

Centre for  
Data Analytics



# Introduction to Constraint Programming

Helmut Simonis

CRT-AI CP Week 2025



# Introduction to Constraint Programming

# Licence

This work is licensed under the Creative Commons Attribution-Noncommercial-Share Alike 3.0 Unported License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc-sa/3.0/> or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.



# Acknowledgments

This publication has emanated from research conducted with the financial support of Science Foundation Ireland under Grant number 12/RC/2289-P2 at Insight the SFI Research Centre for Data Analytics at UCC, which is co-funded under the European Regional Development Fund.

A version of this material was developed as part of the ECLiPSe ELearning course:

<https://eclipseclp.org/ELearning/index.html>.

Support from Cisco Systems and the Silicon Valley Community Foundation is gratefully acknowledged.

# Objectives

- Overview of Core Constraint Programming
- Three Main Concepts
  - Constraint Propagation
  - Global Constraints
  - Customizing Search
- Topics will be treated in more detail in later parts of the school
- Based on Examples, not Formal Description

# Outline

- Why Constraint Programming?
- Constraint Propagation
- Global Constraints
- Customizing Search

# Tutorial Based on ECLiPSe ELearning Course

- Self-study course in constraint programming
- Supported by Cisco Systems and Silicon Valley Community Foundation
- Multi-media format, video lectures, slides, handout etc
- <https://eclipseclp.org/ELearning/index.html>

## Also Part of CRT-AI Constraint Week

- Annual one week course on CP and Optimization in Ireland
- Part of national training program for PhD students in AI
- <https://www.crt-ai.ie/>

# Constraint Programming - in a nutshell

- Declarative description of problems with
  - *Variables* which range over (finite) sets of values
  - *Constraints* over subsets of variables which restrict possible value combinations
  - A *solution* is a value assignment which satisfies all constraints
- Constraint propagation/reasoning
  - Removing inconsistent values for variables
  - Detect failure if constraint can not be satisfied
  - Interaction of constraints via shared variables
  - Incomplete
- Search
  - User controlled assignment of values to variables
  - Each step triggers constraint propagation
- Different domains require/allow different methods

# Constraint Programming is Different

- Declarative Programming
  - Concentrate on what you want
  - Not how to get there
  - Program  $\neq$  Algorithm
  - Program = Model
- Applied to Combinatorial Problems
  - No complete polynomial algorithms known (exist?)
  - CP less ad-hoc than heuristics
  - Models can evolve

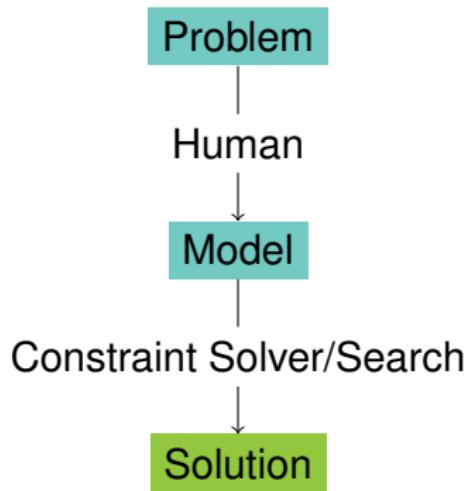
# A Subtractive Process



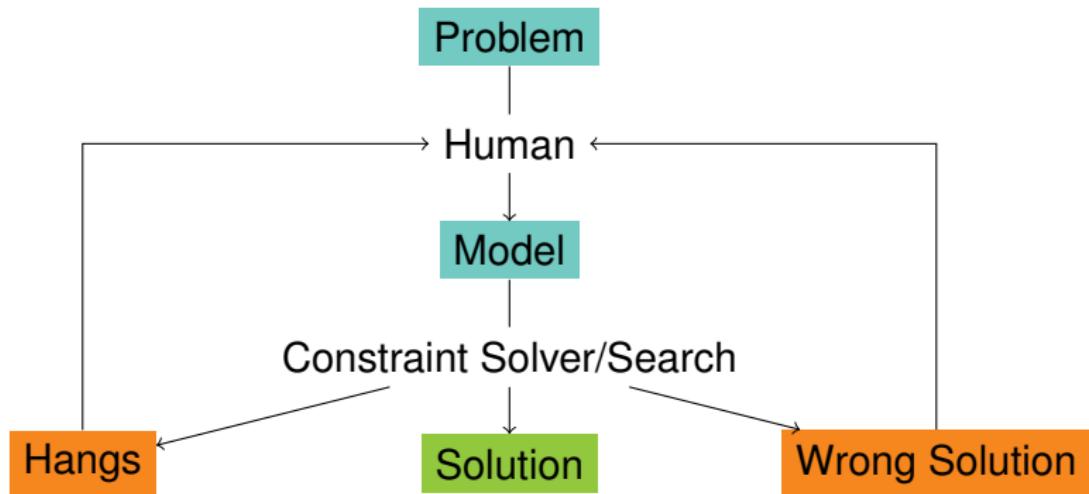
*“Oh, bosh, as Mr. Ruskin says. Sculpture, per se, is the simplest thing in the world. All you have to do is to take a big chunk of marble and a hammer and chisel, make up your mind what you are about to create and chip off all the marble you don’t want.” -Paris Gaulois.*

Source: <https://quoteinvestigator.com/2014/06/22/chip-away/>

# Basic Process



# More Realistic



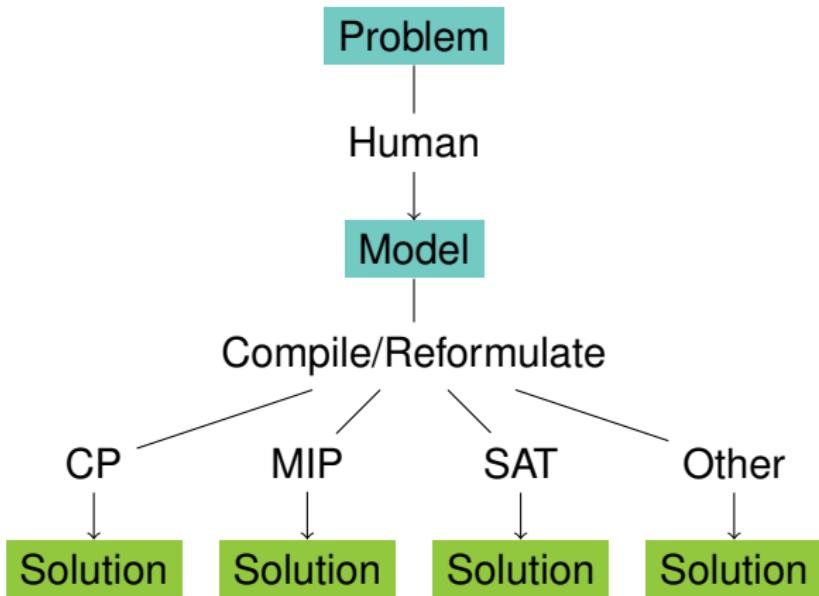
# Dual Role of Model

- Allows Human to Express Problem
  - Close to Problem Domain
  - Constraints as Abstractions
- Allows Solver to Execute
  - Variables as Communication Mechanism
  - Constraints as Algorithms

# Modelling Frameworks

- MiniZinc (NICTA, Monash University, Australia)
- NumberJack (Insight, Ireland)
- EssencePrime/SavilleRow (UK)
- CPMpy (KU Leuven)
- Allow use of multiple back-end solvers
- Compile model into variants for each solver
- A priori solver independent model(CP, MIP, SAT)

# Framework Process



# Why use Puzzles as Examples?

- Easy to understand the problem
- Solvable by hand without specialized knowledge
- Possible to compare automated to manual solving process

*The puzzle, though inanimate, is presented as a solvable problem without lasting negative consequences, a very low-risk low-reward situation. By being a puzzle, the object is attempting to convince the user that it must be completed.*

Source: Every Day Rhetoric

# Part I

## Basic Constraint Propagation

## Example 1: SEND+MORE=MONEY

- Example of Finite Domain Constraint Problem
- Models and Programs
- Constraint Propagation and Search
- Some Basic Constraints: linear arithmetic, alldifferent, disequality
- A Built-in search
- Visualizers for variables, constraints and search

# Outline

Problem

Program

Constraint Setup

Search

Points to Remember

# Problem Definition

## A Crypt-Arithmetic Puzzle

We begin with the definition of the SEND+MORE=MONEY puzzle. It is often shown in the form of a hand-written addition:

$$\begin{array}{r} \text{S} \quad \text{E} \quad \text{N} \quad \text{D} \\ + \text{M} \quad \text{O} \quad \text{R} \quad \text{E} \\ \hline \text{M} \quad \text{O} \quad \text{N} \quad \text{E} \quad \text{Y} \end{array}$$

The puzzle was first proposed by Henry Dudeney in the Strand Magazine from 1924.

# Rules

- Each character stands for a digit from 0 to 9.
- Numbers are built from digits in the usual, positional notation.
- Repeated occurrence of the same character denote the same digit.
- Different characters denote different digits.
- Numbers do not start with a zero.
- The equation must hold.

$$\begin{array}{r} & S & E & N & D \\ + & M & O & R & E \\ \hline M & O & N & E & Y \end{array}$$

# Outline

Problem

Program

Constraint Setup

Search

Points to Remember

# Model

- Each character is a variable, which ranges over the values 0 to 9.
- An *alldifferent* constraint between all variables, which states that two different variables must have different values. This is a very common constraint, which we will encounter in many other problems later on.
- Two *disequality constraints* (variable  $X$  must be different from value  $V$ ) stating that the variables at the beginning of a number can not take the value 0.
- An arithmetic *equality constraint* linking all variables with the proper coefficients and stating that the equation must hold.

# SEND+MORE=MONEY Models

- ECLiPSe [▶ Show](#)
- MiniZinc [▶ Show](#)
- NumberJack [▶ Show](#)
- CPMPy [▶ Show](#)
- Choco-solver [▶ Show](#)

# ECLiPSe Model

```
:  
- lib(ic).  
  
sendmore(Digits) :-  
    Digits = [S,E,N,D,M,O,R,Y],  
    Digits :: [0..9],  
    alldifferent(Digits),  
    S #\= 0,  
    M #\= 0,  
    1000*S + 100*E + 10*N + D  
        + 1000*M + 100*O + 10*R + E  
    #= 10000*M + 1000*O + 100*N + 10*E + Y,  
    labeling(Digits).
```

► Continue

# MiniZinc Model

```
include "alldifferent.mzn";
var 0..9: S;
var 0..9: E;
var 0..9: N;
var 0..9: D;
var 0..9: M;
var 0..9: O;
var 0..9: R;
var 0..9: Y;
constraint S != 0;
constraint M != 0;
constraint
    1000 * S + 100 * E + 10 * N + D
    + 1000 * M + 100 * O + 10 * R + E
    = 10000 * M + 1000 * O + 100 * N + 10 * E + Y;
constraint alldifferent([S,E,N,D,M,O,R,Y]);
solve satisfy;
```

► Continue

# NumberJack Model

(from <https://github.com/eomahony/Numberjack/>)

```
from Numberjack import *

def get_model():
    model = Model()
    s, m = VarArray(2, 1, 9)
    e, n, d, o, r, y = VarArray(6, 0, 9)
    model.add(      s*1000 + e*100 + n*10 + d +
                    m*1000 + o*100 + r*10 + e ==
                    m*10000 + o*1000 + n*100 + e*10 + y)
    model.add(AllDiff((s, e, n, d, m, o, r, y)))
    return s, e, n, d, m, o, r, y, model

def solve(param):
    s, e, n, d, m, o, r, y, model = get_model()
    solver = model.load(param['solver'])
    solver.setVerbosity(param['verbose'])
    solver.solve()
```

▶ Continue

# CPMpy Model (from <https://github.com/CPMpy/>)

```
from cpmpy import *
import numpy as np

s,e,n,d,m,o,r,y = intvar(0,9, shape=8)
model = Model(
    AllDifferent([s,e,n,d,m,o,r,y]),
    (sum([s,e,n,d] * np.array([1000, 100, 10, 1])) + \
     sum([m,o,r,e] * np.array([1000, 100, 10, 1])) + \
     sum([m,o,n,e,y] * np.array([10000, 1000, 100, 10, 1])), 
    s > 0,
    m > 0,
)
model.solve()
```

▶ Continue

# Choco-solver Model (from <https://choco-solver.org/>)

```
Model model = new Model("SEND+MORE=MONEY");
IntVar S = model.intVar("S", 1, 9, false);
IntVar E = model.intVar("E", 0, 9, false);
IntVar N = model.intVar("N", 0, 9, false);
IntVar D = model.intVar("D", 0, 9, false);
IntVar M = model.intVar("M", 1, 9, false);
IntVar O = model.intVar("O", 0, 9, false);
IntVar R = model.intVar("R", 0, 9, false);
IntVar Y = model.intVar("Y", 0, 9, false);

model.allDifferent(new IntVar[]{S, E, N, D, M, O, R, Y}).post();

IntVar[] ALL = new IntVar[]{
    S, E, N, D,
    M, O, R, E,
    M, O, N, E, Y};
int[] COEFFS = new int[]{
    1000, 100, 10, 1,
    1000, 100, 10, 1,
    -10000, -1000, -100, -10, -1};
model.scalar(ALL, COEFFS, "=" , 0).post();

Solver solver = model.getSolver();
solver.showStatistics();
solver.showSolutions();
solver.findSolution();
```

▶ Continue

## A Note on Syntax

- Some formulations may seem simpler than others
- This largely is an artifact of a very simple problem
- In most models, you do not write down constraints one by one
- You create constraints based on data
- Ease of integration becomes more important than syntax
- Debugging tools for those who need a debugger :-)

# Choice of Model

- This is *one* model, not *the* model of the problem
- Many possible alternatives
- Choice often depends on your constraint system
  - Constraints available
  - Reasoning attached to constraints
- Not always clear which is the *best* model
- Often: Not clear what is the *problem*

# Running the Program (MiniZinc IDE)

The screenshot shows the MiniZinc IDE interface. The main window displays the code for the SENDMORE puzzle:

```
1 include "alldifferent.mzn";
2 var 0..9: S;
3 var 0..9: E;
4 var 0..9: N;
5 var 0..9: D;
6 var 0..9: M;
7 var 0..9: O;
8 var 0..9: R;
9 var 0..9: Y;
10 constraint S != 0;
11 constraint M != 0;
12 constraint 1000 * S + 100 * E + 10 * N + D
13     + 1000 * M + 100 * O + 10 * R + E
14     = 10000 * M + 1000 * D + 100 * N + 10 * E + Y;
15 constraint alldifferent([S,E,N,D,M,O,R,Y]);
16
17 solve satisfy;
18
```

The output window below shows the results of running the solver:

```
Compiling sendmore.mzn
Running sendmore.mzn
S = 9;
E = 5;
N = 6;
D = 7;
M = 1;
O = 8;
R = 3;
Y = 2;
=====
Finished in 3s 103msec
```

At the bottom left, it says "Line: 16, Col: 1". At the bottom right, it says "3s 103msec".

# Question

- But how did the program come up with this solution?
- We show solution with ECLiPSe, other solvers vary slightly

# Outline

Problem

Program

## Constraint Setup

Domain Definition

AllDifferent Constraint

Disequality Constraints

Equality Constraint

Search

Points to Remember

# Domain Definition

```
var 0..9: S;  
var 0..9: E;  
var 0..9: N;  
var 0..9: D;  
var 0..9: M;  
var 0..9: O;  
var 0..9: R;  
var 0..9: Y;
```

# Domain Visualization

	0	1	2	3	4	5	6	7	8	9
S										
E										
N										
D										
M										
O										
R										
Y										

# Domain Visualization

Rows =  
Variables

	0	1	2	3	4	5	6	7	8	9
S										
E										
N										
D										
M										
O										
R										
Y										

# Domain Visualization

Columns = Values

	0	1	2	3	4	5	6	7	8	9
S										
E										
N										
D										
M										
O										
R										
Y										

# Domain Visualization

	0	1	2	3	4	5	6	7	8	9
S										
E										
N										
D										
M			Cells= State							
O										
R										
Y										

# Alldifferent Constraint

```
include "alldifferent.mzn";  
  
constraint alldifferent([S,E,N,D,M,O,R,Y]);
```

- Built-in alldifferent predicate included
- No initial propagation possible
- *Suspends*, waits until variables are changed
- When variable is fixed, remove value from domain of other variables
- *Forward checking*

# Alldifferent Visualization

Uses the same representation as the domain visualizer

	0	1	2	3	4	5	6	7	8	9
S										
E										
N										
D										
M										
O										
R										
Y										

# Disequality Constraints

```
constraint S != 0;  
constraint M != 0;
```

Remove value from domain

$$S \in \{1..9\}, M \in \{1..9\}$$

Constraints solved, can be removed

# Domains after Disequality

	0	1	2	3	4	5	6	7	8	9
S										
E										
N										
D										
M										
O										
R										
Y										

# Equality Constraint

- Normalization of linear terms
  - Single occurrence of variable
  - Positive coefficients
- Propagation

# Normalization

1000*S+	100*E+	10*N+	D
+1000*M+	100*O+	10*R+	E
10000*M+	1000*O+	100*N+	10*E+ Y

# Normalization

1000*S+	100*E+	10*N+	D
+1000*M+	100*O+	10*R+	E
<b>10000*M+</b>	1000*O+	100*N+	10*E+

---

# Normalization

1000*S+	100*E+	10*N+	D
+	100*O+	10*R+	E
<b>9000*M+</b>	<b>1000*O+</b>	<b>100*N+</b>	<b>10*E+ Y</b>

# Normalization

1000*S+	100*E+	10*N+	D
+	<b>100*O+</b>	10*R+	E
9000*M+ <b>1000*O+</b>	100*N+	10*E+	Y

# Normalization

1000*S+	100*E+	10*N+	D
	+ 10*R+	E	
9000*M+	<b>900*O+</b>	100*N+	10*E+ Y

# Normalization

1000*S+	100*E+	<b>10*N+</b>	D
		+ 10*R+	E
9000*M+	900*O+	<b>100*N+</b>	10*E+ Y

# Normalization

$$\begin{array}{r} 1000^*S+ \quad 100^*E+ \quad D \\ \quad \quad \quad + \quad 10^*R+ \quad E \\ \hline 9000^*M+ \quad 900^*O+ \quad \mathbf{90^*N+} \quad 10^*E+ \quad Y \end{array}$$

# Normalization

$$\begin{array}{r} 1000^*S+ \quad \mathbf{100^*E+} \quad \quad \quad D \\ \quad \quad \quad + \quad 10^*R+ \quad \mathbf{E} \\ \hline 9000^*M+ \quad 900^*O+ \quad 90^*N+ \quad \mathbf{10^*E+} \quad Y \end{array}$$

# Normalization

$$\begin{array}{r} 1000^*S+ \quad 91^*E+ \quad \quad \quad D \\ \quad \quad \quad + \quad 10^*R \\ \hline 9000^*M+ \quad 900^*O+ \quad 90^*N+ \quad \quad \quad Y \end{array}$$

# Simplified Equation

$$1000 * S + 91 * E + 10 * R + D = 9000 * M + 900 * O + 90 * N + Y$$

# Propagation

$$1000 * S^{1..9} + 91 * E^{0..9} + 10 * R^{0..9} + D^{0..9} = \\ 9000 * M^{1..9} + 900 * O^{0..9} + 90 * N^{0..9} + Y^{0..9}$$

# Propagation

$$\underbrace{1000 * S^{1..9} + 91 * E^{0..9} + 10 * R^{0..9} + D^{0..9}}_{1000..9918} =$$
$$\underbrace{9000 * M^{1..9} + 900 * O^{0..9} + 90 * N^{0..9} + Y^{0..9}}_{9000..89919}$$

# Propagation

$$\underbrace{1000 * S^{1..9} + 91 * E^{0..9} + 10 * R^{0..9} + D^{0..9}}_{9000..9918} =$$
$$\underbrace{9000 * M^{1..9} + 900 * O^{0..9} + 90 * N^{0..9} + Y^{0..9}}_{9000..9918}$$

# Propagation

$$\underbrace{1000 * S^{1..9} + 91 * E^{0..9} + 10 * R^{0..9} + D^{0..9}}_{9000..9918} =$$
$$\underbrace{9000 * M^{1..9} + 900 * O^{0..9} + 90 * N^{0..9} + Y^{0..9}}_{9000..9918}$$

Deduction:

$$M = 1, S = 9, O \in \{0..1\}$$

# Propagation

$$\underbrace{1000 * S^{1..9} + 91 * E^{0..9} + 10 * R^{0..9} + D^{0..9}}_{9000..9918} =$$
$$\underbrace{9000 * M^{1..9} + 900 * O^{0..9} + 90 * N^{0..9} + Y^{0..9}}_{9000..9918}$$

Deduction:

$$M = 1, S = 9, O \in \{0..1\}$$

Why? 

## Consider lower bound for $S$

$$\underbrace{1000 * S^{1..9} + 91 * E^{0..9} + 10 * R^{0..9} + D^{0..9}}_{9000..9918} = \underbrace{9000 * M^{1..9} + 900 * O^{0..9} + 90 * N^{0..9} + Y^{0..9}}_{9000..9918}$$

- Lower bound of equation is 9000
- Rest of lhs (left hand side) ( $91 * E^{0..9} + 10 * R^{0..9} + D^{0..9}$ ) is atmost 918
- $S$  must be greater or equal to  $\frac{9000 - 918}{1000} = 8.082$ 
  - otherwise lower bound of equation not reached by lhs
- $S$  is integer, therefore  $S \geq \lceil \frac{9000 - 918}{1000} \rceil = 9$
- $S$  has upper bound of 9, so  $S = 9$

## Consider upper bound of $M$

$$\underbrace{1000 * S^{1..9} + 91 * E^{0..9} + 10 * R^{0..9} + D^{0..9}}_{9000..9918} = \underbrace{9000 * M^{1..9} + 900 * O^{0..9} + 90 * N^{0..9} + Y^{0..9}}_{9000..9918}$$

- Upper bound of equation is 9918
- Rest of rhs (right hand side)  $900 * O^{0..9} + 90 * N^{0..9} + Y^{0..9}$  is at least 0
- $M$  must be smaller or equal to  $\frac{9918-0}{9000} = 1.102$
- $M$  must be integer, therefore  $M \leq \lfloor \frac{9918-0}{9000} \rfloor = 1$
- $M$  has lower bound of 1, so  $M = 1$

## Consider upper bound of $O$

$$\underbrace{1000 * S^{1..9} + 91 * E^{0..9} + 10 * R^{0..9} + D^{0..9}}_{9000..9918} = \underbrace{9000 * M^{1..9} + 900 * O^{0..9} + 90 * N^{0..9} + Y^{0..9}}_{9000..9918}$$

- Upper bound of equation is 9918
- Rest of rhs (right hand side)  $9000 * 1 + 90 * N^{0..9} + Y^{0..9}$  is at least 9000
- $O$  must be smaller or equal to  $\frac{9918 - 9000}{900} = 1.02$
- $O$  must be integer, therefore  $O \leq \lfloor \frac{9918 - 9000}{900} \rfloor = 1$
- $O$  has lower bound of 0, so  $O \in \{0..1\}$

# Propagation of equality: Result

	0	1	2	3	4	5	6	7	8	9
S	-	-	-	-	-	-	-	-	-	*
E										
N										
D										
M	*	*	-	-	-	-	-	-	-	-
O			*	*	*	*	*	*	*	*
R										
Y										

# Propagation of alldifferent

	0	1	2	3	4	5	6	7	8	9
S	-	-	-	-	-	-	-	-	-	*
E										
N										
D										
M	*	-	-	-	-	-	-	-	-	-
O		*	*	*	*	*	*	*	*	*
R										
Y										

# Propagation of alldifferent

	0	1	2	3	4	5	6	7	8	9
S										*
E										
N										
D										
M		*								
O										
R										
Y										

# Propagation of alldifferent

	0	1	2	3	4	5	6	7	8	9
S										█
E		█								
N		█								
D		█								
M	█	█	█							
O		█								
R		█								
Y		█								

# Propagation of alldifferent

	0	1	2	3	4	5	6	7	8	9
S										█
E										
N										
D										
M			█							
O	★									
R										
Y										

# Propagation of alldifferent

	0	1	2	3	4	5	6	7	8	9
S										█
E	█									
N	█									
D	█									
M			█							
O	█	*								
R	█									
Y	█									

# Propagation of alldifferent

	0	1	2	3	4	5	6	7	8	9
S										█
E										
N										
D										
M		█								
O	█									
R										
Y										

$$O = 0, [E, R, D, N, Y] \in \{2..8\}$$

# Waking the equality constraint

- Triggered by assignment of variables
- or update of lower or upper bound

# Removal of constants

$$1000 * 9 + 91 * E^{2..8} + 10 * R^{2..8} + D^{2..8} = \\ 9000 * 1 + 900 * 0 + 90 * N^{2..8} + Y^{2..8}$$

# Removal of constants

$$\begin{aligned}1000 * 9 + 91 * E^{2..8} + 10 * R^{2..8} + D^{2..8} = \\9000 * 1 + 900 * 0 + 90 * N^{2..8} + Y^{2..8}\end{aligned}$$

# Removal of constants

$$91 * E^{2..8} + 10 * R^{2..8} + D^{2..8} = 90 * N^{2..8} + Y^{2..8}$$

## Propagation of equality (Iteration 1)

$$\frac{91 * E^{2..8} + 10 * R^{2..8} + D^{2..8}}{204..816} = \frac{90 * N^{2..8} + Y^{2..8}}{182..728}$$

# Propagation of equality (Iteration 1)

$$\underbrace{91 * E^{2..8} + 10 * R^{2..8} + D^{2..8}}_{204..728} = 90 * N^{2..8} + Y^{2..8}$$

## Propagation of equality (Iteration 1)

$$\underbrace{91 * E^{2..8} + 10 * R^{2..8} + D^{2..8}}_{204..728} = 90 * N^{2..8} + Y^{2..8}$$

$$N \geq 3 = \lceil \frac{204 - 8}{90} \rceil, E \leq 7 = \lfloor \frac{728 - 22}{91} \rfloor$$

## Propagation of equality (Iteration 2)

$$91 * E^{2..7} + 10 * R^{2..8} + D^{2..8} = 90 * N^{3..8} + Y^{2..8}$$

## Propagation of equality (Iteration 2)

$$\underbrace{91 * E^{2..7} + 10 * R^{2..8} + D^{2..8}}_{204..725} = \underbrace{90 * N^{3..8} + Y^{2..8}}_{272..728}$$

## Propagation of equality (Iteration 2)

$$\underbrace{91 * E^{2..7} + 10 * R^{2..8} + D^{2..8}}_{272..725} = 90 * N^{3..8} + Y^{2..8}$$

## Propagation of equality (Iteration 2)

$$\underbrace{91 * E^{2..7} + 10 * R^{2..8} + D^{2..8}}_{272..725} = 90 * N^{3..8} + Y^{2..8}$$
$$E \geq 3 = \lceil \frac{272 - 88}{91} \rceil$$

## Propagation of equality (Iteration 3)

$$91 * E^{3..7} + 10 * R^{2..8} + D^{2..8} = 90 * N^{3..8} + Y^{2..8}$$

## Propagation of equality (Iteration 3)

$$\underbrace{91 * E^{3..7} + 10 * R^{2..8} + D^{2..8}}_{295..725} = \underbrace{90 * N^{3..8} + Y^{2..8}}_{272..728}$$

## Propagation of equality (Iteration 3)

$$\underbrace{91 * E^{3..7} + 10 * R^{2..8} + D^{2..8}}_{295..725} = 90 * N^{3..8} + Y^{2..8}$$

## Propagation of equality (Iteration 3)

$$\underbrace{91 * E^{3..7} + 10 * R^{2..8} + D^{2..8}}_{295..725} = 90 * N^{3..8} + Y^{2..8}$$
$$N \geq 4 = \lceil \frac{295 - 8}{90} \rceil$$

## Propagation of equality (Iteration 4)

$$91 * E^{3..7} + 10 * R^{2..8} + D^{2..8} = 90 * N^{4..8} + Y^{2..8}$$

## Propagation of equality (Iteration 4)

$$\underbrace{91 * E^{3..7} + 10 * R^{2..8} + D^{2..8}}_{295..725} = \underbrace{90 * N^{4..8} + Y^{2..8}}_{362..728}$$

## Propagation of equality (Iteration 4)

$$\underbrace{91 * E^{3..7} + 10 * R^{2..8} + D^{2..8}}_{362..725} = 90 * N^{4..8} + Y^{2..8}$$

## Propagation of equality (Iteration 4)

$$\underbrace{91 * E^{3..7} + 10 * R^{2..8} + D^{2..8}}_{362..725} = 90 * N^{4..8} + Y^{2..8}$$
$$E \geq 4 = \lceil \frac{362 - 88}{91} \rceil$$

## Propagation of equality (Iteration 5)

$$91 * E^{4..7} + 10 * R^{2..8} + D^{2..8} = 90 * N^{4..8} + Y^{2..8}$$

## Propagation of equality (Iteration 5)

$$\underbrace{91 * E^{4..7} + 10 * R^{2..8} + D^{2..8}}_{386..725} = \underbrace{90 * N^{4..8} + Y^{2..8}}_{362..728}$$

## Propagation of equality (Iteration 5)

$$\underbrace{91 * E^{4..7} + 10 * R^{2..8} + D^{2..8}}_{386..725} = 90 * N^{4..8} + Y^{2..8}$$

## Propagation of equality (Iteration 5)

$$\underbrace{91 * E^{4..7} + 10 * R^{2..8} + D^{2..8}}_{386..725} = 90 * N^{4..8} + Y^{2..8}$$

$$N \geq 5 = \lceil \frac{386 - 8}{90} \rceil$$

## Propagation of equality (Iteration 6)

$$91 * E^{4..7} + 10 * R^{2..8} + D^{2..8} = 90 * N^{5..8} + Y^{2..8}$$

## Propagation of equality (Iteration 6)

$$\underbrace{91 * E^{4..7} + 10 * R^{2..8} + D^{2..8}}_{386..725} = \underbrace{90 * N^{5..8} + Y^{2..8}}_{452..728}$$

## Propagation of equality (Iteration 6)

$$\underbrace{91 * E^{4..7} + 10 * R^{2..8} + D^{2..8}}_{452..725} = 90 * N^{5..8} + Y^{2..8}$$

## Propagation of equality (Iteration 6)

$$\underbrace{91 * E^{4..7} + 10 * R^{2..8} + D^{2..8}}_{452..725} = 90 * N^{5..8} + Y^{2..8}$$

$$N \geq 5 = \lceil \frac{452 - 8}{90} \rceil, E \geq 4 = \lceil \frac{452 - 88}{91} \rceil$$

No further propagation at this point

# Domains after setup

	0	1	2	3	4	5	6	7	8	9
S										■
E										
N										
D										
M		■								
O	■									
R										
Y										

# Outline

Problem

Program

Constraint Setup

## Search

Step 1

Step 2

Further Steps

Solution

Points to Remember

# Search

solve satisfy;

- Try to find a feasible solution, choice left to solver
- Naive search strategy shown here
  - Try variable in order given
  - Try values starting from smallest value in domain
  - When failing, backtrack to last open choice
  - *Chronological Backtracking*
  - *Depth First search*

# Search Tree Step 1

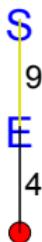
```
graph TD; S((S)) --- 9[9]; 9 --- E((E))
```

S  
9  
E

Variable  $S$  already fixed

## Step 2, Alternative $E = 4$

Variable  $E \in \{4..7\}$ , first value tested is 4



# Assignment $E = 4$

	0	1	2	3	4	5	6	7	8	9
S										
E					★	-	-	-		
N										
D										
M										
O										
R										
Y										

# Propagation of $E = 4$ , equality constraint

$$91 * 4 + 10 * R^{2..8} + D^{2..8} = 90 * N^{5..8} + Y^{2..8}$$

## Propagation of $E = 4$ , equality constraint

$$\underbrace{91 * 4 + 10 * R^{2..8} + D^{2..8}}_{386..452} = \underbrace{90 * N^{5..8} + Y^{2..8}}_{452..728}$$

## Propagation of $E = 4$ , equality constraint

$$\underbrace{91 * 4 + 10 * R^{2..8} + D^{2..8}}_{452} = 90 * N^{5..8} + Y^{2..8}$$

## Propagation of $E = 4$ , equality constraint

$$\underbrace{91 * 4 + 10 * R^{2..8} + D^{2..8}}_{452} = 90 * N^{5..8} + Y^{2..8}$$

$$N = 5, Y = 2, R = 8, D = 8$$

# Result of equality propagation

	0	1	2	3	4	5	6	7	8	9
S										■
E					■					
N						★	-	-	-	
D			-	-	-	-	-	-	★	
M		■								
O	■									
R			-	-	-	-	-	-	★	
Y			★	-	-	-	-	-	-	

# Propagation of alldifferent

	0	1	2	3	4	5	6	7	8	9
S										
E										
N										
D			-	-	-	-	-	-		*
M			*							
O	*									
R			-	-	-	-	-	-	*	
Y			*	-	-	-	-	-	-	

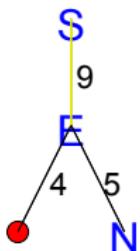
# Propagation of alldifferent

	0	1	2	3	4	5	6	7	8	9
S										
E										
N						*	-	-		
D			-	-	-	-	-	-	*	
M										
O	*									
R			-	-	-	-	-	-	*	
Y			*	-	-	-	-	-		

Alldifferent fails!

