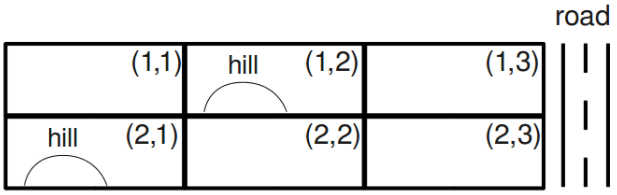
1. (44 points) (Campus Layout) You are asked to determine the layout of a new, small college. The campus will have four structures: an administration structure (A), a bus stop (B), a classroom (C), and a dormitory (D). Each structure (including the bus stop) must be placed somewhere on the grid shown below.



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. We recommend you work out the solutions to the following questions on a sheet of scratch paper, and then enter your results below.

(a) Which of the constraints above are unary constraints?

√ ***(i)*** □ (ii) □ (iii) □ (iv) √ ***(v)*** √ ***(vi)*** □ (vii) □ None of these

(b) Select the domains of all variables after unary constraints have been applied.

A: √ ***(1,1)*** □ (1,2) √ ***(1,3)*** □ (2,1) √ ***(2,2)*** √ ***(2,3)***

B: □ (1,1) □ (1,2) √ ***(1,3)*** □ (2,1) □ (2,2) √ ***(2,3)***

C: √ ***(1,1)*** √ ***(1,2)*** √ ***(1,3)*** √ ***(2,1)*** √ ***(2,2)*** √ ***(2,3)***

D: □ (1,1) √ ***(1,2)*** √ ***(1,3)*** √ ***(2,1)*** □ (2,2) √ ***(2,3)***

(c) Let's start from the table above (the answer to Part b) and enforce arc consistency. Initally, the queue contains all arcs (in alphabetical order). Let's examine what happens when enforcing A → B. After enforcing unary constraints, the domains of A and B are:

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

Which of the following contains the correct domains after enforcing A → B? Pay attention to which variable's domain changes and which side of the arc it's on.

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

i

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

ii

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

iii

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

iv

□ i √ ***ii*** □ iii □ iv

(d) Starting from the answer to Part b (in which unary constraints are enforced), select the domains of all variables after A → B is enforced.

A: □ (1,1) □ (1,2) √ ***(1,3)*** □ (2,1) √ ***(2,2)*** √ ***(2,3)***
B: □ (1,1) □ (1,2) √ ***(1,3)*** □ (2,1) □ (2,2) √ ***(2,3)***
C: √ ***(1,1)*** √ ***(1,2)*** √ ***(1,3)*** √ ***(2,1)*** √ ***(2,2)*** √ ***(2,3)***
D: □ (1,1) √ ***(1,2)*** √ ***(1,3)*** √ ***(2,1)*** □ (2,2) √ ***(2,3)***

(e) You should verify that enforcing consistency for A → C, A → D, B → A, B → C, B → D, and C → A do not change the domains of any variables. After enforcing these arcs, the next is C → B. Continuing from the previous parts, select the domains of all variables after C → B is enforced.

A: □ (1,1) □ (1,2) √ ***(1,3)*** □ (2,1) √ ***(2,2)*** √ ***(2,3)***
B: □ (1,1) □ (1,2) √ ***(1,3)*** □ (2,1) □ (2,2) √ ***(2,3)***
C: □ (1,1) √ ***(1,2)*** √ ***(1,3)*** □ (2,1) √ ***(2,2)*** √ ***(2,3)***
D: □ (1,1) √ ***(1,2)*** √ ***(1,3)*** √ ***(2,1)*** □ (2,2) √ ***(2,3)***

(f) What arcs got added to the queue while enforcing C → B? Remember that the queue contained C → D, D → A, D → B, and D → C prior to enforcing C → B.

□ A → B √ ***A → C*** □ A → D □ B → A √ ***B → C*** □ B → D
□ C → A □ C → B □ C → D □ D → A □ D → B □ D → C
□ None of the above

(g) Continuing from the previous parts, select the domains of all variables after enforcing arc consistency until the queue is empty. Remember that the queue currently contains C → D, D → A, D → B, D → C, and any arcs that were added while enforcing C → B.

