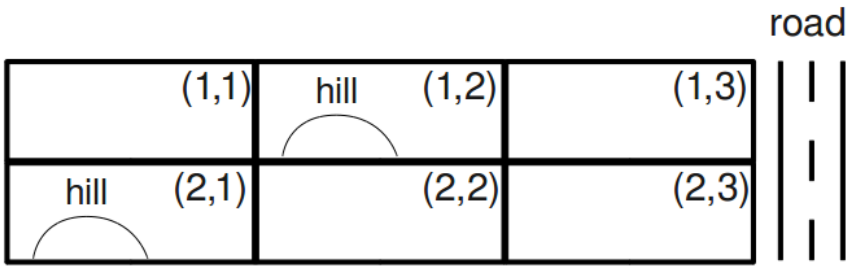
CIS 471/571: Introduction to Artificial Intelligence, Fall 2020

Written Assignment 2: Solution
Deadline: Oct 24th, 2020

Instruction: You may discuss these problems with classmates, but please complete the write-ups individually. (This applies to BOTH undergraduates and graduate students.) Remember the collaboration guidelines set forth in class: you may meet to discuss problems with classmates, but you may not take any written notes (or electronic notes, or photos, etc.) away from the meeting. Your answers must be **typewritten**, except for figures or diagrams, which may be hand-drawn.

Q1. Campus Layo

You are asked to determine the layout of a new, small college. The ca s: an administration structure (A), a bus stop (B), a classroom (C), and a dorm structure (including the bus stop) must be placed somewhere on the g



The layout must satisfy the following constraints:

(i) The bus stop (B) must be adjacent to the road.

(ii) The administration structure (A) and the classroom (C) must both be adjacent to the bus stop (B).

(iii) The classroom (C) must be adjacent to the dormitory (D).

(iv) The administration structure (A) must not be adjacent to the dormitory (D).

(v) The administration structure (A) must not be on a hill.

(vi) The dormitory (D) must be on a hill or adjacent to the road.

(vii) All structures must be in different grid squares.

Here, adjacent means that the structures must share a grid edge, not just a corner.

Q1.1. Provide the domains of all variables after unary constraints have been applied.
Answer.

- A: (1,1), (1,3), (2,2), (2,3)
- B: (1,3), (2,3)
- C: all locations
- D: (1,2), (1,3), (2,1), (2,3)

Explanation.
i limits (B, C) to (1,3) or (2,3) because those are the only positions adjacent to the road.
v removes (1,2) and (2,1) from the domain of (A) because those two positions have hills
vi removes (1,1) and (2, a
hill, nor adjacent to the road
(C) has no unary const plied.

Q1.2. Starting from the answer to Q1.1 (in which unary constra
domains of all variables after $A \rightarrow B$ is enforced.
Answer.

- A: (1,3), (2,2), (2,3)
- B: (1,3), (2,3)
- C: all locations
- D: (1,2), (1,3), (2,1), (2,3)

Explanation. After an arc is enforced, every value in the domain of the tail has at least one possible value in the domain of the head such that the two values satisfy all constraints with each other. If a value in the tail's domain does not satisfy this, that value can be removed, because it is guaranteed not to be a part of a solution without backtracking on some previously selected variable. In this case, (1,1) in A's domain is not adjacent to either (1,3) or (2,3), so B has no possible values and (1,1) can be removed from A's domain.

Q1.3. Continuing from the answer to Q1.2, provide the domains of all variables after $C \to B$ is enforced.
Answer.

- A: (1,3), (2,2), (2,3)
- B: (1,3), (2,3)
- C: (1,2), (1,3), (2,2), (2,3)
- D: (1,2), (1,3), (2,1), (2,3)

Explanation. (1,1) and (2,1) are removed from C, because if either value is assigned to C, there is no possible value in B's domain that satisfies all of the constraints. A and D's domains do not change.

