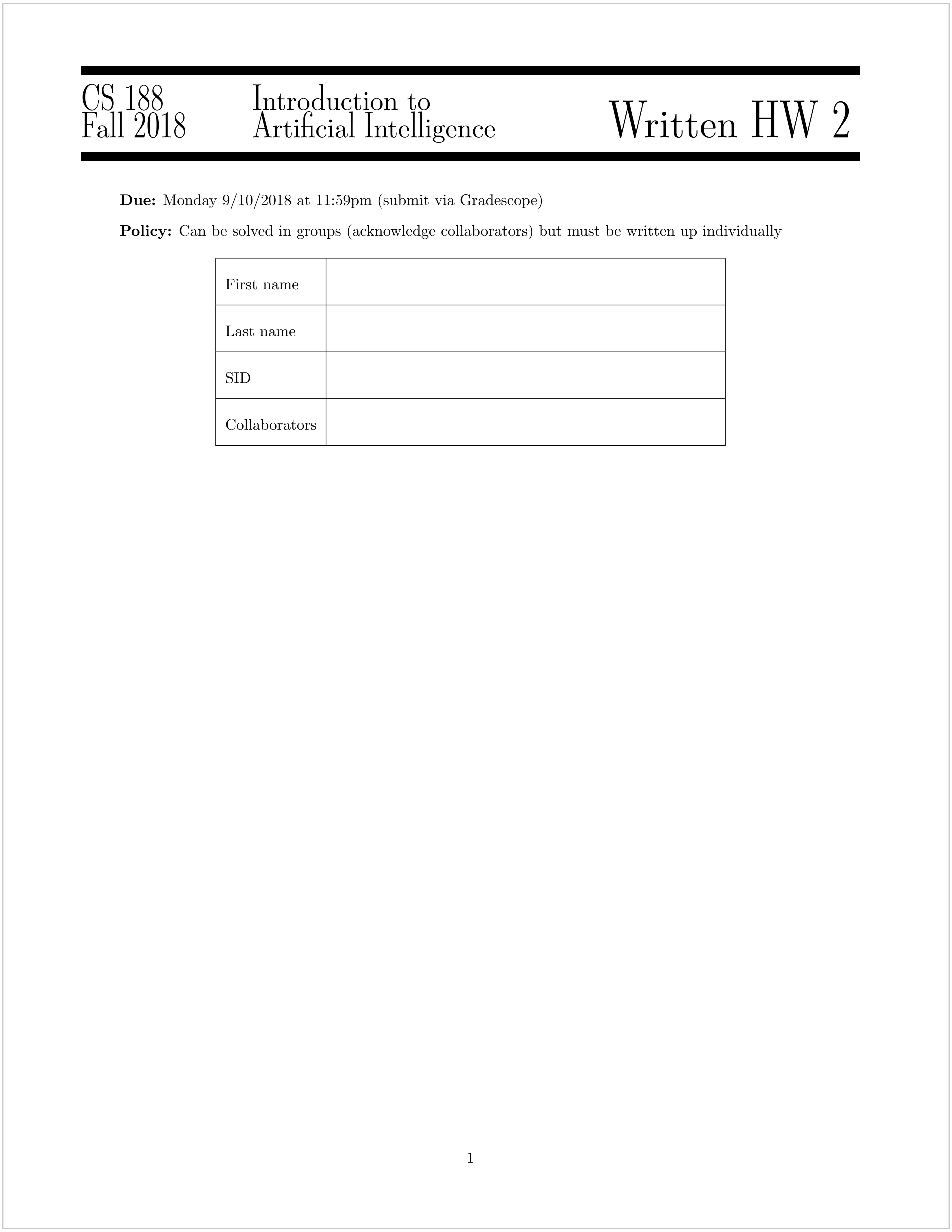
HW2: Written

Friday, March 15, 2019 11:18 AM

CS_188_Fa...



CS 188
Fall 2018

Introduction to
Artificial Intelligence

Written HW 2

Due: Monday 9/10/2018 at 11:59pm (submit via Gradescope)

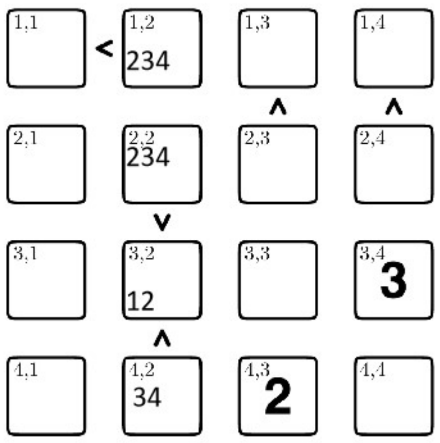
Policy: Can be solved in groups (acknowledge collaborators) but must be written up individually

First name	
Last name	
SID	
Collaborators	

Q1. CSP Futoshiki

Futoshiki is a Japanese logic puzzle that is very simple, but can be quite challenging. You are given an n x n grid, and must place the numbers $1, \ldots n$ in the grid such that every row and column has exactly one of each. Additionally, the assignment must satisfy the inequalities placed between some adjacent squares.