## Step 2, Alternative $E = 5$

Return to last open choice,  $E$ , and test next value



# Assignment $E = 5$

	0	1	2	3	4	5	6	7	8	9
S										
E					-	*	-	-		
N										
D										
M										
O										
R										
Y										

# Propagation of alldifferent

	0	1	2	3	4	5	6	7	8	9
S										
E					-	*	-	-		
N										
D										
M										
O										
R										
Y										

# Propagation of alldifferent

	0	1	2	3	4	5	6	7	8	9
S										
E						*				
N										
D										
M										
O										
R										
Y										

# Propagation of alldifferent

	0	1	2	3	4	5	6	7	8	9
S										■
E						■				
N										
D										
M		■								
O	■									
R										
Y										

$$N \neq 5, N \geq 6$$

# Propagation of equality

$$91 * 5 + 10 * R^{2..8} + D^{2..8} = 90 * N^{6..8} + Y^{2..8}$$

# Propagation of equality

$$\underbrace{91 * 5 + 10 * R^{2..8} + D^{2..8}}_{477..543} = \underbrace{90 * N^{6..8} + Y^{2..8}}_{542..728}$$

# Propagation of equality

$$\underbrace{91 * 5 + 10 * R^{2..8} + D^{2..8}}_{542..543} = 90 * N^{6..8} + Y^{2..8}$$

# Propagation of equality

$$\underbrace{91 * 5 + 10 * R^{2..8} + D^{2..8}}_{542..543} = 90 * N^{6..8} + Y^{2..8}$$

$$N = 6, Y \in \{2, 3\}, R = 8, D \in \{7..8\}$$

# Result of equality propagation

	0	1	2	3	4	5	6	7	8	9
S										
E										
N										
D			✗	✗	✗		✗			
M										
O										
R			-	-	-		-	-		✗
Y					✗		✗	✗	✗	

# Propagation of alldifferent

	0	1	2	3	4	5	6	7	8	9
S										
E										
N							*	-	-	
D		*	*	*		*				
M										
O	*									
R			-	-	-		-	-	*	
Y					*		*	*	*	

# Propagation of alldifferent

	0	1	2	3	4	5	6	7	8	9
S										
E										
N										
D										
M										
O										
R										
Y										

# Propagation of alldifferent

	0	1	2	3	4	5	6	7	8	9
S										
E										
N										
D										
M										
O										
R										
Y										

# Propagation of alldifferent

	0	1	2	3	4	5	6	7	8	9
S										
E										
N										
D										
M										
O										
R										
Y										

$$D = 7$$

# Propagation of equality

$$91 * 5 + 10 * 8 + 7 = 90 * 6 + Y^{2..3}$$

# Propagation of equality

$$\underbrace{91 * 5 + 10 * 8 + 7}_{542} = \underbrace{90 * 6 + Y^{2..3}}_{542..543}$$

# Propagation of equality

$$\underbrace{91 * 5 + 10 * 8 + 7 = 90 * 6 + Y^{2..3}}_{542}$$

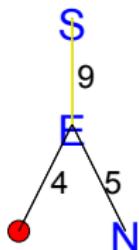
# Propagation of equality

$$\underbrace{91 * 5 + 10 * 8 + 7 = 90 * 6 + Y^{2..3}}_{542}$$
$$Y = 2$$

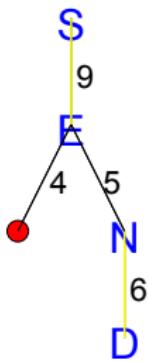
# Last propagation step

	0	1	2	3	4	5	6	7	8	9
S										
E										
N										
D										
M										
O										
R										
Y										

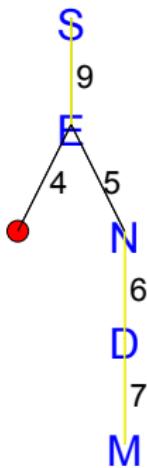
# Further Steps: Nothing more to do



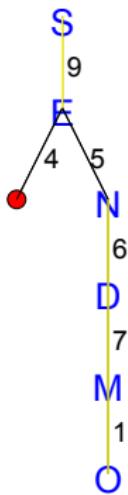
## Further Steps: Nothing more to do



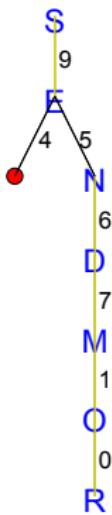
## Further Steps: Nothing more to do



## Further Steps: Nothing more to do



# Further Steps: Nothing more to do



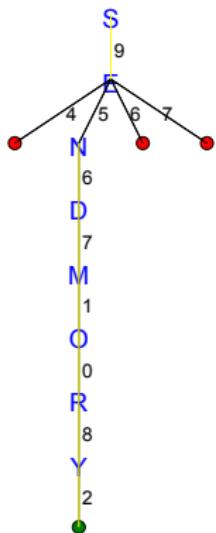
# Further Steps: Nothing more to do



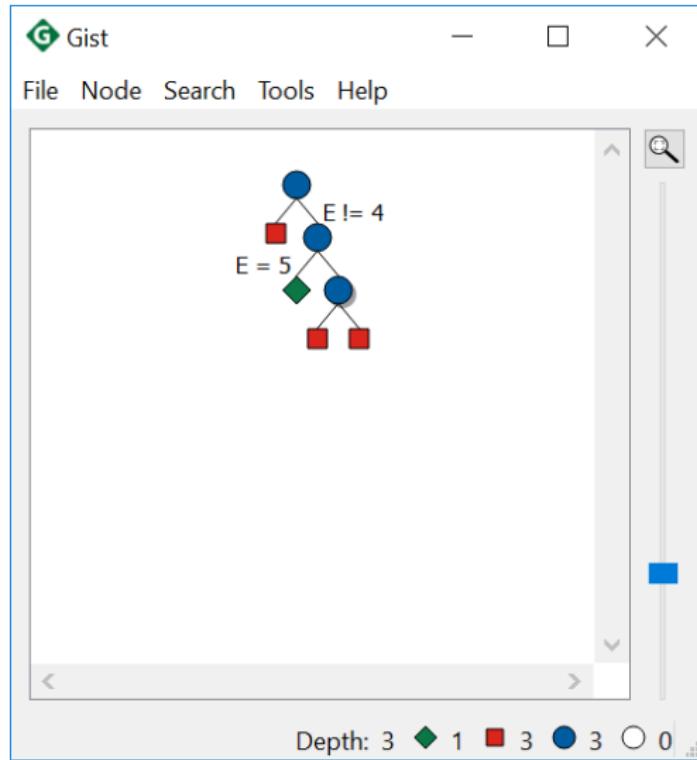
# Further Steps: Nothing more to do



# Complete Search Tree



# Search Tree with Gecode/GIST



# Some Differences

- Uses binary branching
  - var equal value, var not equal value
- Solutions in green, failure leafs in red, internal nodes in blue
- By default, shows all failed sub trees collapsed
- By default, uses different search strategy

# Solution

$$\begin{array}{r} 9 & 5 & 6 & 7 \\ + & 1 & 0 & 8 & 5 \\ \hline 1 & 0 & 6 & 5 & 2 \end{array}$$

# Outline

Problem

Program

Constraint Setup

Search

Points to Remember

## Points to Remember

- Constraint models are expressed by variables and constraints.
- Problems can have many different models, which can behave quite differently. Choosing the best model is an art.
- Constraints can take many different forms.
- Propagation deals with the interaction of variables and constraints.
- It removes some values that are inconsistent with a constraint from the domain of a variable.
- Constraints only communicate via shared variables.

## Points to Remember

- Propagation usually is not sufficient, search may be required to find a solution.
- Propagation is data driven, and can be quite complex even for small examples.
- The default search uses chronological depth-first backtracking, systematically exploring the complete search space.
- The search choices and propagation are interleaved, after every choice some more propagation may further reduce the problem.

## For Puzzle Purists Only

- We did not follow the puzzler ethos!
- We should solve the puzzle without making choices
- Even a case analysis should be avoided
- The puzzle has a single solution, we should be able to deduce the solution
- In the process shown, we are limited by the underlying assumptions
  - Treat each constraint on its own, they interact only by domains of variables
  - We only use the constraints that we stated in the model
- Can we do better than that?

▶ Skip

# Better Reasoning

- Three possible approaches (possibly many more)
  - Full domain reasoning for arithmetic, not just bound reasoning
  - Interaction of *sum* and *alldifferent* constraints
  - Deduced implied constraint

## Looking at more than just bounds

- We only considered the smallest and largest values that can be achieved in the sum constraint
- We can do more
  - Can any of the values between be expressed as the sum of the terms
  - Consider holes in the domains, and in the range of possible values for LHS and RHS
- Usually not done in actual solvers for arithmetic constraints
- Easy to do with Dynamic Programming

## Consider the interaction of multiple constraints

- Usually ignored, as only interaction is via domains of shared variables
- Here: Sum and *alldifferent* interact
  - When considering the bounds, we cannot assume that each variable takes its smallest/largest value independently
  - Find feasible assignment that minimizes/maximizes the total weight
  - To do this properly, we need some non-trivial reasoning
- Do we do this combined reasoning automatically, or only when prompted by the modeler?

# Deduced Implied Constraints

- Look at the partially solved puzzle

$$\begin{array}{r} 9\text{END} \\ +10\text{RE} \\ \hline \end{array}$$

- 10NEY
- In the hundreds position, we have  $E + 0 + C_{10} = N + 10 * C_{100}$ , with  $C_{10}$  the 0/1 carry from the tens position
  - NB: No carry  $C_{100}$  into the thousands,  $C_{100} = 0$
  - $N$  must be equal to  $E + 1$  with  $C_{10} = 1$
  - If  $C_{10} = 0$ , then  $N = E$ , not possible
  - We can substitute  $N = E + 1$  into our main equation, but keep  $N = E + 1$  as well

# Expert Mode Reasoning

Starting with

$$91 * E^{4..7} + 10 * R^{2..8} + D^{2..8} = 90 * N^{5..8} + Y^{2..8}$$

# Expert Mode Reasoning

we get

$$91 * E^{4..7} + 10 * R^{2..8} + D^{2..8} = 90 * E^{4..7} + 90 + Y^{2..8}$$

# Expert Mode Reasoning

Eliminating duplicate occurrences of E

$$\underbrace{E^{4..7} + 10 * R^{2..8} + D^{2..8}}_{26..95} = \underbrace{90 + Y^{2..8}}_{92..98}$$

shared range 92..95

# Expert Mode Reasoning

To reach 92, R must be equal to 8, therefore N, E, D, Y must be less than 8

As  $N = E + 1$ , E must be less than 7

$$E^{4..6} + 10 * 8 + D^{2..7} = 90 + Y^{2..7}$$

# Expert Mode Reasoning

Simplification yields

$$\underbrace{E^{4..6} + D^{2..7}}_{6..13} = \underbrace{10 + Y^{2..7}}_{12..17}$$

shared range 12..13

## Expert Mode Reasoning

To reach 12 on LHS, E must be greater than 4, D must be greater than 5

To reach 13 in RHS, Y must be smaller than 4

$$E^{5..6} + D^{6..7} = 10 + Y^{2..3}$$

## Expert Mode Reasoning

As both N and D must be in 6..7, no other variable can use those values

NB: This requires better reasoning than *forward checking* on the *alldifferent* constraint

If we remove 6 from the domain of E, E must be  $E = 5$ , and thus  $N = 6$ , due to  $N = E + 1$

*Alldifferent* forces  $D = 7$ , leaving

$$5 + 7 = 10 + Y^{2..4}$$

## Expert Mode Reasoning

This only leaves  $Y = 2$

$$5 + 7 = 10 + 2$$

That is a solution, the constraint is satisfied

## Expert Mode Summary

- Often there is more propagation that can be done
- Can be difficult/expensive to do
- Balancing
  - How much work it done at each step of search?
  - How many steps of search you need?
- For hard problems, doing all possible propagation may be exponential
- Not aware that any CP system does the full reasoning shown here

# This is how people solve the puzzle by hand



Send More Money | Puzzle only a genius can solve | MAX MATH GAMES



Max Math Games  
1.28K subscribers

Subscribe

14

1

Share

Download

Clip

...

2K views 3 years ago #mathsandphysicsfun

- When writing the first version of this puzzle for CHIP (in 1986), we wanted to mimic the way we solve the puzzle by hand

# Part II

## Global Constraints

## Example 2: Sudoku

- Global Constraints
  - Powerful modelling abstractions
  - Non-trivial propagation
  - Different consistency levels
- Example: Sudoku puzzle

# Outline

Problem

Initial Propagation (Forward Checking)

Improved Reasoning

Search

Other Global Constraints

# Problem Definition

## Sudoku

Fill in numbers from 1 to 9 so that each row, column and 3x3 block contain each number exactly once

4	1 2 3	8	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3
1 2 3	2 3 4	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3
4 5 6	4 5 6	4 5 6	1 7	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6
7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9
1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3
4 5 6	4 5 6	4 5 6	8	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6
7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9
1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3
4 5 6	4 5 6	6	4 5 6	4 5 6	8	2	5	4 5 6
7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9
1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3
4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6
7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9
1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3
4 5 6	4 5 6	3	7	6	4	5	6	4 5 6
7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	9	4 5 6	7 8 9
1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3
2	7	1 2	4 5	4 5	5	4 5 6	4 5 6	4 5 6
7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9
1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3
4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6
7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9
1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3
4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	6	4 5 6	4 5 6
7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9

# Problem Definition

## Sudoku

Fill in numbers from 1 to 9 so that each row, column and 3x3 block contain each number exactly once

4	5	2	3	8	1	2	3	4
1	2	3	2	3	1	2	3	4
4	5	6	7	8	9	1	2	3
7	8	9	1	2	3	4	5	6
1	2	3	2	3	1	2	3	4
4	5	6	7	8	9	1	2	3
7	8	9	1	2	3	4	5	6
1	2	3	2	3	1	2	3	4
4	5	6	7	8	9	1	2	3

6	7	2	3	8	1	4	5	9
1	2	3	2	3	1	2	3	4
4	5	6	7	8	9	1	2	3
7	8	9	1	2	3	4	5	6
1	2	3	2	3	1	2	3	4
4	5	6	7	8	9	1	2	3
7	8	9	1	2	3	4	5	6
1	2	3	2	3	1	2	3	4
4	5	6	7	8	9	1	2	3

4	2	8	5	6	3	1	7	9
3	5	9	1	7	2	4	6	8
7	6	1	4	8	9	5	3	2
1	4	6	3	9	8	2	5	7
5	9	2	7	4	1	3	8	6
8	3	7	6	2	5	9	4	1
2	7	4	9	5	6	8	1	3
6	8	3	2	1	4	7	9	5
9	1	5	8	3	7	6	2	4

# Model

- A variable for each cell, ranging from 1 to 9
- A  $9 \times 9$  matrix of variables describing the problem
- Preassigned integers for the given hints
- all different constraints for each row, column and  $3 \times 3$  block

# Sudoku Models

- ECLiPSe [▶ Show](#)
- MiniZinc [▶ Show](#)
- NumberJack [▶ Show](#)
- CPMPy [▶ Show](#)
- Choco-solver [▶ Show](#)

# ECLIPSe Sudoku Model (from <https://eclipseclp.org/>)

```
:- lib(ic).
:- import alldifferent/1 from ic_global.

top :-  
    problem(Board),  
    print_board(Board),  
    sudoku(Board),  
    labeling(Board),  
    print_board(Board).

sudoku(Board) :-  
    dim(Board, [N,N]),  
    Board :: 1..N,  
    ( for(I,1,N), param(Board) do  
        alldifferent(Board[I,*]),  
        alldifferent(Board[*,I])  
    ),  
    NN is integer(sqrt(N)),  
    ( multifor([I,J],1,N,NN), param(Board,NN) do  
        alldifferent(concat(Board[I..I+NN-1, J..J+NN-1]))  
    ).  
  
print_board(Board) :-  
    dim(Board, [N,N]),  
    ( for(I,1,N), param(Board,N) do  
        ( for(J,1,N), param(Board,I) do  
            X is Board[I,J],  
            ( var(X) -> write(" _") ; printf(" %2d", [X]) )  
        ), nl  
    ), nl.
```

# ECLiPSe Data Definition

```
problem([](
```

- [ ] (4, \_, 8, \_, \_, \_, \_, \_, \_, \_),
- [ ] (\_, \_, \_, 1, 7, \_, \_, \_, \_, \_),
- [ ] (\_, \_, \_, \_, 8, \_, \_, \_, 3, 2),
- [ ] (\_, \_, 6, \_, \_, 8, 2, 5, \_),
- [ ] (\_, 9, \_, \_, \_, \_, \_, 8, \_),
- [ ] (\_, 3, 7, 6, \_, \_, 9, \_, \_),
- [ ] (2, 7, \_, \_, 5, \_, \_, \_, \_),
- [ ] (\_, \_, \_, \_, 1, 4, \_, \_, \_),
- [ ] (\_, \_, \_, \_, \_, \_, 6, \_, 4))).

► Continue

# MiniZinc Sudoku Model

```
int: s;
int: n=s*s;
array[1..n,1..n] of var 1..n: puzzle;

include "sudoku.dzn";

predicate alldifferent(array[int] of var int: x) =
  forall(i,j in index_set(x) where i < j)
    (x[i] != x[j]);

constraint forall(i in 1..n)
  (alldifferent([puzzle[i,j] | j in 1..n]));
constraint forall(j in 1..n)
  (alldifferent([ puzzle[i,j] | i in 1..n]));
constraint forall(i,j in 1..s)
  (alldifferent([puzzle[s*(i-1)+p, s*(j-1)+q] |
    p,q in 1..s]));
solve satisfy;
```

# MiniZinc Output

```
output [ "sudoku:\\n" ] ++
[ show(puzzle[i,j]) ++
if j = n then
  if i mod s = 0 /\ i < n then "\\n\\n"
  else "\\n"
  endif
else
  if j mod s = 0 then " "
  else " "
  endif
endif
| i,j in 1..n];
```

# MiniZinc Data File (sudoku.dzn)

```
s=3;  
puzzle=[|  
    4, _, 8, _, _, _, _, _, _, _ |  
    _, _, _, 1, 7, _, _, _, _, _ |  
    _, _, _, _, 8, _, _, _, 3, 2 |  
    _, _, 6, _, _, 8, 2, 5, _ |  
    _, 9, _, _, _, _, _, _, 8, _ |  
    _, 3, 7, 6, _, _, 9, _, _ |  
    2, 7, _, _, 5, _, _, _, _, _ |  
    _, _, _, _, 1, 4, _, _, _, _ |  
    _, _, _, _, _, 6, _, 4, _ |  
];
```

▶ Continue

# NumberJack Sudoku Model

```
from Numberjack import *

def get_model(N, clues):
    grid = Matrix(N*N, N*N, 1, N*N)

    sudoku = Model([AllDiff(row) for row in grid.row],
                  [AllDiff(col) for col in grid.col],
                  [AllDiff(grid[x:x + N, y:y + N]) for x in range(0, N*N, N)
                   for y in range(0, N * N, N)],
                  [(x == int(v)) for x, v in
                   zip(grid.flat, "".join(open(clues)).split()) if v != '*'])
                 )
    return grid, sudoku

def solve(param):
    N = param['N']
    clues = param['file']

    grid, sudoku = get_model(N, clues)

    solver = sudoku.load(param['solver'])
    solver.setVerbosity(param['verbose'])
    solver.setTimeLimit(param['tcutoff'])

    solver.solve()
```

# NumberJack Data File

```
4 * 8 * * * * *
* * * 1 7 * * * *
* * * * 8 * * 3 2
* * 6 * * 8 2 5 *
* 9 * * * * 8 *
* 3 7 6 * * 9 * *
2 7 * * 5 * * * *
* * * * 1 4 * * *
* * * * * 6 * 4
```

▶ Continue

# CPMpy Sudoku Model (from <https://github.com/CPMpy/>)

```
import numpy as np
from cpmpy import *

# Variables
puzzle = intvar(1,9, shape=given.shape, name="puzzle")

model = Model(
    # Constraints on values (cells that are not empty)
    puzzle[given!=e] == given[given!=e], # numpy's indexing, vectorized equality
    # Constraints on rows and columns
    [AllDifferent(row) for row in puzzle],
    [AllDifferent(col) for col in puzzle.T], # numpy's Transpose
)

# Constraints on blocks
for i in range(0,9, 3):
    for j in range(0,9, 3):
        model += AllDifferent(puzzle[i:i+3, j:j+3]) # python's indexing

model.solve()
```

# CPMy Data Definition

```
e = 0 # value for empty cells
given = np.array([
    [4, e, 8, e, e, e, e, e, e],
    [e, e, e, 1, 7, e, e, e, e],
    [e, e, e, 8, e, e, 3, 2],
    [e, e, 6, e, e, 8, 2, 5, e],
    [e, 9, e, e, e, e, e, 8, e],
    [e, 3, 7, 6, e, e, 9, e, e],
    [2, 7, e, e, 5, e, e, e, e],
    [e, e, e, e, 1, 4, e, e, e],
    [e, e, e, e, e, e, 6, e, 4]
])

```

▶ Continue

# Choco-solver Sudoku Model

```
Model model = new Model("Sudoku");
int blockSize = 3;
int m = blockSize*blockSize;

IntVar[][] vars = new IntVar[m][m];
for(int i=0;i<m;i++) {
    for(int j=0;j<m;j++) {
        vars[i][j] = model.intVar("X"+i+"_"+j, 1, m);
        if (data[i][j]>0) {
            model.arithm(vars[i][j], "=", data[i][j]).post();
        }
    }
}
for(int i=0;i<m;i++) {
    model.allDifferent(row(i,m,vars)).post();
    model.allDifferent(column(i,m,vars)).post();
}
for(int i=0;i<m;i+=blockSize) {
    for(int j=0;j<m;j+=blockSize) {
        model.allDifferent(block(i,j,blockSize,vars)).post();
    }
}
Solver solver = model.getSolver();
solver.solve();
```

# Choco-solver Data

```
int[][] data = new int[m][m] {  
    {4, 0, 8, 0, 0, 0, 0, 0, 0},  
    {0, 0, 0, 1, 7, 0, 0, 0, 0},  
    {0, 0, 0, 0, 8, 0, 0, 3, 2},  
    {0, 0, 6, 0, 0, 8, 2, 5, 0},  
    {0, 9, 0, 0, 0, 0, 0, 8, 0},  
    {0, 3, 7, 6, 0, 0, 9, 0, 0},  
    {2, 7, 0, 0, 5, 0, 0, 0, 0},  
    {0, 0, 0, 0, 1, 4, 0, 0, 0},  
    {0, 0, 0, 0, 0, 0, 6, 0, 4}  
};
```

# Choco-solver Utilities

```
IntVar[] row(int row, int size, IntVar[][] array){  
    return array[row];  
}  
  
IntVar[] column(int col,int size,IntVar[][] array){  
    IntVar[] column = new IntVar[size];  
    for(int i=0; i<size; i++){  
        column[i] = array[i][col];  
    }  
    return column;  
}  
  
IntVar[] block(int x,int y,int blockSize,IntVar[][] array){  
    IntVar[] block = new IntVar[blockSize*blockSize];  
    int k=0;  
    for(int i=0;i<blockSize;i++){  
        for(int j=0;j<blockSize;j++){  
            block[k++] = array[x+i][y+j];  
        }  
    }  
    return block;  
}
```

» Continue

# Domain Visualizer

- Problem shown as matrix
  - Each cell corresponds to a variable
  - Instantiated: Shows integer value (large)
  - Uninstantiated: Shows values in domain

# Outline

Problem

Initial Propagation (Forward Checking)

Improved Reasoning

Search

Other Global Constraints

# Initial State (Forward Checking)

4	1 2 3	8	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3
4	5 6	8	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6
7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9
1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3
4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6
7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9
1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3
4 5 6	4 5 6	8	4 5 6	4 5 6	4 5 6	4 5 6	3	2	
7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	
1 2 3	1 2 3	6	1 2 3	1 2 3	8	2	5		
4 5 6	4 5 6	7 8 9	4 5 6	4 5 6	7 8 9	7 8 9	7 8 9		
7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9		
1 2 3	1 2 3	9	1 2 3	1 2 3	1 2 3	1 2 3	8		
4 5 6	4 5 6	7 8 9	4 5 6	4 5 6	7 8 9	7 8 9	7 8 9		
7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9		
1 2 3	1 2 3	3	7	6	1 2 3	1 2 3	9	1 2 3	1 2 3
4 5 6	4 5 6	7 8 9	4 5 6	4 5 6	7 8 9	7 8 9	7 8 9	4 5 6	4 5 6
7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	
2	7	1 2 3	1 2 3	5	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3
4 5 6	4 5 6	7 8 9	4 5 6	4 5 6	7 8 9	7 8 9	7 8 9	4 5 6	4 5 6
7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9
1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3
4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6
7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9
1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	6	1 2 3	
4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	
7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	

## Propagation Steps (Forward Checking)

► Skip Animation

## Propagation Steps (Forward Checking)

Back to Start

► Skip Animation

## Propagation Steps (Forward Checking)

4	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3
	5 6	8	5 6	5 6	5 6	5 6	5 6	5 6
	7 9	7 9 7	9 7	9 7	9 7	9 7	9 7	9 7
1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3
4 5 6	4 5 6	4 5 6	4 5 6	1	7	4 5 6	4 5 6	4 5 6
7 8 9	7 8 9	7 8 9	7 8 9			7 8 9	7 8 9	7 8 9
1 2 3	1 2 3	1 2 3	1 2 3	8	1 2 3	1 2 3	1 2 3	1 2 3
4 5 6	4 5 6	4 5 6	4 5 6			4 5 6	4 5 6	4 5 6
7 8 9	7 8 9	7 8 9	7 8 9			7 8 9	7 8 9	7 8 9
1 2 3	1 2 3	1 2 3	1 2 3	6	8 2	1 2 3	1 2 3	1 2 3
4 5 6	4 5 6	4 5 6	4 5 6		5	4 5 6	4 5 6	4 5 6
7 8 9	7 8 9	7 8 9	7 8 9			7 8 9	7 8 9	7 8 9
1 2 3	1 2 3	1 2 3	1 2 3	9	8	1 2 3	1 2 3	1 2 3
4 5 6	4 5 6	4 5 6	4 5 6			4 5 6	4 5 6	4 5 6
7 8 9	7 8 9	7 8 9	7 8 9			7 8 9	7 8 9	7 8 9
1 2 3	3	7	6	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3
4 5 6	3	7	6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6
7 8 9				7 8 9	7 8 9	7 8 9	7 8 9	7 8 9
2	7	1 2 3	1 2 3	5	1 2 3	1 2 3	1 2 3	1 2 3
		4 5 6	4 5 6		4 5 6	4 5 6	4 5 6	4 5 6
		7 8 9	7 8 9		7 8 9	7 8 9	7 8 9	7 8 9
1 2 3	1 2 3	1 2 3	1 2 3	1	4	1 2 3	1 2 3	1 2 3
4 5 6	4 5 6	4 5 6	4 5 6			4 5 6	4 5 6	4 5 6
7 8 9	7 8 9	7 8 9	7 8 9			7 8 9	7 8 9	7 8 9
1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3
4 5 6	4 5 6	4 5 6	4 5 6	6	4 5 6	4 5 6	4 5 6	4 5 6
7 8 9	7 8 9	7 8 9	7 8 9			7 8 9	7 8 9	7 8 9
1 2 3	1 2 3	1 2 3	1 2 3	4	6	1 2 3	1 2 3	1 2 3
4 5 6	4 5 6	4 5 6	4 5 6			4 5 6	4 5 6	4 5 6
7 8 9	7 8 9	7 8 9	7 8 9			7 8 9	7 8 9	7 8 9

