CS 471, Intro to Artificial Intelligence
Fall 2022
Jose Renteria

Written Assignment 1

Due Wednesday, October 26, 2022

Q1. Campus Layout

1.1. Provide the domains of all variables after unary constraints have been applied

The unary constraints are:

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

Below are highlighted in green the domains of variables: A,B,C,D after the above unary constraints have been applied.

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

1.2. Provide the domains of all variables after A → B is enforced.

After enforcing unary constraints, the domains of A and B are as such:

A	B
(1, 1) (1, 3) (2, 2) (2, 3)	(1, 3) (2, 3)

The domains after enforcing A → B are as such:

A	B
(1, 3) (2, 2) (2, 3)	(1, 3) (2, 3)

The domains of all variables after enforcing A → B are:

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

1.3. Provide the domains of all variables after C → B is enforced.

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

(1, 1) and (2,1) are removed from C, since if either value gets assigned to C, there exists no possible value in the domain of B that satisfies all constraints. The domains of A and D remain unchanged.

1.4. What arcs got added to the queue while enforcing C → B?

When a domain changes after enforcing an arc, all arcs which end at that variable must be added to the queue: A→C, B→C. This is done because some values in the domains of other variables might have depended on the removed values in order to remain consistent, so they need to be reevaluated.

1.5. Provide the domains of all variables after enforcing arc consistency until the queue is empty.

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

Here we repeat the same process as the previous problems for each arc in our queue. The only value that ends up getting removed when we enforce the arcs is (2,3) from domain D, since we enforced D→A.

1.6. Which variable gets assigned next?

Using the minimum remaining values (MRV) heuristic, we find that B is the variable which gets assigned next. This is because it only has 2 possible values in its domain, while D and A both have 3, and C has 4.

1.7. Which value has the largest number of values remaining (and therefore is the least constraining value)?

The least constraining value will be (2,3). Since assigning (1,3) to B provides only two remaining values: A or D with value (2,3) and C with (1, 2). If we assign (2,3) to B, this will provide 3 remaining values: A with value (1,3), C with value (2, 2) and D with value (2, 1). Since 2 < 3, (2, 3) will be our least constraining value here.

1.8. Provide the domains of all variables after assignment of the least-constraining value to the variable you selected and enforcing arc consistency

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

These are simply the domains we found in **Q1.8** when computing which was the least constraining value.

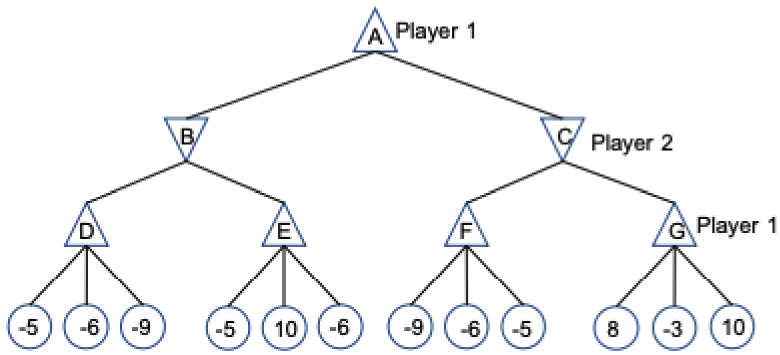
Q2. CSP Properties (T/F)

1. Even when using arc consistency, backtracking might be needed to solve a CSP
    a. **True**. Arc consistency as a standalone method runs the possibility of being insufficient. We can assume a case of a CSP with three variables, where the only constraint involves every variable. Because the constraints are not binary (i.e. have more than two options) arc consistency will not reduce any of the domains, and thus it is necessary to use backtracking search.
2. Even when using forward checking, backtracking might be needed to solve a CSP
    a. **False**. Consider a CSP with variables $A \in \{1,2\}$, $B \in \{1,2\}$, $C \in \{3,4\}$. The only constraint is that $A+B > C$. Using both LCV and MRV, with a tiebreaker of choosing the lowest value, the first assignment would be $A = 1$, yet the only solution is $A = 2$, $B = 2$, $C = 3$.
3. When using backtracking search with the same rules to select unassigned variables and to order value assignments, arc consistency will always give the same solution as forward checking, if the CSP has a solution.
    a. **False**. If the CSP has multiple solutions this is possible, we can use arc consistency to reduce the domains in such a way that the least constraining value is a different value. Consider a CSP with three variables A,B,C each with domain $\{1,2,3\}$. The constraints are that no two variables contain the same value, and that B must be less than both A and C. Using MRV and LCV, with a tiebreaker of selecting the lowest value, arc consistency will yield a solution of A=2, B=1, C=3. While a forward checking solution will yield A=3, B=1, C=2. These solutions obviously differ, which disproves the claim.
4. For a CSP with binary constraints that has no solution, some initial values may still pass arc consistency before any variable is assigned.
    a. **True**. Consider the following CSP: Three variables, A, B, C each with domains $\{1,2\}$. The only constraint is that no two variables can have the same value. Since arc consistency considers only two variables at a time, no values will be eliminated from any domain after enforcing arc consistency.
5. If the CSP has no solution, it is guaranteed that enforcement of arc consistency results in at least one domain being empty.
    a. **False**. Consider the CSP with three variables: A,B,C each with domains $\{1,2\}$. The only constraint is that no two variables can have the same value. Since arc consistency considers only two variables at a time, no values will be eliminated from any domain after enforcing arc consistency.
6. If the CSP has a solution, then after enforcing arc consistency, you can directly read off the solution from resulting domains.
    a. **False**. Consider the above CSP, except each variable now has domain $\{1,2,3\}$. Arc consistency will still not remove any value from any domain.