A: □ (1,1) □ (1,2) √ ***(1,3)*** □ (2,1) √ ***(2,2)*** √ ***(2,3)***
B: □ (1,1) □ (1,2) √ ***(1,3)*** □ (2,1) □ (2,2) √ ***(2,3)***
C: □ (1,1) √ ***(1,2)*** √ ***(1,3)*** □ (2,1) √ ***(2,2)*** √ ***(2,3)***
D: □ (1,1) √ ***(1,2)*** √ ***(1,3)*** √ ***(2,1)*** □ (2,2) □ (2,3)

(h) 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?

□ A √ ***B*** □ C □ D

(i) 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)?

□ (1,1) □ (1,2) □ (1,3) □ (2,1) □ (2,2) √ ***(2,3)***

(j) After assigning a variable, backtracking search with arc consistency enforces arc consistency before proceeding to the next variable.

Select 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.

A: □ (1,1) □ (1,2) √ ***(1,3)*** □ (2,1) □ (2,2) □ (2,3)
B: □ (1,1) □ (1,2) □ (1,3) □ (2,1) □ (2,2) √ ***(2,3)***
C: □ (1,1) □ (1,2) □ (1,3) □ (2,1) √ ***(2,2)*** □ (2,3)
D: □ (1,1) □ (1,2) □ (1,3) √ ***(2,1)*** □ (2,2) □ (2,3)

(k) Is the answer to the previous part a solution to the CSP?

√ ***Yes*** □ No

Explanation.

(a) Unary constraints are constraints that involve a single variable. (i) only constrains (B), (v) only constrains (A), and (vi) only constrains (D). All the other constraints involve more than one variable.

(b) (i) limits (B) 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 (2,2) from the domain of (D) because those two positions are neither on a hill, nor adjacent to the road; (C) has no unary constraints, and thus can still be any value after unary constraints are applied.

(c) 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.

(d) These are the same as part 2, except with (1,1) removed from A's domain due to enforcing A $\rightarrow$ B.

(e) (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.

(f) When a domain changes after enforcing an arc, all arcs that end at that variable must be re-added to the queue: A$\rightarrow$C, B$\rightarrow$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.

(g) Repeat the same process as parts 4 and 5 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$\rightarrow$A.

(h) B only has 2 possible values, while A and D have 3, and C has 4.

(i) 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 (shown in solution to part 10). Since 3 ¿ 2, (2,3) is the least constraining value.

(j) Assigning (2,3) to B leaves 3 other values: A with value (1,3), C with value (2,2) and D with value (2,1). These domains were determined while finding the least constraining value from part (i).

(k) Since each variable has exactly one value in its domain after enforcing arc consistency, the answer is a solution to the CSP.

2. (10 points) (CSP Properties)

(a) Select all of the following statements about CSPs that are true.

- √ ***Even when using arc consistency, backtracking might be needed to solve a CSP.***
- √ ***Even when using forward checking, backtracking might be needed to solve a CSP.***
- □ None of the above.

(b) Select all of the following statements about CSPs that are true.

- □ 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.
- √ ***For a CSP with binary constraints that has no solution, some initial values may still pass arc consistency before any variable is assigned.***
- □ None of the above.

Explanation.

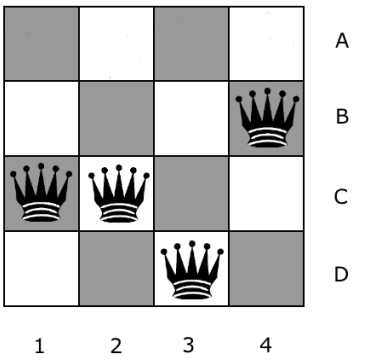
(a) Arc consistency is often not sufficient on its own to solve a CSP. Consider a CSP with three variables, where the only constraint involves all three variables. Because it is not a binary constraint, arc consistency will not reduce any of the domains, and normal backtracking search is necessary. Explicitly, let the variables and domains be $A \in \{1, 2\}$, $B \in \{1, 2\}$, and $C \in \{3, 4\}$ and let the only constraint be $A + B > C$. Using MRV and LCV, and breaking ties by choosing the lowest value/variable, the first assignment would be $A = 1$, while the only solution is $A = 2, B = 2, C = 3$.