Back to Start

► Skip Animation

# Propagation Steps (Forward Checking)

4	1 2 3	5 6	8	1 2 3	5 6	5 6	5 6	5 6	5 6	5 6	5 6
7	9	7	9	7	9	7	9	7	9	7	9
1	3	1 2 3	1 2 3	1	7	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3
5	6	4 5 6	4 5 6	5	6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6
7	8 9	7 8 9	7 8 9	8	1	7	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6
1	3	1 2 3	1 2 3	1 2 3	8	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3
5	6	4 5 6	4 5 6	4 5 6	8	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6
7	8 9	7 8 9	7 8 9	7	8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9
1	3	1 2 3	1 2 3	6	1 2 3	1 2 3	8	2	5	1 2 3	1 2 3
5	6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	8	2	5	4 5 6	4 5 6
7	8 9	7 8 9	7 8 9	7	8 9	7 8 9	7	8 9	7	8 9	7 8 9
1	3	9	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	8	1 2 3	1 2 3
5	6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	8	4 5 6	4 5 6
7	8 9	7 8 9	7 8 9	7	8 9	7 8 9	7	8 9	7	8 9	7 8 9
1	3	3	7	6	1 2 3	1 2 3	9	1 2 3	1 2 3	1 2 3	1 2 3
5	6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	9	4 5 6	4 5 6	4 5 6	4 5 6
7	8 9	7 8 9	7 8 9	7	8 9	7 8 9	7	8 9	7	8 9	7 8 9
2	7	1 2 3	1 2 3	5	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3
4	5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6
7	8 9	7 8 9	7 8 9	7	8 9	7 8 9	7	8 9	7	8 9	7 8 9
1	3	1 2 3	1 2 3	1 2 3	1	4	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3
5	6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6
7	8 9	7 8 9	7 8 9	7	8 9	7 8 9	7	8 9	7	8 9	7 8 9
1	3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	6	1 2 3	1 2 3	4 5 6	4 5 6
5	6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	6	4 5 6	4 5 6	4 5 6	4 5 6
7	8 9	7 8 9	7 8 9	7	8 9	7 8 9	7	8 9	7	8 9	7 8 9

◀ Back to Start

► Skip Animation

# Propagation Steps (Forward Checking)

4	1 2 3 5 6	8	1 2 3 5 6					
7	9		7	9	7	9	7	9
3	2 3	2 3	2 3	2 3	2 3	2 3	2 3	2 3
5 6	4 5 6	4 5 6	1	7	4 5 6	4 5 6	4 5 6	4 5 6
8 9	8 9	8 9			8 9	8 9	8 9	8 9
1	3 1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	3	2
5 6	4 5 6	4 5 6	8	4 5 6	4 5 6	4 5 6		
7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9		
1	3 1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	
5 6	4 5 6	4 5 6	6	4 5 6	4 5 6	8	2	5
7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9		
1	3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	
5 6	4 5 6	4 5 6	9	4 5 6	4 5 6	4 5 6	4 5 6	
7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9		
1	3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	
5 6	4 5 6	4 5 6	3	7	4 5 6	4 5 6	4 5 6	
7 8 9	7 8 9	7 8 9	6	7 8 9	7 8 9	9	7 8 9	
2	7	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	
4 5 6	4 5 6	4 5 6	5	4 5 6	4 5 6	4 5 6	4 5 6	
7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9		
1	3 1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	
5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	
7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9		
1	3 1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	
5 6	4 5 6	4 5 6	1	4	4 5 6	4 5 6	4 5 6	
7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9		
1	3 1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	
5 6	4 5 6	4 5 6	6	4 5 6	4 5 6	4 5 6	4 5 6	
7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9		

◀ Back to Start

▶ Skip Animation

# Propagation Steps (Forward Checking)

4	1 2 5 6	8	1 2 3 5 6 9 7	1 2 3 5 6 9 7	1 2 3 5 6 9 7	1 2 3 5 6 9 7	1 2 3 5 6 9 7	1 2 3 5 6 9 7
3	2 5 6 8 9	2 3 4 5 6 8 9	1 7	2 3 4 5 6 8 9	2 3 4 5 6 8 9			
1 5 6 8 9	3 1 2 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	8	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	3	2	
1 5 6 7 8 9	3 1 2 4 5 6 8	1 2 3 4 5 6 7 8 9	6	1 2 3 4 5 6 7 8 9	8 2 5			
1 5 6 7 8 9	3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	8	4 5 6 7 8 9	
1 5 6 7 8 9	3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	7	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	9	4 5 6 7 8 9	
2 5 6 7 8 9	7	1 2 3 4 5 6 7 8 9	6	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	9	4 5 6 7 8 9	
1 5 6 7 8 9	2 7 8	1 2 3 4 5 6 7 8 9	5	1 2 3 4 5 6 7 8 9				
1 5 6 7 8 9	3 4 5 6 8	1 2 3 4 5 6 7 8 9	4	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	6	4 5 6 7 8 9	4

◀ Back to Start

► Skip Animation

# Propagation Steps (Forward Checking)

4	1 2 5 6	8	1 2 3 5 6 7	1 2 3 5 6 9				
3	2 5 6 8 9	1	2 3 4 5 6 8 9	7	2 3 4 5 6 8 9			
1	1 5 6 7 9	8	1 4 5 6 7 9	1 4 5 6 9	1 4 5 6 7 9	1 4 5 6 9	3	2
1	3 1 2 5 6 4 5 6 7 8 9 8	6	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	8	2	5	1 2 3 4 5 6 7 8 9
1	3 5 6 7 8 9	9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	8	4 5 6 7 8 9	1 2 3 4 5 6 7 8 9
1	3 5 6 7 8 9	3 7 6	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9
2	7	5	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	5	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9
1	3 1 2 5 6 4 5 6 7 8 9 8	1	2 3 1 2 3 4 5 6 4 5 6 7 8 9 7 8 9	4 5 6 4 5 6 7 8 9 7 8 9	1 2 3 1 2 3 4 5 6 4 5 6 7 8 9 7 8 9	1 2 3 1 2 3 4 5 6 4 5 6 7 8 9 7 8 9	1 2 3 1 2 3 4 5 6 4 5 6 7 8 9 7 8 9	1 2 3 1 2 3 4 5 6 4 5 6 7 8 9 7 8 9
1	3 1 2 5 6 4 5 6 7 8 9 8	6	1 2 3 1 2 3 4 5 6 4 5 6 7 8 9 7 8 9	1 2 3 1 2 3 4 5 6 4 5 6 7 8 9 7 8 9	6	1 2 3 1 2 3 4 5 6 4 5 6 7 8 9 7 8 9	4	1 2 3 1 2 3 4 5 6 4 5 6 7 8 9 7 8 9

◀ Back to Start

► Skip Animation

# Propagation Steps (Forward Checking)

4	1 2 5 6	8	1 2 3 5 6 7	1 2 3 5 6 9 7	1 2 3 5 6 9			
3	2 5 6 8 9	2 3 4 5 8	1 7	2 3 4 5 6 8 9				
1	1 5 6 7 9	1 4 5 6 9 7 9	1 8	1 4 5 6 7 9 7	1 4 5 6 7 9 7	3 4 5 6 8 9	2 4 5 6 7 8 9	2 4 5 6 7 8 9
1	3 1 2 5 6 7 8 9	1 2 3 4 5 6 8	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9
1	3 5 6 7 8 9	9 4 5 9 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	8 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9
1	3 5 6 7 8 9	3 7 4 5 9 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	9 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9
2	7	2 7 4 5 9 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	5 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9
1	3 1 2 5 6 7 8 9	1 2 3 4 5 6 8	1 2 3 4 5 6 9 7 8 9	1 2 3 4 5 6 7 8 9				
1	3 1 2 5 6 7 8 9	1 2 3 4 5 6 8	1 2 3 4 5 6 9 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	6 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	4 4 5 6 7 8 9

◀ Back to Start

► Skip Animation

# Propagation Steps (Forward Checking)

4	1 2 5 6	8	1 2 3 5 6 7 9	1 2 3 5 6 9 7	1 2 3 5 6 9 9			
3	2 5 6 8 9	2 3 4 5 8	1 7	2 3 4 5 6 8 9				
1	1 5 6 7 9	1 4 5 6 9 7 9	1 8	1 4 5 6 7 9 7 9	1 4 5 6 7 9 7 9	1 4 5 6 7 9 7 9	3 2	1 3 4 7 9
1	3 1 4	6 4 7 9	1 3 1 3 4 4 7 9 7 9	8 2	2 5 4 4 7 9	1 3 4 7 9	1 3 4 7 9	1 3 4 7 9
1	3 5 6 7 8 9	9 4 5 9 7 8 9	1 2 3 4 5 6 9 7 8 9	1 2 3 4 5 6 9 7 8 9	1 2 3 4 5 6 9 7 8 9	1 2 3 4 5 6 9 7 8 9	8 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9
1	3 5 6 7 8 9	3 7 6 4 5 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9
2	7	5 9 7 8 9	1 2 3 4 5 6 9 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9
1	3 1 2 5 6 4 5 6 7 8 9 8	1 2 3 4 5 6 9 7 8 9	1 2 3 4 5 6 9 7 8 9	1 2 3 4 5 6 9 7 8 9	1 2 3 4 5 6 9 7 8 9	1 2 3 4 5 6 9 7 8 9	1 2 3 4 5 6 9 7 8 9	1 2 3 4 5 6 9 7 8 9
1	3 1 2 5 6 4 5 6 7 8 9 8	6 4 9 7 8 9	1 2 3 4 5 6 9 7 8 9	1 2 3 4 5 6 9 7 8 9	1 2 3 4 5 6 9 7 8 9	1 2 3 4 5 6 9 7 8 9	4 4	1 2 3 4 5 6 7 8 9

◀ Back to Start

► Skip Animation

# Propagation Steps (Forward Checking)

4	1 2 5 6	8	2 3 5	1 2 3 5 6	1 2 3 5 6	1 2 3 5 6	1 2 3 5 6	1 2 3 5 6
3	2 5 6 4 5 8 9	2 3 4 5 8	1 7	2 3 4 5 6 8 9	2 3 4 5 6 8 9	2 3 4 5 6 8 9	2 3 4 5 6 8 9	2 3 4 5 6 8 9
1	1 5 6 4 5 6 7 9	1 4 5 9 7 9	8	1 4 5 6 7 9 7 9	1 4 5 6 7 9 7 9	3	2	
1	3 1 4	6	3 1 4 4 7 9 7 9	8	2 5	1 3 4		
1	3 5 6 7 8 9	9	1 2 3 4 5 9 7 8 9	2 3 4 5 7 8 9 7 8 9	1 2 3 4 5 6 7 8 9 7 8 9	1 2 3 4 5 6 7 8 9 7 8 9		
1	3 5 6 7 8 9	3 7 6	1 2 3 4 5 6 7 8 9 7 8 9	1 2 3 4 5 6 7 8 9 7 8 9	9	1 2 3 4 5 6 7 8 9 7 8 9		
2	7	1 2 3 4 5 9 7 8 9	5	1 2 3 4 5 6 7 8 9 7 8 9	1 2 3 4 5 6 7 8 9 7 8 9	1 2 3 4 5 6 7 8 9 7 8 9		
1	3 1 2 5 6 4 5 6 7 8 9	1 2 3 4 5 9 7 8 9	1 4	1 2 3 4 5 6 7 8 9 7 8 9	1 2 3 4 5 6 7 8 9 7 8 9	1 2 3 4 5 6 7 8 9 7 8 9		
1	3 1 2 5 6 4 5 6 7 8 9	1 2 3 4 5 9 7 8 9	6	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	4		

◀ Back to Start

► Skip Animation

# Propagation Steps (Forward Checking)

4	1 2 5 6	8	2 3 5 5 6 5 6 5 6 5 6 5 6	1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3	1 2 3 2 3 2 3 4 5 6 4 5 6 4 5 6 4 5 6	1 2 3 2 3 2 3 4 5 6 4 5 6 4 5 6 4 5 6	1 2 3 2 3 2 3 4 5 6 4 5 6 4 5 6 4 5 6
3	2 5 6 4 5 6 8 9	2 3 1 9	7 9 7 9 7 9	1 2 3 5 6 4 5 6 8 9	2 3 4 5 6 8 9 8 9 8 9 8 9	2 3 4 5 6 4 5 6 8 9 8 9 8 9	2 3 4 5 6 4 5 6 8 9 8 9 8 9
1	1 5 6 4 5 6 7 9	1 8	1 4 5 9 7 9	1 4 5 6 7 9 7	1 4 5 6 7 9 7 9 7 9	1 4 5 6 7 9 7 9	2 3 1 3 4 7 9
1	3 1 4 7 9	6	3 1 4 7 9 7	3 1 3 4 5 4 5 7 9 7 9	8 2 8 2 9 9	2 5 5 7 9	1 3 4 7 9
1	3 5 6 7	9	1 2 3 4 5 7	2 3 1 2 3 4 5 6 4 5 6 7 7 7	1 2 3 1 2 3 4 5 6 4 5 6 7 7 7	1 2 3 4 5 6 8 7	1 2 3 4 5 6 8 7
1	3 5 6 7 8 9	3 7	6	1 2 3 1 2 3 4 5 6 4 5 6 7 8 9 7 8 9	9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9
2	7	1 2 3 4 5 9 7 8 9	2 3 4 5 7 8 9	1 2 3 4 5 6 7 8 9	5 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9
1	3 1 2 5 6 4 5 6 7 8 9	1 2 3 4 5 8	2 3 4 5 9 7 8 9	1 2 3 1 2 3 4 5 6 4 5 6 7 8 9 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9
1	3 1 2 5 6 4 5 6 7 8 9	1 2 3 4 5 8	2 3 4 5 9 7 8 9	1 2 3 1 2 3 4 5 6 4 5 6 7 8 9 7 8 9	6 7 8 9	1 2 3 4 5 6 7 8 9	4 7 8 9

◀ Back to Start

► Skip Animation

# Propagation Steps (Forward Checking)

4	1 2 5 6	8	2 3 5 7 9	2 3 6 9 7	1 2 3 5 6 9 7	1 2 3 5 6 9 7	1 2 3 5 6 9
3	2 5 6 8 9	2 3 4 5 8	1 7 9	2 3 4 5 6 8 9	2 3 4 5 6 8 9	2 3 4 5 6 8 9	2 3 4 5 6 8 9
1	1 5 6 7 9	1 4 5 9 7 9	1 8 7 9	1 4 5 6 7 9 7 9	1 4 5 6 7 9 7 9	3 1 3 4 8 2 5 1 3 4 5 6 7 7	2 1 3 4 8 2 5 1 3 4 5 6 7 7
1	3 1 4 7 9	6 4 7 9	3 4 9	3 4 6 7 9 7	8 4 5 6 9 7 8 9	2 4 5 6 7 8 9	5 4 5 6 7 8 9
1	3 5 6 7 8 9	9 4 5 7	1 2 3 4 5 9	2 3 4 6 7	1 2 3 4 5 6 7	1 2 3 4 5 6 7	1 2 3 4 5 6 7
1	3 5 6 7 8 9	3 4 5 9	7 8 9	2 3 4 6 4 5 6 9 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9
2	7	6	1 2 3 4 5 9 7 8 9	5	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9
1	3 1 2 5 6 4 5 6 7 8 9	1 2 3 4 5 8	2 3 4 5 9 7 8 9	1 2 3 4 6 4 5 6 9 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9
1	3 1 2 5 6 4 5 6 7 8 9	1 2 3 4 5 8	2 3 4 5 9 7 8 9	1 2 3 4 6 4 5 6 9 7 8 9	6 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	4 7 8 9

◀ Back to Start

► Skip Animation

# Propagation Steps (Forward Checking)

4	1 2 5 6	8	2 3 5 7 9	2 3 6 9 7	1 2 3 5 6 9 7	1 2 3 5 6 9 7	1 2 3 5 6 9 7	1 2 3 5 6 9 7
3	2 5 6 8 9	2 3 4 5 8	1 7 9	2 3 4 5 6 8 9				
1	1 5 6 7 9	1 4 5 9 7 9	1 8 7 9	1 4 5 6 7 9 7	1 4 5 6 7 9 7	3	2	
1	3 1 4 7 9	1 6 4 5 7 9	3 4 9	3 8 9	2 8 2	5	1 3 4 7 9	
1	3 5 6 7	1 2 3 4 5 7	2 3 4 5 7	2 3 4 6 4 5 6 7 7	1 2 3 4 5 6 7	8	1 2 3 4 5 6 7	
1	5 8	3 7 4 5 9 7 8 9	7 6 4 5 8	2 4 5 8	1 2 9 4 5 8	1 2 4 5 8	1 2 4 5 8	
2	7	1 2 3 4 5 9 7 8 9	2 3 4 5 8 9	5	1 2 3 4 5 6 7 8 9			
1	3 1 2 5 6 4 5 6 7 8 9	1 2 3 4 5 6 9 7 8 9	2 3 4 5 9 7 8 9	1 4 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	
1	3 1 2 5 6 4 5 6 7 8 9	1 2 3 4 5 6 9 7 8 9	2 3 4 5 9 7 8 9	6	1 2 3 4 5 6 7 8 9	4		

◀ Back to Start

► Skip Animation

# Propagation Steps (Forward Checking)

4	1 2 5 6	8	2 3 5 7 9	2 3 6 9 7	1 2 3 5 6 9 7			
3	2 5 6 8 9	2 3 4 5 8	1 7 9	2 3 5 6 9	2 3 4 5 6 8 9	2 3 4 5 6 8 9	2 3 4 5 6 8 9	2 3 4 5 6 8 9
1	1 5 6 7 9	1 4 5 9 7 9	1 8 7 9	1 5 6 7 9	1 4 5 6 9 7 9	3	2	
1	3 1 4 7 9	6	3 4 4 7 9	3 4 6 9	8 5 6 7 7	2 4 5 6 8	5 4 7 9	1 3 4 7 9
1	3 5 6 7	9	1 2 3 4 5 7	2 3 4 6 7	1 2 3 5 6 7	1 2 3 4 5 6 8	1 2 3 4 5 6 7	
1	5 8	3 7 6	4	2 5	1 2 9	1 2 4 5 8	1 2 4 5 8	
2	7	1 2 3 4 5 9 7 8 9	2 3 4 5 8 9	5	1 2 3 5 6 7 9	1 2 3 4 5 6 8 9	1 2 3 4 5 6 7 8 9	
1	3 1 2 5 6 4 5 6 7 8 9	1 2 3 4 5 6 8	2 3 4 5 9 7 8 9	1 4	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	
1	3 1 2 5 6 4 5 6 7 8 9	1 2 3 4 5 6 8	2 3 4 5 9 7 8 9	2 3 4 6 9 7 9	1 2 3 5 6 9 7 9	6 4 5 6 7 8 9	4 7 8 9	

◀ Back to Start

► Skip Animation

# Propagation Steps (Forward Checking)

4	1 2 5 6	8	2 3 5 7 9	2 3 6 9 7	1 2 3 5 6 9 7	1 2 3 5 6 9 7	1 2 3 5 6 9 7 9
3	2 5 6 8 9	2 3 4 5 8	1 7 9	2 3 5 6 4 5 6 9 8 9	2 3 4 5 6 8 9	2 3 4 5 6 8 9	2 3 4 5 6 8 9
1	1 5 6 7 9	1 4 5 9 7 9	1 8 7 9	1 5 6 4 5 6 7 9 7 9	1 5 6 4 5 6 9 7 9	3 2	2
1	3 1 4 7 9	1 6 4 5 7 9	3 4 9	3 8 2 9	2 8 2 9	5 4 7 9	1 3 4 7 9
1	3 5 6 7	1 2 3 4 5 7	2 3 4 5 7	2 3 4 6 7	1 2 3 5 6 4 5 6 7 7	8 4 5 6 7	1 2 3 4 5 6 7
1	5 8	3 7 4 9	7 6 4 9 8 9	2 4 5	1 2 5 9	1 2 4 5 8	1 2 4 5 8
2	7	1 3 4 9	3 4 8 9	5	1 3 1 6 4 9 8 9	3 1 6 4 8 9	1 3 6 4 8 9
1	3 1 2 5 6 4 5 6 7 8 9	1 2 3 4 5 9 7 8 9	2 3 4 5 9 7 8 9	1 2 3 4 6 9 7 9	1 2 3 4 5 6 4 5 6 7 8 9 7 8 9	1 2 3 4 5 6 4 5 6 7 8 9 7 8 9	1 2 3 4 5 6 4 5 6 7 8 9 7 8 9
1	3 1 2 5 6 4 5 6 7 8 9	1 2 3 4 5 9 7 8 9	2 3 4 5 9 7 9	1 2 3 5 6 9 7 9	6 4 5 6 7 8 9	4 7 8 9	4

◀ Back to Start

► Skip Animation

# Propagation Steps (Forward Checking)

4	1 2 5 6	8	2 3 5 9	2 3 6 9	1 2 3 5 6 9	1 5	3 7	1 2 3 5 6 9	1 2 3 5 6 9
3 5 6 8 9	2 4 5 8	2 3 4 5 9	1 7	2 3 5 6 9	3 4 5 8	2 3 4 5 6 8 9	2 3 4 5 6 8 9	2 3 4 5 6 8 9	2 3 4 5 6 8 9
1 5 6 7 9	1 4 5 9	1 4 5 9	8	1 5 6 7	1 4 5 9	3 4 5 7	2 3 4 7	2 3 4 7	2 3 4 7
1 5 6 7 9	3 1 4	6	3 4 5 9	3 4 6 9	8 5 6 7	2 3 4 7	5 6 7	2 3 4 7	1 3 4 7
1 5 6 7	3 4 5 7	9	1 2 3 4 5 7	2 3 4 6 7	1 2 3 5 6 7	1 3 8	1 2 3 4 5 6 7	1 2 3 4 5 6 7	1 2 3 4 5 6 7
1 5 6 8	3 4	7	6	2 4	1 2 5	9	1 2 4 5 8	1 2 4 5 8	1 2 4 5 8
2 7	1 3 4	7	1 3 4	3 9	1 3 6 4	9	1 2 4 6 8	1 3 4 6 8	1 3 4 6 8
1 5 6 7 8 9	1 2 4 5 6 8	1 2 3 4 5 6 9	1 2 3 4 5 6 9	2 3 4 6 9	1 2 3 5 6 9	1 4 5 7 8	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9
1 5 6 7 8 9	3 1 2 5 6 4 5 6 8	1 2 3 4 5 6 9	1 2 3 4 5 6 9	2 3 1 2 3 4 6 5 6 9 7 9	1 2 3 4 5 6 7 8 9	6 4	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	4

◀ Back to Start

► Skip Animation

# Propagation Steps (Forward Checking)

4	1 2 5 6	8	2 3 5 7 9	2 3 6 9 7	1 2 3 5 6 9 7	1 5 7	1 2 3 5 6 7 9	1 2 3 5 6 9 7 9
3	2 5 6 8 9	2 3 4 5 8	1 7	2 3 5 6 9 8	3 4 5 8 9	2 3 4 5 6 8 9	2 3 4 5 6 8 9	2 3 4 5 6 8 9
1	1 5 6 7 9	1 4 5 9 7 9	1 8	1 5 6 7 9 7	1 4 5 3	1 3	2	
1	3 1 4	6	3 4 4 7 9	3 8	2 5	1 3 4		
1	3 5 6 7	9	1 2 3 4 5 7	2 3 4 6 7	1 2 3 5 6 7	1 3 4 5 8	1 2 3 4 5 6 7	
1	5 8	3 7	6 4 9 8 9	2 4 5	1 2 5 9	1 2 4 5 8	1 2 4 5 8	
2	7	1 3 4 9	3 4 8 9	5	1 3 1 3 1 3 6 4 9 8 8 9	1 2 3 4 6 4 6 8 9 8 9	1 2 3 4 6 4 6 8 9 8 9	
3	2 5 6 7 8 9	2 3 5 8	2 3 5 9 7 8 9	1 4	2 3 5 7 8 9	2 3 5 6 7 8 9	2 3 5 6 7 8 9	
1	3 1 2 5 6 4 5 6 7 8 9	1 2 3 4 5 9 7 8 9	2 3 4 6 9 7 9	1 2 3 5 6 9 7 9	1 2 3 4 5 6 6	1 2 3 4 5 6 7 8 9	4	

◀ Back to Start

► Skip Animation

# Propagation Steps (Forward Checking)

4	1 2 5 6	8	2 3 5 9	2 3 6 9	1 2 3 5 6 9	1 5	3 7	1 2 6 7	1 2 3 5 6 9
3 5 6 8 9	2 4 5 8	2 3 4 5 9	1 7	2 3 5 6 9	3 4 5 8	2 4 6 9	2 3 4 5 6 8 9		
1 5 6 7 9	1 4 5 6 9	1 4 5 9	8	1 5 6 4 5 7 9 7	1 5 6 4 5 7 9 7	3 4 5 9	2 4 5 6 7 9		
1 5 6 7 9	3 1 4	6	3 4 4 7 9	3 8	2 8 2 5	5 8	1 3 4 7 7 9		
1 5 6 7	3 4 5	9	1 2 3 4 5 7	2 3 4 6 7	1 2 3 5 6 4 5 7 7	3 8	1 2 3 4 5 6 7		
1 5 8	3 4	7	6	2 4	1 2 5	9	1 2 4 5 8		
2	7	1 3 4 9	3 4 8 9	5	1 3 1 3 6 4 9 8	1 4	1 3 4 6 4 6 9 8 9		
3 5 6 7 8 9	2 5 6 8	2 3 5 9	2 3 5 9	1 4	2 3 1 2 3 5 6 7 8	3 6	2 3 6 5 6 7 8 9		
1 5 6 7 8 9	3 1 2 4 5 6 8	1 2 3 4 5 9	2 3 4 5 9 7 8 9	2 3 4 6 9 7 9	1 2 3 5 6 7 9	6 4 6 7 9	1 2 4 6 7 9		

◀ Back to Start

► Skip Animation

# Propagation Steps (Forward Checking)

4	1 2 5 6	8	2 3 5 9	2 3 6 9	1 2 3 5 6 9	1 5	3 7	1 2 6 7	1 2 3 5 6 9
3 5 6 8 9	2 4 5 8	2 3 4 5 9	1 7	2 3 5 6 9	3 4 5 8	2 4 6 9	2 3 4 5 6 8 9		
1 5 6 7 9	1 4 5 6 9	1 4 5 9	8	1 5 6 4 5 7 9 7	1 5 6 4 5 9 7	3 4 5 9	2 4 6 9		
1 5 6 7 9	3 1 4	6	3 4 4 7 9	3 8	2 5	1 3 4	1 3 4 7 9		
1 5 6 7	3 4 5	9	1 2 3 4 5 7	2 3 4 6 7	1 2 3 5 6 4 5 7 7	8	1 2 3 4 5 6 7		
1 5 8	3 4	7	6	2 4	1 2 5	9	1 2 4 5 8		
2	7	1 3 4 9	3 4 8 9	5	1 3 1 3 6 4 9 8	1 4 9	1 3 4 6 4 6 9 8 9		
3 5 6 7 8 9	2 5 6 8	2 3 5 9	2 3 8 9	1 7 8 9	2 3 1 2 3 5 9 7 9	6	2 3 6 5 6 7 8 9		
1 5 7 8 9	3 1 2 5 5 8	1 2 3 5 5 9	2 3 9 7 8 9	2 3 9 7 9	1 2 3 5 9	4			

◀ Back to Start

► Skip Animation

# Propagation Steps (Forward Checking)

4	1 2 5 6	8	2 3 5 9	2 3 6 9	1 2 3 5 6 9	1 3 5 7	1 2 6 7	1 3 5 6 9
3	2 5 6 4 5 8 9	2 3 4 5 8 9	1 7	2 3 5 6 9 8	3 4 5 9	2 4 6 9	3 5 6 8 9	
1	1 5 6 4 5 6 7 9	1 4 5 9 7 9	8	1 5 6 4 5 7 9 7	1 3 3	2	1 3 7 9	
1	3 1 4 7 9	6	3 4 5 7 9	3 4 6 9	8 5 6 7	2 4 5 9	5 1 3 7 9	
1	3 5 6 7	9	1 2 3 4 5 7	2 3 4 6 7	1 2 3 5 6 4 5 7 7	3 8	1 3 5 6 7	
1	5 8	3 7	6	2 4 5	1 2 5 9	1 2 4 9	1 5 8	
2	7	1 3 4 9	3 4 8 9	5	1 3 1 3 6 4 9 8	1 3 4 6 9	1 3 6 6 8 9	
3	2 5 6 7 8 9	2 3 5 8	2 3 5 9 7 8 9	1 2 3 9 7 9	3 5 7 8	2 6 7 9	3 5 6 8 9	
1	3 1 2 5 5 7 8 9	1 2 3 5 9	2 3 5 9 7 8 9	1 2 3 5 9 7 9	6 1 2 7 9	1 2 4	1 3 7 9	

◀ Back to Start

► Skip Animation

# Propagation Steps (Forward Checking)

4	1 2 5 6	8	2 3 5 9	2 3 6 9	1 2 3 5 6 9	1 5	3 7	1 2 6 9	1 3 5 6 9
3 5 6 9	2 5 6 9	1	7	2 3 5 6 4 5 9 8	3 4 6 9 9	2 4 6 9 9	3 5 6 8 9	3 5 6 8 9	3 5 6 8 9
1 5 6 7 9	1 5 6 9	8	1 4 5 7 9	1 5 6 4 5 7 9 7	1 3 3	2 3 7 9	1 3 5 6 7 9	1 3 5 6 7 9	2 3 7 9
1 3 7	3 1 4 7 9	6	3 4 4 7 9	3 4 6 9	8 2 7	2 5 9	5 7 9	1 3 5 6 7 9	1 3 5 6 7 9
1 3 5 6 7	9 4 5 7	2 3 4 5 7	2 3 4 6 7	1 2 3 5 6 4 5 7 7	1 3 8	1 2 3 7	1 3 5 6 7	1 3 5 6 7	1 3 5 6 7
1 5 8	3 7 9	7 6	2 4 4	1 2 5 5	9	1 2 4 4	1 5 8	1 5 8	1 5 8
2	7	1 3 4 9	3 4 8 9	5	1 3 1 3 6 4 9 8	1 4 4	1 3 6 6 9 8 9	1 3 6 6 9 8 9	1 3 6 6 9 8 9
3 5 6 7 8 9	2 5 6 8	2 3 5 9	2 3 5 7 8 9	1 2 3 1 2 3 5 9 7 9	3 5 7 8	2 6 7 9	3 5 6 7 8 9	3 5 6 7 8 9	3 5 6 7 8 9
1 5 7 8 9	1 2 5 5 8	1 2 3 5 5 9	2 3 5 7 8 9	1 2 3 5 9 7 9	6	1 2 7 9	1 2 7 9	4	4

