Week 11 – In-Class Assignment

1. Pure Backtracking

Assume we use pure backtracking to search for a solution to this problem. We use a fixed variable ordering in the search (V1, V2, V3, V4) and the values are considered in the order shown in the picture above. Then, show the order in which individual variable assignments are considered by backtracking (this is analogous to the order in which nodes are expanded by depth-first search).

Iterations:

1. 1R
2. 1R, 2B
3. 1R, 2B, 3B (4X)
4. 1R, 2B, 3G (4X)
5. 1R, 2G
6. 1R, 2G, 3B (4X)
7. 1R, 2G, 3G
8. 1R, 2G, 3G, 4B (Solved!)
9. 1B
10. 1B, 2G

2. Backtracking with Forward Checking

Repeat assuming we use backtracking with forward checking to search for a solution to this problem. We use the same variable ordering and value ordering as before. Show the order in which assignments are considered by BT-FC. Whenever propagating after an assignment causes a domain to become empty, that causes backup in the search. Assume that the search continues even after finding a valid solution. Write each assignment (up to 10) as before.

1. 1R
2. 1R, 2B (4X)
3. 1R, 2G
4. 1R, 2G, 3B (4X)
5. 1R, 2G, 3G, 4B (Solved!)
6. 1B
7. 1B, 2G
8. 1B, 2G, 3G
9. 1B, 2G, 23G, 4B (Solved!)
10. 1G

3. Arc Consistency

Write the value in each of the indicated value domains after any changes required to achieve arc consistency for just that arc. Then, assume that the following arcs are done sequentially, with the effects on the domains propagating. Write domains as a sequence of letters (R B). If there are no other values in the domain, write None.