The second choice is also seen to be true by considering the CSP just described.

(b) The first situation can occur if the CSP has multiple solutions, and arc consistency reduces the domains in a way such that a different value is the least constraining value. For example, consider the following CSP: There are three variables: A, B, and C, each with domain $\{1, 2, 3\}$. The constraints are that no two variables can have the same value, and that B must be less than both A and C. Using LCV and MRV, and breaking ties by selecting the lowest value or variable, with arc consistency, the solution will be $A = 2, B = 1, C = 3$. Meanwhile, using forward checking, the solution will be $A = 3, B = 1, C = 2$.

To see that the second option is true, consider the CSP with three variables, A, B, and C, each with domain $\{1, 2\}$, and the only constraint being that no two variables have the same value. Enforcing arc consistency will not eliminate any value from any domain, because arc consistency only considers two variables at a time.

3. (5 points) (4-Queens) The min-conflicts algorithm attempts to solve CSPs iteratively. It starts by assigning some value to each of the variables, ignoring the constraints when doing so. Then, while at least one constraint is violated, it repeats the following: (1) randomly choose a variable that is currenly violating a constraint, (2) assign to it the value in its domain such that after the assignment the total number of constraints violated is minimized (among all possible selections of values in its domain).



In this question, you are asked to execute the min-conflicts algorithm on a simple problem: the 4-queens problem in the figure shown below. Each queen is dedicated to its own column (i.e. we have variables Q_1, Q_2, Q_3, and Q_4 and the domain for each one of them is $\{A, B, C, D\}$). In the configuration shown below, we have $Q_1 = C$, $Q_2 = C$, $Q_3 = D$, $Q_4 = B$. Two queens are in conflict if they share the same row, diagonal, or column (though in this setting, they can never share the same column).

You will execute min-conflicts for this problem three times, starting with the state shown in the figure above. When selecting a variable to reassign, min-conflicts chooses a conflicted variable at random. For this problem, assume that your random number generator always chooses the leftmost conflicted queen. When moving a queen, move it to the square in its column that leads to the fewest conflicts with other queens. If there are ties, choose the topmost square among them.

We recommend you work out the solutions to the following questions on a sheet of scratch paper, and then enter your results below.

(a) Starting with the queens in the configuration shown in the above figure, which queen will be moved, and where will it be moved to?

Queen: √ ***1*** □ 2 □ 3 □ 4

Position: √ ***A*** □ B □ C □ D

(b) Continuing from the previous part, which queen will be moved next, and where will it be moved to?

Queen: □ 1 √ ***2*** □ 3 □ 4

Position: √ ***A*** □ B □ C □ D

(c) Continuing from the previous part, which queen will be moved next, and where will it be moved to?

Queen: √ ***1*** □ 2 □ 3 □ 4

Position: □ A □ B √ ***C*** □ D

Explanation.

(a) Queens 1,2, and 3 are conflicted, so according to the specification, the leftmost queen is selected: Queen 1. It is moved to position A, because there are no conflicts with position A.

(b) Queens 2 and 3 are conflicted, so Queen 2 is selected. Positions A and C have one conflict, so the topmost is selected: position A.

(c) Now Queens 1 and 2 are conflicted, so Queen 1 is chosen again. This time position C has no conflicts, so it is moved back to position C.