◀ Back to Start

► Skip Animation

# Propagation Steps (Forward Checking)

4	1 2 5 6	8	2 3 5 9	2 3 6 9	2 3 5 6	1 5	3 7	1 2 6 7	1 3 5 6
3 5 6 9	2 5 6 9	2 3 5 9	1 7	2 3 5 6	4 5 9 8	3 4 6	2 4 9	3 5 6 8 9	3
1 5 6 7 9	1 5 6 9	1 5 4 5 9	8	1 5 6 4 5 9 7	1 3 7	3 2 5	2 1 3	2 1 3	2
1 3 4 7 9	3 1 4	6	3 4 7 9	3 9	8 2 5	1 3	1 3 7 9	1 3 5 6 7	3
1 3 5 6 7	9	1 2 3 4 5 7	2 3 4 5 7	2 3 4 6 7	1 2 3 5 6 7	1 3 4 5 7	8 1 2	1 3 5 6 7	3
1 5 8	3 7	7 6	4	2 5	1 2 5	9 4	1 2 4	1 3 5 8	5
2	7	1 3 4 9	3 4 8 9	5	1 3 6 4 9 8	1 3 4 6 9	1 2 4 6 9	1 3 6 6 8 9	3
3 5 6 7 8 9	2 5 6 8	2 3 5 9	2 3 8 9	1 7 8 9	2 3 7 8	3 5 7 9	2 6 7 9	3 5 6 7 8 9	3
1 5 7 8 9	3 5 8	1 2 5 9	1 2 3 5 7 8 9	2 3 9 7 9	1 2 3 5 9 7 9	6 1 2 7 9	1 2 7 9	1 2 5 7 9	4

◀ Back to Start

► Skip Animation

# Propagation Steps (Forward Checking)

4	1 2 5 6	8	2 3 5 9	2 3 6 9	2 3 5 6	1 5	1 7	1 9	1 6 5 6 9
3 5 6 9	2 5 6 9	1 7	2 3 5 6 9 8	4 5	4 5	4	6	9	5 6 8 9
1 5 6 7 9	1 5 6 9	8	1 5 6 9 7	4 5	3 9	3	2	1 3	
1 5 6 7 9	4 7 9	6 4 5 7	3 4 9	3 9	8	2	5	1 7 9	
5 9 2 7	2 3 4 5 7	2 3 4 6 7	1 2 3 5 6 7	1 3 4 5 7	1 3	8	1 2 4 5 7	1 3 5 6 7	
8 3 7 2 7	7 6 1 3 4 9	6 4	1 2 5	9	1 2 4	1 2 4	1 5 8	1 3 6 9	
3 5 6 7 8 9	2 5 6 8	2 3 5 9	2 3 7 8 9	1 7 8 9	3 5 7 8	2 7 9	2 7 9	3 6 5 6 7 8 9	
1 5 7 8 9	1 2 5 9	1 2 3 5 9	2 3 7 8 9	2 3 9 7 9	1 2 3 5 9	6 7 9	1 2 7 9	4	

◀ Back to Start

► Skip Animation

# Propagation Steps (Forward Checking)

4	1 2 5 6	8	2 3 5 9	2 3 6 9	2 3 5 6	1 5	1 7	1 9	1 6 5 6	9
3 6 9	2 5 6 9	1 7	2 3 5 6 9	4 5 8	4 5 9	4 6	4 9	5 6 8 9		
1 6 7	1 5 6 9	8		1 5 6 9	1 4 5 7	3	2			
1 5	4 9	6 2	3 7 9	3 9	8 2	5				3
8	3 7	7 6	2 4	1 2 5	9	1 2 4	5	1 2 4	1 5	
2 3	7 6 9	1 3 4 9	3 4 8 9	5	1 3 6 4 9 8	3 4 6	1 2 4 9	1 3 6 6 8 9		
3 7	2 9	5 6 8	3 9 7 8 9	1 2 3 5 9 7 8 9	2 3 1 2 3 5 9 7 9	3 5 7 8	2 7 9	6 7 9 8 9	3 5 6 8 9	
3 7	1 2 9	1 3 8	2 3 9 7 8 9	2 3 9 7 9	1 2 3 5 9	6	1 2 7 9	4		

◀ Back to Start

► Skip Animation

# Propagation Steps (Forward Checking)

4	1 2 5 6	8	2 3 5 9	2 3 6 9	2 3 5 6	1 5	1 7	1 9	5 6 7 9
3 6 9	2 5 6 9	1 7	2 3 5 6 9 8	4 5 9 8	4 6	6 9	5 6 8 9		
1 6 7	1 5 6 9	8	1 5 6 9 7	4 5 9 7	3	2			
1 5	4 9	6 2	3 7 9	3 9	8 2	5	7 9	3	
8 2	3 7	7 6	2 5	5 9	9 4	4	1		
3 7	2 6 9	1 3 4 9	3 4 8 9	1 3 7	3 1 6 4 9 8	3 1 4 9	1 3 6 6 9 8 9		
3 7	1 2 5 9	1 3 5 9	2 3 5 7 8 9	2 3 1 2 3 9 7 9	5 9	3 7 8	2 7 9	6 5 6 7 8 9	3
					6	1 2 7 9		4	

◀ Back to Start

► Skip Animation

# Propagation Steps (Forward Checking)

4	1 2 5 6	8	2 3 5 9	3 6 9	2 3 6 9	1 5 7	1 7 9	6 9 7	5 6 9 9
3 6 9	2 5 6	3 5 9	1 7	2 3 6 9	4 5 8		6 9	5 6 8 9	
1 6 7	1 5 6	1 5 9	4 5 9	8		1 6 4 5 9 7	3	2	
1 5	4 9	6 2	3 4 7	3 4	3 1 7 7	3 7	2 8	5 3	7
8 2	3 7	7 6	2 4	2 5	5 9	9 8	4 6	1 7	3 6
3 7	2 9	1 3 4 9	3 4 8 9	3 5	1 3 1 3 6 4 9 8	1 6 9	3 6	2 8 9	3 6
6 9	5 6 8	5 9 7 8 9	2 3 5 9 7 8 9	1 4	2 3 7 8	3 7 9	2 7 9	5 6 8 9	3 6
3 7	1 2 9	1 3 5	2 3 5 9	3 9 7 9	1 2 3 7 9	6 7 9	1 2 7 9	4	

◀ Back to Start

► Skip Animation

# Propagation Steps (Forward Checking)

4	1 2 5 6	8	2 3 5 9	3 6 9	2 3 6 9 7	1 5	1 7	6 9 7	5 6 9 9
3 6 9	2 5 6	3 5 9	1 7	2 3 6 4 5 9 8		6	5 6 9 9	8 9	
1 6 7	1 5 6	1 5 9	4 5 9	8	1 6 4 5 9 7	3	2		3
1 5	4 9	6 2	3 4 7	3 1 9	8 2 5	3 7			
8 2	3 7	7 6	2 4 9	5	1 3 1 6 4 9 8	8 3 7			3 6
3 6 9	5 6 8	3 5 9 7 8 9	2 3 8 9	1 4 7 8	3 1 2 3 5 7 8 9	3 2 7 9	2 6 7 9 8 9	3 5 6 7 8 9	
3 1 5 9	1 5 8	3 5 9 7 8 9	2 3 8 9	1 4 7 9	6	1 2 7 9	2 6 7 9 8 9	3 5 6 7 8 9	4

◀ Back to Start

► Skip Animation

# Propagation Steps (Forward Checking)

4	1 2 5 6	8	2 3 5 9	3 6 9	2 3 6 9 7	1 5	1 7	6 9 7	5 6 9 9
3 6 9	2 5 6	3 5 9	1 7	2 3 6 4 5 9 8		6	5 6 9 9		
1 6 7	1 5 6	1 5 9	4 5 9	8	1 6 4 5 9 7	3	2		
1 5	4 9	6 2	3 7 9	3 9	8 2	5		3 7	
8 2	3 7	7 6	2 5	2 5	9 4	1			
3 6 9	1 5 6 8	3 5 9 7 8 9	3 2 3 8 9	5	3 1 6 4 9 8	3 1 6	3 9	6 9 8 9	
3 6 9	1 5 6 8	1 5 9 7 8 9	3 2 3 8 9	1 4	5 7 8	3 2	1 2 7 9	6 9 7 8 9	
3 6 9	1 5 6 8	1 5 9 7 8 9	2 3 8 9	3 2 3 9 7 9	6	1 2 7 9	4		

◀ Back to Start

► Skip Animation

# Propagation Steps (Forward Checking)

4	1 2 5 6	8	2 3 5 9	3 6 9	2 3 6 9 7	1 5	1 7	6 9 7	5 6 9 9
3 6 9	2 5 6	3 5 9	1 7	2 3 6 4 5 9 8		6	5 6 9 9		
1 6 7	1 5 6	1 5 9	4 5 9	8	1 6 4 5 9 7	3	2		
1 5	4 9	6 2	3 7 9	3 9	8 2	5		3 7	
8 2	3 7	7 6	2 5	5 9	9 4			3 6 7	
3 6 9	1 5 6 8	3 5 9	2 3 8 9	5	3 1 6 9 8	3 1		9 8 9	
3 6 9	1 5 6 8	1 3 9	2 3 7 8 9	1 4	5 7 8	3 2		3 5 7 8 9	
3 6 9	1 5 6 8	1 3 9	2 3 7 8 9	3 2 3 9 7 9	6	1 2 7 9		4	

◀ Back to Start

# After Setup (Forward Checking)

4	1 2 5 6	8	2 3 5 9	3 6 9	2 3 6 7	1 5	1 7	6 9	5 6 7 9
3 6 9	2 5 6 5 9	3 9	1 7	2 3 6 4 5 9 8			6 9	5 6 8 9	
6 7	1 5 6 5 9	1 9	8	1 6 4 5 9 7	3 3	2 7			
1	4	6	3 7 9	3 9	8 3 1 3	2 3	5 7		3
5	9	2	3 4 7	3 4	7 7	7 7	8 3		6 7
8	3	7	6 7 9	2 8 9	5 5	9 6 9	4 8	1 9	
2	7	1 3 4	3 9	3 8 9	5 5	3 1 6 9	3 1 8		3 8 9
3 6 9	5 6 8	5 9 7 8 9	2 3 9 7 8 9	1 9 7 9	4 7 8	5 7 8	2 7	3 9 7 8 9	5
3 9	1 3 5 8	1 3 5	2 3 9 7 8 9	3 9 7 9	2 3 6 7 9	1 2 6 7 9		4 7 9	

# Outline

## Problem

### Initial Propagation (Forward Checking)

## Improved Reasoning

Bounds Consistency

Domain Consistency

Comparison

## Search

### Other Global Constraints

# Can we do better?

- The alldifferent constraint is missing propagation
  - How can we do more propagation?
  - Do we know when we derive all possible information from the constraint?
- Constraints only interact by changing domains of variables

# A Simpler Example

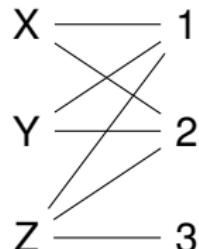
```
include "alldifferent.mzn";  
  
var 1..2:X;  
var 1..2:Y;  
var 1..3:Z;  
  
constraint alldifferent([X, Y, Z]);  
  
solve satisfy;
```

# Using Forward Checking

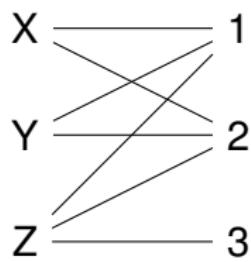
- No variable is assigned
- No reduction of domains
- But, values 1 and 2 can be removed from Z
- This means that Z is assigned to 3

# Visualization of alldifferent as Graph

- Show problem as graph with two types of nodes
  - Variables on the left
  - Values on the right
- If value is in domain of variable, show link between them
- This is called a *bipartite* graph



# A Simpler Example



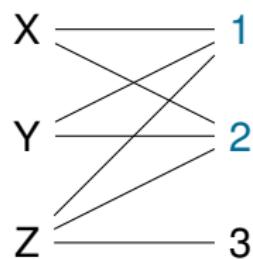
Value Graph for

`var 1..2:X;`

`var 1..2:Y;`

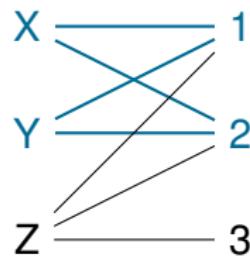
`var 1..3:Z;`

# A Simpler Example



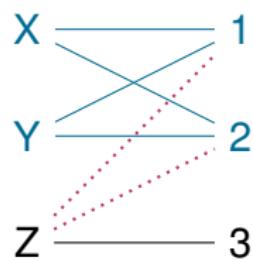
Check interval [1,2]

# A Simpler Example



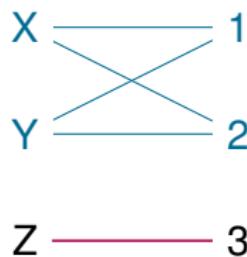
- Find variables completely contained in interval
- There are two: X and Y
- This uses up the capacity of the interval

# A Simpler Example



No other variable can use that interval

# A Simpler Example



Only one value left in domain of Z,  
this can be assigned

## Idea (Hall Intervals)

- Take each interval of possible values, say size  $N$
- Find all  $K$  variables whose domain is completely contained in interval
- If  $K > N$  then the constraint is infeasible
- If  $K = N$  then no other variable can use that interval
- Remove values from such variables if their bounds change
- If  $K < N$  do nothing
- Re-check whenever domain bounds change

# Implementation

- Problem: Too many intervals ( $O(n^2)$ ) to consider
- Solution:
  - Check only those intervals which update bounds
  - Enumerate intervals incrementally
  - Starting from lowest(highest) value
  - Using sorted list of variables
- Complexity:  $O(n \log(n))$  in standard implementations
- Important: Only looks at min/max bounds of variables

# Bounds Consistency

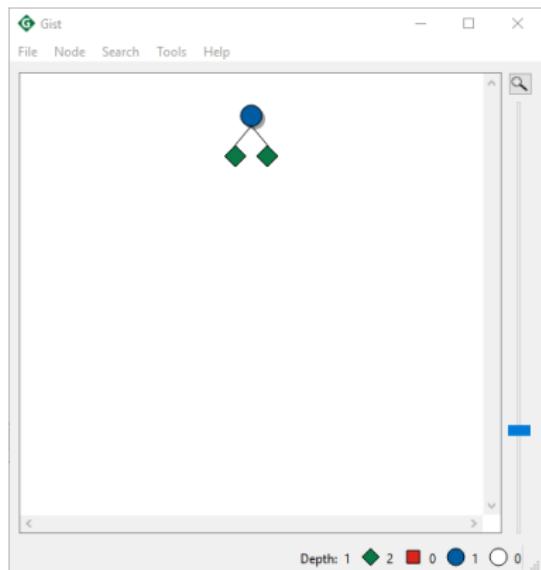
## Definition

A constraint achieves *bounds consistency*, if for the lower and upper bound of every variable, it is possible to find values for all other variables between their lower and upper bounds which satisfy the constraint.

## Annotation: :: bounds

```
include "alldifferent.mzn";  
  
var 1..2:X;  
var 1..2:Y;  
var 1..3:Z;  
  
constraint alldifferent([X,Y,Z]) :: bounds;  
solve satisfy;
```

# Running with Gecode Gist



All Solutions

A screenshot of the Gist Console window titled "Gist Console: Gecode/FlatZinc". It has "Clear" and "Stay on top" buttons. The console displays the following FlatZinc code:

```
X = [1..2];
Y = [1..2];
Z = 3;
```

Node Inspector (Root)

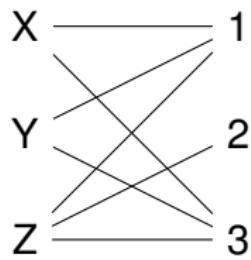
# Can we do even better?

- Bounds consistency only considers min/max bounds
- Ignores “holes” in domain
- Sometimes we can improve propagation looking at those holes

## Another Simple Example

```
include "alldifferent.mzn";  
  
var {1,3}:X; % note enumerated domain  
var {1,3}:Y;  
var 1..3:Z; % note domain as interval  
  
% annotated constraint  
constraint alldifferent([X,Y,Z]) :: bounds;  
solve satisfy;
```

## Another Simple Example



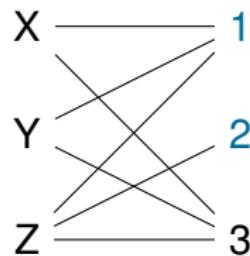
Value Graph for

`var {1, 3}:X;`

`var {1, 3}:Y;`

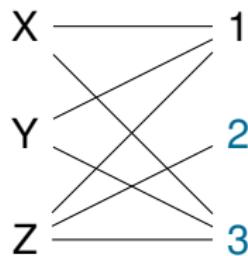
`var 1..3:Z;`

## Another Simple Example



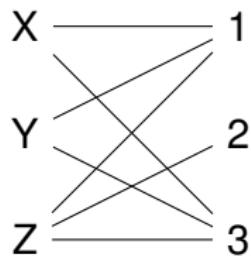
- Check interval [1,2]
- No domain of a variable completely contained in interval
- No propagation

## Another Simple Example



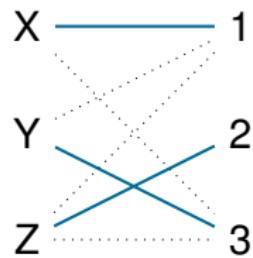
- Check interval [2,3]
- No domain of a variable completely contained in interval
- No propagation

## Another Simple Example



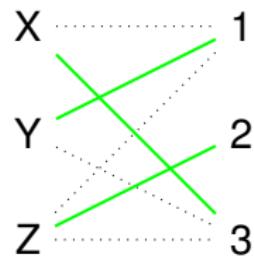
But, more propagation is possible,  
there are only two solutions

## Another Simple Example



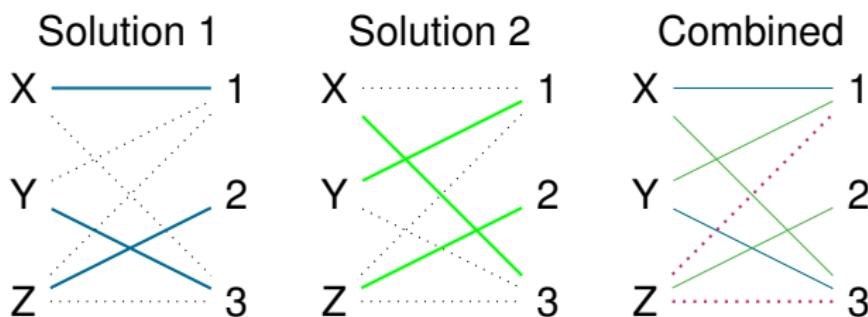
Solution 1: assignment in blue

## Another Simple Example



Solution 2: assignment in green

## Another Simple Example

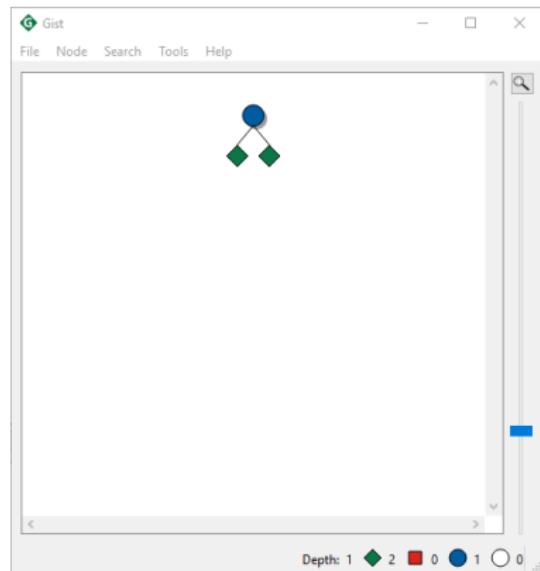


Combining solutions shows that  $Z=1$  and  $Z=3$  are not possible.

## Another Simple Example

Can we deduce this without enumerating solutions?

# Bounds Consistency with Gecode Gist: No Propagation



All Solutions

The screenshot shows the Gist Console window titled "Gist Console: Gecode/FlatZinc". It has a "Clear" button and a "Stay on top" checkbox. The console displays the following FlatZinc code:  
X = {1..3};  
Y = {1..3};  
Z = [1..3];

Node Inspector (Root)

# Solutions and Maximal Matchings

- A *Matching* is subset of edges which do not coincide in any node
- No matching can have more edges than number of variables
- Every solution corresponds to a *maximal matching* and vice versa
- If a link does not belong to some maximal matching, then it can be removed

# Implementation

- Possible to compute all links which belong to some matching
  - Without enumerating all of them!
- Enough to compute **one** maximal matching
- Requires algorithm for *strongly connected components*
- Extra work required if more values than variables
- All links (values in domains) which are not supported can be removed
- Complexity:  $O(n^{1.5}d)$

# Domain Consistency

## Definition

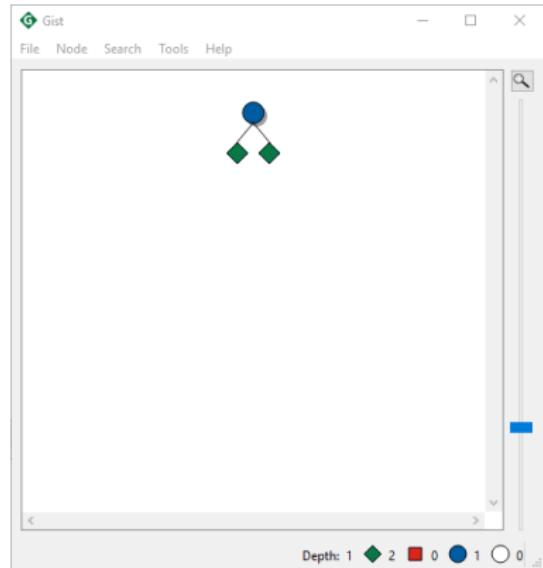
A constraint achieves *domain consistency*, if for every variable and for every value in its domain, it is possible to find values in the domains of all other variables which satisfy the constraint.

- Also called *generalized arc consistency (GAC)*
- or *hyper arc consistency*

# Simple Example Revisited

```
include "alldifferent.mzn";  
  
var {1,3}:X; % note enumerated domain  
var {1,3}:Y;  
var 1..3:Z; % note domain as interval  
  
% note different annotation  
constraint alldifferent([X,Y,Z]) :: domain;  
solve satisfy;
```

# Domain Consistency with Gecode Gist: Propagation



All Solutions

A screenshot of the Gist Console window titled 'Gist Console: Gecode/FlatZinc'. It shows a code snippet:

```
X = {1, 3};  
Y = {1, 3};  
Z = 2;
```

Below the code are standard window control buttons (minimize, maximize, close) and a status bar.

Node Inspector (Root)

## Can we still do better?

- NO! This extracts all information from this one constraint
- We could perhaps improve speed, but not propagation
- But possible to use different model
- Or model interaction of multiple constraints

# Should all constraints achieve domain consistency?

- Domain consistency is usually more expensive than bounds consistency
  - Overkill for simple problems
  - Nice to have choices
- For some constraints achieving domain consistency is NP-hard
  - We have to live with more restricted propagation

# Modified MiniZinc Program

```
int: s;
int: n=s*s;
array[1..n,1..n] of var 1..n: puzzle;

include "sudoku.dzn";

include "alldifferent.mzn";

constraint forall(i in 1..n)
    (alldifferent([puzzle[i,j] | j in 1..n])::domain);
constraint forall(j in 1..n)
    (alldifferent([ puzzle[i,j] | i in 1..n])::domain);
constraint forall(i,j in 1..s)
    (alldifferent([puzzle[s*(i-1)+p, s*(j-1)+q] |
        p,q in 1..s])::domain);

solve satisfy;
```

# Modified Choco-solver Sudoku Model

```
Model model = new Model("Sudoku");
int blockSize = 3;
int m = blockSize*blockSize;

IntVar[][] vars = new IntVar[m][m];
for(int i=0;i<m;i++){
    for(int j=0;j<m;j++) {
        vars[i][j] = model.intVar("X"+i+"_"+j, 1, m);
        if (data[i][j]>0)
            model.arithm(vars[i][j], "=", data[i][j]).post();
    }
}
// Consistency level AC: domain consistency, BC: bounds consistency, default: mix
for(int i=0;i<m;i++){
    model.allDifferent(row(i,m,vars),AC).post();
    model.allDifferent(column(i,m,vars),AC).post();
}
for(int i=0;i<m;i+=blockSize){
    for(int j=0;j<m;j+=blockSize){
        model.allDifferent(block(i, j, blockSize, vars),AC).post();
    }
}
Solver solver = model.getSolver();
solver.solve();
```

# Initial State (Domain Consistency)

4	1 2 3	8	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3
4 5 6	4 5 6	7 8 9	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6
7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9
1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3
4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6
7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9
1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3
4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6
7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9
1 2 3	1 2 3	6	1 2 3	1 2 3	8	2	5	1 2 3	4 5 6
4 5 6	4 5 6	7 8 9	4 5 6	4 5 6	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9
7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9
1 2 3	1 2 3	9	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	4 5 6
4 5 6	4 5 6	7 8 9	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6
7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9
1 2 3	1 2 3	3	7	6	1 2 3	1 2 3	9	1 2 3	1 2 3
4 5 6	4 5 6	7 8 9	4 5 6	4 5 6	7 8 9	7 8 9	7 8 9	7 8 9	4 5 6
7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9
2	7	1 2 3	1 2 3	5	1 2 3	1 2 3	1 2 3	1 2 3	4 5 6
4 5 6	4 5 6	7 8 9	4 5 6	4 5 6	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9
7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9
1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	4 5 6
4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6
7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9
1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	6	1 2 3	4 5 6
4 5 6	4 5 6	7 8 9	4 5 6	4 5 6	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9
7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9

# Propagation Steps (Domain Consistency)

4	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3
4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6
7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9
1 2 3	1 2 3	1 2 3		1 2 3	1 2 3	1 2 3	1 2 3	1 2 3
4 5 6	4 5 6	4 5 6	1	7	4 5 6	4 5 6	4 5 6	4 5 6
7 8 9	7 8 9	7 8 9			7 8 9	7 8 9	7 8 9	7 8 9
1 2 3	1 2 3	1 2 3	1 2 3		1 2 3	1 2 3		1 2 3
4 5 6	4 5 6	4 5 6	4 5 6	8	4 5 6	4 5 6	3	2
7 8 9	7 8 9	7 8 9	7 8 9		7 8 9	7 8 9		
1 2 3	1 2 3		1 2 3	1 2 3		1 2 3	1 2 3	
4 5 6	4 5 6	6	4 5 6	4 5 6	8	2	5	4 5 6
7 8 9	7 8 9		7 8 9	7 8 9				7 8 9
1 2 3		1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	
4 5 6	9	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	8	4 5 6
7 8 9		7 8 9	7 8 9	7 8 9	7 8 9	7 8 9		7 8 9
1 2 3		1 2 3	1 2 3	1 2 3		1 2 3	1 2 3	
4 5 6	3	7	6	4 5 6	4 5 6	9	4 5 6	4 5 6
7 8 9				7 8 9	7 8 9		7 8 9	7 8 9
2	7		5	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3
7 8 9		7 8 9		4 5 6	4 5 6	4 5 6	4 5 6	4 5 6
1 2 3	1 2 3	1 2 3	1 2 3	1 2 3		1 2 3	1 2 3	1 2 3
4 5 6	4 5 6	4 5 6	4 5 6	4 5 6		4 5 6	4 5 6	4 5 6
7 8 9	7 8 9	7 8 9	7 8 9	7 8 9		7 8 9	7 8 9	7 8 9
1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3		1 2 3	
4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	6	4 5 6	4
7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9		7 8 9	

▶ Skip Animation

# Propagation Steps (Domain Consistency)

[◀ Back to Start](#)

► Skip Animation

# Propagation Steps (Domain Consistency)

4	1 2 3 5 6 7 9	8	1 2 3 5 6 7 9	1 2 3 5 6 9 7	1 2 3 5 6 9 9			
1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1	7	1 2 3 4 5 6 7 8 9				
1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	8	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	3	2		
1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	6	1 2 3 4 5 6 7 8 9	8	2	5	1 2 3 4 5 6 7 8 9	
1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	8	4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	
1 2 3 4 5 6 7 8 9	3	7	6	1 2 3 4 5 6 7 8 9	9	4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	
2	7			5	1 2 3 4 5 6 7 8 9			
1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9		1	4	1 2 3 4 5 6 7 8 9			
1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9			6	1 2 3 4 5 6 7 8 9	4		

◀ Back to Start

► Skip Animation

# Propagation Steps (Domain Consistency)

4	1 2 3 5 6	8	1 2 3 5 6					
7 9	7 9	7 9	7 9	7 9	7 9	7 9	7 9	7 9
1 3 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 7 8 9	1 2 3 4 5 6 7 8 9					
1 3 5 6 7 8 9	1 2 3 4 5 6 7 8 9	8	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	3	2		
1 3 5 6 7 8 9	1 2 3 4 5 6 7 8 9	6	1 2 3 4 5 6 7 8 9	8	2	5	1 2 3 4 5 6 7 8 9	
1 3 5 6 7 8 9	9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	8	1 2 3 4 5 6 7 8 9	
1 3 5 6 7 8 9	3 7 9	7 6 5	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	9	1 2 3 4 5 6 7 8 9	
2 7 8 9	7 8 9	7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	5	1 2 3 4 5 6 7 8 9	
1 3 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 4 6	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 4	1 2 3 4 5 6 7 8 9	
1 3 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 4 6	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	4	1 2 3 4 5 6 7 8 9	

