i is queens (1, 5) increments.

~~cons (2, 5)~~

~~place (2, 5)~~ k = 1 ; i = 2

Calls place (1, 2) → True

x [1] = 2

NQueens (2, 5)

Calls place (2, 1) → False

calls place (2, 2) → False

calls place (2, 3) → False

calls place (2, 4) → True

x [2] = 4

NQueens (3, 5)

calls place (3, 1) → True

~~calls place (3,~~ x [3] = 1

NQueens (4, 5)

calls place (4, 1)

calls place (4, 2)

calls place (4, 3) → True

x [4] = 3

NQueens (5, 5)

calls place (5, 1)

calls place (5, 2)

calls place (5, 3)

calls place (5, 4)

calls place (5, 5) → True

x [5] = 5

so $soln_2 = (2, 4, 1, 3, 5)$