Q1.4. What arcs got added to the queue while enforcing $C \to B$? Note that the queue contained $C \to D$, $D \to A$, $D \to B$, and $D \to C$ prior to enforcing $C \to B$.
Answer. $A \to C$, $B \to C$
Explanation. When a domain changes after enforcing an arc, all arcs that end at that variable must be re-added to the queue: $A \to C$, $B \to C$. This is done because some values in other variables' domains might have relied on the removed values to be consistent, so they have to be checked again.

Q1.5. Continuing fr[illegible] consistency until the queue is empty. Be[illegible] $\to D$, $C \to A$, $D \to B$, and $D \to C$, and any arcs that were added while enforcing C B.
Answer.

- A: (1,3), (2,2), (2,3)
- B: (1,3), (2,3)
- C: (1,2), (1,3), (2,2), (2,3)
- D: (1,2), (1,3), (2,1)

Explanation. Repeat the same process as parts 3 and 4 above for each arc in the queue. In this case the only value that gets removed from any domain while enforcing the arcs is (2,3) from D due to enforcing $D \to A$.

Q1.6. If arc consistency had resulted in all domains having a single value left, we would have already found a solution. Similarly, if it had found that any domain had no values left, we would have already found that no solution exists. Unfortunately, this is not the case in our example (as you should have found in the previous part). To solve the problem, we need to start searching. Use the MRV (minimum remaining values) heuristic to choose which variable gets assigned next (breaking any ties alphabetically). Which variable gets assigned next?
Answer. B
Explanation. B only has 2 possible values, while A and D have 3, and C has 4.

Q1.7. Continuing from Q1.6, the variable you selected should have two values left in its domain. We will use the least-constraining value (LCV) heuristic to decide which value to assign before continuing with the search. To choose which value is the least-constraining value, enforce arc consistency for each value (on a scratch piece of paper). For each value, count the total number of values remaining over all variables. Which value has the largest number of values remaining (and therefore is the least constraining value)?

Answer. (2,3)

Explanation. Assigning (1,3) to B leaves only 2 other values: two of the variables will be left with one value and one will have no values. Assigning (2,3) to B leaves 3 other values, one for each of the three unassigned variables. Since $3 > 2$, (2,3) is the least constraining value.

Q1.8. After assigning a variable, backtracking search with arc consistency enforces arc consistency before proceeding to the next variable.

Provide the domains of all variables after assignment of the least-constraining value to the variable you selected and enforcing arc consistency. Note that you already did this computation to determine which value was the LCV.

Answer.

- A: (1,3)
- B: (2,3)
- C: (2,2)
- D: (2,1)

Explanation. Assig) and D with value (2,1). These domains were determined while findin from part (i).

Q2. CSP Properties (True or False Answer) (18 points)

1. Even when using arc consistency, backtracking might be needed to solve a CSP. **True**
2. Even when using forward checking, backtracking might be needed to solve a CSP. **True**
3. When using backtracking search with the same rules to select unassigned variables and to order value assignments (in our case, usually Minimum Remaining Values and Least-Constraining Value, with alphabetical tiebreaking), arc consistency will always give the same solution as forward checking, if the CSP has a solution. **False**
4. For a CSP with binary constraints that has no solution, some initial values may still pass arc consistency before any variable is assigned. **True**
5. If the CSP has no solution, it is guaranteed that enforcement of arc consistency resulted in at least one domain being empty. **False**
6. If the CSP has a solution, then after enforcing arc consistency, you can directly read off the solution from resulting domains. **False**

7. In general, to determine whether the CSP has a solution, enforcing arc consistency alone is not sufficient; backtracking may be required. **True**

8. If after a run of arc consistency during the backtracking search we end up with the filtered domains of *all* of the not yet assigned variables being empty, this means the CSP has no solution. **False**

9. If after a run of arc consistency during the backtracking search we end up with the filtered domains of **all** of the not yet assigned variables being empty, this means the search should backtrack because this particular branch in the search tree has no solution. **True**