◀ Back to Start

► Skip Animation

# Propagation Steps (Domain Consistency)

4	1 2 3 5 6	8	1 2 3 5 6 6 5	1 2 3 6 5 5 6	1 2 3 5 6 5 6	1 2 3 5 6 5 6	1 2 3 5 6 5 6
7 9	7 9	7 9 7	9 7 9	9 7 9	9 7 9	9 7 9	9 7 9
3 5 6 8 9	2 3 4 5 6 8 9	2 3 4 5 6 8 9	1 7	2 3 4 5 6 8 9			
1 5 6 7 8 9	3 1 2 3 4 5 6 4 5 6 7 8 9 7 8 9	1 2 3 4 5 6 4 5 6 7 8 9 7 8 9	8	1 2 3 4 5 6 4 5 6 7 8 9 7 8 9	1 2 3 4 5 6 4 5 6 7 8 9 7 8 9	3	2
1 5 6 7 8 9	3 1 2 3 4 5 6 4 5 6 7 8 9 7 8 9	1 2 3 4 5 6 4 5 6 7 8 9 7 8 9	6	1 2 3 4 5 6 4 5 6 7 8 9 7 8 9	8 2 5	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9
1 5 6 7 8 9	3 1 2 3 4 5 6 4 5 6 7 8 9 7 8 9	1 2 3 4 5 6 4 5 6 7 8 9 7 8 9	9	1 2 3 4 5 6 4 5 6 7 8 9 7 8 9	8	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9
1 5 6 7 8 9	3 7 4 5 6 4 5 6 7 8 9 7 8 9	1 2 3 4 5 6 4 5 6 7 8 9 7 8 9	6	1 2 3 4 5 6 4 5 6 7 8 9 7 8 9	9	1 2 3 4 5 6 4 5 6 7 8 9 7 8 9	1 2 3 4 5 6 7 8 9
2 7	1 2 3 4 5 6 4 5 6 7 8 9 7 8 9	1 2 3 4 5 6 4 5 6 7 8 9 7 8 9	5	1 2 3 4 5 6 4 5 6 7 8 9 7 8 9	1 2 3 4 5 6 4 5 6 7 8 9 7 8 9	1 2 3 4 5 6 4 5 6 7 8 9 7 8 9	1 2 3 4 5 6 4 5 6 7 8 9 7 8 9
1 5 6 7 8 9	3 1 2 3 4 5 6 4 5 6 7 8 9 7 8 9	1 2 3 4 5 6 4 5 6 7 8 9 7 8 9	1 4	1 2 3 4 5 6 4 5 6 7 8 9 7 8 9	1 2 3 4 5 6 4 5 6 7 8 9 7 8 9	1 2 3 4 5 6 4 5 6 7 8 9 7 8 9	1 2 3 4 5 6 4 5 6 7 8 9 7 8 9
1 5 6 7 8 9	3 1 2 3 4 5 6 4 5 6 7 8 9 7 8 9	1 2 3 4 5 6 4 5 6 7 8 9 7 8 9	6	1 2 3 4 5 6 4 5 6 7 8 9 7 8 9	6	1 2 3 4 5 6 7 8 9	4

◀ Back to Start

► Skip Animation

# Propagation Steps (Domain Consistency)

4	1 2 5 6	8	1 2 3 5 6 7 9	1 2 3 5 6 9 7	1 2 3 5 6 9 9			
3	2	2 3	1	7	2 3	2 3	2 3	2 3
5 6	4 5 6	4 5 6	1	7	4 5 6	4 5 6	4 5 6	4 5 6
8 9	8	8 9			8 9	8 9	8 9	8 9
1	3 1 2	1 2 3	1 2 3	8	1 2 3	1 2 3	3	2
5 6	4 5 6	4 5 6	4 5 6	8	4 5 6	4 5 6	4 5 6	4 5 6
7 8 9	8	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9
1	3 1 2	1 2 3	1 2 3	6	1 2 3	1 2 3	1 2 3	1 2 3
5 6	4 5 6	4 5 6	4 5 6	8	2	5	4 5 6	4 5 6
7 8 9	8	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9
1	3	9	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3
5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6
7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9
1	3	3	7	6	1 2 3	1 2 3	9	1 2 3
5 6	4 5 6	4 5 6	4 5 6	7 8 9	4 5 6	4 5 6	4 5 6	4 5 6
7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9
2	7		5	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3
4 5 6	4 5 6	4 5 6	4 5 6	7 8 9	4 5 6	4 5 6	4 5 6	4 5 6
7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9
1	3 1 2	1 2 3	1 2 3	1	4	1 2 3	1 2 3	1 2 3
5 6	4 5 6	4 5 6	4 5 6	7 8 9	4 5 6	4 5 6	4 5 6	4 5 6
7 8 9	8	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9
1	3 1 2	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	6	4
5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	4 5 6	7 8 9	
7 8 9	8	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	7 8 9	

◀ Back to Start

► Skip Animation

# Propagation Steps (Domain Consistency)

4	1 2 5 6	8	1 2 3 5 6 7	1 2 3 5 6 9				
3	2 5 6 8 9	1	2 3 4 5 6 8 9	7	2 3 4 5 6 8 9			
1	1 5 6 7 9	8	1 4 5 6 7 9	3	2			
1	3 1 2 5 6 7 8 9	6	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	8	2	5	1 2 3 4 5 6 7 8 9
1	3 5 6 7 8 9	9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	8	4 5 6 7 8 9	1 2 3 4 5 6 7 8 9
1	3 5 6 7 8 9	3 7 6	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	9	4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9
2	7	5	1 2 3 4 5 6 7 8 9					
1	3 1 2 5 6 7 8 9	1	2 3 1 2 3 4 5 6 4 5 6 7 8 9 7 8 9	4 5 6 4 5 6 7 8 9 7 8 9	1 2 3 1 2 3 4 5 6 4 5 6 7 8 9 7 8 9	1 2 3 1 2 3 4 5 6 4 5 6 7 8 9 7 8 9	1 2 3 1 2 3 4 5 6 4 5 6 7 8 9 7 8 9	1 2 3 1 2 3 4 5 6 4 5 6 7 8 9 7 8 9
1	3 1 2 5 6 7 8 9	6	1 2 3 1 2 3 4 5 6 4 5 6 7 8 9 7 8 9	1 2 3 1 2 3 4 5 6 4 5 6 7 8 9 7 8 9	6	4 5 6 7 8 9	4	1 2 3 1 2 3 4 5 6 4 5 6 7 8 9 7 8 9

◀ Back to Start

► Skip Animation

# Propagation Steps (Domain Consistency)

4	1 2 5 6	8	1 2 3 5 6 7	1 2 3 5 6 9 7	1 2 3 5 6 9			
3	2 5 6 8 9	2 3 4 5 8	1 7	2 3 4 5 6 8 9				
1	1 5 6 7 9	1 4 5 6 9 7 9	1 8	1 4 5 6 7 9 7	1 4 5 6 9 7 9	3 4 5 6 8 9	2 4 5 6 7 8 9	2 4 5 6 7 8 9
1	3 1 2 5 6 7 8 9	1 2 3 4 5 6 8	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9
1	3 5 6 7 8 9	9 4 5 9 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	8 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9
1	3 5 6 7 8 9	3 7 4 5 9 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	9 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9
2	7	2 7 4 5 9 7 8 9	1 2 3 4 5 6 9 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	5 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9
1	3 1 2 5 6 7 8 9	1 2 3 4 5 6 8	1 2 3 4 5 6 9 7 8 9	1 2 3 4 5 6 7 8 9				
1	3 1 2 5 6 7 8 9	1 2 3 4 5 6 8	1 2 3 4 5 6 9 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	6 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	4 4 5 6 7 8 9

◀ Back to Start

► Skip Animation

# Propagation Steps (Domain Consistency)

4	1 2 5 6	8	1 2 3 5 6 7	1 2 3 5 6 9	1 2 3 5 6 9	1 2 3 5 6 9	1 2 3 5 6 9	1 2 3 5 6 9	1 2 3 5 6 9
3	2 5 6 8 9	2 3 4 5 9	1 7	2 3 4 5 6 8 9	2 3 4 5 6 8 9	2 3 4 5 6 8 9	2 3 4 5 6 8 9	2 3 4 5 6 8 9	2 3 4 5 6 8 9
1	1 5 6 7 9	1 4 5 6 9 7 9	1 8	1 4 5 6 7 9 7 9	1 4 5 6 7 9 7 9	1 4 5 6 7 9 7 9	3 4 5 6 7 8 9	2 4 5 6 7 8 9	2 4 5 6 7 8 9
1	3 1 4 7 9	6 4 7 9 7 9	1 3 1 3 4 4 8 2 5	1 2 3 4 5 6 7 8 9 7 8 9	1 2 3 4 5 6 7 8 9 7 8 9	1 2 3 4 5 6 7 8 9 7 8 9	1 3 4 4 7 8 9	2 5 4 4 7 8 9	2 5 4 4 7 8 9
1	3 5 6 7 8 9	9 4 5 9 7 8 9	1 2 3 4 5 6 9 7 8 9	1 2 3 4 5 6 9 7 8 9	1 2 3 4 5 6 9 7 8 9	1 2 3 4 5 6 9 7 8 9	8 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9
1	3 5 6 7 8 9	3 4 5 9 7 8 9	7 6 7 8 9 7 8 9	1 2 3 4 5 6 7 8 9 7 8 9	1 2 3 4 5 6 7 8 9 7 8 9	1 2 3 4 5 6 7 8 9 7 8 9	9 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9
2	7	2 7 4 5 9 7 8 9	1 2 3 4 5 6 5	1 2 3 4 5 6 5	1 2 3 4 5 6 5	1 2 3 4 5 6 5	2 7 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9
1	3 1 2 5 6 4 5 6 7 8 9 8	1 2 3 4 5 6 9 7 8 9	1 2 3 4 5 6 1	1 2 3 4 5 6 1	1 2 3 4 5 6 1	1 2 3 4 5 6 1	4 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9
1	3 1 2 5 6 4 5 6 7 8 9 8	1 2 3 4 5 6 9 7 8 9	1 2 3 4 5 6 9	1 2 3 4 5 6 9	1 2 3 4 5 6 9	1 2 3 4 5 6 9	6 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	4 4 5 6 7 8 9

◀ Back to Start

► Skip Animation

# Propagation Steps (Domain Consistency)

4	1 2 5 6	8	2 3 5	1 2 3 5 6	1 2 3 5 6	1 2 3 5 6	1 2 3 5 6	1 2 3 5 6
3 5 6 8 9	2 4 5 8	2 3 1 9	7 9 7	9 7 9 7	9 7 9 7	9 7 9 7	9 7 9 7	9 7 9 7
1 5 6 7 9	1 4 5 9	1 8	1 4 5 6	1 4 5 6	1 4 5 6	1 4 5 6	3	2
1 5 6 7 9	3 1 4	6	3 1 4	3 1 8	3 1 8	3 1 8	2	5
1 5 6 7 8 9	3 4	9	7 8 9	7 8 9	7 8 9	7 8 9	8	7 8 9
1 5 6 7 8 9	3 7	6	2 3 4 5	1 2 3 4 5 6	1 2 3 4 5 6	1 2 3 4 5 6	9	1 2 3 4 5 6
2 5 6 7 8 9	7	5	2 3 4 5	1 2 3 4 5 6	1 2 3 4 5 6	1 2 3 4 5 6	9	1 2 3 4 5 6
1 5 6 7 8 9	1 2 8	1	2 3 9 7 8 9	1 2 3 7 8 9	1 2 3 7 8 9	1 2 3 7 8 9	4	1 2 3 7 8 9
1 5 6 7 8 9	1 2 8	6	2 3 9 7 8 9	1 2 3 7 8 9	1 2 3 7 8 9	1 2 3 7 8 9	4	1 2 3 7 8 9

◀ Back to Start

► Skip Animation

# Propagation Steps (Domain Consistency)

4	1 2 5 6	8	2 3 5 7	1 2 3 5 6 9 7	1 2 3 5 6 9 7	1 2 3 5 6 9 7	1 2 3 5 6 9 7	1 2 3 5 6 9
3	2 5 6 8 9	2 3 4 5 8	1 7 9	2 3 4 5 6 8 9	2 3 4 5 6 8 9	2 3 4 5 6 8 9	2 3 4 5 6 8 9	2 3 4 5 6 8 9
1	1 5 6 7 9	1 4 5 9 7	8	1 4 5 6 7 9	1 4 5 6 7 9	3	2	
1	3 1 4 7 9	6	3 1 3 4 4 7 9 7 9	8 2	2 5	1 3 4 7 9		
1	3 5 6 7	9	1 2 3 4 5 7	2 3 1 2 3 1 2 3 4 5 6 4 5 6 4 5 6 7 7 7 7	1 2 3 4 5 6 8	1 2 3 4 5 6 7		
1	3 5 6 7 8 9	3 7	6	1 2 3 1 2 3 4 5 6 4 5 6 7 8 9 7 8 9	9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	
2	7	1 2 3 4 5 9 7 8 9	5	1 2 3 4 5 6 7 8 9 7 8 9	9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	
1	3 1 2 5 6 4 5 6 7 8 9	1 2 3 4 5 9 7 8 9	1 4	1 2 3 4 5 6 7 8 9 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	
1	3 1 2 5 6 4 5 6 7 8 9	1 2 3 4 5 9 7 8 9	6	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	4		

◀ Back to Start

► Skip Animation

# Propagation Steps (Domain Consistency)

4	1 2 5 6	8	2 3 5 7 9	2 3 6 9 7	1 2 3 5 6 9 7	1 2 3 5 6 9 7	1 2 3 5 6 9 7	1 2 3 5 6 9
3	2 5 6 8 9	2 3 4 5 8	1 7 9	2 3 4 5 6 8 9	2 3 4 5 6 8 9	2 3 4 5 6 8 9	2 3 4 5 6 8 9	2 3 4 5 6 8 9
1	1 5 6 7 9	1 4 5 9 7 9	1 8 7 9	1 4 5 6 7 9 7 9	1 4 5 6 7 9 7 9	3	2	
1	3 1 4 7 9	6	3 4 4 7 9	3 4 6 4 5 6 9 7 8 9	8 2 5	1 3 4 7 9		
1	3 5 6 7	9	1 2 3 4 5 7	2 3 4 6 4 5 6 7 9 7 9	1 2 3 4 5 6 4 5 6 9 7 8 9	8	4 5 6 7	
1	3 5 6 7 8 9	3 7 6	1 2 3 4 5 9 7 8 9	2 3 1 2 3 4 6 4 5 6 9 7 8 9	1 2 3 1 2 3 4 5 6 4 5 6 9 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	
2	7	1 2 3 4 5 9 7 8 9	6	5	1 2 3 1 2 3 4 5 6 4 5 6 7 8 9 7 8 9	1 2 3 1 2 3 4 5 6 4 5 6 7 8 9 7 8 9	1 2 3 1 2 3 4 5 6 4 5 6 7 8 9 7 8 9	
1	3 1 2 5 6 4 5 6 7 8 9	1 2 3 4 5 9 7 8 9	2 3 4 5 9 7 8 9	1 2 3 4 6 4 5 6 9 7 8 9	1 2 3 4 5 6 4 5 6 9 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	
1	3 1 2 5 6 4 5 6 7 8 9	1 2 3 4 5 9 7 8 9	2 3 4 5 9 7 8 9	1 2 3 4 6 4 5 6 9 7 8 9	6	1 2 3 4 5 6 7 8 9	4	

◀ Back to Start

► Skip Animation

# Propagation Steps (Domain Consistency)

4	1 2 5 6	8	2 3 5 7 9	2 3 6 9 7	1 2 3 5 6 9 7	1 2 3 5 6 9 7	1 2 3 5 6 9 7	1 2 3 5 6 9 7
3 5 6 8 9	2 4 5 8	2 3 4 5 9	1 7 9	2 3 4 5 6 8 9				
1 5 6 7 9	1 4 5 9	1 4 5 9	8 7 9	1 4 5 6 7 9	1 4 5 6 7 9	3 4 5 6 9	2 4 5 6 7	1 3 4 7 9
1 5 6 7	3 1 4 9	1 4 6 7 9	3 4 9	8 7 9	2 8 9	5 7 9	5 7 9	1 3 4 7 9
1 5 6 7	3 4 5 9	1 2 3 4 5 7	2 3 4 9	1 2 3 6 7	1 2 3 4 5 6 7	8 7	1 2 3 4 5 6 7	1 2 3 4 5 6 7
1 5 8 7	3 7 6 4	1 2 3 4 5 8	2 4 8	1 2 3 4 5 8	9 7	1 2 4 5 8	1 2 4 5 8	1 2 4 5 8
2 7	7 4 5 9 7 8 9	1 2 3 4 5 9 7 8 9	5 5	1 2 3 4 5 6 7 8 9				
1 5 6 7 8 9	3 1 2 4 5 6 8	1 2 3 4 5 9 7 8 9	1 4 9	1 2 3 4 6 9 7 8 9	1 2 3 4 5 6 7 8 9			
1 5 6 7 8 9	3 1 2 4 5 6 8	1 2 3 4 5 9 7 8 9	4 4 5 9 7 8 9	1 2 3 6 9 7 8 9	6 7 8 9	4 7 8 9	4 7 8 9	4 7 8 9

◀ Back to Start

► Skip Animation

# Propagation Steps (Domain Consistency)

4	1 2 5 6	8	2 3 5 7 9	2 3 6 9 7	1 2 3 5 6 9 7	1 2 3 5 6 9 7	1 2 3 5 6 9 7	1 2 3 5 6 9 7
3 5 6 8 9	2 4 5 6 8	2 3 4 5 9	1 7	2 3 5 6 9	2 3 4 5 6 8 9	2 3 4 5 6 8 9	2 3 4 5 6 8 9	2 3 4 5 6 8 9
1 5 6 7 9	1 4 5 6 9	1 4 5 9 7 9	8	1 5 6 7 9 7 9	1 4 5 6 9	3	2	
1 5 6 7 9	3 1 4	1 6 7 9	3 4 9	8	2 5	2 5	1 3 4 7 9	
1 5 6 7	3 9	1 2 3 4 5 7	2 3 4 5 9	2 3 4 6 7	1 2 3 5 6 7	1 2 3 4 5 6 8	1 2 3 4 5 6 7	
1 5 6 8	3 7	7 6 4	2 5	1 2 5	9	1 2 4 5 8	1 2 4 5 8	
2 7	1 2 3 4 5 9 7 8 9	2 3 4 5 9 7 8 9	5	1 2 3 5 6 7 9	1 2 3 4 5 6 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	
1 5 6 7 8 9	3 1 2 4 5 6 8	1 2 3 4 5 9 7 8 9	2 3 4 5 9 7 8 9	1 2 3 4 6 9 7 9	1 2 3 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	
1 5 6 7 8 9	3 7	1 2 3 4 5 9 7 8 9	2 3 4 5 9 7 8 9	6	1 2 3 4 5 6 7 8 9	4		

◀ Back to Start

► Skip Animation

# Propagation Steps (Domain Consistency)

4	1 2 5 6	8	2 3 5 7 9	2 3 6 9 7	1 2 3 5 6 9 7	1 2 3 5 6 9 7	1 2 3 5 6 9 7 9
3 5 6 8 9	2 4 5 8	2 3 4 5 9	1 7	2 3 5 6 9	2 3 4 5 6 8 9	2 3 4 5 6 8 9	2 3 4 5 6 8 9
1 5 6 7 9	1 4 5 9	1 4 5 9	8	1 5 6 4 5 6 7 9 7 9	1 3	1 3	2
1 5 6 7 9	3 1 4	6	3 4 7 9	3 8 9	2 8 2 7 9	5 4 7 9	1 3 4 7 9
1 5 6 7	3 4 5 7	1 2 3 4 5 7	2 3 4 6 7	1 2 3 5 6 4 5 6 7 7	1 2 3 8	1 2 3 4 5 6 7	1 2 3 4 5 6 7
1 5 8	3 7	7	6 4	2 1 2 5	9 9	1 2 4 5 8	1 2 4 5 8
2 7	1 3 4 9	1 3 4 8 9	3 5	1 3 1 6 4 9	3 1 6 4 8 9	3 1 6 4 8 9	1 3 6 4 8 9
1 5 6 7 8 9	1 2 4 5 6 8	1 2 3 4 5 9 7 8 9	2 3 4 6 9 7 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9
1 5 6 7 8 9	3 1 2 4 5 6 8	1 2 3 4 5 9 7 8 9	2 3 4 6 9 7 9	1 2 3 5 6 9	6 7 8 9	1 2 3 4 5 6 7 8 9	4

◀ Back to Start

► Skip Animation

# Propagation Steps (Domain Consistency)

4	1 2 5 6	8	2 3 5 7 9	2 3 6 9 7	1 2 3 5 6 9 7	1 5	3 7	1 2 3 5 6 7 9	1 2 3 5 6 7 9
3 5 6 8 9	2 4 5 6 8	2 3 4 5 9	1 7	2 3 5 6 9 8	3 4 5 6 8 9	2 3 4 5 6 8 9	2 3 4 5 6 8 9	2 3 4 5 6 8 9	2 3 4 5 6 8 9
1 5 6 7 9	1 4 5 6 9	1 4 5 7 9	8	1 5 6 4 5 7 9 7	1 3 7	3 2	1 3 4 7 9	1 3 4 5 6 7	2
1 5 6 7 9	3 1 4	6	3 4 7 9	3 4 6 9	8 5 6 4 5 7 7 7	2 5	1 3 4 7 9	1 3 4 5 6 7	3
1 5 6 7	3 4 5	9	1 2 3 4 5 7	2 3 4 6 7	1 2 3 5 6 4 5 7 7 7	3 8	1 2 3 4 5 6 7	1 2 3 4 5 6 7	1 2 3 4 5 6 7
1 5 6 8	3 4	7	6	2 4	1 2 5	9	1 2 4 5 8	1 2 4 5 8	1 2 4 5 8
2 7	1 3 4	7	3 4 9	5	1 6 4 9 8	9	1 3 4 6 4 8 9	1 3 4 6 4 8 9	1 3 4 6 4 8 9
1 5 6 7 8 9	1 2 4 5 6 8	1 2 3 4 5 9 7 8 9	2 3 4 5 9 7 9	1 2 3 4 6 9 7 9	1 2 3 4 5 6 7 8 9	1 6	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9
1 5 6 7 8 9	3 1 2 4 5 6 8	1 2 3 4 5 9 7 8 9	2 3 4 5 9 7 9	2 3 4 6 9 7 9	1 2 3 4 5 6 7 8 9	6 4	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	4

◀ Back to Start

► Skip Animation

# Propagation Steps (Domain Consistency)

4	1 2 5 6	8	2 3 5 7 9	2 3 6 9 7	1 2 3 5 6 9 7	1 5 7	1 2 3 5 6 7 9	1 2 3 5 6 9 7 9
3 5 6 8 9	2 4 5 6 8	2 3 4 5 9	1 7	2 3 5 6 9 8	3 4 5 6 8 9	2 3 4 5 6 8 9	2 3 4 5 6 8 9	2 3 4 5 6 8 9
1 5 6 7 9	1 4 5 6 9	1 4 5 7 9	8	1 5 6 7 9	1 4 5 7	3 4	2 4	2 4
1 5 6 7 9	3 1 4	6	3 4 7 9	3 4 9	8 2	2 5	1 3 4	1 3 4
1 5 6 7	3 4 5	1 2 3 4 5 7	2 3 4 5 6	2 3 5 6 7	1 2 3 4 5 7	8 7	1 2 3 4 5 6 7	1 2 3 4 5 6 7
1 5 6 8	3 4	7	6	2 4	1 2 5	9 4 5 8	1 2 4 5 8	1 2 4 5 8
2	7	1 3 4 9	3 4 8 9	5	1 3 6 4 9 8	1 3 4 6 4 8 9	1 3 4 6 4 8 9	1 3 4 6 4 8 9
3 5 6 7 8 9	2 5 6 8	2 3 5 9 7 8 9	2 3 5 9 7 8 9	1 4	2 3 5 6 7 8 9	3 5 6 7 8 9	2 3 5 6 7 8 9	2 3 5 6 7 8 9
1 5 6 7 8 9	3 1 2 4 5 6 8	1 2 3 4 5 9 7 8 9	2 3 4 6 9 7 9	1 2 3 5 6 9 7 9	6 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	4 7 8 9	4 7 8 9

◀ Back to Start

► Skip Animation

# Propagation Steps (Domain Consistency)

4	1 2 5 6	8	2 3 5 7 9	2 3 6 9 7	1 2 3 5 6 9 7	1 5	3 7	1 2 6 7 9	1 2 3 5 6 7 9
3 5 6 8 9	2 4 5 6 8	2 3 4 5 9	1 7	2 3 5 6 9 8	3 4 5 9	2 4 6 9	2 3 4 5 6 8 9		
1 5 6 7 9	1 4 5 6 9	1 4 5 9 7 9	8	1 5 6 4 5 7 9 7	1 3 4 5 7 7	3 2	1 2 3 4 5 6 7	2	
1 7 9	3 1 4	1 6 7 9	3 4 9	3 8 9	2 8 2 5 7 7	5	1 3 4 7 9	1 2 3 4 5 6 7	
1 5 6 7	3 9 4 5 7	1 2 3 4 5 7	2 3 4 6 7	1 2 3 5 6 4 5 7 7	1 3 8	1 2 3 4 5 6 7	1 2 3 4 5 6 7		
1 5 8	3 7 4 9	7 6 8 9	6 4	1 2 5	9	1 2 4	1 2 4 5 8	1 2 4 5 8	
2	7 4 9	1 3 4 9	3 4 8 9	5	1 3 1 3 6 4 9 8	1 4	1 3 4 6 9	1 3 4 6 8 9	
3 5 6 7 8 9	2 5 6 8 9	2 3 5 9 7 8 9	2 3 5 9 7 8 9	1 4 9 7 9	3 5 7 8	2 6 7 9	2 3 6 5 6 7 8 9		
1 5 6 7 8 9	3 1 2 4 5 6 8	1 2 3 4 5 9 7 8 9	2 3 4 6 9 7 8 9	2 3 1 2 3 5 6 9 7 9	6 4 7 9	1 2 4 6 7 9	1 2 3 4 5 6 7 9	4	

◀ Back to Start

► Skip Animation

# Propagation Steps (Domain Consistency)

4	1 2 5 6	8	2 3 5 7 9	2 3 6 9 7	1 2 3 5 6 9 7	1 5	3 7	1 2 6 7 9	1 2 3 5 6 7 9
3 5 6 8 9	2 4 5 6 8	2 3 4 5 9	1 7	2 3 5 6 9 8	3 4 5 9	2 4 6 9	2 3 4 5 6 8 9		
1 5 6 7 9	1 4 5 6 9	1 4 5 7 9	8	1 5 6 4 5 7 9 7	1 5 6 4 5 7 9 7	3 4 5 9	2 4 5 7 9		
1 5 6 7 9	3 1 4	1 6 7 9	3 4 4 7 9	3 4 6 9	8 5 6 4 5 7 9 7	2 4 5 9	5 4 5 7 9	1 3 4 7 9	
1 5 6 7	3 4 5 7	1 2 3 4 5 7	2 3 4 5 7	2 3 1 2 3 4 6 5 6 4 5 7 7 7	1 3 5 6 4 5 7 7 7	8 4 5 6 7	1 2 3 4 5 6 7		
1 5 8 7	3 7 9	7 6 4	6 7 9 4	2 1 2 5	9 4 5 9	1 2 4 9	1 2 4 5 8	1 2 4 5 8	
2 7	1 3 4 9	1 3 4 8 9	3 5	1 6 4 9 8	1 3 1 3 6 4 4 9 8 9	1 4 6 4 9 9 8	1 3 4 6 4 9 8 9	1 3 4 6 4 9 8 9	
3 5 6 7 8 9	2 5 6 8	2 3 5 9 7 8 9	2 3 5 9 7 8 9	2 3 1 2 3 5 9 7 9	3 5 7 8	2 6 7 9	2 3 5 6 7 8 9		
1 5 7 8 9	3 1 2 5 5 8	1 2 3 5 9 7 8 9	2 3 5 9 7 8 9	2 3 1 2 3 5 9 7 9	6 7 9	1 2 7 9	4 7 9		

◀ Back to Start

► Skip Animation

# Propagation Steps (Domain Consistency)

4	1 2 5 6	8	2 3 5 9	2 3 6 9	1 2 3 5 6 9	1 3 5 7	1 2 6 7	1 3 5 9
3 5 6 8 9	2 3 4 5 8	1 7	2 3 5 6 9 8	3 4 5 9	2 3 4 6 9	2 3 5 6 8 9	2 3 4 6 9	2 3 5 6 8 9
7	1 1 4 5 6 9	4 5 7 9	8	1 1 5 6 4 5 7 9 7	3	2	3	2
1 3 7 9	4 6 7 9	4 5 7 9	8	2 5 7 9	1 3 7 9	1 3 7 9	1 3 7 9	1 3 7 9
1 3 5 6 7	9	1 2 3 4 5 7	2 3 4 6 7	1 2 3 5 6 4 5 7 7	8	1 2 5 4 5 7	1 3 5 6 7	1 3 5 6 7
1 5 8	3 7 6	4 5 9 8 9	2 4	1 2 5 4 5 9	9	1 2 4 4 5 7	1 3 5 8	1 3 5 8
2 7	1 3 3 4 4 9	5	1 3 3 6 4 9	1 3 3 6 4 8	1 3 1 6 4 6 9	1 3 1 6 4 6 9	1 3 6 6 4 6 9	1 3 6 6 4 6 9
3 5 6 7 8 9	2 2 5 5 6 9	2 3 5 8 9	1 7 8 9	2 3 1 2 3 5 7 9	3 5 6 7 8 9	2 6 6 7 9 7 8 9	3 5 6 7 8 9	3 5 6 7 8 9
1 3 5 7 8 9	1 2 5 5 5 9	1 2 3 5 2 3 9 7 8 9	2 3 9 7 9	1 2 3 5 5 9	6	1 2 7 7 9	4	1 2 7 7 9

◀ Back to Start

► Skip Animation

# Propagation Steps (Domain Consistency)

4	1 2 5 6	8	2 3 5 7 9	2 3 6 9 7	1 2 3 5 6 9 7	1	3	1 2 6 7 9	1 3 5 6 7 9
3 5 6 9	2 5 6 9	1	7	2 3 5 6 4 5 9 8	3	2	3	4 6 9	5 6 8 9
7	1 5 6 9	4	8	1 5 6 4 5 9	1	3	2	5	1 3 7 9
1 9	4 6	7 9	3 9	3 9	8	2	5	1 7 9	3
1 5 6	9	1 2 3 4 5 7	2 3 4 5 6	2 3 4 6 5 6 7	1 2 3 5 6 4 5 7	1 3 8	1 3 5 6 7	1 5 6 7	1 3
1 5 8	3	7	6	4	2 1 2 5	9	1 2 4	1 5 8	1 5 8
2	7	1 3 4 9	3 8 9	5	1 6 4 9 8	3	1 4 9	1 3 6 9	1 3 6 8 9
3 5 6 8 9	2 5 6 8	2 3 5 9 7 8 9	2 3 5 9 7 8 9	1	4 5 7 8	3 5 7 8	2 6 7 9	3 6 7 8 9	3
1 5 8	1 2 5 9	1 2 3 5 9 7 8 9	2 3 5 9 7 9	2 3 5 9 7 9	1 2 3 5 9 7 9	6	1 2 7 9	4	1 3

