Clearer Definition

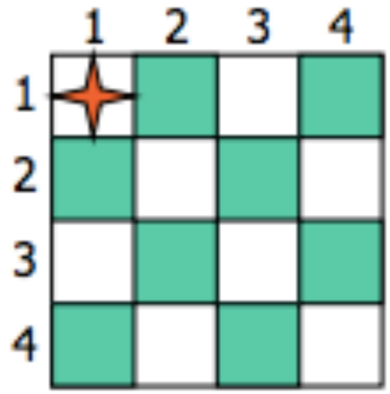
Definition: Arc consistency

- A constraint C_xy is said to *be arc consistent* wrt x if for each value v of x there is an allowed value of y
- Similarly, we define that C_xy is arc consistent wrt y
- A binary CSP is arc consistent iff every constraint C_xy is arc consistent wrt x as well as y
- When a CSP is not arc consistent, we can make it arc consistent, e.g. by using AC3
  - This is also called "enforcing arc consistency"

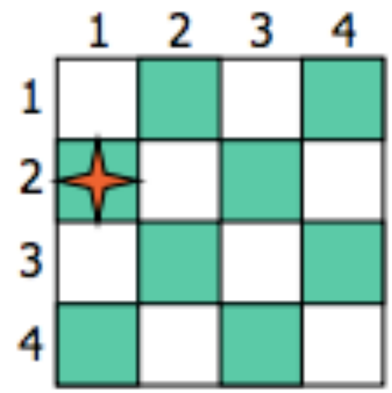
X is the tail

Chess as a CSP

Let's define the 4-queens problem as a CSP with the variable Xi denoting the position (row) of the queen on column i.



X1 = 1



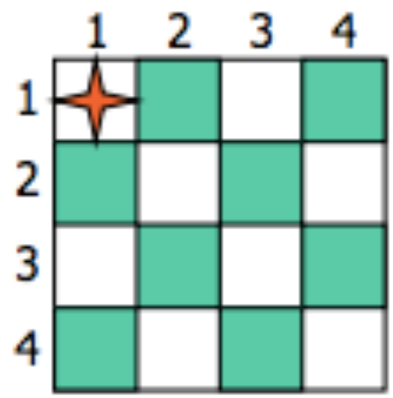
X1 = 2

Remember the constraints: two queens attack each other when the are in the same row, the same column or on the same diagonal. We want to place n=4 queens on the board so no queen is attacking another.

CSP Challenge Question 1

Suppose we set X1 = 1

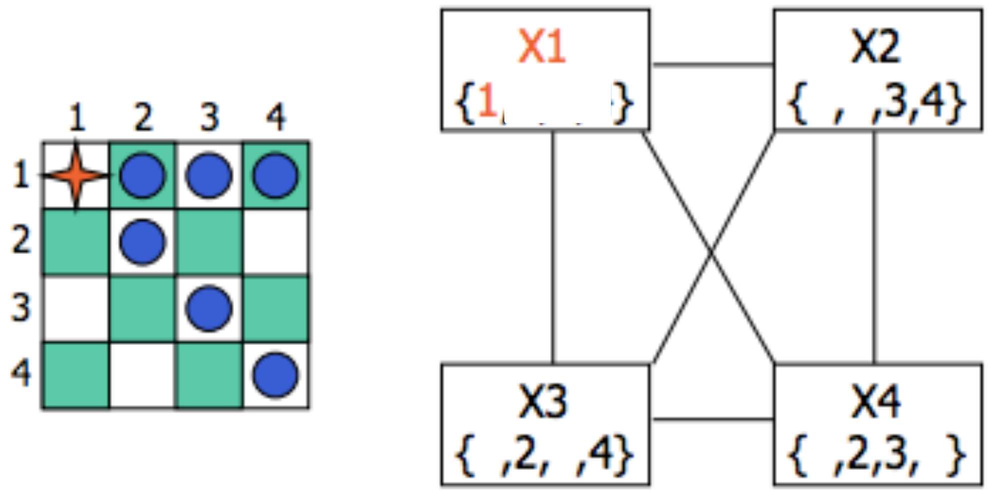
Show the effect of forward checking on the domains of the remaining variables
(I suggest crossing off values in the lists below:)