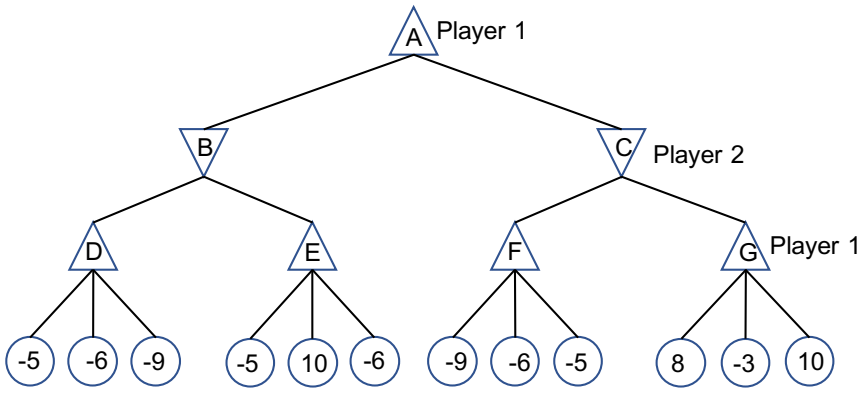
10. If after a run of arc consistency during the backtracking search we end up with the filtered domain of *one* of the not yet assigned variables being empty, this means the CSP has no solution. **False**

11. If after a run of arc consistency during the backtracking search we end up with the filtered domains of all of the not yet assigned variables each having exactly one value left, this means we have found a solution. **True**

12. If after a run of arc consistency during the backtracking search we end up with the filtered domains of all of the not yet assigned variables each having more than one value left, this means we have found a whole space of solutions and we can just pick any combination of values still left in the domains and that will be a solution. **False**

Q3. Adversarial S

Q3.1. Search Algorithms. (20 points) Cons n below. Triangles that point up, such as at the top node (root), represent choices for th player; triangles that point down represent choices for the minimiz the maximizing player are listed for each leaf node, represented by the val bottom of the tree.



1. (10 points) Assuming both players act optimally, carry out the minimax search algorithm. Enter the values for the letter nodes.

Answer. D=-5, E= 10, F = -5, G = 10, B = -5, C = -5, A = -5

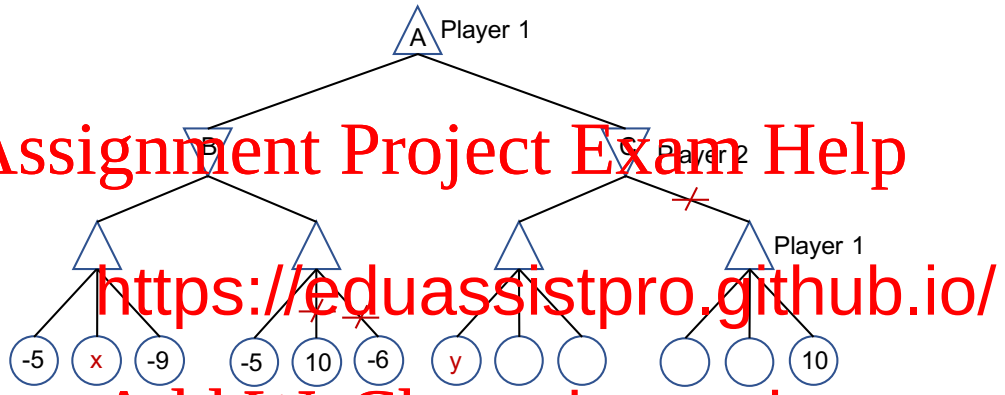
2. (10 points) Assuming both players act optimally, use alpha-beta pruning to find the value of the root node. The search goes from left to right; when choosing which child to visit first, choose the left-most unvisited child.

   - Enter the values of the letter nodes.

     Answer. D = -5, E= -5, B = -5, F = -5, C = -5, G = None, A = -5
   - Select the leaf nodes that don't get visited due to pruning.