◀ Back to Start

► Skip Animation

# Propagation Steps (Domain Consistency)

4	1 2 5 6	8	2 3 5 9	2 3 6 9	2 3 5 6	1 5 9 7	3 7 7 9	1 2 6 9	1 3 5 6
3 5 6 9	2 5 6 9	2 3 5 9	1 7	2 3 5 6	4 5 9 8	3 5	2 4 6 9	3 5 6 8 9	
7	1 5 6 9	1 5 9	4 8		1 5 9	3 5	2 3 5	2 5 7 9	
1 9	3 4	6 7 9	3 9	3 9	8 7 9	2 7	5 8	1 3 7 9	
1 5 6	3 9	1 2 3 4 5	2 3 5	2 3 4 6	1 2 3 5 6	1 3 4 5	3 8	1 3 5 6 7	
1 5 8	3 7	7 9	6 8 9	2 5	1 2 5	9 9 8	1 2 4	1 3 5 8	
2	7 4	1 3 9	3 8 9	5 7 8 9	1 7 8	3 5	1 4	1 3 6 9	6 8 9
3 5 6 8 9	2 5 6 8	2 3 5	2 3 7 8 9	1 9 7 8 9	2 3 5	3 7 8	2 7	3 6 7 8 9	
1 5 8	3 5 8	1 2 5	1 2 3 9 7 8 9	2 3 9 7 9	1 2 3 5	6 9	1 2 7	4 9	

◀ Back to Start

► Skip Animation

# Propagation Steps (Domain Consistency)

4	1 2 5 6	8	2 3 5 9	2 3 6 9	2 3 5 6	1 5	1 7	1 9	1 6 5 6 9
3 5 6 9	2 5 6 9	2 3 5 9	1 7	2 3 5 6	4 5 9 8	4 6	4 9	5 6 8 9	
7	1 5 6 9	1 5	4 9	8	5 6 9	1 5	3 7	2 9	
1 4	6 7	3 9	3 9	8	2 7	5 7	1 3 9		
5 9	2 7	2 3 5 4	2 3 6 7	1 2 3 5 6 7	1 4 5 7	3 8	1 5 6 7		
8 3	7 6	4 8	2 5	1 2 5	9 4	1 2 4	1 5 8		
2 7	4 8	3 8 9	5 9	1 6 9	3 4 8	1 4 9	1 6 9	3 6 8 9	
3 5 6 8 9	2 5 6 8	2 3 5 7	2 3 8 9	1 2 3 9 7	3 5 7 8	2 6 7 9	3 6 7 8 9		
1 5 8	3 5 9	1 2 5 9	1 2 3 5 7 8 9	2 3 1 2 3 9 7 9	6 5	1 2 7 9	4 9		

◀ Back to Start

► Skip Animation

# Propagation Steps (Domain Consistency)

4	2	8	5	6	3	1	1 7	1 9	5 6 9
3 6 9	5	3 5 9	1	7	2	4	6	8	
7	6	1	4	8	9	5	3	2	
1	4	6	7 9	3 9	3 9	8	2	5	3 7 9
5	9	2	7 3	4	1	1 4 7	3	8	6
8	3	7	6	2	5	9	4	1	
2	7	4	8 9	5	6	8	1	1 3 6 8 9	
6	8	2	1	4	5 7 8	3	2	6 9	5
3 9	1	5	8	2 3 9	7	6	2	4	

◀ Back to Start

► Skip Animation

# Propagation Steps (Domain Consistency)

4	2	8	5	6	3	1	7	9	7	9
3	5	3	1	7	2	4	6	8		
9	9									
7	6	1	4	8	9	5	3	2		
1	4	6	7	9	9	8	2	5	7	9
5	9	2	7	3	4	1	7	3	8	6
8	3	7	6	2	5	9	4	1		
2	7	4	3	9	5	6	8	1		3
6	8	3	2	1	4	7	3	7	9	5
3	1	5	8	3	7	6	2	4		
9			9							

◀ Back to Start

► Skip Animation

# Propagation Steps (Domain Consistency)

4	2	8	5	6	3	1	7	9	7	9
3	5	3	1	7	2	4	6	8		
9	9									
7	6	1	4	8	9	5	3	2		
1	4	6	7	9	9	8	2	5	7	3
5	9	2	7	3	4	1	7	3	8	6
8	3	7	6	2	5	9	4	1		
2	7	4	3	9	5	6	8	1	9	3
6	8	2	1	4	7	3	7	9	5	
3	1	5	8	9	7	6	2	4		

◀ Back to Start

► Skip Animation

# Propagation Steps (Domain Consistency)

4	2	8	5	6	3	1	7	9	7	9
3	5	3	1	7	2	4	6	8		
9	9									
7	6	1	4	8	9	5	3	2		
1	4	6	7	9	9	8	2	5	7	3
5	9	2	7	3	4	1	7	3	8	6
8	3	7	6	2	5	9	4	1		
2	7	4	3	9	5	6	8	1	9	3
6	8	2	1	4	7	3	7	9	5	
3	1	5	8	9	7	6	2	4		

◀ Back to Start

► Skip Animation

# Propagation Steps (Domain Consistency)

4	2	8	5	6	3	1	7	9	7	9
3	5	3	1	7	2	4	6	8		
9	9									
7	6	1	4	8	9	5	3	2		
1	4	6	7	9	9	8	2	5		3
5	9	2	3	4	1	7	3	8	6	
8	3	7	6	2	5	9	4	1		
2	7	4	3	5	6	8	1		3	9
6	8	3	2	1	4	7	3	7	9	5
3	1	5	8	3	7	6	2	4		
9			9							

◀ Back to Start

► Skip Animation

# Propagation Steps (Domain Consistency)

4	2	8	5	6	3	1	7	9	7	9
3 9	5	3 9	1	7	2	4	6	8		
7	6	1	4	8	9	5	3	2		
1	4	6	7 9	3 9	3 9	8	2	5	7	3
5	9	2	7 9	4	1	7	3	8	6	
8	3	7	6	2	5	9	4	1		
2	7	4	3 9	5	6	8	1	3 9		
6	8	3 9	2	1	4	7	3 9	7 9	5	
3 9	1	5	8	3 9	7	6	2	4		

◀ Back to Start

# After Setup (Domain Consistency)

4	2	8	5	6	3	1			
3 9	5	3 9	1	7	2	4	6	8	
7	6	1	4	8	9	5	3	2	
1	4	6	7 9	3 9	3	8	2	5	3 7
5	9	2	7 9	4	1	7	3	8	6
8	3	7	6	2	5	9	4	1	
2	7	4	3 9	5	6	8	1		3 9
6	8	3 9	2	1	4	7 9	7 9	5	
3 9	1	5	8	3 9	7	6	2	4	

# Comparison

Forward Checking

4	1 2	8	2 3	3	2 3	1	1	7	6	5 6
3	2	5 8	3	9	6	5	7	7	6	8
6	5 8	5	1	7	2 3	6	4 5	8	5 6	9
9	8	9	9	9	9	8	9	8 9	8 9	9
1	6	5 6	5	4 5	8	1	6 5	3	2	3
7	9	9	9	9	9	7	7	7	6	6
1	4	6	7	9	9	8	2	5	7	3
5	9	2	4	7	7	7	7	8	7	3
8	3	7	6	2	5	9	4	1	7	3
2	7	4	3	3	3	5 1	9 1	1	7	3
3	6	5 6	5	3	2 3	6	1	7	6	8
6	8	8	8	7	8 9	1	4	2	5	3
9	1	1	5	2	3	2	2	3	1	2
9	8	5	9	7	8 9	9	6	7	6	4

Bounds Consistency

4	1 2	8	5	6	2 3	1	1			
3	2	5	3	9	6	4 5	8	5 6	6	5
6	5	8	1	7	2 3	4 5	8	3	8 9	9
9	8	9	9	9	9	8	9	8 9	8 9	9
7	6	1	5	4	8	9	1	5	3	2
1	4	6	7	9	9	8	2	5	7	3
5	9	2	4	7	7	9	1	8	6	3
8	3	7	6	2	5	9	4	1	7	3
2	7	4	3	3	3	5 6	8	3	8 9	9
6	8	8	8	7	8 9	1	4	7	6	3
9	1	1	5	2	3	2	2	3	1	2
9	8	5	9	7	8 9	9	6	7	6	4

Domain Consistency

4	2	8	5	6	3	1	7	9	7	9
3	5	1	7	2	4	6	8	9	7	9
7	6	1	4	8	9	5	3	2	7	9
1	4	6	7	8	9	2	5	7	8	9
5	9	2	4	1	7	8	6	3	5	7
8	3	7	6	2	5	9	4	1	7	3
2	7	4	3	3	3	5 6	8	3	8 9	9
6	8	2	1	4	7	6	8	1	7	9
9	1	5	8	7	6	2	4	3	5	7

## Typical?

- This does not always happen
- Sometimes, two methods produce same amount of propagation
- Possible to predict in certain special cases
- In general, tradeoff between speed and propagation
- Not always fastest to remove inconsistent values early
- But often required to find a solution at all

# Outline

Problem

Initial Propagation (Forward Checking)

Improved Reasoning

Search

Other Global Constraints

## Simple search routine

- Enumerate variables in given order
- Try values starting from smallest one in domain
- Complete, chronological backtracking
- Advantage: Results can be compared with each other
- Disadvantage: Usually not a very good strategy

# Asking for Naive Search in MiniZinc

```
solve :: int_search(  
    puzzle,  
    input_order,  
    indomain_min)  
satisfy;
```

# Search Tree (Forward Checking)

2

4	1	2	3	4	5	6	7	8	9	1	2	3	4	5	6	7	8	9
3	2	5	6	8	9	5	6	7	8	2	3	4	5	6	7	8	9	5
6	5	6	5	9	9	1	7	8	9	6	4	5	6	7	8	9	8	9
9	7	8	9	1	2	3	4	5	6	7	8	9	4	5	6	7	2	3
1	6	5	6	1	2	3	4	5	8	1	2	3	4	5	6	7	3	2
7	9	8	9	9	9	9	9	9	9	7	8	9	7	8	9	7	8	9
1	4	6	8	9	7	8	9	2	5	1	4	6	7	8	9	2	5	3
5	9	2	8	7	8	9	3	4	5	7	8	9	7	8	9	7	8	6
8	3	7	2	1	2	3	4	5	6	3	4	5	6	7	8	9	4	1
2	7	4	5	3	4	5	6	7	8	1	2	3	4	5	6	7	8	3
7	8	9	8	9	9	8	9	7	8	9	8	9	7	8	9	7	8	9
3	6	5	6	5	3	2	3	4	5	3	1	2	3	4	5	6	7	3
9	8	7	8	9	9	7	8	9	1	4	5	6	7	8	9	7	8	9
3	1	5	6	1	2	3	4	5	6	3	2	3	4	5	6	7	8	4
9	8	7	8	9	9	7	8	9	9	7	8	9	7	8	9	7	8	4

# Search Tree (Forward Checking)

2  
1  
4

4	1	8	2 3	3	2 3	1	1	6	5 6
3	2	3	5	6	6	5	7	7	9 7
6	5 6	5	9	1	7	2 3	4 5	6	5 6
9		9				6	9 8	9	8 9
1	6	1	5	4 5	8	1	4 5	3	2
6	5 6	5	9	9		6	9 7	7	
7	9				3	3			3
1	4	6	7	8	8	2	5	7	
5	9	2	4	4	3	1	3	8	6
8	3	7	6	2	5	9	4	1	
2	7	1	3	3	3	1	3		3
		4			6				
		9	8 9	5	9	8	9	9	8 9
3		3	2 3		3	2	5		3
6	5 6	5	9 7 8 9	1	4	5	5		
9	8					7	8	7	9 7 8 9
3	1	1	3	2 3	3	2 3	1 2		
5		5					6	7	9 4
9	8	9	7 8 9	9	7	9	7	9	4

# Search Tree (Forward Checking)

2  
1  
4  
2  
5

4	1	8	2	3	2	3	5	6	5	6	5	6	5	6
3	2	3		6	6	6	5	6	4	5	6	4	5	6
6	5	6	5	9	1	7	7	8	9	8	9	8	9	8
9			9				7		7		7		7	
7	6	5	6	5	4	5	8	6	4	5	3	2	5	7
								1						
1	4	6		7	8	9	8	2	5					
5	9	2	4	7	3	4	3	1	3	3	8			3
							7	7			7			6
8	3	7	6	2	5	9	4	1						
2	7	1	3	3	4	5	3	1	3	1				3
		9	8	9	9	8	9	6	8	9				8
3		3	2	3				9	8	7	9	7	8	9
6	5	6	5	9	7	8	9	1	4	5	2	3	5	3
9	8							7	8	7	9	7	8	9
3		1	3	2	3		3	2	3	1	2			
5		5		9	7	8	9	8	7	9	6	7	9	4
9	8													

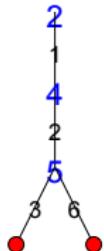
# Search Tree (Forward Checking)

2  
1  
4  
2  
5  
3



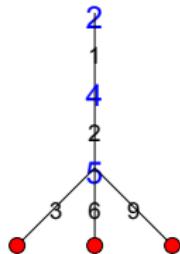
4	1	8	2	3	3	3	5	6	5	6	5	6
3	2	3			6	6	7	7	7	9	7	9
6	5	6	5	1	7	6	4	5	6	8	9	8
9	9	9	9			9	8	8	9	8	9	8
6	5	6	5	4	5	8	6	4	5	3	2	3
7	9	9	9	9	9	9	7	7	7	7	7	7
1	4	6			8	2	5					3
5	9	2	4	4	3	3	1	3	3	8	7	6
8	3	7	6	2	5	9	4	1				3
2	7	1	3	3	3	3	1	3	1			3
		4	9	8	9	5	6	8	9			3
3	6	5	3	3	3	1	4	5	2	5	1	3
6	5	6	5	9	7	8	9	7	8	7	9	8
9	8	9	9	9	8	9	8	7	9	7	8	9
3	5	1	3	3	3	3	2	3	1	2		4
6	5	5	5	9	7	8	9	9	7	9	7	9
9	8	9	9	9	8	9	9	7	9	7	9	4

# Search Tree (Forward Checking)



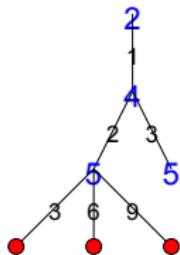
4	1	8	2	6	3	3	5	6	5	6	9
3	2	3		6	6	6	7	7	7	9	7
6	5	6	5	9	1	7	6	4	5	6	5
9			9				9	8	9	8	9
7	6	5	6	5	4	5	8	6	4	5	3
9				9	9	9	7		3	2	
1	4	6			3	3	8	2	5		7
5	9	2	4	4	3	3	1	3	3		3
7	7	7	7	7	7	7	7	7	8	7	6
8	3	7	6	2	5	9	4	1			
2	7	1	3	3	3	3	1	3	1		3
		4	9	8	9	5	6	8	9	8	9
3	6	5	6	5	3	3	1	3	2	3	3
9	8		9	7	8	9	9	8	7	9	8
3	5	6	5	5	1	4	5	5	1	2	4
9	8		9	7	8	9	9	7	9	7	9

# Search Tree (Forward Checking)



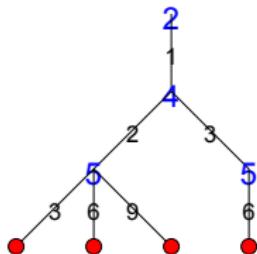
4	1	8	2	9	3	3	6	5	6	5	6
3	2	3	6	7	9	7	7	9	7	9	7
6	5	6	5	1	7	6	4	5	6	5	6
9	9	9	9	8	8	9	8	9	8	9	8
7	6	5	6	5	4	5	6	4	5	3	2
9	9	9	9	9	8	9	7	7	7	7	7
1	4	6	7	8	9	8	2	5	7	3	3
5	9	2	4	4	3	3	1	3	3	8	7
8	3	7	6	2	5	9	4	1	3	3	3
2	7	1	3	3	4	3	1	3	1	9	8
9	9	9	8	9	9	8	8	9	8	9	8
3	6	5	6	5	3	3	6	5	2	3	3
8	8	8	9	7	8	9	7	8	7	9	8
3	5	6	5	1	3	3	5	4	1	2	4
9	8	9	9	7	8	9	8	7	9	7	9

# Search Tree (Forward Checking)



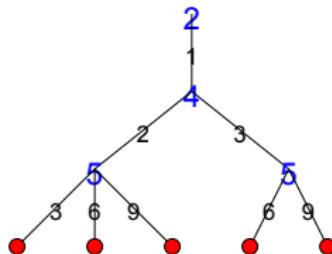
4	1	8	3	3	2	3	5	6	5	6	5	6	5	6
6	5	6	5	6	4	5	9	8	7	6	4	5	8	9
9	9	9	9	9	8	7	7	7	7	7	7	7	7	7
7	6	5	6	5	4	5	9	8	7	6	4	5	9	7
9	9	9	9	9	8	7	7	7	7	7	7	7	7	7
1	4	6	7	8	9	3	3	3	3	3	3	3	3	3
5	9	2	4	4	4	3	3	3	3	3	3	3	3	3
8	3	7	6	2	5	9	4	1	1	1	1	1	1	1
2	7	1	3	3	3	3	3	3	3	3	3	3	3	3
9	9	9	8	9	8	9	9	9	9	9	8	9	8	9
3	6	5	6	5	3	2	3	1	4	1	2	3	2	3
8	8	8	9	8	9	7	8	9	1	4	5	6	5	6
9	9	9	9	9	9	7	8	9	9	7	8	9	7	8
3	5	5	5	5	1	3	2	3	3	2	3	1	2	3
9	8	9	9	7	8	9	9	7	9	7	9	6	7	9
9	8	9	9	7	8	9	9	7	9	7	9	4	4	4

# Search Tree (Forward Checking)



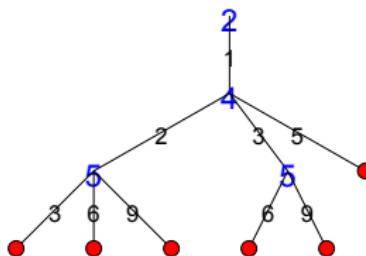
4	1	8	3	6	2	6	5	6	5	6
3	2	3	9	9	7	9	7	9	7	9
6	5	6	5	9	1	7	6	4	5	6
9	9	9	9	9	9	9	8	9	8	9
7	6	5	6	5	4	5	8	6	4	5
9	9	9	9	9	9	9	7	9	7	9
1	4	6	7	9	8	2	5	7	3	3
5	9	2	4	4	3	1	3	3	8	7
8	3	7	6	2	5	9	4	1	3	3
2	7	1	3	4	3	1	3	1	9	8
9	8	7	9	8	9	8	9	8	9	8
3	6	5	6	5	3	2	5	3	2	3
9	8	9	7	8	9	1	4	7	8	7
3	5	6	5	9	7	8	9	8	7	9
9	8	9	7	8	9	9	7	9	7	9
6	6	5	5	9	7	8	9	8	7	9
9	8	9	9	9	9	9	9	9	9	9

# Search Tree (Forward Checking)



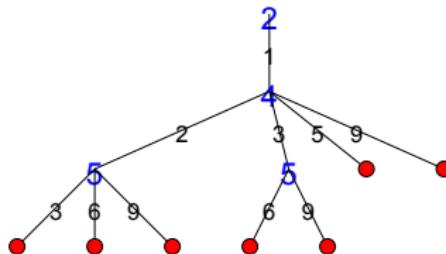
4	1	8	3	9	2	6	5	6	5	6	5	6
3	2	3	6	9	1	7	4	5	6	5	6	5
6	5	6	5	9	8	2	4	5	6	5	6	5
9	9	9	9	9	3	1	3	1	3	1	3	1
7	7	9	9	9	8	2	5	3	2	5	7	3
6	5	6	5	9	8	2	5	3	2	5	7	3
9	9	9	9	9	8	2	5	3	2	5	7	3
1	4	6	7	9	8	2	5	3	1	3	1	3
5	9	2	4	4	7	7	7	7	7	7	7	7
8	3	7	6	2	5	9	4	1	3	1	3	1
2	7	1	3	4	9	8	9	5	6	8	9	8
3	6	5	6	5	9	7	8	9	1	4	7	8
6	9	8	8	8	9	7	8	9	3	2	5	3
9	9	8	8	8	9	7	8	9	6	7	9	4

# Search Tree (Forward Checking)



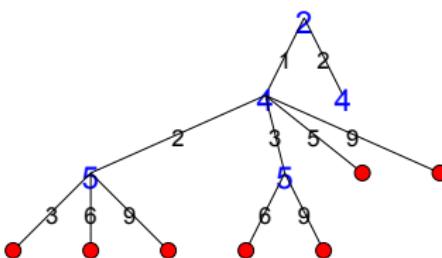
4	1	8	2 3	3	2 3	5	6	5	6	5	6
3	2	9	6	6	9	7	2 3	4 5	6	5	6
6	5 6	5	9	1	7	8	6 4 5	9 8	6	5	6
9							9 7		9	8	9
6	5 6	5	4 5	8			1	6 4 5	3	2	3
7	9	9	9	3	3		7	9 7	3	2	3
1	4	6	7	8	9	8	2	5	7		
5	9	2	4	3	1	3	7	7	8	7	6
8	3	7	6	2	5	9	4	1			
2	7	1	3	3	1	3	1	6	8	9	8 9
		4	9	8 9	5	5	6	9	8	9	3
3	6	5 6	5	3	2 3	1	4	5	2	5	3
9	8	9	9	7 8 9			7	8	7	9	8 9
3	5	1	3	2 3	3	2 3	6	1 2	7	9	4
9	8	9	7 8 9	9	7	9	6	7	9	8	4

# Search Tree (Forward Checking)



4	1	8	2 3 6 9	3 6 9	2 3 6 9	5	6	5 6 9
3	2	3			2 3 6 9	4 5 9	6	5 6 9
6	5 6	5			6 4 5 9	8	6	5 6 9
9	9	9			9 7	7	9	8 9
6	5 6	5 4 5 9	8		6 4 5 9	3	2	3
7	9	9			9 7			
1	4	6			1			
5	9	2			8 2 5 7			
8	3	7	6	2 5	9 4 1			
2	7	1 3 4	3	5	3 1 3 1 6			
3	6	5 6 8	3 2 3 9 8 9	1 4	9 8 9 7 8 7	2 9 8 9	3 2 5	3
9	8	9	7 8 9		7 8 7	9 7 8 9	6	7 9 4
3	5 6	5			1 2			
9	8	9	7 8 9		9 7 9			

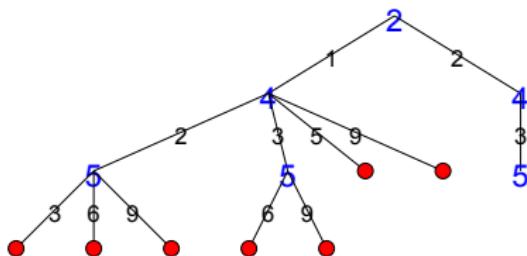
## Search Tree (Forward Checking)



A 9x9 Sudoku grid with the following values filled in:

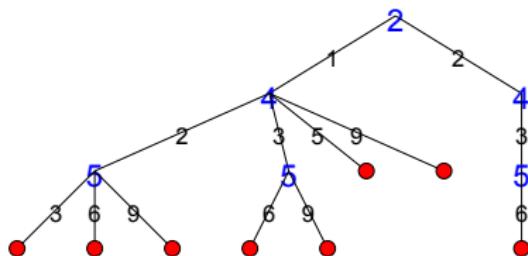
4	2	8	2	3	3	2	3	1
3	2	5	5	6	6	5	6	5
6	5	6	9	9	9	7	7	9
9	1	7	2	3	6	4	5	6
1	6	5	4	5	4	8	4	5
7	9	9	9	9	9	7	3	2
1	4	6	3	3	8	2	5	3
5	9	2	7	3	3	1	3	3
8	3	7	6	2	5	9	4	1
2	7	1	3	3	3	1	3	3
3	4	9	8	8	5	9	8	9
6	5	6	3	2	3	5	2	3
9	8	9	7	8	9	7	8	9
3	1	1	3	2	3	1	2	3
9	8	5	5	6	6	7	9	4
9	8	7	8	9	9	7	9	8

# Search Tree (Forward Checking)



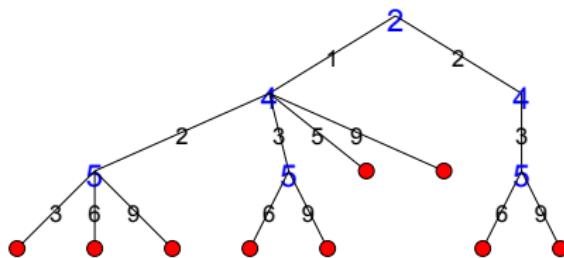
4	2	8	3	3	1	5	1	5	6
3	6	5	9	6	6	5	7	7	9
6	5	6	9	1	4	5	8	8	5
9	9	9	9	7	6	4	5	3	2
1	5	6	5	8	1	5	3	2	3
7	9	9	9	3	3	3	3	3	3
1	4	6	7	8	2	5	7	7	6
5	9	2	4	3	1	3	3	8	7
8	3	7	6	2	5	9	4	1	3
2	7	1	3	3	3	1	3	1	3
		4	7	8	6	6	8	9	8
3	6	5	5	9	8	9	7	7	9
9	8	8	9	7	8	9	7	7	8
3	1	1	3	2	3	3	2	3	3
5	5	5	9	7	8	5	5	1	2
9	8	9	9	7	8	9	7	9	4

# Search Tree (Forward Checking)



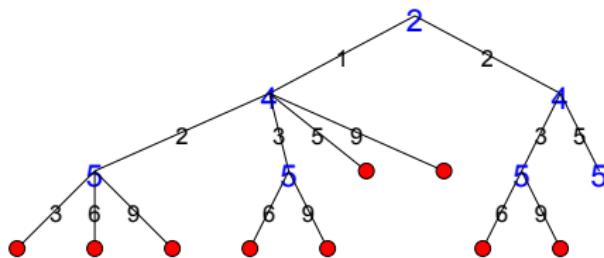
4	2	8	3	6		1	5	1		5	6
3	6	5	3	9	2	6	4	5	6	5	9
6	5	6	5	9	1	7	9	8	6	5	8
9	9	9	9	9	8	8	6	4	5	3	2
1	5	1	4	5	8	1	5	3	2	5	7
6	5	6	6	9	3	8	2	5	7	3	6
7	9	7	9	9	1	8	2	5	7	3	6
1	4	6	7	9	8	1	5	3	2	5	7
5	9	2	4	4	3	1	3	3	8	7	6
8	3	7	6	2	5	9	4	1	3	2	3
2	7	1	3	4	5	3	1	3	1	5	6
6	5	6	5	9	8	6	9	8	7	9	8
9	8	9	7	8	9	7	8	7	9	7	8
3	6	3	2	1	4	3	2	5	5	6	3
6	5	6	5	9	1	4	5	6	7	8	9
9	8	9	7	8	9	8	7	9	6	7	9
3	1	1	3	2	3	2	3	1	2	4	4
5	5	5	9	7	9	9	7	9	7	9	4
9	8	9	7	8	9	8	7	9	7	9	4

# Search Tree (Forward Checking)



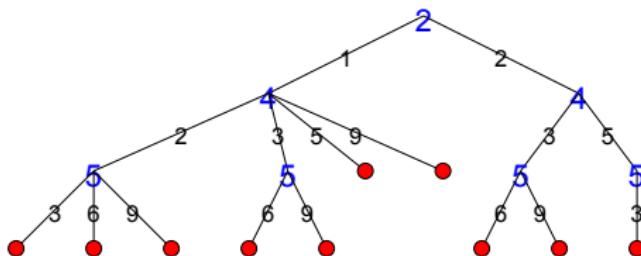
4	2	8	3	9	6	1	5	1	1
3	6	5	3	9	2	6	4	5	6
6	5	6	5	9	1	7	9	8	5
9	9	9	9	9	8	8	8	9	9
1	5	1	4	5	8	1	4	5	3
6	5	6	5	9	9	6	4	5	2
7	9	7	9	9	7	7	7	7	3
1	4	6	7	9	8	2	5	7	1
5	9	2	4	4	3	1	3	3	3
8	3	7	6	2	5	9	4	1	1
2	7	1	3	4	5	3	1	3	3
6	5	6	5	9	8	6	4	5	5
9	8	8	9	8	9	9	8	9	8
3	6	3	2	1	4	3	2	5	3
6	5	6	5	9	7	8	7	8	7
9	8	8	9	7	8	9	7	9	8
3	1	1	3	2	3	2	3	1	2
5	5	5	5	9	7	9	7	9	4
9	8	9	8	9	7	9	7	9	4

# Search Tree (Forward Checking)



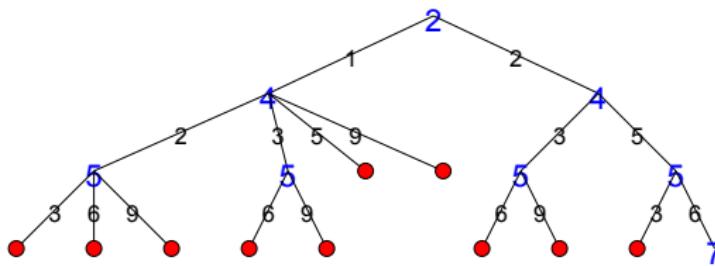
4	2	8	5	3	3	1	1	6	5	6	5	6	5	6
3	6	5	9	6	5	9	7	2	3	6	4	5	8	5
6	5	6	9	1	7	8	9	6	4	5	9	7	3	2
9	7	9	8	8	8	3	3	1	1	6	4	5	9	7
1	4	6	7	8	9	3	3	1	3	3	2	5	7	3
5	9	2	4	5	4	3	1	3	3	7	7	7	8	7
8	3	7	6	2	5	9	4	1	3	1	3	1	2	3
2	7	1	4	5	5	6	6	9	8	8	9	9	8	9
3	6	5	6	9	8	9	8	9	7	8	7	9	7	8
9	8	8	9	7	8	9	1	4	5	5	5	5	5	5
3	1	5	5	9	7	8	9	8	7	9	1	2	3	4
6	8	9	8	9	7	8	9	9	7	9	7	9	7	8