X1 = 1

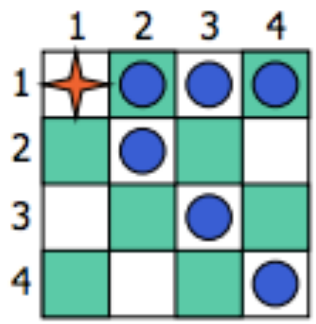
Domain X2 = {1, 2, 3, 4}
Domain X3 = {1, 2, 3, 4}
Domain X4 = {1, 2, 3, 4}

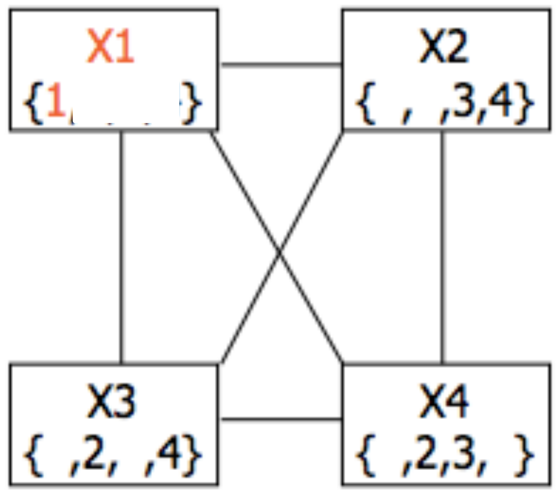
Answer 1



Forward checking will delete values from the domains of all other variables, as shown

Question 2

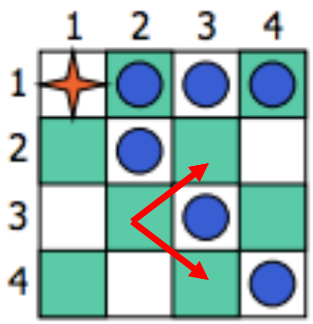


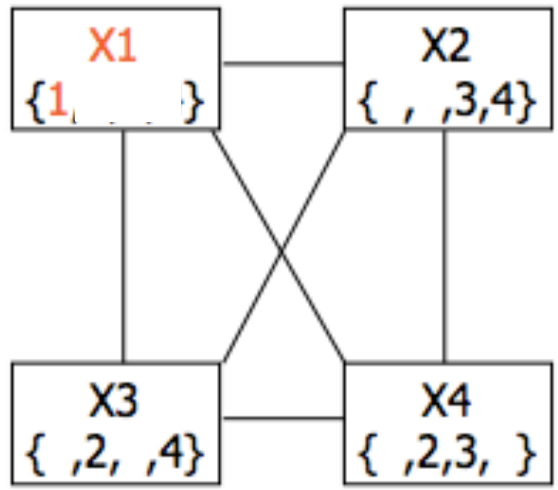


Is this CSP now arc consistent?

(for the purposes of this question – assume that there is one constraint between each pair of queens that rules out all attacks)

Answer 2

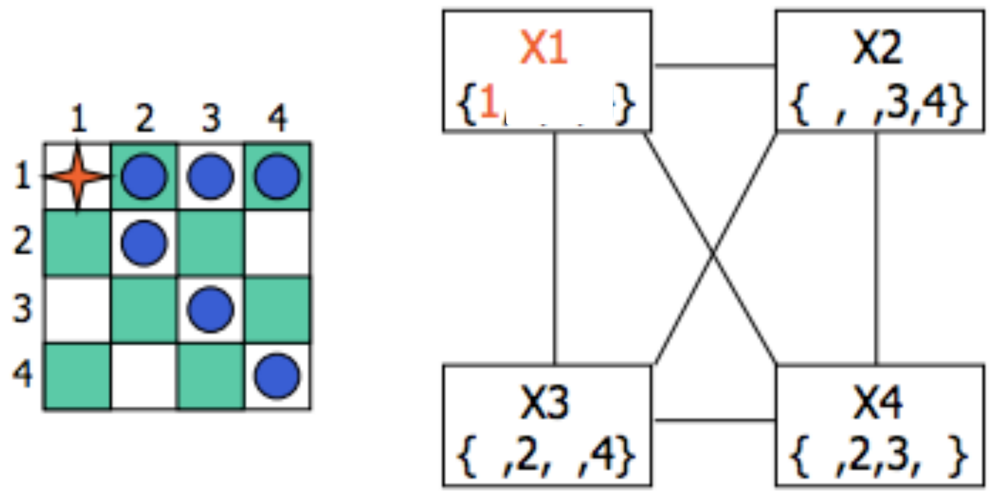




No, the constraint between X2 and X3 is not consistent with respect to X2 There exists a value in the domain of X2 (specifically X2=3) such that NO value for X3 will work.