7. In general, to determine whether the CSP has a solution, enforcing arc consistency alone is not sufficient; backtracking may be required.
    a. **True**. As we can see from previous conclusions, arc consistency alone is not sufficient, and it is usually going to be necessary to use a backtracking search.
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.
    a. **False**. These cases imply that along the current branch of the tree there exists no solution for this CSP, and that we must perform backtracking search. In the special case where there are no variables assigned, this statement must be true.
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.
    a. **True**. If we have an empty domain, this means no value in the domain can satisfy all of the constraints of the currently assigned variables, which means we must go back and change one of these assignments. In the case where no variables have been assigned, this means the CSP has no solution.
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
    a. **False**. See statement 8. In the special case where there are no variables assigned, this statement must also be true.
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.
    a. **True**. If there is exactly one value left in every domain, this implies that these values must satisfy all binary constraints of our problem. Thus, for every variable, the assignment of the single remaining value has at least one possible value for every other variable, and one of those values is the one that remains in the domain.
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.
    a. **False**. This case implies we must continue our search. For example, consider a CSP we mentioned earlier, with three variables: A, B, C, each with two domain values {1,2}, and the only constraint being that no two variables can have the same value. Arc consistency provides no effective filtering, but we do not yet know if there exists a solution here.

Q3. Adversarial Search

3.1



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

   In order for the maximizing player to act optimally, they will attempt to maximize utility at each step, and the minimizing player will try to minimize the utility that the maximizing player would get.

   Both will play their best moves at each step.

   D = max(-5, -6, -9) = -5
   E = max(-5, 10, -6) = 10
   B = min(-5, 10) = -5

   F = max(-9, -6, -5) = -5
   G = max(8, -3, 10) = 10
   C = min(-5, 10) = -5

   A = max(-5, -5) = -5

2. D = max(-5, -6, -9) = -5
   E = max(-5, 10, X) = 10
   B = min(-5, 10) = -5

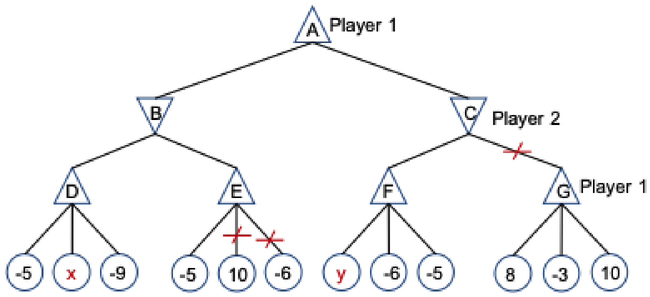
   Since 10 > -5, -6 does not have to be visited because node B will choose the min value of its children.

   F = max(-9, -6, -5) = -5
   G = max(8, -3, 10) = 10

C = min(-5, 10) = -5

A = max(-5, -5) = -5

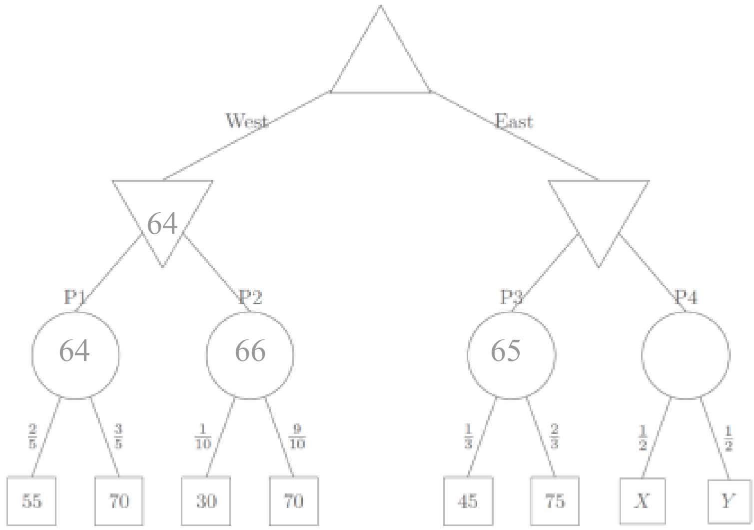
3.2.



For what values of x and y such that the pruning shown in the figure will be achieved?

y < *-6*
x <= -5

Q4. Search with Uncertain Outcomes



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

   i. see graph above

2. In order to move East, $X + Y$ must be > 128
   To reach P4, $X + Y > 129$
   In order to pick A over B, value(A) $>$ value(B). Also, the expected value of our parent node of X is $(X+Y)/2$.
   Implying,

   $min(65, (X+Y)/2) > 64$

   $(X+Y)/2 > 64$

   So, $X + Y > 129$ implies value(A) $>$ value(B)

   To ensure we reach X or Y, we also have $(X+Y)/2 < 65$/
   Implying the following inequality,

   $128 < X+Y < 130$

   So, $X,Y \in \mathbb{N} \Rightarrow X + Y = 129$