     Answer. E-(10), E-(-6), C-G

Q3.2. Unknown Leaf Nodes. (10 points) Consider the following game tree with the same setting as Q3.1 except two of the leaves has an unknown payoff, x and y.



For what values of x and y such that the pruning show search goes from left to right; when choosing which child to visit first, choose the left-most un-visited child. Please specify your answer in one of the following forms:

- Write All if x can take on all values
- Write None if x has no possible values
- Use an inequality in the form $x < \{value\}$, $x > \{value\}$, or $\{value1\} < x < \{value2\}$ to specify an interval of values. As an example, if you think x can take on all values larger than 16, you should enter $x > 16$.

Answer. $x \leq -5$, $y \leq -5$

Explanation. In order for the two children of E to be pruned, then:

$$\max(-5, -9, x) \leq -5 \implies x \leq -5$$

Now we have $B = -5$. As a result, in order for the right-child of C to be pruned:

$$\max(y, -6. - 5) \leq -5 \implies y \leq -5.$$

Q4. Search with Uncertain Outcomes (20 points)

The following questions are completely unrelated to the above parts. Pacman is playing a tricky game. There are 4 portals to food dimensions. But, these portals are guarded by a ghost. Furthermore, neither Pacman nor the ghost know for sure how many pellets are behind each portal, though they know what options and probabilities there are for all but the last portal. Pacman moves first, either moving West or East. After which, the ghost can block **1** of the portals available. You have the following gametree. The maximizer node is Pacman. The minimizer nodes are ghosts and the portals are chance nodes with the probabilities indicated on the edges to the food. In the event of a tie, the left action is taken. Assume Pacman and the ghosts play optimally.

1. (10 points) Fill in values for the nodes that do not depend on X and Y.

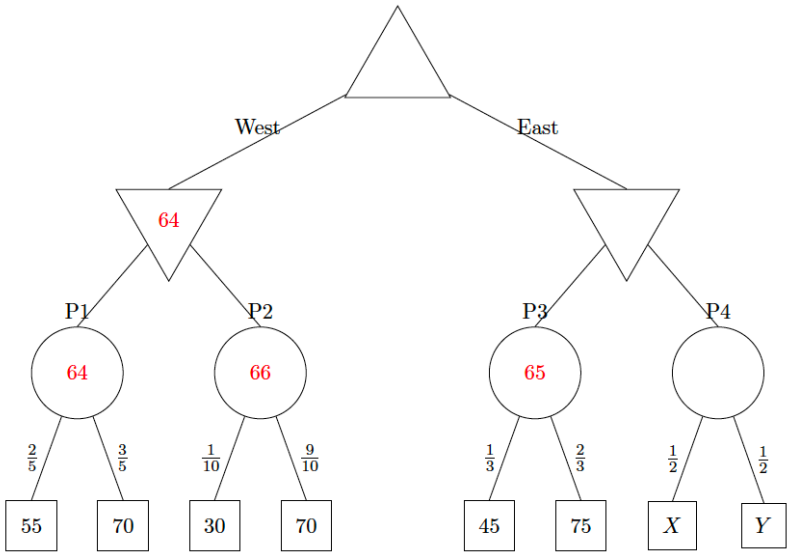
   Answer. See figure below

2. (10 points) What conditions must X and Y satisfy for Pacman to move East? What about to definitely reach the P4? Keep in mind that X and Y denote numbers of food pellets and must be **whole numbers**: $X, Y \in \{0, 1, 2, 3, \dots\}$.

   Answer. To move East: $X + Y > 128$.
   To reach $P4$: $X + Y = 129$

   Explanation. First, in order to pick a node A over a node B, then value(A) > value(B). Therefore, in order for Pacman to pick East, then:

$$\min(65, \frac{X+Y}{2}) > 64 \implies X + Y > 128$$



In addition, in order for Ghost to pick P4 over P3, then

$$\frac{X+Y}{2} < 65 \Rightarrow X+Y < 130$$

By combining the a