

**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

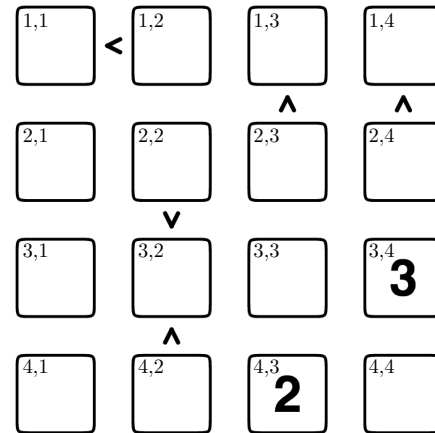
**Submission:** Your submission should be a PDF that matches this template. Each page of the PDF should align with the corresponding page of the template (page 1 has name/collaborators, question 1 begins on page 2, etc.). Do not reorder, split, combine, or add extra pages. The intention is that you print 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	Shenao
Last name	Zhang
SID	3034487184
Collaborators	None

# Q1. CSP Futoshiki

Futoshiki is a Japanese logic puzzle that is very simple, but can be quite challenging. You are given an  $n \times n$  grid, and must place the numbers  $1, \dots, n$  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  $i$ th row and  $j$ th 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 specified 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  $n \geq 3$ .

Domain of  $X$ : 1,2,3,4 for  $X$  not in unary constraints. And  $X(4,3)=2, X(3,4)=3$ .

$X(4,2) > X(3,2); X(2,2) > X(3,2); X(2,3) > X(1,3); X(2,4) > X(1,4); X(1,2) > X(1,1)$

Domain doesn't contain unary constraints And the 2-ary constraints also contains different row/column elements cannot be the same.

- (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.

$X(2,4): 1,2,3,4; X(1,4): 1,2,3$

No 3 in both cases because  $X(3,4)$  is already 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.

Yes, because from  $X(2,2), X(4,2)$  to  $X(3,2)$ , we can find that  $X(2,2)$  (OR  $X(4,2)$ ): 2,3,4  
 $X(3,2)$ : 1,2,3; from  $X(1,2)$  to  $X(1,1)$ ,  $X(1,2)$ : 2,3,4; and so  $X(1,2), X(2,2), X(4,2)$  are 2,3,4.  
 So  $X(3,2)$ : 1.

2

Since we only inspect column 2,  $X(1,2)$  can be 1.  
 (Not comparing  $X(1,2)$  and  $X(1,1)$ )

## 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

True

- (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

$O(1)$

$O(nd^2)$

$O(n^2d^3)$

$O(d^n)$

$\infty$

correct

- (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

$O(1)$

$O(nd^2)$

$O(n^2d^3)$

$O(d^n)$

$\infty$

It should be the same as (b).

- (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

$O(1)$

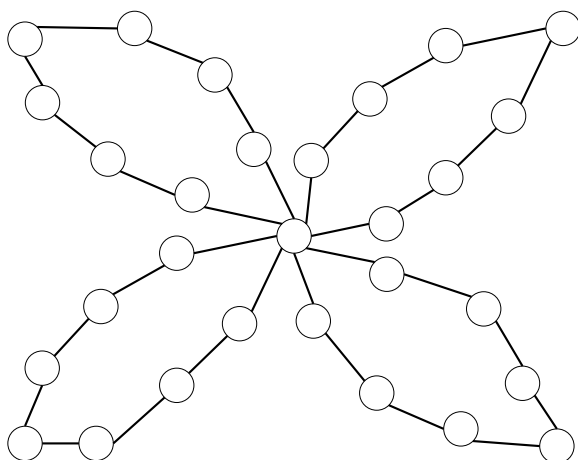
$O(nd^2)$

$O(n^2d^3)$

$O(d^n)$

$\infty$

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



Answer is 0. Applying arc consistency to a tree-structured CSP guarantees that no backtracking is required, if variables are assigned starting at the root and moving down towards the leaves.

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

Remove the middle node, get four tree structure. Then backtrack in a tree-structured CSP.

Correct