CSCI 561: Foundations of Artificial Intelligence
Summer 2017

Homework 2

Due on June 26, 2017 at 11:59 PM

Programming Assignment: Adversarial Constraint Satisfaction by Game-tree Search

Inspired by the work of Brown et al.[1], the goal of this assignment is to implement a game at which two agents alternate in assigning values to the variables of a constraint satisfaction problem (CSP). The type of CSP here is limited to the map coloring problem where a set of states and their neighbors are given and the goal is to color every state in a way that it has a different color than its neighbors. The variables of the CSP are the states, and the colors represent the finite domain of such variables. The game is played by two minimax agents which behave in the same way, except that they follow a defined set of preferences: for each agent, every color is assigned a weight and the goal is it to make assignments (i.e. take actions) that maximize the total sum of weights.

In this assignment, you need to combine **arc-consistency** (AIMA Fig.6.3) and **alpha-beta pruning** (AIMA Fig.5.7). **The goal is to compute the next best action for player 1** by employing Minimax (with alpha-beta pruning) to take future outcomes of both players' turns into account.

The map coloring game obeys the following rules:

1. There are two players, Player 1 and the opponent Player 2. Each player takes turns as in chess or tic-tac-toe. That is, Player 1 takes a move to color a node, then Player 2, then back to Player 1 and so forth. In the game, you could assume Player 1 will always start first (i.e. Player 1 is the MAX player, Player 2 is the MIN player).
2. The players can only color **neighbors of nodes that have already been colored** in the map. **Adjacent nodes could not have the same color.**
3. The score of each player is defined as the total sum of weights of their colored nodes.
4. The terminal state of the game is that no more nodes could be colored in the map based on rule 2. It could be either all nodes in the map have been colored, or no possible assignment could be made according to rule 2.

---

[1] Brown, K. N., Little, J., Creed, P. J., & Freuder, E. C. (2004). Adversarial Constraint Satisfaction by Game-Tree Search. In *ECAI* (pp. 151–155). Retrieved from http://www.frontiersinai.com/ecai/ecai2004/ecai04/pdf/p0151.pdf

5. The evaluation function of the colored map state could be defined as Score_Player1 – Score_Player2. The leaf node values are always calculated from this evaluation function. Although there might be a better evaluation function, you should comply with this rule for simplicity.

Node Expanding rule in alpha-beta pruning algorithm:

When expanding a node in the game tree (i.e. choosing an action), you need to expand first based on the alphabetical order of the state name, and then based on the color name (if state names are equal). For example, if you have the (state, color) pair (CA,G) and (Q,R) , you need to first expand (CA,G).And If you have the pair (CA,G) and (CA,R), you need expand (CA,G) first.

Example

As an example, let us consider the map of Mainland Australia. Two players take turns at coloring the states by maintaining the map coloring consistency, i.e. neighboring states must have different colors. We will define red (R), green (G) and blue (B) as the possible colors both players can use. The two players have different preferences for colors which we specify by giving weights to these colors. In the following, player 1 has the weights 10, 5 and 0 for the colors R, G and B, respectively. This means, player 1 prefers to assign red (10) over blue (5). Player 2 has the weights 0, 2 and 8 for colors R, G and B.

In the following we will start from an initial assignment in which player 1 colored the state WA red(dotted fill below) and player 2 colored the state SA green(solid fill below). After these two assignments, there remain the states NT, Q, NSW, and V to be colored since they are the neighboring states of the current assigned states WA and SA. Player 1 could now choose any one of these states to assign a color such that is is consistent with the map coloring(adjecent nodes could not have the same color). A good move could be color state Q red because red is player 1's top preference and this assignment is consistent with the map coloring. A drawback of this assignment could be that in the future, states NT and NSW must be colored blue which does not yield any utility for player 1, but the maximum utility (8) for player 2.

Players 1 and 2 continue to take turns until no states can be assigned anymore, i.e. all states neighboring the currently assigned states are colored. To compute the utility, the following terminal evaluation function is used:

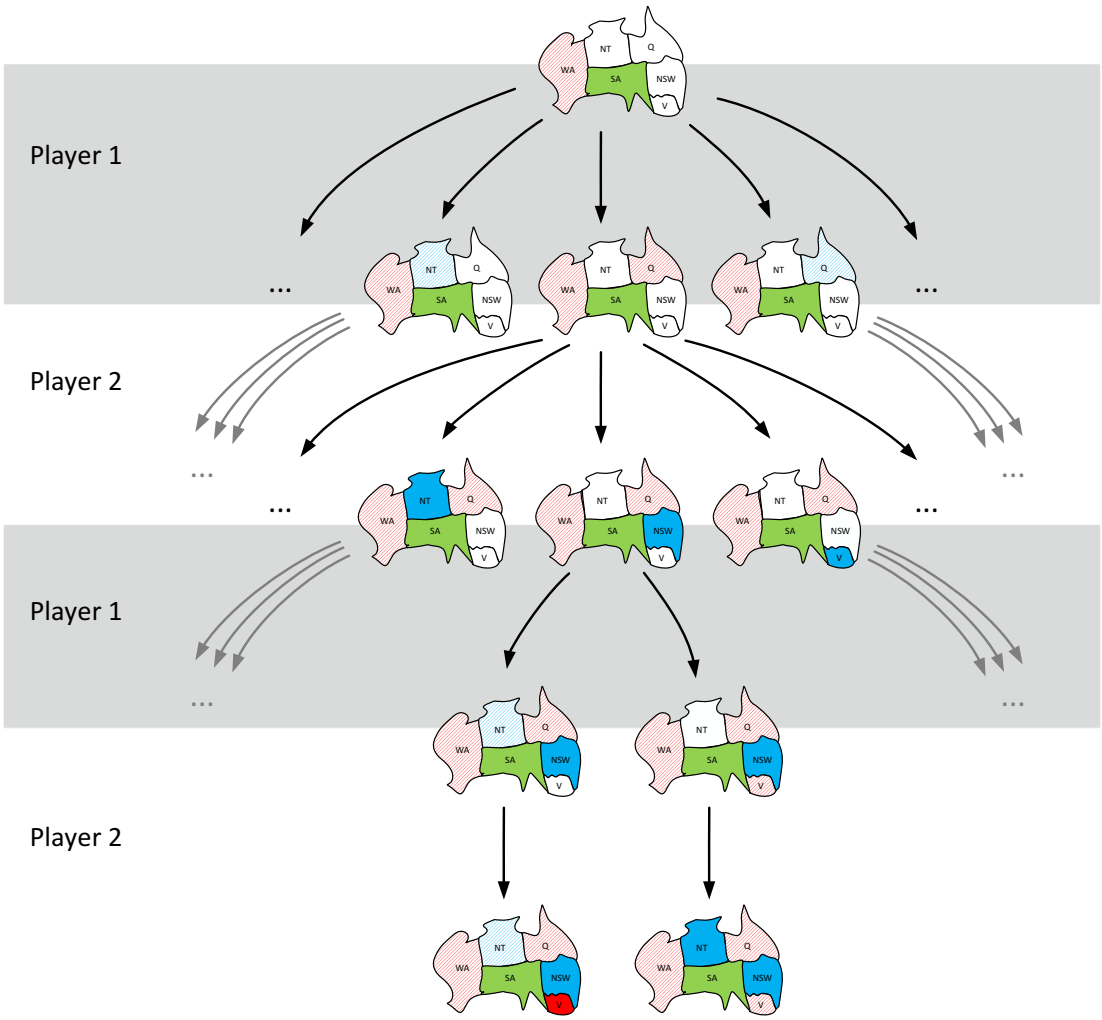


Figure 1 Example game tree for the map coloring game given an initial assignment. Player 1's assignments are visualized in striped colors, player 2's assignments are in solid colors.

The terminal evaluation for player 1 in the **left** leaf of the game tree in Figure 1 is:

$$Player1 = 2\,red + 1\,blue = 20$$

$$Player2 = 1\,red + 1\,green + 1\,blue = 10$$

$$TerminalState = Player1 - Player2 = 10$$

The terminal evaluation for player 1 in the **right** leaf of the game tree in Figure 1 is:

$$Player1 = 3\,red = 30$$

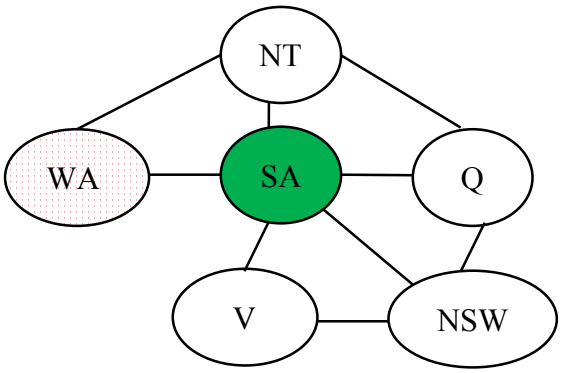
$$Player2 = 1\,green + 2\,blue = 18$$

$$TerminalState = Player1 - Player2 = 12$$

Therefore, the latter game yielded a higher utility for player 1. If this leaf was the overall highest utility, player 1 should have chosen to color state Q in red as the next best move based on the minimax tree computation. In this assignment, the depth of the generated game tree is limited (as given in the input file) such that minimax will not always evaluate all possible moves until no more moves can be made according to rule 2. (i.e. the assignment is complete = game over).