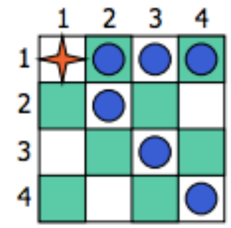
Furthermore, the constraint between X4 and X3 is not consistent with respect to X4, because X4=3 also leaves X3 with no legal values

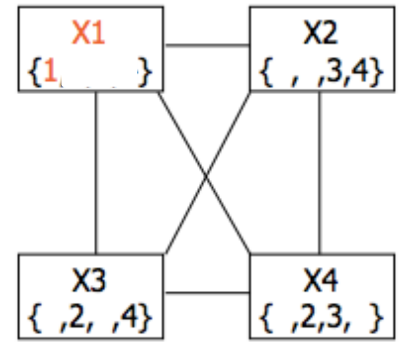
Question 3



Simulate the behavior of AC3 to make the CSP arc consistent
First subquestion, what goes on the queue?

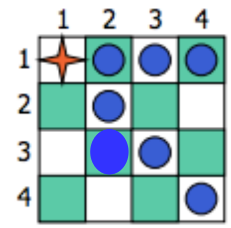
Answer 4

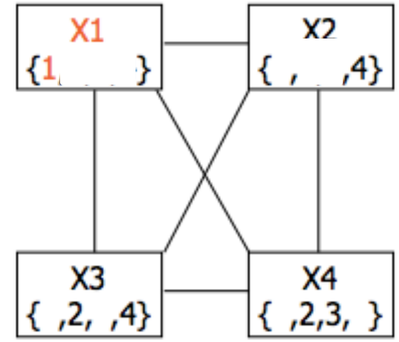




For each pair of variables, you need to put a directed constraint.
I'll write X2→X3 to mean the constraint wrt X2 (ie X2 is the tail)
For this example, let's ignore constraints with X1 because those constraints are consistent (as a result of forward checking) and can't become inconsistent because we've chose a single value for X1.
So the queue might be
<X2→X3, X2→X4, X3→X2, X3→X4, X4→X2, X4→X3>
We've already established that X2→X3 is inconsistent.
What does AC-3 do to fix this?

Answer 5

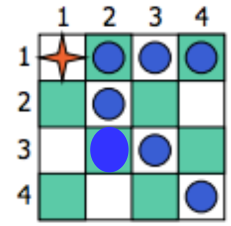


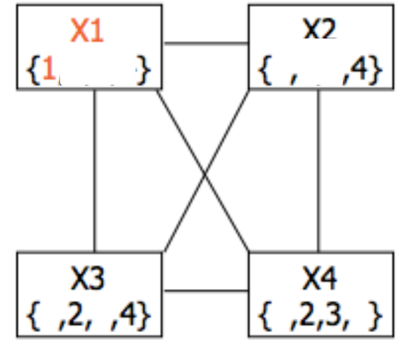


AC-3 deletes from the domain of … X2..
So now Domain(X2) = {4}
AC-3 also adds some more constraints onto the queue, X3→X2 and X4→X2, but since they are already there there is no change. So the queue is
<X2→X4, X3→X2, X3→X4, X4→X2, X4→X3>

Is X2→X4 consistent?

Answer 6

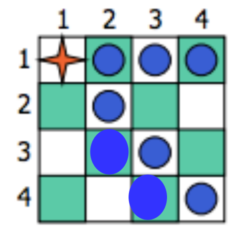


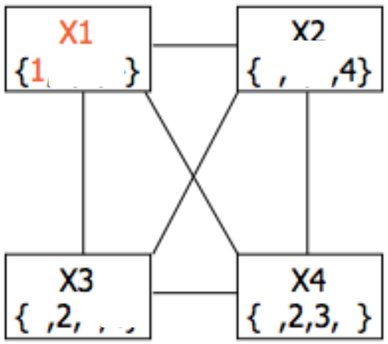


Yep. Now the queue is
<X3→X2, X3→X4, X4→X2, X4→X3>

Is X3→X2 consistent?

