File - C:\Users\Matthias\Documents\Spring 2018\Ai\Assignments\Assignment04\backtrack.py

```
1  import csp_lib.sudoku as CSP
2  from csp_lib.backtrack_util import (first_unassigned_variable,
3                                      unordered_domain_values,
4                                      no_inference)
5
6  def backtracking_search(csp,
7                          select_unassigned_variable=first_unassigned_variable,
8                          order_domain_values=unordered_domain_values,
9                          inference=no_inference):
10     """backtracking_search
11     Given a constraint satisfaction problem (CSP),
12     a function handle for selecting variables,
13     a function handle for selecting elements of a domain,
14     and a set of inferences, solve the CSP using backtrack search
15     """
16     '''Instead of passing more variables just nest functions
17     e.g.
18     #return backtrack({}, csp, select_unassigned_variable, order_domain_values, inference)
19     #return backtrack({}, csp)
20     #def backtrack(assignment, csp, select_unassigned_variable, order_domain_values, inference):
21     '''
22
23     def backtrack(assignment):
24         """Attempt to backtrack search with current assignment
25             Returns None if there is no solution.  Otherwise, the
26             csp should be in a goal state.
27         """
28         # assert isinstance(csp, CSP)
29         # if csp.nassigns == len(csp.variables):
30         if len(assignment) == len(csp.variables):
31             return assignment
32         variable = select_unassigned_variable(assignment, csp)
33         for value in order_domain_values(variable, assignment, csp):
34             csp.assign(variable, value, assignment)  # assigns variables dictionary{}, doesn't
   manipulate curr_domains
35             removals = csp.suppose(variable, value)  # list of tuples; tuple (var, domain_item_removed
   )  this is incase the assignemnt doesnt work out, its a way to go back
36             if csp.nconflicts(variable, value, assignment) ==  :
37                 # i.e. if removals = [] then its curr_domain[var] is len(1) meaning it's assigned; else
   removals == [(0,1), (0,2), (0,3), (0,4), (0,5), (0,6), (0,7), (0,8)] # Note that the value chosen for
   assignment in this case would be 9
38                 inferences = inference(csp, variable, value, assignment, removals)  # is this
   no_inference because we have are maintaining arc consistency therefor we dont need to forward check
39                 # todo - last thing would be to make mac work
40                 if inferences:  # if inferences != failure:
41                     # todo - add inferences to assignment
42                     result = backtrack(assignment)  # recursive call if consistency is kept
43                     if result is not None:
44                         return result
45                 ''' essentially every line below in def is the else condition of (if inferences:) b/c
   return statement inside if condition scope'''
46                 #csp.restore(removals) # essentially if variable assignement was not right # todo - make
    sure this updates assignment at fix the domains of the removals # log this so we can fix domains of var
    asssigment manip
47             csp.restore(removals)  # essentially if variable assignement was not right # todo - make
   sure this updates assignment at fix the domains of the removals # log this so we can fix domains of var
   asssigment manip
48         csp.unassign(variable, assignment)
49         return None
50
51     # Call with empty assignments, variables accessed
52     # through dynamic scoping (variables in outer
53     # scope can be accessed in Python) i.e. csp and function handles
54     result = backtrack({})  # initial backtrack function(search procedure) called/invoked
55     assert result is None or csp.goal_test(result)
56     return result
57
58
```