To the right is an instance of this problem, for size $n = 4$. Some of the squares have known values, such that the puzzle has a unique solution. (The letters mean nothing to the puzzle, and will be used only as labels with which to refer to certain squares). Note also that inequalities apply only to the two adjacent squares, and do not directly constrain other squares in the row or column.



Let's formulate this puzzle as a CSP. We will use 4^2 variables, one for each cell, with X_{ij} as the variable for the cell in the ith row and jth column (each cell contains its i, j label in the top left corner). The only unary constraints will be those assigning the known initial values to their respective squares (e.g. $X_{34} = 3$).

(a) Complete the formulation of the CSP using only binary constraints (in addition to the unary constraints specificed above. In particular, describe the domains of the variables, and all binary constraints you think are necessary. You do not need to enumerate them all, just describe them using concise mathematical notation. You are not permitted to use n-ary constraints where $n \geq 3$.

All variables in a row require have an Alldiff constraint. Xij != Xik

Unary constraints are the "given" values

All variables in a column have an Alldiff constraint: Xij != Xlj

Binary constraints are also added for inequalities. For ineq $X_{ij} > X_{kl}$, well, just use that.

The domains range from 1 thru 4

(b) After enforcing unary constraints, consider the binary constraints involving X_{14} and X_{24}. Enforce arc consistency on just these constraints and state the resulting domains for the two variables.

X14 = {1, 2}
X24 = {2, 4}

(c) Suppose we enforced unary constraints and ran arc consistency on this CSP, pruning the domains of all variables as much as possible. After this, what is the maximum possible domain size for any variable? [*Hint*: consider the least constrained variable(s); you should *not* have to run every step of arc consistency.]

X21 has the potential to remain with a domain size of 4

(d) Suppose we enforced unary constraints and ran arc consistency on the initial CSP in the figure above. What is the maximum possible domain size for a variable adjacent to an inequality?

3. If it's a LT ineq, it can't be min
4. If it's GT ineq, it can't be max of it's r

(e) By inspection of column 2, we find it is necessary that $X_{32} = 1$, despite not having found an assignment to any of the other cells in that column. Would running arc consistency find this requirement? Explain why or why not.

No. Arc consistency checks pairs of variables at a time. Specifically it considers each possible assignment of a given variable, and then checks to see if for all variables it is constrained with, it leaves a nonempty domain for future assignment

Q2. CSPs: Properties

(a) When enforcing arc consistency in a CSP, the set of values which remain when the algorithm terminates does not depend on the order in which arcs are processed from the queue.

[✓ **True**] **False**

It will add variables back into the queue if it becomes modified.

(b) In a general CSP with n variables, each taking d possible values, what is the maximum number of times a backtracking search algorithm might have to backtrack (i.e. the number of the times it generates an assignment, partial or complete, that violates the constraints) before finding a solution or concluding that none exists? (circle one)

0 $O(1)$ $O(nd^2)$ $O(n^2d^3)$ [✓ $O(d^n)$] ∞

It is possible we may searc
Conclude there is no solut

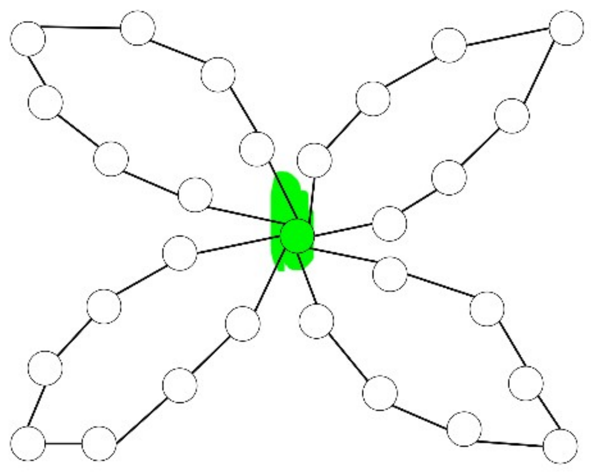
(c) What is the maximum number of times a backtracking search algorithm might have to backtrack in a general CSP, if it is running arc consistency and applying the MRV and LCV heuristics? (circle one)

0 $O(1)$ $O(nd^2)$ $O(n^2d^3)$ [✓ $O(d^n)$] ∞

(d) What is the maximum number of times a backtracking search algorithm might have to backtrack in a *tree-structured* CSP, if it is running arc consistency and using an optimal variable ordering? (circle one)

[✓ 0] $O(1)$ $O(nd^2)$ $O(n^2d^3)$ $O(d^n)$ ∞

(e) **Constraint Graph** Consider the following constraint graph:



We can try to force the problem to be a tree.
For the center node, we assign it some value. Then we apply
Tree-structured CSP search

We may need to try all values of the center node. But this remains a polynomial time
Algorithm.

What is Cutset Cond.

In two sentences or less, describe a strategy for efficiently solving a CSP with this constraint structure.

ch the entire search space and
ion. The entire search space is of size d^n