# Search Tree (Forward Checking)



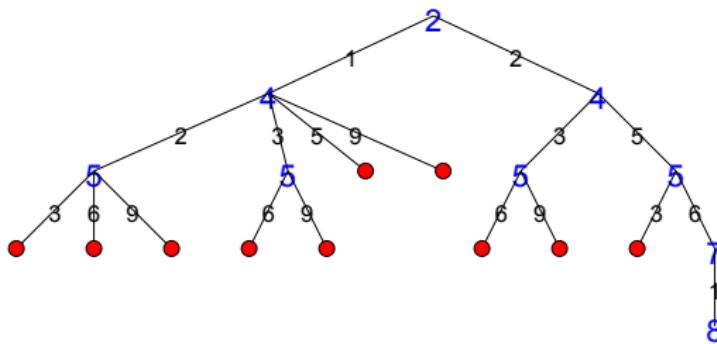
4	2	8	5	3	3	1	1	6	6	8
3	6	5	5	1	2	3	6	4	5	9
6	5	6	9	7	7	7	8	9	5	8
9	9	9	9	8	8	8	7	7	9	9
1	5	1	4	8	1	1	6	4	5	3
6	5	9	9	8	3	3	3	9	7	2
7	9	9	9	7	7	7	7	7	7	3
1	4	6	7	8	8	8	2	5	7	1
5	9	2	4	3	3	3	1	3	3	3
8	3	7	6	2	5	9	4	1	1	3
2	7	1	3	3	3	3	1	3	1	3
4	9	8	9	8	9	9	6	5	4	5
3	6	5	5	1	4	3	2	5	7	3
8	8	9	7	8	9	7	8	7	9	8
3	1	1	3	2	3	3	2	1	2	3
5	5	5	9	7	8	9	6	7	9	4
9	8	9	7	8	9	8	7	9	7	9

# Search Tree (Forward Checking)



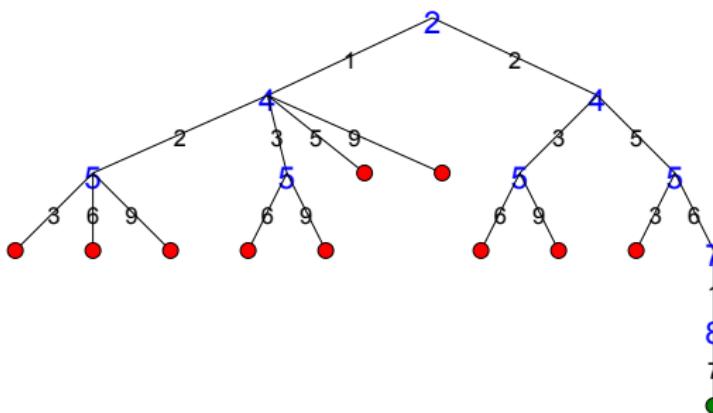
4	2	8	5	6	3	1	1	6	9
3	6	5	5	1	7	2	4	5	8
6	5	6	5	9	4	8	9	4	5
9	9	9	9	7	7	7	7	9	8
1	5	6	1	4	8	9	1	5	3
6	5	6	5	9	7	7	7	3	2
7	9	9	9	7	7	7	7	7	3
1	4	6	7	8	9	8	2	5	7
5	9	2	4	5	6	1	3	8	6
8	3	7	6	2	5	9	4	1	3
2	7	1	3	3	5	6	1	3	3
4	9	8	9	8	9	8	9	9	8
3	5	6	5	3	2	3	5	2	3
6	8	9	9	7	8	9	1	4	5
9	8	8	9	7	8	9	7	8	9
3	1	1	3	2	3	3	1	2	3
5	5	5	9	7	8	9	7	6	5
9	8	8	9	7	8	9	7	9	4

# Search Tree (Forward Checking)



4	2	8	5	6	3	1	7	9	7	9	7	9
3	6	5	6	5	9	1	7	2	4	5	6	5
6	1	5	6	1	5	4	8	9	1	5	3	2
7	6	5	6	5	9	3	3	3	7	8	7	3
1	4	6	7	8	9	8	2	5	7	8	6	3
5	9	2	7	3	4	1	7	8	7	8	7	6
8	3	7	6	2	5	9	4	1	3	1	2	3
2	7	1	4	3	3	5	6	7	8	9	8	5
3	6	5	6	5	9	8	9	1	4	5	2	3
9	8	5	6	5	9	8	9	7	8	7	9	7
3	1	5	6	5	9	8	9	7	6	1	2	4
9	8	5	6	5	9	8	9	8	7	6	9	4

## Search Tree (Forward Checking)



4	2	8	5	6	3	1	7	9
3	5	9	1	7	2	4	6	8
7	6	1	4	8	9	5	3	2
1	4	6	3	9	8	2	5	7
5	9	2	7	4	1	3	8	6
8	3	7	6	2	5	9	4	1
2	7	4	9	5	6	8	1	3
6	8	3	2	1	4	7	9	5
9	1	5	8	3	7	6	2	4

# Search Tree (Bounds Consistency)

2

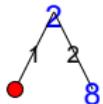
4	1 2	8	5	6	2 3	1	1		
3	2	3			2 3	4 5	6	5	9
	5		1	7		8	9	8	9
9		9							
7	6	4	8	9	1	5	3	2	
1	4	6	7	8	9	8	2	5	3
5	9	2	7	3	4	1	7	8	6
8	3	7	6	2	5	9	4	1	
2	7	4	8	9	5	6	1	3	1
6	5	3	2 3	8	9	8	9	8	9
8	9	9	8	9	1	4	5	2	3
3	1	1	2	3	7	8	7	9	7 8 9
5	5	5	8	9	7	6	1	2	4
9	8								

# Search Tree (Bounds Consistency)

2  
1

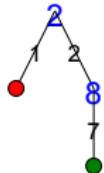
4	1	8	5	6	2	3	1	1		
3	2	3	1	7	2	3	4	5	6	5
9	5	9	1	7	7	8	9	8	9	9
7	6	5	4	8	9	1	5	3	2	3
1	4	6	7	8	9	8	2	5	7	
5	9	2	7	3	4	1	7	8	6	
8	3	7	6	2	5	9	4	1		
2	7	4	8	9	5	6	1	3	1	3
6	5	3	2	3	1	4	5	2	5	3
3	1	1	2	3	7	8	7	9	7	8
9	5	5	8	9	7	6	1	2	4	

# Search Tree (Bounds Consistency)



4	2	8	5	6	3	1	1		
3 9	5	3 9	1	7	2	4	6	5 8 9	
7	6	1	4	8	9	5	3	2	
1	4	6	7 8 9	3 9	8	2	5	7 3	
5	9	2	7 8 9	3 4	1	7 8	8	6	
8	3	7	6	2	5	9	4	1	
2	7	4	8 9	3 5	6	8	1	7 8 9	
6	8	2	1	4	7 8 9	3 2	5	7 8 9	
3 9	1	5	8	7 9	6	2	4		

# Search Tree (Bounds Consistency)



4	2	8	5	6	3	1	7	9
3	5	9	1	7	2	4	6	8
7	6	1	4	8	9	5	3	2
1	4	6	3	9	8	2	5	7
5	9	2	7	4	1	3	8	6
8	3	7	6	2	5	9	4	1
2	7	4	9	5	6	8	1	3
6	8	3	2	1	4	7	9	5
9	1	5	8	3	7	6	2	4

# Search Tree (Domain Consistency)

8

4	2	8	5	6	3	1			
3 9	5	3 9	1	7	2	4	6	8	
7	6	1	4	8	9	5	3	2	
1	4	6	7 8 9	3 9	8	2	5	7 3	
5	9	2	7	3 9	4	1	7 3	8	6
8	3	7	6	2	5	9	4	1	
2	7	4	3 9	5	6	8	1	7 9 3	
6	8	2	1	4	7	7 9	5		
3 9	1	5	8	3 9	7	6	2	4	

# Search Tree (Domain Consistency)

8  
7  
6

4	2	8	5	6	3	1	7	9
3	5	9	1	7	2	4	6	8
7	6	1	4	8	9	5	3	2
1	4	6	3	9	8	2	5	7
5	9	2	7	4	1	3	8	6
8	3	7	6	2	5	9	4	1
2	7	4	9	5	6	8	1	3
6	8	3	2	1	4	7	9	5
9	1	5	8	3	7	6	2	4

# Trading Propagation Against Search

- If we perform more propagation, search is more constrained
- Fewer values left, fewer alternatives to explore in search
- Best compromise is not obvious
- But can be learned from examples or during search
- Annotations are optional
  - Some MiniZinc back-end solvers do the search they want, not the one you specify
  - Some solvers simply do not work in a way that these search annotations apply

# Outline

Problem

Initial Propagation (Forward Checking)

Improved Reasoning

Search

Other Global Constraints

# Are there other Global Constraints?

- alldifferent is the most commonly used constraint
- Propagation methods can be explained
- But there are many more

# Global Constraint Catalog

- <https://sofdem.github.io/gccat/>
- Description of 354 global constraints, 2800 pages
- Not all of them are widely used
- Detailed, meta-data description of constraints in Prolog

# Families of Global Constraints

- Value Counting
  - alldifferent, global cardinality
- Scheduling
  - cumulative
- Properties of Sequences
  - sequence, no\_valley
- Graph Properties
  - circuit, tree

# Common Algorithmic Techniques

- Bi-Partite Matching
- Flow Based Algorithms
- Automata
- Task Intervals
- Reduced Cost Filtering
- Decomposition

# Part III

## Customizing Search

## What we want to introduce

- Importance of search strategy, constraints alone are not enough
- Two schools of thought
  - Black-box solver, solver decides by itself
  - Human control over process
- Dynamic variable ordering exploits information from propagation
- Variable and value choice
- Hard to find strategy which works all the time
- Different way of improving stability of search routine

# Example Problem

- N-Queens puzzle
- Rather weak constraint propagation
- Many solutions, limited number of symmetries
- Easy to scale problem size

# Outline

Problem

Program

Naive Search

Improvements

# Problem Definition

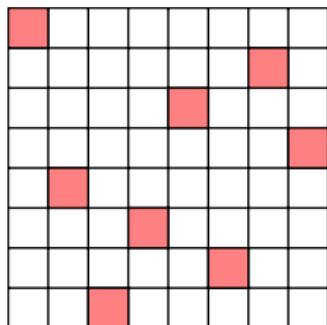
## 8-Queens

Place 8 queens on an  $8 \times 8$  chessboard so that no queen attacks another. A queen attacks all cells in horizontal, vertical and diagonal direction. Generalizes to boards of size  $N \times N$ .

# Problem Definition

## 8-Queens

Place 8 queens on an  $8 \times 8$  chessboard so that no queen attacks another. A queen attacks all cells in horizontal, vertical and diagonal direction. Generalizes to boards of size  $N \times N$ .



Solution for board size  $8 \times 8$

# Outline

Problem

Program  
Model

Naive Search

Improvements

# Basic Model

- Cell based Model
  - A 0/1 variable for each cell to say if it is occupied or not
  - Constraints on rows, columns and diagonals to enforce no-attack
  - $N^2$  variables,  $6N - 2$  constraints
- Column (Row) based Model
  - A 1..N variable for each column, stating position of queen in the column
  - Based on observation that each column must contain exactly one queen
  - $N$  variables,  $N^2/2$  binary constraints

# Model

assign  $[X_1, X_2, \dots, X_N]$

s.t.

$$\forall 1 \leq i \leq N : X_i \in 1..N$$

$$\forall 1 \leq i < j \leq N : X_i \neq X_j$$

$$\forall 1 \leq i < j \leq N : X_i + j \neq X_j + i$$

$$\forall 1 \leq i < j \leq N : X_i + i \neq X_j + j$$

# Outline

Problem

Program

Naive Search

Improvements

# Nqueens Models

- ECLiPSe [▶ Show](#)
- MiniZinc [▶ Show](#)
- NumberJack [▶ Show](#)
- CPMPy [▶ Show](#)
- Choco-solver [▶ Show](#)

# ECLIPSe N-Queens Model

```
:- lib(lists).
:- lib(ic).

top:-
    queens(8,Board),
    search(Board, 0, input_order, indomain, complete, []),
    writeln(Board).

queens(N, Board) :-
    length(Board, N),
    Board :: 1..N,
    ( fromto(Board, [Q1|Cols], Cols, []) do
        ( foreach(Q2, Cols), param(Q1), count(Dist, 1,_) do
            noattack(Q1, Q2, Dist)
        )
    ).

noattack(Q1,Q2,Dist) :-
    Q2 #\= Q1,
    Q2 - Q1 #\= Dist,
    Q1 - Q2 #\= Dist.
```

▶ Continue

# MiniZinc N-Queens Model

```
int: n=8;
array[1..n] of var 1..n: queens;
constraint
  forall(i, j in 1..n where i < j) (
    queens[i] != queens[j] /\ 
    queens[i] + i != queens[j] + j /\ 
    queens[i] - i != queens[j] - j
  )
;
solve :: int_search(
  queens,
  input_order,
  indomain_min)
satisfy;
```

» Continue

# NumberJack N-Queens Model

```
from Numberjack import *

def get_model(N):
    queens = VarArray(N, N)
    model = Model(
        AllDiff(queens),
        AllDiff([queens[i] + i for i in range(N)]),
        AllDiff([queens[i] - i for i in range(N)])
    )
    return queens, model

def solve(param):
    queens, model = get_model(param['N'])
    solver = model.load(param['solver'])
    solver.setHeuristic(param['heuristic'], param['value'])
    solver.setVerbosity(param['verbose'])
    solver.setTimeLimit(param['tcutoff'])
    solver.solve()
```

▶ Continue

# CPMy N-Queens Model

```
def nqueens_naive(n=8):
    queens = IntVar(1,n, shape=n)

    model = Model()
    for i in range(n):
        for j in range(i):
            model += [queens[i] != queens[j],
                      queens[i] + i != queens[j] + j,
                      queens[i] - i != queens[j] - j,
                      ]
```

► Continue

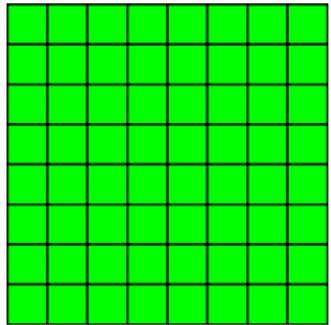
# Choco-solver N-Queens Program

```
int n = 8;
Model model = new Model(n + "-queens problem");
IntVar[] vars = new IntVar[n];
for(int q = 0; q < n; q++){
    vars[q] = model.intVar("Q_"+q, 1, n);
}
for(int i = 0; i < n-1; i++){
    for(int j = i + 1; j < n; j++){
        model.arithm(vars[i], "!=" ,vars[j]).post();
        model.arithm(vars[i], "!=" , vars[j], "-", j - i).post();
        model.arithm(vars[i], "!=" , vars[j], "+", j - i).post();
    }
}
Solution solution = model.getSolver().findSolution();
if(solution != null){
    System.out.println(solution.toString());
}
```

► Continue

# Default Strategy

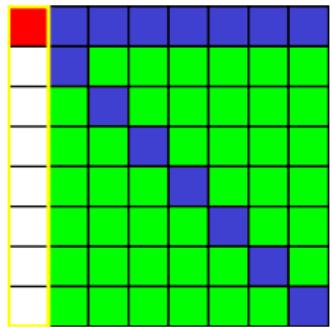
1



▶ Skip Animation

# Default Strategy

1  
1  
2

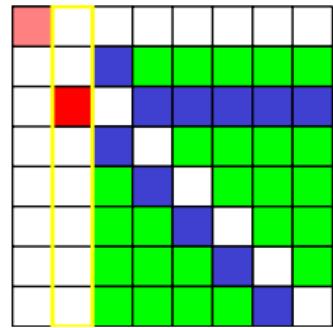


◀ Back to Start

► Skip Animation

# Default Strategy

1  
1  
2  
3  
3

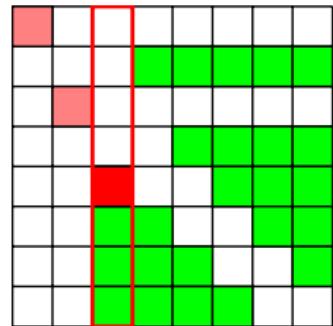


◀ Back to Start

► Skip Animation

# Default Strategy

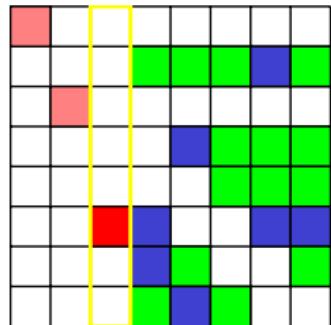
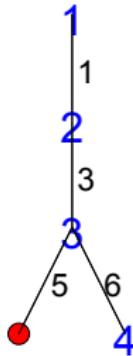
1  
1  
2  
3  
3  
5



◀ Back to Start

▶ Skip Animation

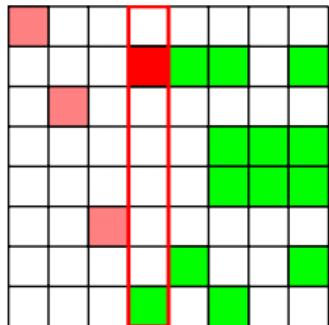
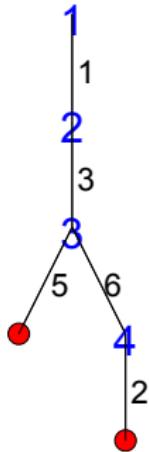
# Default Strategy



◀ Back to Start

► Skip Animation

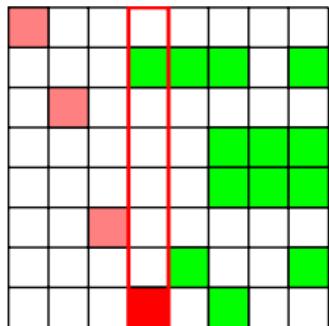
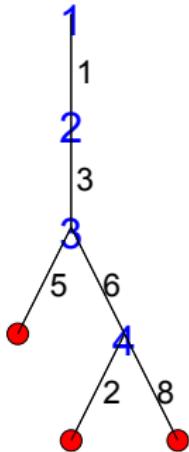
# Default Strategy



◀ Back to Start

► Skip Animation

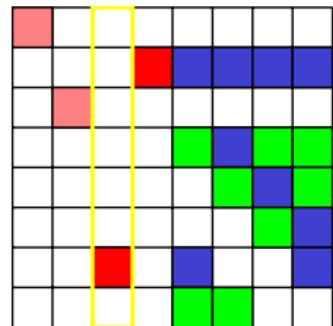
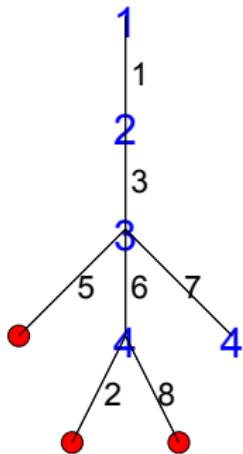
# Default Strategy



◀ Back to Start

► Skip Animation

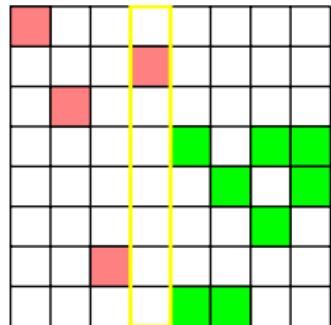
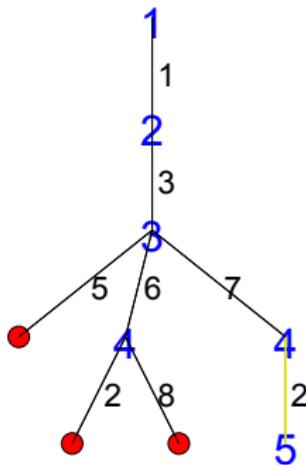
# Default Strategy



◀ Back to Start

► Skip Animation

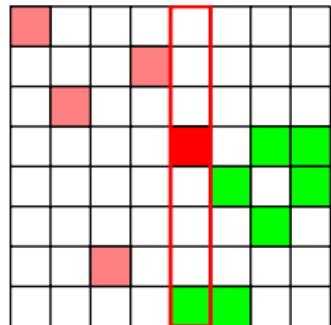
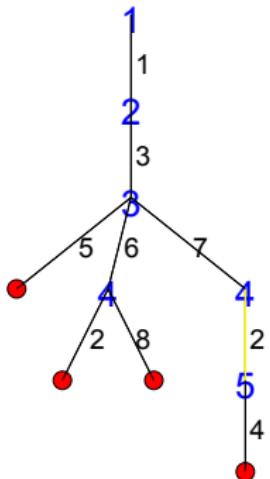
# Default Strategy



◀ Back to Start

► Skip Animation

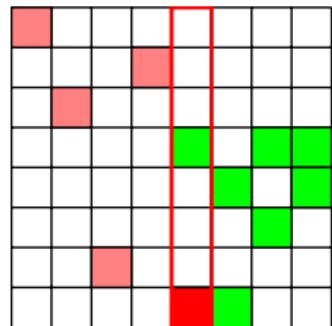
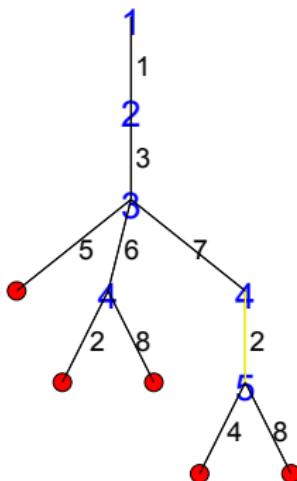
# Default Strategy



◀ Back to Start

► Skip Animation

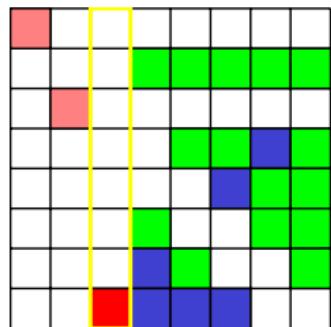
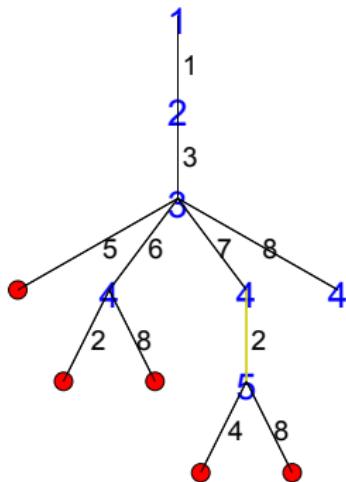
# Default Strategy



◀ Back to Start

► Skip Animation

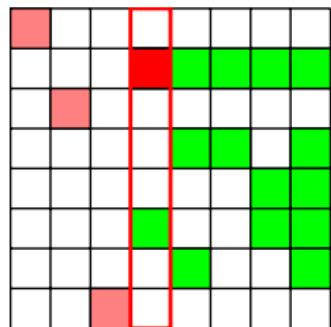
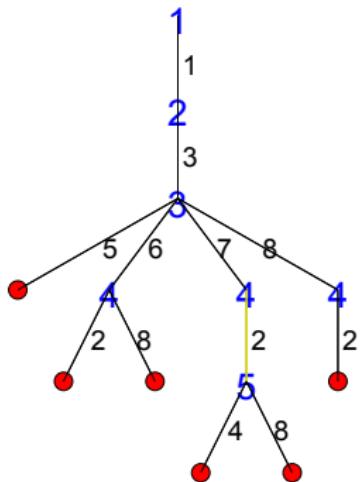
# Default Strategy



◀ Back to Start

► Skip Animation

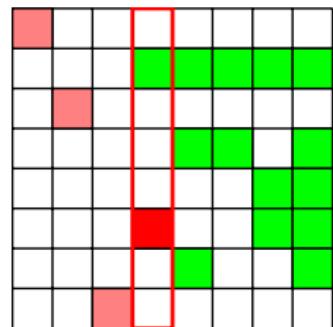
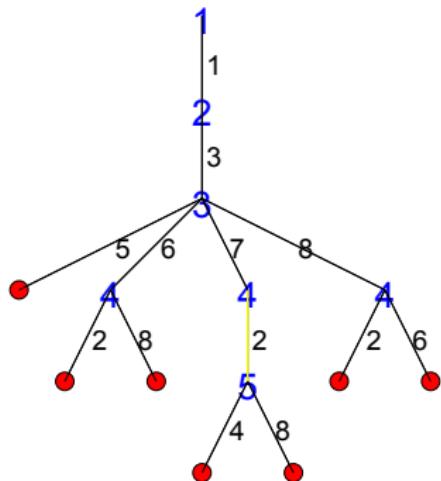
# Default Strategy



◀ Back to Start

► Skip Animation

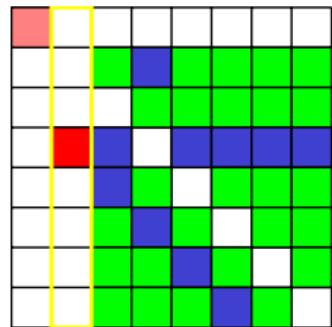
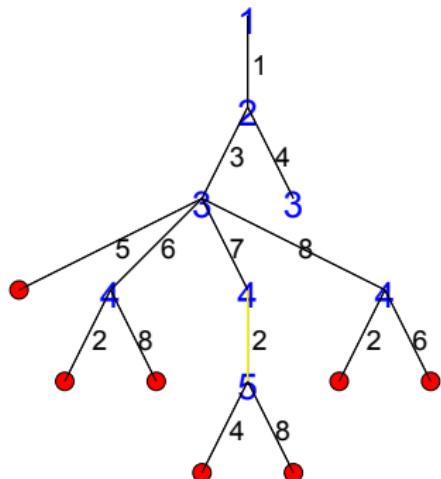
# Default Strategy



◀ Back to Start

► Skip Animation

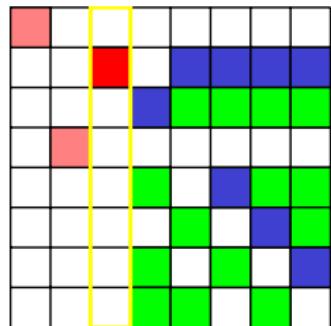
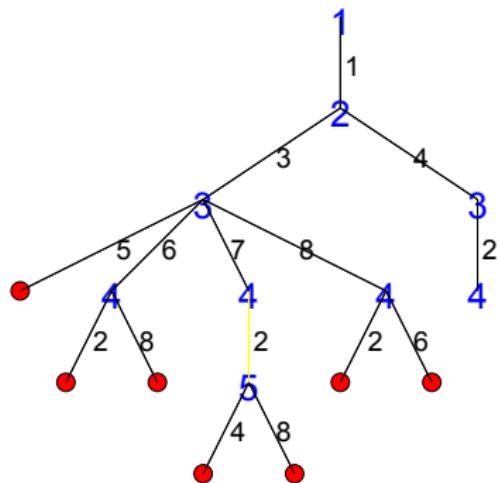
# Default Strategy



◀ Back to Start

► Skip Animation

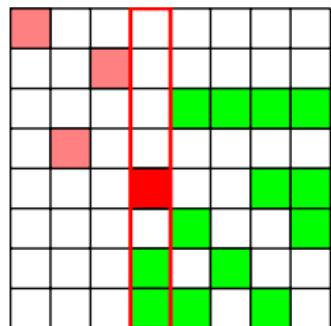
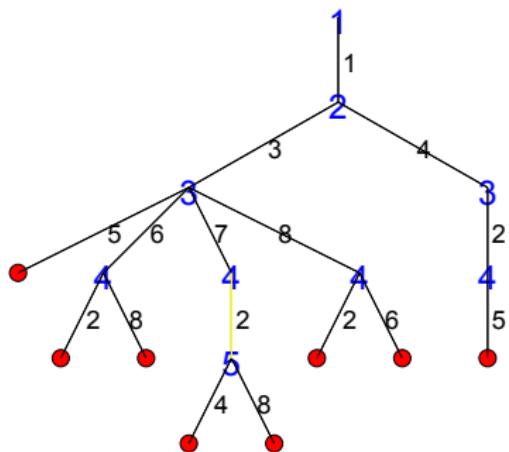
# Default Strategy



◀ Back to Start

► Skip Animation

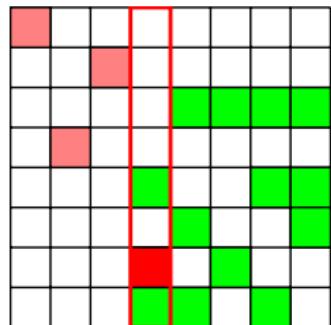
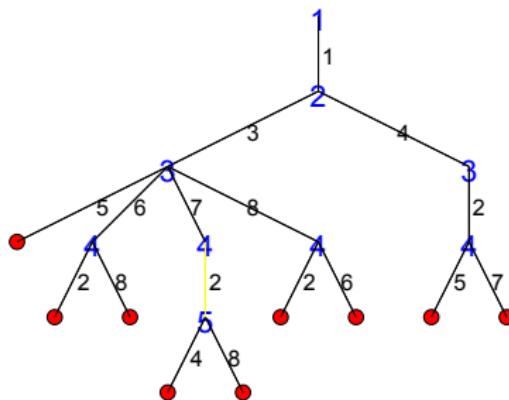
# Default Strategy



◀ Back to Start

► Skip Animation

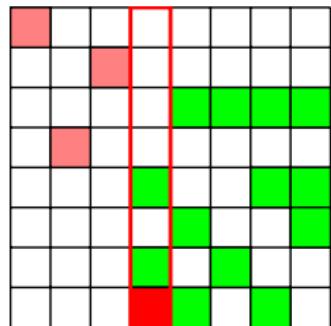
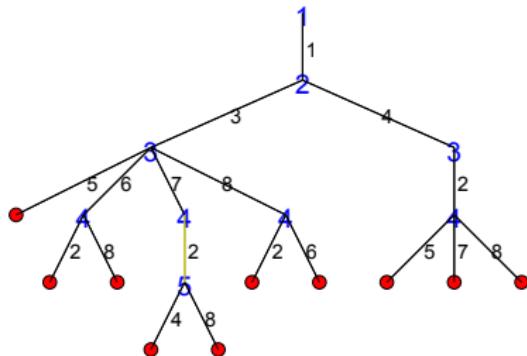
# Default Strategy



