CS 188 Spring 2019 Introduction to Artificial Intelligence Written HW 5

Due: Monday 3/4/2019 at 11:59pm (submit via Gradescope)

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

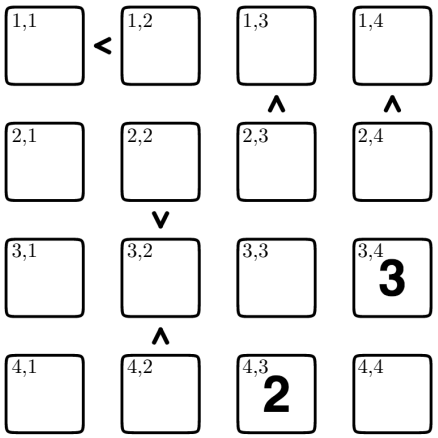
Submission: Your submission should be a PDF that matches this template. Each page of the PDF should align with the corresponding page of the template (page 1 has name/collaborators, question 1 begins on page 2, etc.). Do not reorder, split, combine, or add extra pages. The intention is that you print out the template, write on the page in pen/pencil, and then scan or take pictures of the pages to make your submission. You may also fill out this template digitally (e.g. using a tablet.)

First name	Ran
Last name	Liao
SID	3034504227
Collaborators	None

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 $n \geq 3$.

Xij = {1, 2, 3, 4}
Xik != Xjk
Xki != Xkj
Xab and Xcd satisfy specified inequality constrain if they are adjacent

X11 < X12, X13 < X23, X14 < X24, X32 < X22, X32 < X42
Miss the inequality constrains

(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, 3}
X24 = {2, 3, 4}

x34 = 3 Therefore, x14 and x24 cannot be 3

(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.]

4

correct

(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

correct

(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. This requires check constrains involving more than 2 variables.

correct

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**

(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 $O(1)$ $O(nd^2)$ $O(n^2d^3)$ $O(d^n)$ ∞

(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)

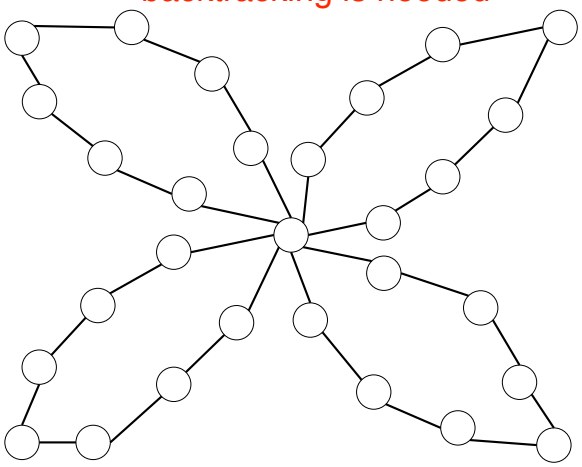
0 $O(1)$ $O(nd^2)$ $O(n^2d^3)$ $O(d^n)$ ∞

(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:

Tree-structured CSP has no cycles, therefore no backtracking is needed



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

Run backtrace search with arc consistence check starting from the vertex in the center

Treat the center vertex as a cutset. And then solve 4 tree-structured CSP.