4. (14 points) (Arc Consistency) Consider the problem of arranging the schedule for an event. There are three time slots: 1, 2, and 3. There are three presenters: A, B, and C. The variables for the CSP will then be A, B, and C, each with domain $\{1, 2, 3\}$. The following constraints need to be satisfied:

   - A, B, and C all need to take on different values
   - $A < C$

   (a) Enforce consistency for the arc $A \to C$, and then select which values remain for each variable. $\surd$ $A : 1$ $\surd$ $A : 2$ $\square$ $A : 3$ $\surd$ $B : 1$ $\surd$ $B : 2$ $\surd$ $B : 3$ $\surd$ $C : 1$ $\surd$ $C : 2$ $\surd$ $C : 3$

   (b) Starting from the result of the previous step, enforce consistency for the arc $B \to A$, and then select which values remain for each variable.
   $\surd$ $A : 1$ $\surd$ $A : 2$ $\square$ $A : 3$ $\surd$ $B : 1$ $\surd$ $B : 2$ $\surd$ $B : 3$ $\surd$ $C : 1$ $\surd$ $C : 2$ $\surd$ $C : 3$

   (c) Starting from the result of the previous step, enforce consistency for the arc $C \to A$, and then select which values remain for each variable.
   $\surd$ $A : 1$ $\surd$ $A : 2$ $\square$ $A : 3$ $\surd$ $B : 1$ $\surd$ $B : 2$ $\surd$ $B : 3$ $\square$ $C : 1$ $\surd$ $C : 2$ $\surd$ $C : 3$

Explanation.

(a) The constraint $A < C$ means that if A has value 3, C has no possible values, so 3 is removed from A's domain.

(b) Each value for B leaves A with at least one possible value in its domain ($1 \to \{2\}$, $2 \to \{1\}$, $3 \to \{1, 2\}$), so the domains do not change.

(c) The constraint $A < C$ means that if C has value 1, A has no possible values, so 1 is removed from C's domain.

5. (6 points) (Arc Consistency Properties) Assume you are given a CSP and you enforce arc consistency. Which of the following are true?

   $\square$ If the CSP has no solution, it is guaranteed that enforcement of arc consistency resulted in at least one domain being empty.

   $\square$ If the CSP has a solution, then after enforcing arc consistency, you can directly read off the solution from resulting domains.

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

   $\square$ None of the above.

Explanation. To see that the first option is false, consider the CSP with three variables, A, B, C, each with domains $\{1, 2\}$, and with one constraint—that no two variables have the same value. Because arc consistency only considers two variables at a time, enforcing it does not eliminate any values from the domains of this CSP.

To see why the second choice is false, consider the CSP just described, but now let each variable have domain $\{1, 2, 3\}$. Arc consistency again fails to remove values from the domains.

The third option is true. It is generally necessary to backtrack even when running arc consistency.

6. (12 points) (Backtracking Arc Consistency) We are given a CSP with only binary constraints. Assume we run backtracking search with arc consistency as follows. Initially, when presented with the CSP, one round of arc consistency is enforced. This first round of arc consistency will typically result in variables having pruned domains. Then we start a backtracking search using the pruned domains. In this backtracking search we use filtering through enforcing arc consistency after every assignment in the search.

   (a) Which of the following are true about this algorithm?

   - □ 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.
   - □ 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.
   - √ ***None of the above.***

   (b) Which of the following are true about this algorithm?

   - √ ***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.***
   - √ ***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 search should backtrack because this particular branch in the search tree has no solution.***
   - □ None of the above.

   (c) Which of the following are true about this algorithm?

   - √ ***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.***
   - □ 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.
   - √ ***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 can't know yet whether there is a solution somewhere further down this branch of the tree, and search has to continue down this branch to determine this.***

Explanation. If any domain is empty, that means that no value in that domain can satisfy all of the constraints with the currently assigned variables. This means that we have to go back and change at least one of those assignments. In the special case where no variables have been assigned, this means that the CSP has no solution. This explains the answers to (a) and (b).

To see that the first statement in (c) is true, note that if there is exactly one value left in every domain, that means that those values satisfy all of the binary constraints of the problem. Explicitly, 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.

To see that the second statement in (c) is false, note that this means we have to continue searching. For example, consider the CSP used in earlier solutions with three variables, A, B, and C, each of which has two values in its domain {1, 2}, with the only constraint being that no two variables can have the same value. Arc consistency does not filter anything out, but we don't know yet whether or not there is a solution. For the same reason, the third option is true.