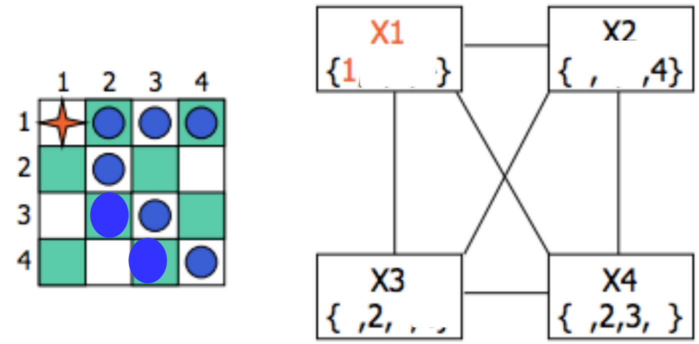
Answer 7





Nope. We need to delete X3=4.
That means we need to add some stuff to the queue.
So what's the queue become?

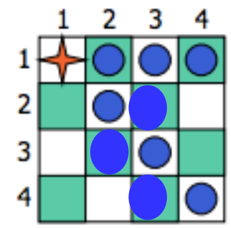
Answer 8

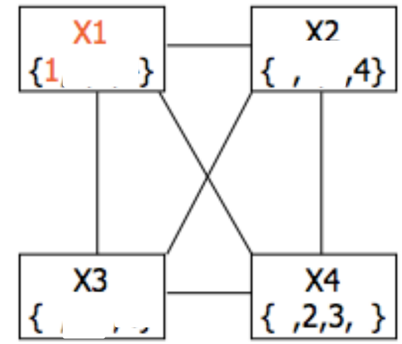


We add X2→X3 and X4→X3 but the latter was already there so we get
<X3→X4, X4→X2, X4→X3, X2→X3>

Now what happens when we process the next constraint?

Answer 9

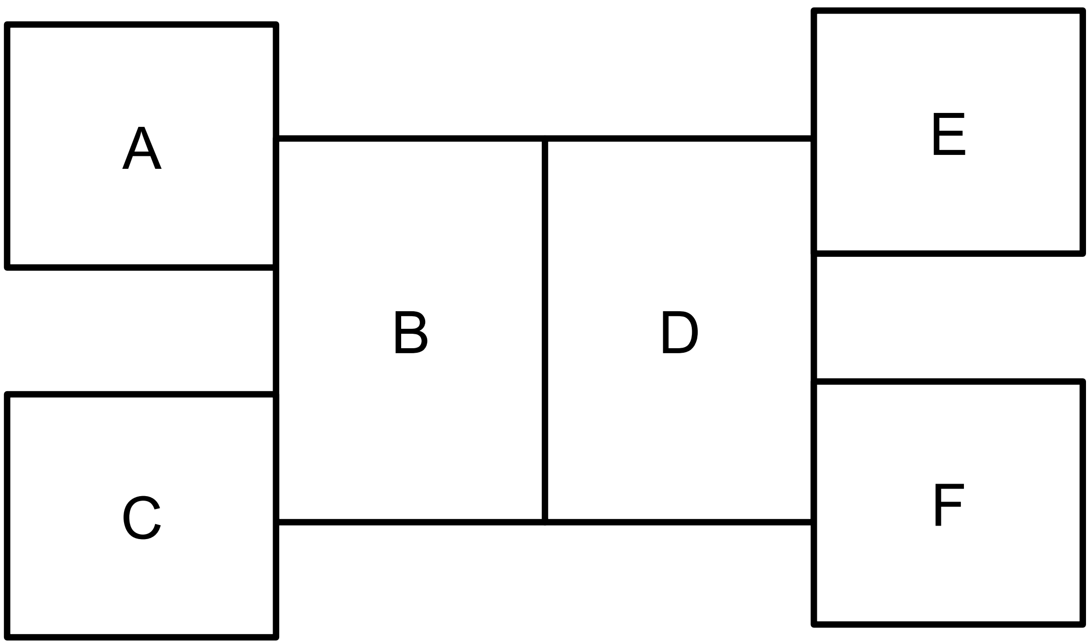




X3→X4 is inconsistent so we need to remove X3=2, but now X3's domain is empty, which means that the CSP is unsolvable. So the very first decision to Assign X1=1 was a mistake.

