

Assignment 3: 22-Queen Problem

In this assignment, you are going to use **backtracking** to solve n-queen problem and n is required to be 22 in this assignment. Your program will place 22 queens on a 22 x 22 chess board while the queens are not attacking each other. You must give 4 solutions.

Backtracking Ideas:

- Idea 1: One variable at a time
 - Variable assignments are commutative, so fix ordering
 - I.e., [WA = red then NT = green] same as [NT = green then WA = red]
 - Only need to consider assignments to a single variable at each step
- Idea 2: Check constraints as you go
 - I.e., consider only values which do not conflict previous assignments
 - Might have to do some computation to check the constraints
 - “Incremental goal test”

```

function BACKTRACKING-SEARCH(csp) returns solution/failure
    return RECURSIVE-BACKTRACKING({ }, csp)

function RECURSIVE-BACKTRACKING(assignment, csp) returns soln/failure
    if assignment is complete then return assignment
    var ← SELECT-UNASSIGNED-VARIABLE(VARIABLES[csp], assignment, csp)
    for each value in ORDER-DOMAIN-VALUES(var, assignment, csp) do
        if value is consistent with assignment given CONSTRAINTS[csp] then
            add { var = value } to assignment
            result ← RECURSIVE-BACKTRACKING(assignment, csp)
            if result ≠ failure then return result
            remove { var = value } from assignment
    return failure
    
```

Requirements:

1. The given ipynb file must be used in this assignment.
2. You need to print out at least **four** of the solutions. The result should be in this format (row, column). Each pair shows a queen's position.
3. **Backtracking** should be used to check when you are placing a queen at a position.
4. Your code should be capable of solving other n-queen problems. For example, if n is changed to 10, your code also will solve 10-queen problem.

Example Output for 4-queens Problem

(0,1) (1,3) (2,0) (3,2)

(0,2) (1,0) (2,3) (3,1)