Input Format

```
R, G, B
WA: R-1, SA: G-2
3
R: 10, G: 5, B: 0
R: 0, G: 2, B: 8
SA: WA, NT, Q, NSW, V
NT: WA, SA, Q
NSW: Q, V, SA
WA: SA, NT
Q: NT, SA, NSW
V: SA, NSW
```



The first line of the input are the possible colors. Colors can be strings (one or more characters) consisting of only alphabetical characters [a-zA-Z].

The second line shows the initial map coloring by the two players. For example, in the input file above, `WA: R-1, SA: G-2` means state WA was assigned color 'R' by player 1, state SA was assigned color 'G' by player 2. There can only be integers 1 or 2, indicating player 1 or 2.

The third line is the maximum depth (integer) of the game search tree. The root has depth 0.

The fourth and the fifth lines specify the preferences of player 1 and player 2, respectively. For example, `R: 10, G: 5, B: 0` in line four means player one has preferences 10, 5, 0 to assign colors R, G, B, respectively. The higher the value, the higher is the preference to use this color for an assignment. The numbers are integers.

The rest of lines represent the graph, for example, `SA: WA, NT, Q, NSW, V` means that state SA has 5 neighbors: WA, NT, Q, NSW, and V

Note: Node names (i.e. states on the map) can be any strings consisting of only alphanumerical characters [a-zA-Z0-9]. All test cases will follow the same format, including the placement of whitespace and new-line characters.

Output Format

The output.txt file contains the trace of steps the alpha beta pruning algorithm takes to compute the best action for player 1). A log entry line is added when (1) the algorithm visits a node, or (2) the value of a node is updated from its children. Each line contains the node name (i.e. the name of the state in the map), the assigned color, the depth of the node, the current minimax value, and the alpha and beta values. For example, the first line of the example output below shows the root node. The root node is always the max node. In the input above, in the root node, it would print the last move's color assignment in the initial map assignment. In the above example, we have player 2 (min player) colors state SA with color G as shown in the input file(second line above). At this step, we are at the top of the minimax

tree, such that the depth is 0 (This is the last move of the initial assignment which is shown in the second line of the input file.). For max node, it would print the info of the previous assignment of min player, and for the min node, it would print the info of the previous assignment of max player. Initially, the minimax value, as well as the alpha and beta values for max and min nodes are not yet computed, therefore they are initialized by $-\infty, -\infty, \infty,$ (printed as `-inf, -inf, inf`) for max node, and $\infty, -\infty, \infty$(printed as `inf, -inf, inf`) for min node, respectively.
Once the tree has been fully generated and the minimax value of the root has been updated, the best action needs to be returned. This is done in the last line which contains the best action in form of the selected node name and assigned color, and the minimax value for player 1. As can be seen below, the algorithm determined to color node Q with color R as the best next action for the max player, yielding an evaluation score of 18.
In the output below, comments are highlighted red and should not be included in the generated output file.