V1 – V2 : D1 = RGB; D2 = GB

V1 – V3 : D1 = RGB; D3 = GB

V2 – V4 : D2 = G; D4 = B

V3 – V4 : D3 = G; D4 = B

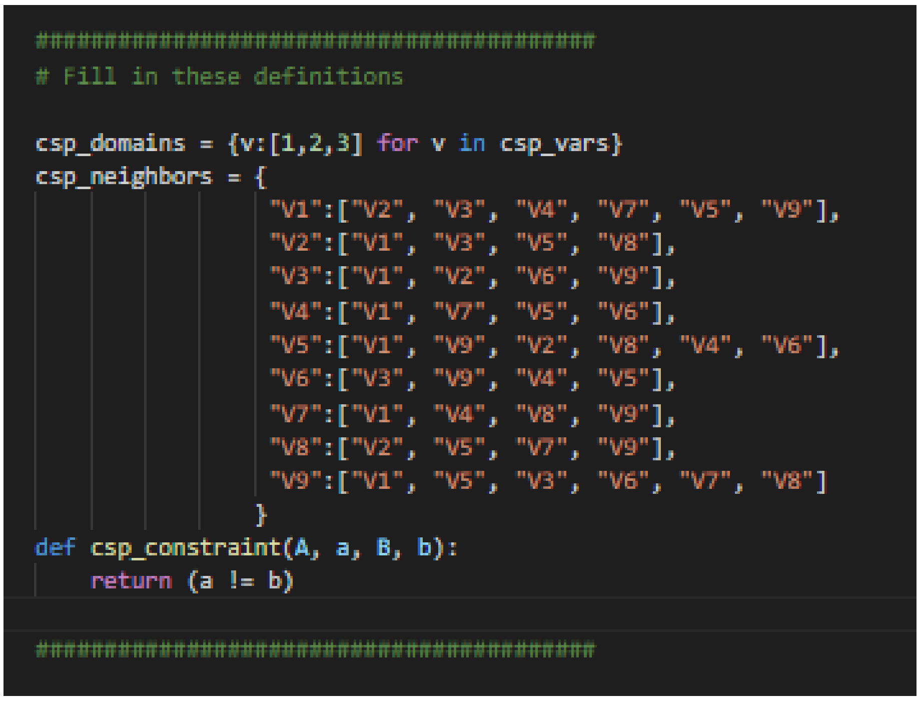
V1 – V2 : D1 = RB; D2 = G

V1 – V3 : D1 = RB; D3 = G

4. Thinking of CSP

1. Constraint propagation without search is sufficient to find a satisfying set of assignments in most problems.	**True**
2. If constraint propagation leaves all variables with non-empty domains, there is at least one solution.	**False**
3. If constraint propagation leaves some variable with an empty domain, there is no solution.	**True**
4. If constraint propagation leaves all variables with singleton domains, there is a unique solution.	**True**
5. Constraint propagation takes on the order of ed^2 arc tests for one pass through all the arcs in a constraint graph with e edges and d values per domain.	**False**
6. When doing backtracking with Forward Checking (BT-FC), then Forward checking during backtracking requires verifying that the newly assigned value for a variable is consistent with all previously assigned values of variables.	**False**
7. Backtracking search with full constraint propagation (until arc-consistency is reached) generally results in an answer using fewer arc-tests than search with forward-checking only.	**True**
8. Backtracking with Forward Checking (BT-FC) never explores more possible variable values than pure backtracking.	**True**
9. BT-FC never tests more constraint arcs than pure backtracking.	**True**
10. Dynamic ordering of variables based on domain size (e.g. minimum remaining values) is generally a very effective strategy to speed up solving CSP (whether or not all the answers are desired).	**False**
11. Dynamic ordering of values based on impact on neighboring variables (e.g. least constraining value) is generally a very effective strategy to speed up solving CSP when all possible answers are desired and not just one.	**True**

5. Semi-Magic Square

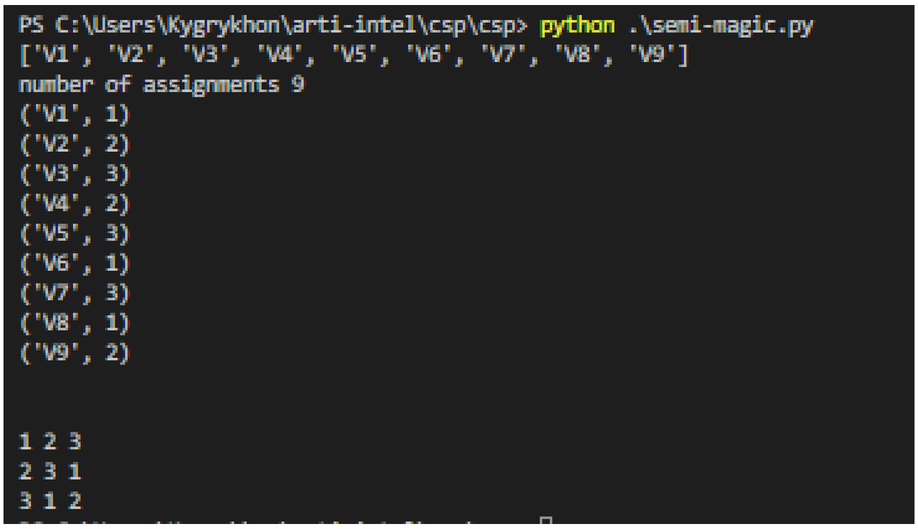


```python
#############################################
# Fill in these definitions

csp_domains = {v:[1,2,3] for v in csp_vars}
csp_neighbors = {
                "V1":["V2", "V3", "V4", "V7", "V5", "V9"],
                "V2":["V1", "V3", "V5", "V8"],
                "V3":["V1", "V2", "V6", "V9"],
                "V4":["V1", "V7", "V5", "V6"],
                "V5":["V1", "V9", "V2", "V8", "V4", "V6"],
                "V6":["V3", "V9", "V4", "V5"],
                "V7":["V1", "V4", "V8", "V9"],
                "V8":["V2", "V5", "V7", "V9"],
                "V9":["V1", "V5", "V3", "V6", "V7", "V8"]
               }
def csp_constraint(A, a, B, b):
    return (a != b)

#############################################
```

Result:



```
PS C:\Users\Kygrykhon\arti-intel\csp\csp> python .\semi-magic.py
['V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9']
number of assignments 9
('V1', 1)
('V2', 2)
('V3', 3)
('V4', 2)
('V5', 3)
('V6', 1)
('V7', 3)
('V8', 1)
('V9', 2)


1 2 3
2 3 1
3 1 2
```