**import** time  
**import** math  
**import** heapq  
**from** collections **import** deque  
**import** random  
  
  
**class** nQueens:  
 **def** \_\_init\_\_(self, state=**None**, choices=**None**, n=8, parent=**None**):

“”” creates an nQueens board where state is a list of n integers,

one per column,

and choices is a list of sets,

n is the size

parent is the state predecessor in a search

“””

**def** assign(self, var, value):

“”” updates the state by setting state[var] to value

also propgates constraints and updates choices

“””

**def** goal\_test(self):

“”” returns True iff state is the goal state “””

**def** get\_next\_unassigned\_var(self):

“”” returns the index of a column that is unassigned and

has valid choices available “””

**def** get\_choices\_for\_var(self, var):

“”” returns choices[var], the list of available values

for variable var, possibly sorted “””

**def** \_\_str\_\_(self):

“”” returns a string representation of the object “””

###---------------------------------------------------------------

**def** dfs\_search(board):

“”” sets board as the initial state and returns a

board containing an nQueens solution

or None if none exists

“””