In fact, following the pseudocode, AC3 will keep running and remove some more stuff – a bit pointless. But I'll stop here.

Part II – Tree structured CSPs



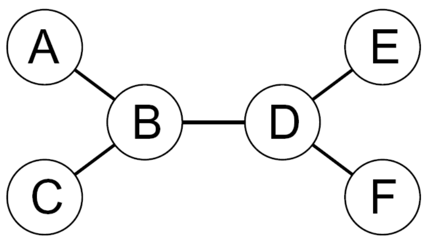
Let's color this!

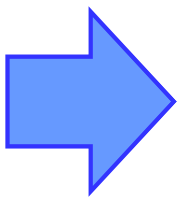
Tree-Structured CSPs

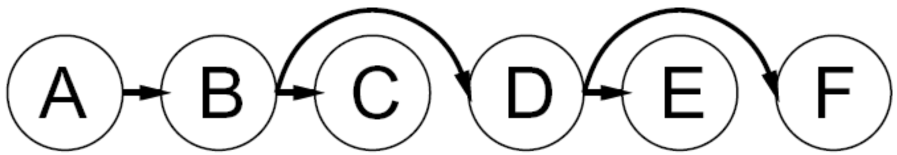
- Algorithm for tree-structured CSPs:
  1. Order: Choose a root variable, order variables so that parents precede children
  2. Remove backward: For i = n : 2, apply RemoveInconsistent(Parent(X_i),X_i)
  3. Assign forward: For i = 1 : n, assign X_i consistently with Parent(X_i)

Tree-Structured CSPs

- Algorithm for tree-structured CSPs:
  1. Order: Choose a root variable, order variables so that parents precede children



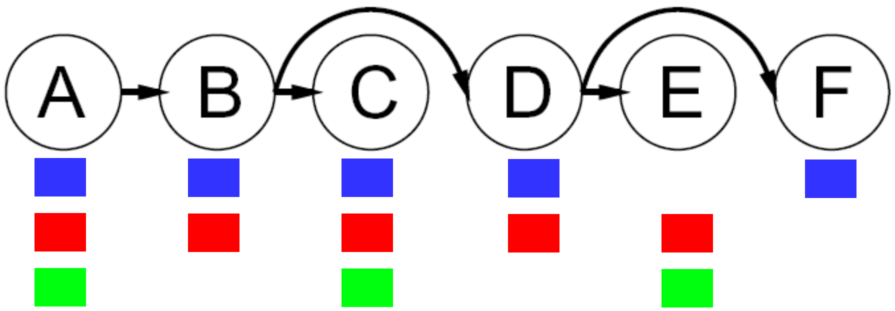




My choice to start with A as the root is arbitrary – could have started with anything else.
It also doesn't matter if B comes before C in the ordering etc.

Question 10

- Algorithm for tree-structured CSPs:
  1. Order: Choose a root variable, order variables so that parents precede children
  2. Remove backward: For i = n : 2, apply RemoveInconsistent(Parent(X_i),X_i)

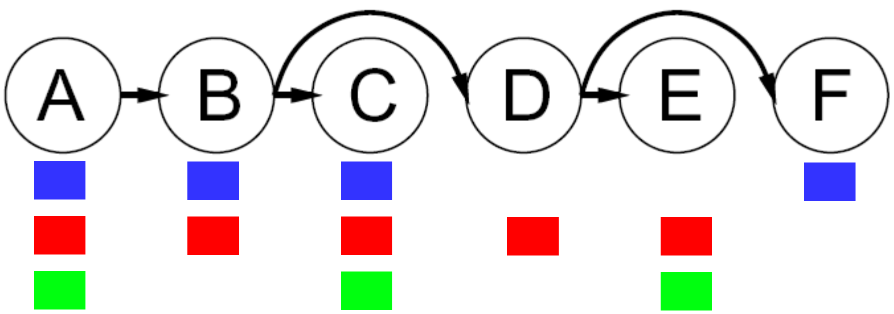


Suppose that the initial legal colors are as I show above

Simulate step 2 of the algorithm (I suggest cross off colors in the diagram above)