#<Node, Color, Depth, Value, Alpha, Beta>

```
SA, G, 0, -inf, -inf, inf
NSW, B, 1, inf, -inf, inf
NT, B, 2, -inf, -inf, inf
Q, R, 3, 10, -inf, inf
NT, B, 2, 10, 10, inf
V, R, 3, 10, 10, inf
NT, B, 2, 10, 10, inf
NSW, B, 1, 10, -inf, 10
Q, R, 2, -inf, -inf, 10
NT, B, 3, 8, -inf, 10
Q, R, 2, 8, 8, 10
V, R, 3, 18, 8, 10
Q, R, 2, 18, 8, 10
NSW, B, 1, 10, -inf, 10
V, R, 2, -inf, -inf, 10
NT, B, 3, 8, -inf, 10
V, R, 2, 8, 8, 10
Q, R, 3, 18, 8, 10
V, R, 2, 18, 8, 10
NSW, B, 1, 10, -inf, 10
SA, G, 0, 10, 10, inf
NSW, R, 1, inf, 10, inf
NT, B, 2, -inf, 10, inf
V, B, 3, 10, 10, inf
NT, B, 2, 10, 10, inf
NSW, R, 1, 10, 10, inf
SA, G, 0, 10, 10, inf
NT, B, 1, inf, 10, inf
NSW, B, 2, -inf, 10, inf
Q, R, 3, 10, 10, inf
NSW, B, 2, 10, 10, inf
V, R, 3, 10, 10, inf
NSW, B, 2, 10, 10, inf
NT, B, 1, 10, 10, inf
SA, G, 0, 10, 10, inf
Q, B, 1, inf, 10, inf
NSW, R, 2, -inf, 10, inf
V, B, 3, 8, 10, inf
NSW, R, 2, 8, 10, inf
Q, B, 1, 8, 10, inf
SA, G, 0, 10, 10, inf
Q, R, 1, inf, 10, inf
NSW, B, 2, -inf, 10, inf
NT, B, 3, 10, 10, inf
NSW, B, 2, 10, 10, inf
V, R, 3, 20, 10, inf
```

```
NSW, B, 2, 20, 20, inf
Q, R, 1, 20, 10, 20
NT, B, 2, -inf, 10, 20
NSW, B, 3, 10, 10, 20
NT, B, 2, 10, 10, 20
V, B, 3, 10, 10, 20
NT, B, 2, 10, 10, 20
V, R, 3, 20, 10, 20
NT, B, 2, 20, 10, 20
Q, R, 1, 20, 10, 20
V, B, 2, -inf, 10, 20
NT, B, 3, 10, 10, 20
V, B, 2, 10, 10, 20
Q, R, 1, 10, 10, 20
SA, G, 0, 10, 10, inf
V, B, 1, inf, 10, inf
NSW, R, 2, -inf, 10, inf
NT, B, 3, 8, 10, inf
NSW, R, 2, 8, 10, inf
Q, B, 3, 8, 10, inf
NSW, R, 2, 8, 10, inf
V, B, 1, 8, 10, inf
SA, G, 0, 10, 10, inf
V, R, 1, inf, 10, inf
NSW, B, 2, -inf, 10, inf
NT, B, 3, 10, 10, inf
NSW, B, 2, 10, 10, inf
Q, R, 3, 20, 10, inf
NSW, B, 2, 20, 20, inf
V, R, 1, 20, 10, 20
NT, B, 2, -inf, 10, 20
NSW, B, 3, 10, 10, 20
NT, B, 2, 10, 10, 20
Q, R, 3, 20, 10, 20
NT, B, 2, 20, 10, 20
V, R, 1, 20, 10, 20
Q, B, 2, 10, 10, 20
V, R, 1, 10, 10, 20
SA, G, 0, 10, 10, inf
#<Best_Node_of_First_Move, Best_Color_of_First_Move, Utility>
NSW, B, 10
```

Grading Notice:

Please follow the instructions carefully. Any deviations from the instructions will lead your grade to be zero for the assignment. If you have any doubts, please use the discussion board on Piazza. Do not assume anything that is not explicitly stated.

- You must use **PYTHON** (Python 2.7) to implement your code. You must not use any other Python libraries besides the default libraries provided by the Python 2.7 environment (Python Standard Library). You must implement any other functions or modules by yourself.
- You need to create a file named "**hw2cs561s17.py**". The command to run your program will be as follows: *(When you submit the homework on labs.vocareum.com, the following command will be executed automatically by the grading script.)*

  `python hw2cs561s17.py –i <inputFile>`

  where <*inputFile*> is the filename of the input file that your program needs to read.
- The input and output files use UNIX line endings ("\n").
- The generated output file needs to be named as "**output.txt**".
- You will use labs.vocareum.com to submit your code. Please refer to http://help.vocareum.com/article/30-getting-started-students to get started with the

system. Please only upload your code to the "/work" directory. Don't create any subfolders or upload any other files.

- If we are unable to execute your code successfully, you will not receive any credits.
- You will get partial credit based on the percentage of test cases that your program gets right for each task.
- Per test case, the output file is considered correct only if it exactly matches the solution (string-based matching).
- String comparisons (alphabetical or alphanumerical) must follow the ASCII ordering of the characters.
- Your program should handle all test cases within a reasonable time (not more than a few seconds for each sample test case). The complexity of test cases is similar to, but not necessarily the same as, the ones provided in the assignment description.

The deadline for this assignment is **June 26, 2017 at 11:59 PM PST**. **No late homework will be accepted.** Any late submissions will not be graded. Any emails for late submission will be ignored.