GWID: G41223192 Name: Neng Zhou

1. Problem:

CSP Tile Placement:
Input: A file contains landscape, tiles and targets
Output: Solution of placing the tiles

2. My Algorithm:

Code:

https://github.com/Vizards8/CSCI_6511_AI/blob/main/P2_CSP_Tile_Placement.py

Explanation of my code:

Below are some notions I used in my code:
variables: The positions I need to place tiles
domains: Three different tiles
tiles: The given number of every tile
targets: The given number of visible bushes

Search Algorithm: backtrack
MRV: I choose the variable with the minimum remaining domain value
LCV: For each domain, I try to assign it and count the number of ruling out remaining unassigned variables. Then I choose the smallest one to assign first.
AC3: First, I generated all pairs of unassigned variables, for example, X_i and X_j. Then, I traverse all domains of X_i. If there is no domain in X_j that can satisfy the constraint, I delete this domain in X_i.

To solve this problem, I used a general backtrack search algorithm. All other components like MRV, LCV, AC-3 are implemented in backtrack function step by step. First, I implemented MRV to find a variable with the minimum remaining values in its domain. Then, I implemented LCV to sort the domain values and assigned a value with the least constraining. After that, I implemented arc consistency using AC-3 to remove more possible domain values. After all these prune operations, I finished

this step which is assign the most suitable value to the most suitable variable. The remaining work is done by backtrack search algorithm by keep trying and backtracking until it finds the solution.

3. Results:

Here are some results of the given test cases.

```
------ Analyze File ------
file name: test_cases/tilesproblem_1326658913086500.txt
------ Find Result ------
0 4 OUTER_BOUNDARY
1 4 OUTER_BOUNDARY
2 4 OUTER_BOUNDARY
3 4 OUTER_BOUNDARY
4 4 OUTER_BOUNDARY
5 4 EL_SHAPE
6 4 EL_SHAPE
7 4 EL_SHAPE
8 4 FULL_BLOCK
9 4 FULL_BLOCK
10 4 FULL_BLOCK
11 4 FULL_BLOCK
12 4 FULL_BLOCK
13 4 FULL_BLOCK
14 4 FULL_BLOCK
15 4 EL_SHAPE
16 4 EL_SHAPE
17 4 EL_SHAPE
18 4 FULL_BLOCK
19 4 OUTER_BOUNDARY
20 4 FULL_BLOCK
21 4 EL_SHAPE
22 4 FULL_BLOCK
23 4 FULL_BLOCK
24 4 FULL_BLOCK
time cost: 0.066s
------ Check Result ------
PASS!

Process finished with exit code 0
```

```
------ Analyze File ------
file name: test_cases/tilesproblem_1326658924404900.txt
------ Find Result ------
0 4 EL_SHAPE
1 4 EL_SHAPE
2 4 EL_SHAPE
3 4 EL_SHAPE
4 4 FULL_BLOCK
5 4 FULL_BLOCK
6 4 OUTER_BOUNDARY
7 4 OUTER_BOUNDARY
8 4 OUTER_BOUNDARY
9 4 FULL_BLOCK
10 4 OUTER_BOUNDARY
11 4 FULL_BLOCK
12 4 OUTER_BOUNDARY
13 4 EL_SHAPE
14 4 OUTER_BOUNDARY
15 4 FULL_BLOCK
16 4 OUTER_BOUNDARY
17 4 FULL_BLOCK
18 4 FULL_BLOCK
19 4 FULL_BLOCK
20 4 FULL_BLOCK
21 4 OUTER_BOUNDARY
22 4 EL_SHAPE
23 4 OUTER_BOUNDARY
24 4 FULL_BLOCK
time cost: 0.438s
------ Check Result ------
PASS!

Process finished with exit code 0
```

```
------ Analyze File ------
file name: test_cases/tilesproblem_1326658926570700.txt
------ Find Result ------
0 4 OUTER_BOUNDARY
1 4 EL_SHAPE
2 4 EL_SHAPE
3 4 EL_SHAPE
4 4 EL_SHAPE
5 4 EL_SHAPE
6 4 EL_SHAPE
7 4 EL_SHAPE
8 4 FULL_BLOCK
9 4 EL_SHAPE
10 4 FULL_BLOCK
11 4 EL_SHAPE
12 4 FULL_BLOCK
13 4 FULL_BLOCK
14 4 OUTER_BOUNDARY
15 4 FULL_BLOCK
16 4 FULL_BLOCK
17 4 FULL_BLOCK
18 4 FULL_BLOCK
19 4 FULL_BLOCK
20 4 EL_SHAPE
21 4 FULL_BLOCK
22 4 EL_SHAPE
23 4 FULL_BLOCK
24 4 FULL_BLOCK
time cost: 0.071s
------ Check Result ------
PASS!

Process finished with exit code 0
```

4. Conclusion:

This project is much more challengeable than project 1. Unlike the graph coloring problem which is told in class, I'm really new to the tile placing problem. First, I quickly wrote a general backtracking solution. Sadly, it just ran for a long time and didn't give

me an answer. To be honest, I spent many days drawing the landscape and tiles in excel to see whether my algorithm goes wrong. But nothing changed. The backtrack itself is correct. Then I tried to implement MRV and LCV. To my surprise, this time it quickly gives an answer and it's correct. After that, I implemented arc consistency using AC-3. This can further reduce the domain. Now I have backtrack + MRV + LCV + AC-3. I ran some test cases and it can solve them in a few seconds. In this project, I have learnt how much powerful it is for all these prune functions. It really reduces much redundant choices and lead the algorithm to the solution. In conclusion, the algorithm with MRV, LCV and AC-3 is very effective in solving the CSP (Constraint Satisfaction Problem).