Answer 10

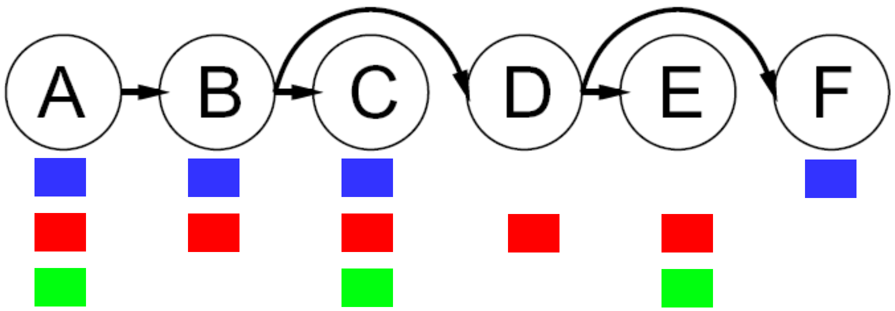
- Algorithm for tree-structured CSPs:
  1. Order: Choose a root variable, order variables so that parents precede children
  2. Remove backward: For i = n : 2, apply RemoveInconsistent(Parent(X_i),X_i)



When processing D→F, we need to remove blue from the domain of D
What about when we process D→E?

Answer 11

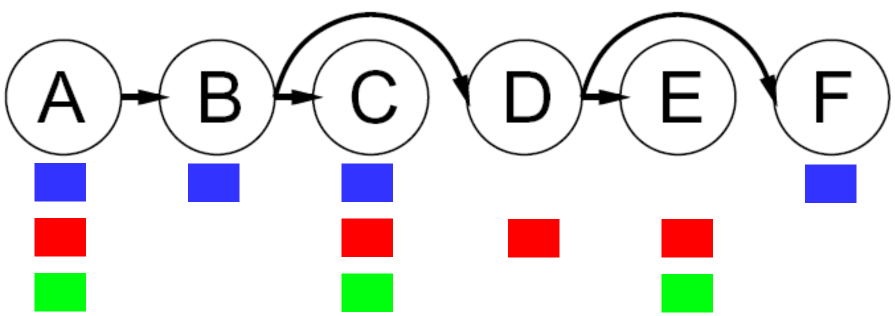
- Algorithm for tree-structured CSPs:
  1. Order: Choose a root variable, order variables so that parents precede children
  2. Remove backward: For i = n : 2, apply RemoveInconsistent(Parent(X_i),X_i)



When processing D→E, we don't do anything.
We would only remove something from the parent, D, but red is consistent, because we can make E green. So we just leave it as is.
What about B→D?

Answer 12

- Algorithm for tree-structured CSPs:
  1. Order: Choose a root variable, order variables so that parents precede children
  2. Remove backward: For i = n : 2, apply RemoveInconsistent(Parent(X_i),X_i)

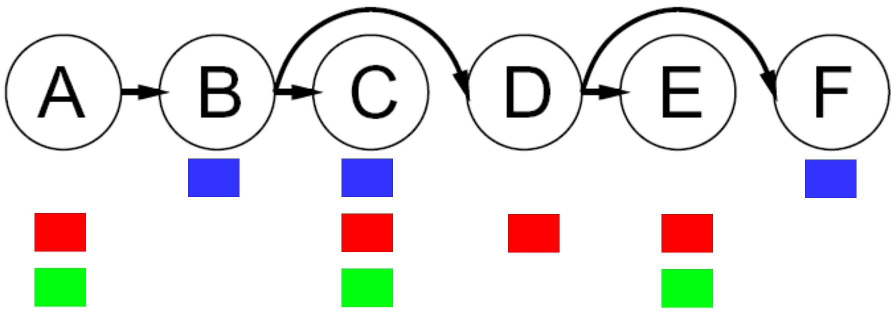


When processing B→C, we don't do anything.

What about A→B?

Answer 13

- Algorithm for tree-structured CSPs:
  1. Order: Choose a root variable, order variables so that parents precede children
  2. Remove backward: For i = n : 2, apply RemoveInconsistent(Parent(X_i),X_i)
  3. Assign forward: For i = 1 : n, assign X_i consistently with Parent(X_i)



Right, we delete blue from A.

Now simulate step 3.

Any choice for A is ok. B will be blue. C can be red or green. D is red, E will be green… It all works!!