◀ Back to Start

► Skip Animation

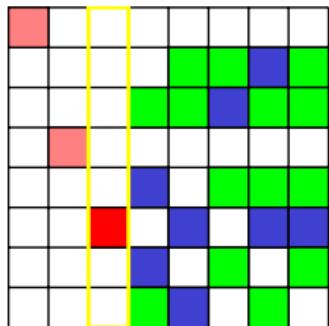
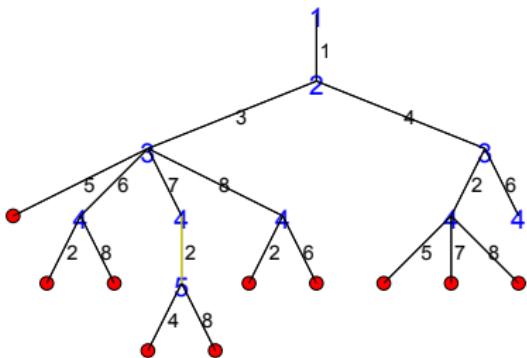
# Default Strategy



◀ Back to Start

► Skip Animation

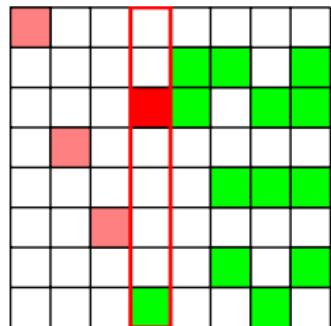
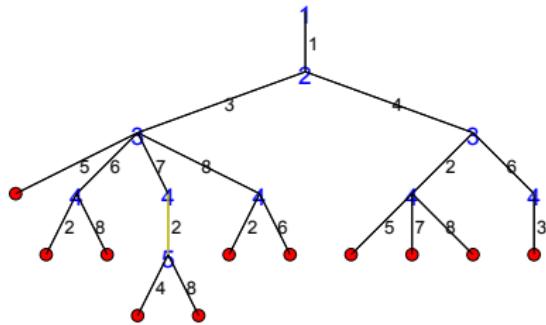
# Default Strategy



◀ Back to Start

► Skip Animation

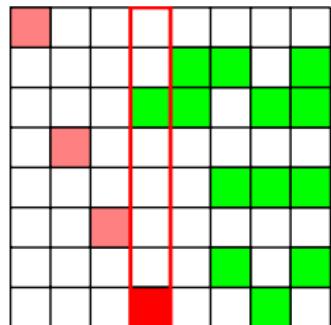
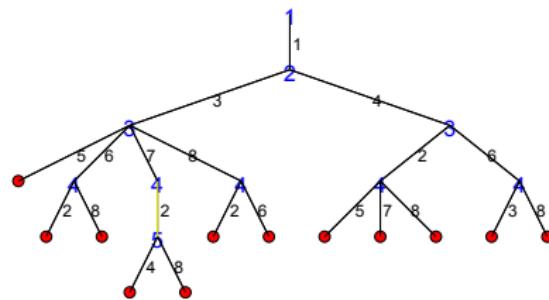
# Default Strategy



◀ Back to Start

► Skip Animation

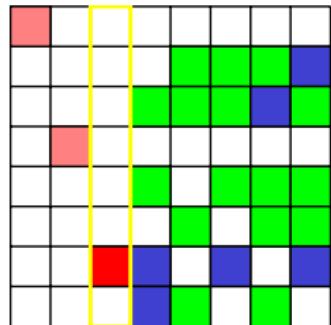
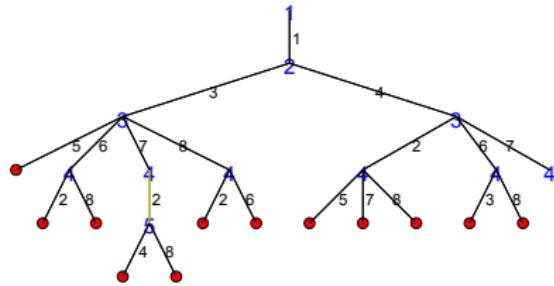
# Default Strategy



◀ Back to Start

► Skip Animation

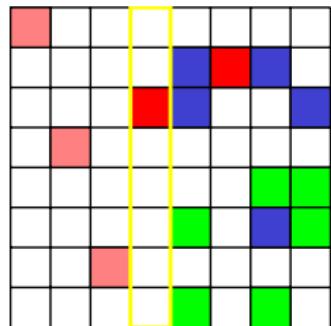
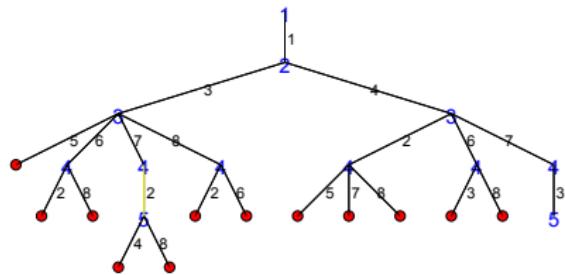
# Default Strategy



◀ Back to Start

► Skip Animation

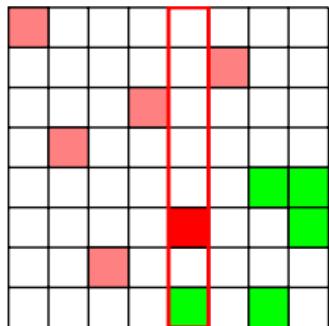
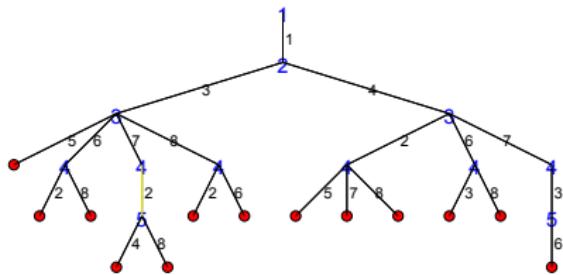
# Default Strategy



◀ Back to Start

► Skip Animation

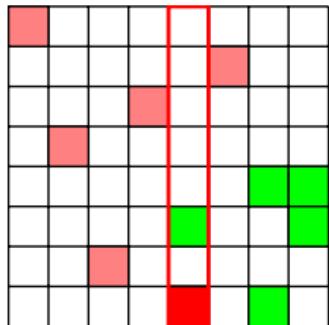
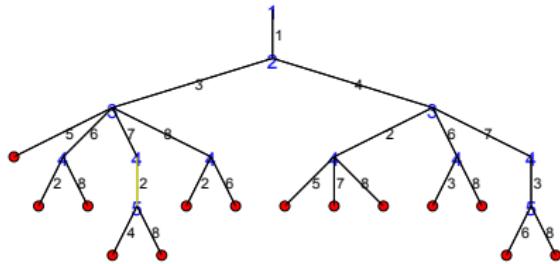
# Default Strategy



◀ Back to Start

► Skip Animation

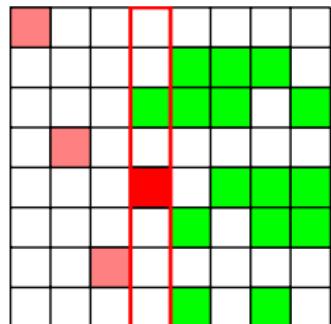
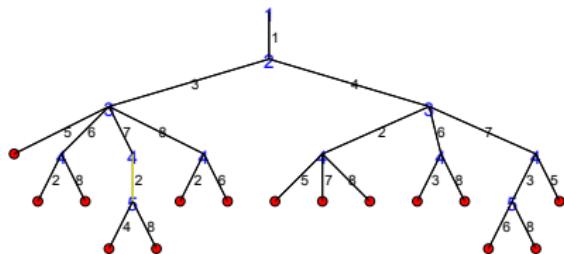
# Default Strategy



◀ Back to Start

► Skip Animation

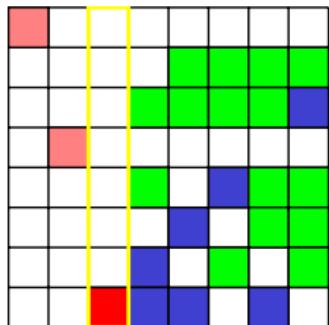
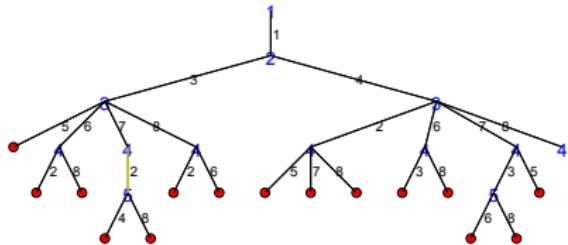
# Default Strategy



◀ Back to Start

► Skip Animation

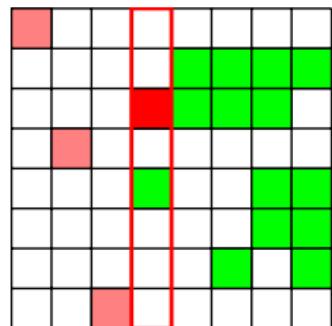
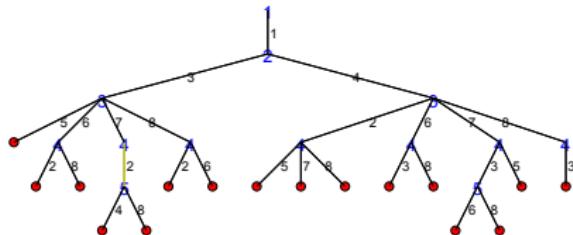
# Default Strategy



◀ Back to Start

► Skip Animation

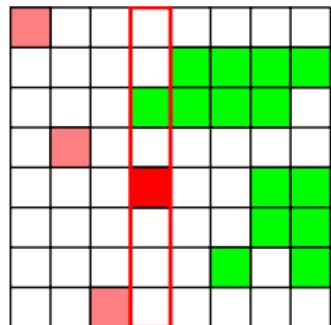
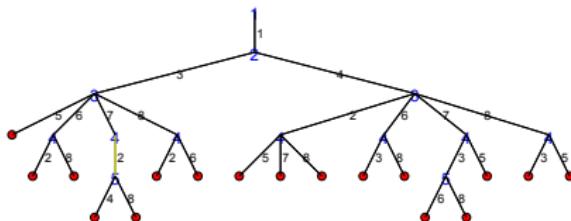
# Default Strategy



◀ Back to Start

► Skip Animation

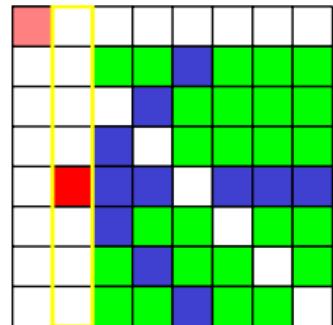
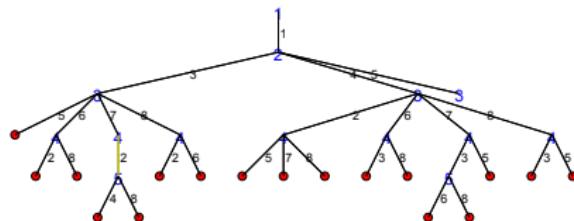
# Default Strategy



◀ Back to Start

► Skip Animation

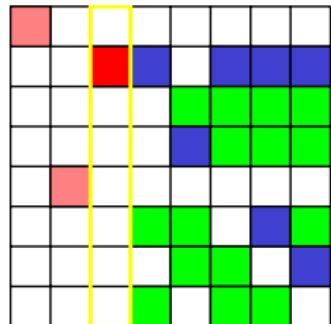
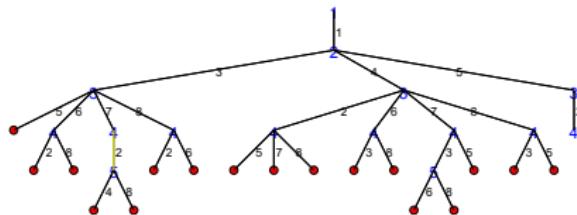
# Default Strategy



◀ Back to Start

► Skip Animation

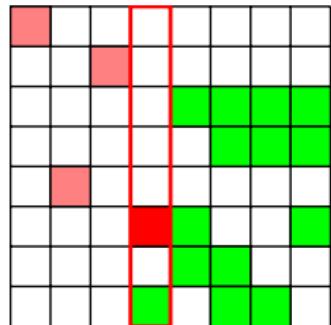
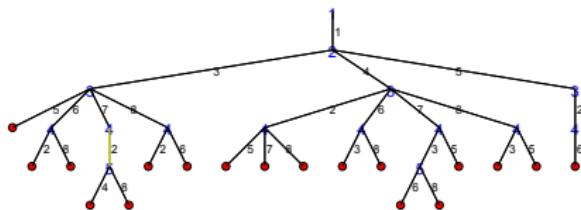
# Default Strategy



◀ Back to Start

► Skip Animation

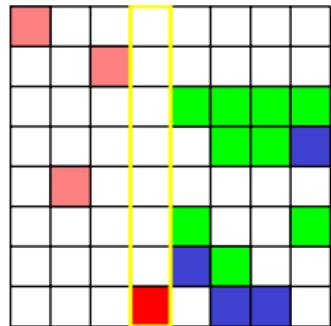
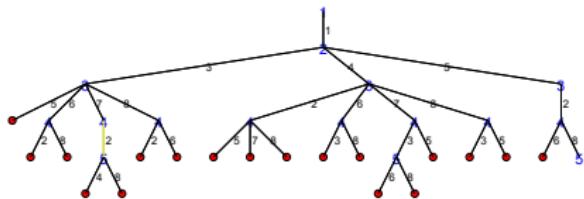
# Default Strategy



◀ Back to Start

► Skip Animation

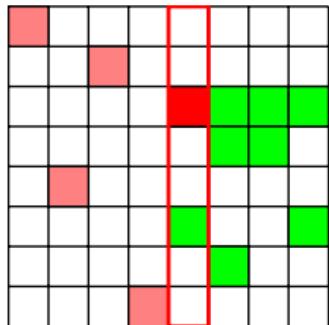
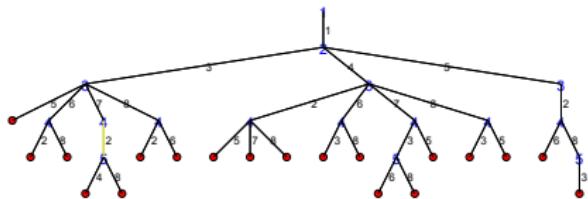
# Default Strategy



◀ Back to Start

► Skip Animation

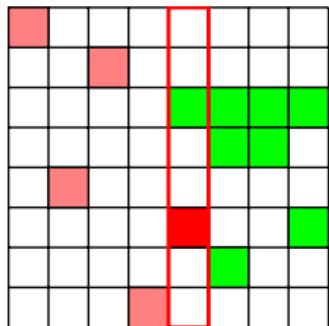
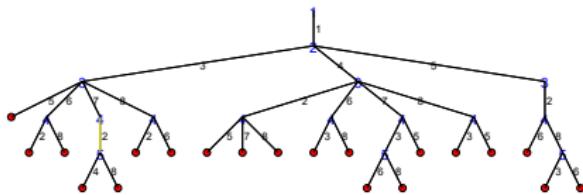
# Default Strategy



◀ Back to Start

► Skip Animation

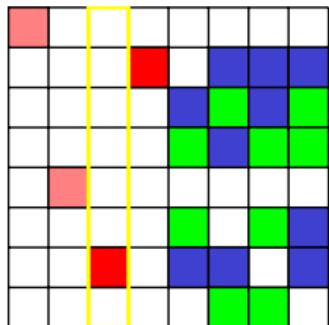
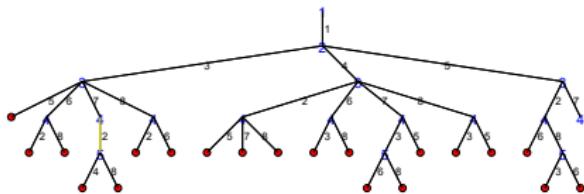
# Default Strategy



◀ Back to Start

► Skip Animation

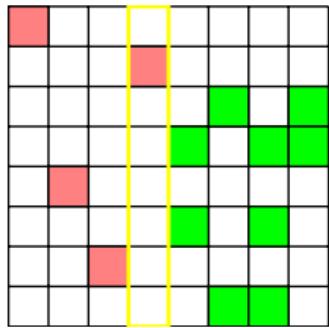
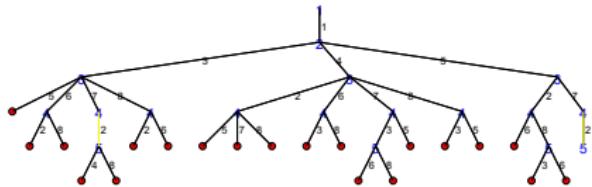
# Default Strategy



◀ Back to Start

► Skip Animation

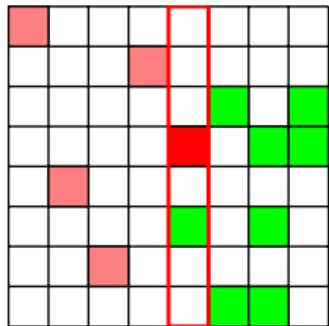
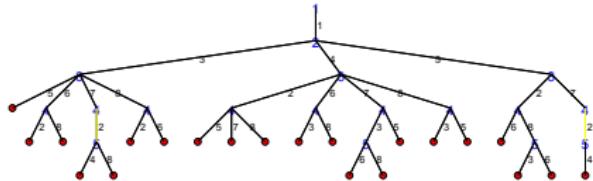
# Default Strategy



[◀ Back to Start](#)

[▶ Skip Animation](#)

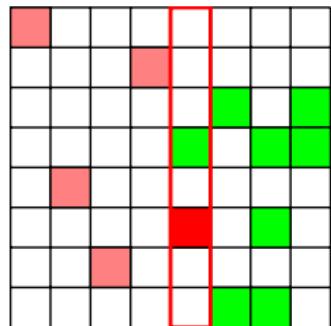
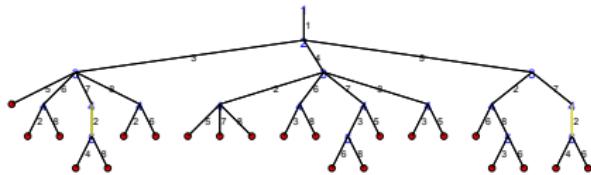
# Default Strategy



◀ Back to Start

► Skip Animation

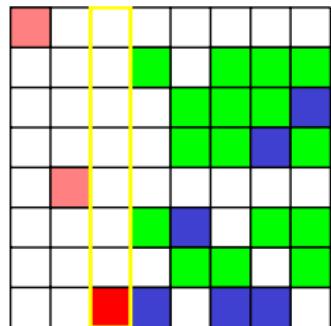
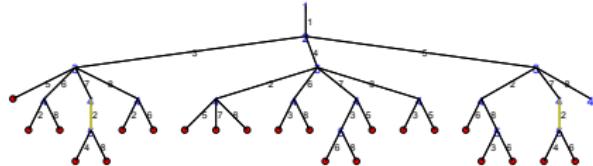
# Default Strategy



◀ Back to Start

► Skip Animation

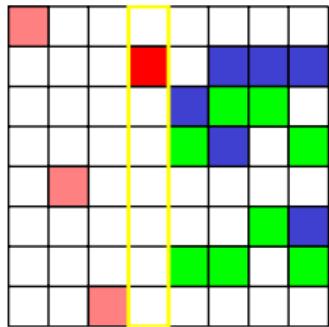
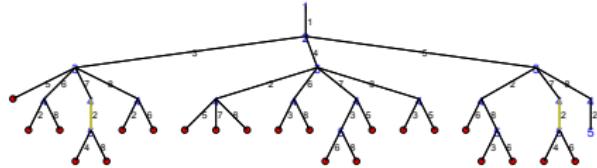
# Default Strategy



◀ Back to Start

► Skip Animation

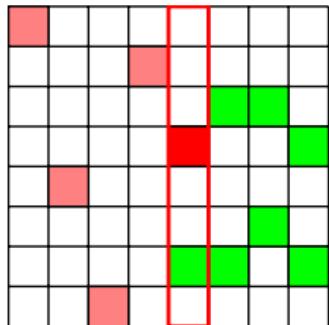
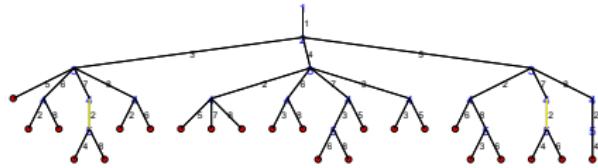
# Default Strategy



◀ Back to Start

► Skip Animation

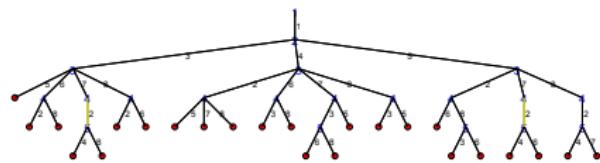
# Default Strategy



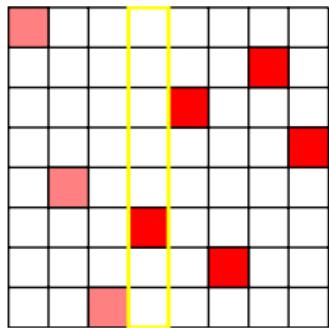
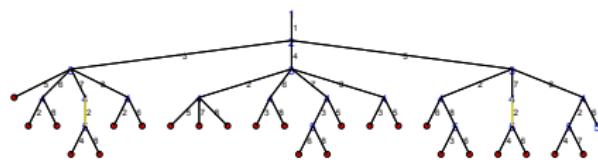
◀ Back to Start

► Skip Animation

# Default Strategy



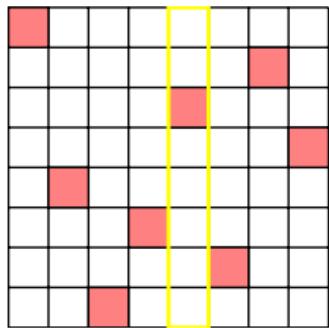
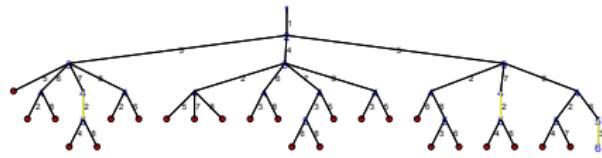
# Default Strategy



◀ Back to Start

► Skip Animation

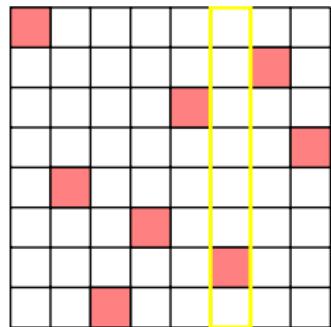
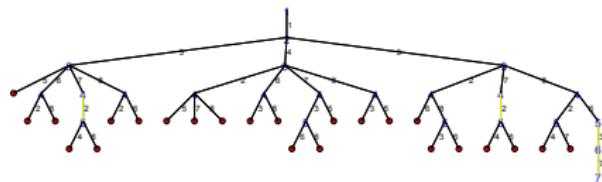
# Default Strategy



◀ Back to Start

► Skip Animation

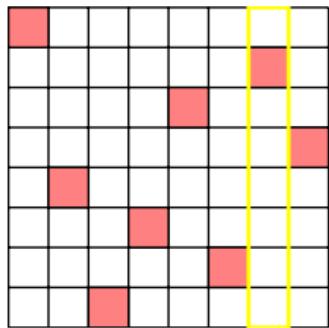
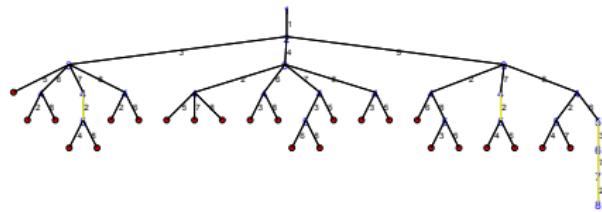
# Default Strategy



◀ Back to Start

► Skip Animation

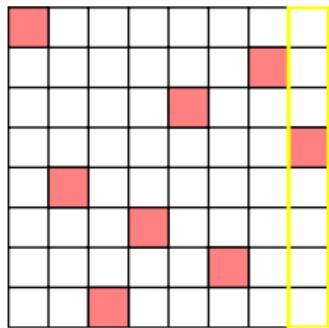
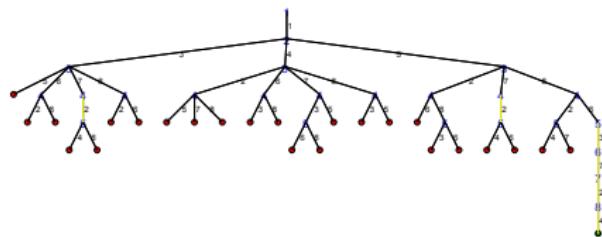
# Default Strategy



◀ Back to Start

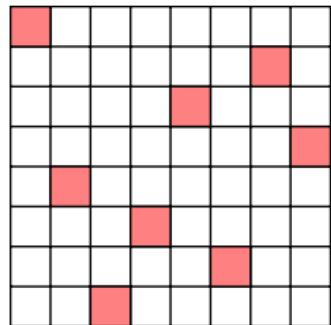
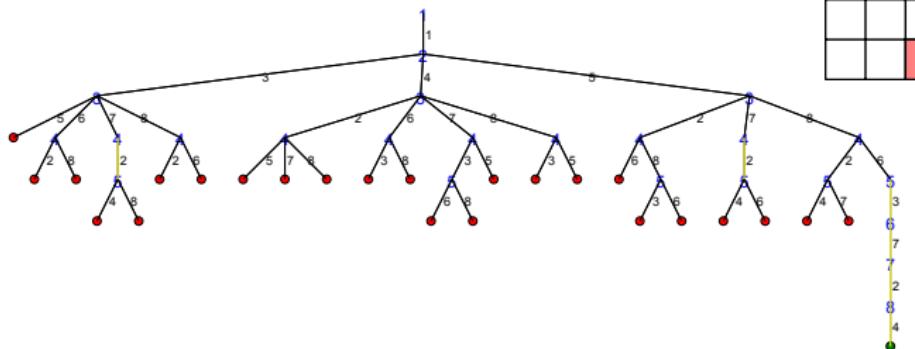
► Skip Animation

# Default Strategy



[◀ Back to Start](#)

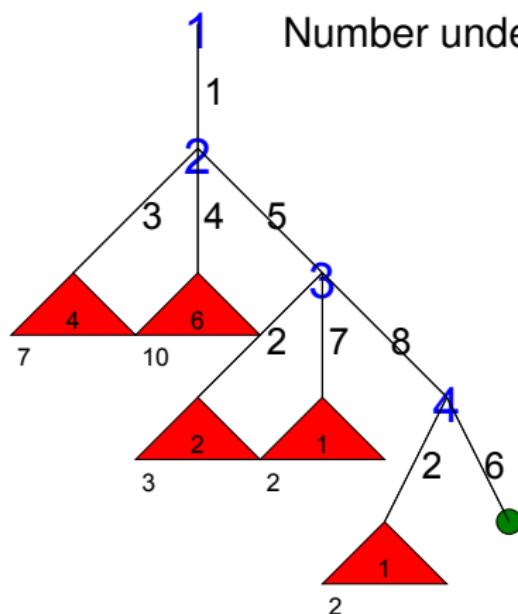
## First Solution



# Observations

- Even for small problem size, tree can become large
- Not interested in all details
- Ignore all automatically fixed variables
- For more compact representation abstract failed sub-trees

# Compact Representation



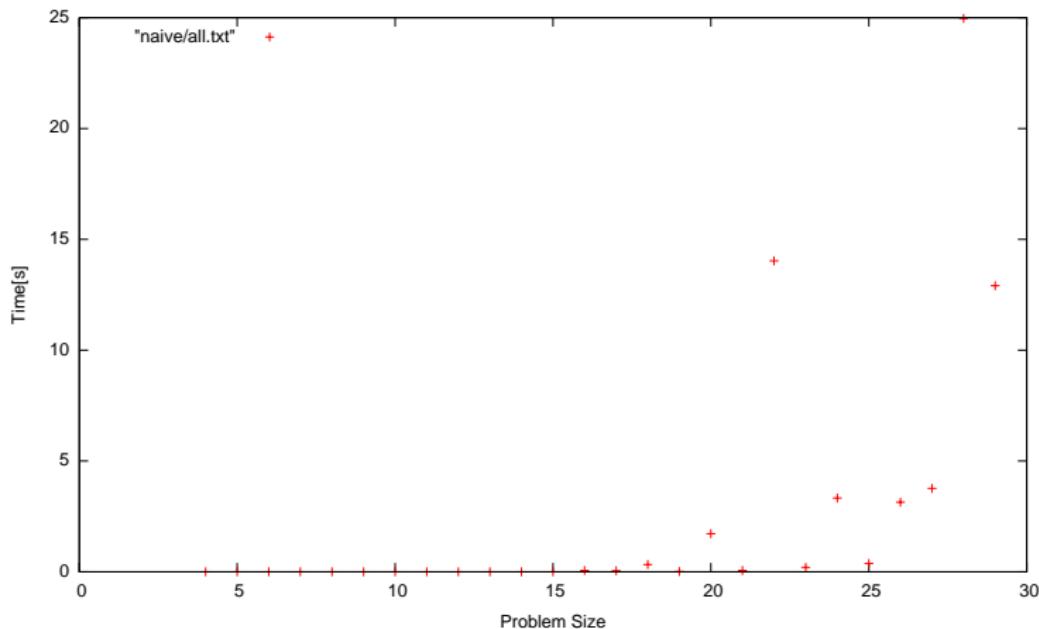
Number inside triangle:  
Number under triangle:

Number of choices  
Number of failures

# Exploring other board sizes

- How stable is the model?
- Try all sizes from 4 to 100
- Timeout of 100 seconds

# Naive Strategy, Problem Sizes 4-100



# Observations

- Time very reasonable up to size 20
- Sizes 20-30 times very variable
- Not just linked to problem size
- No size greater than 30 solved within timeout

# Outline

Problem

Program

Naive Search

## Improvements

Dynamic Variable Choice

Weighted Degree

Improved Heuristics

Making Search More Stable

# Possible Improvements

- Better constraint reasoning
  - Remodelling problem with 3 all different constraints
  - Global reasoning as described before
- Better control of search
  - Static vs. dynamic variable ordering
  - Better value choice
  - Not using complete depth-first chronological backtracking

# Static vs. Dynamic Variable Ordering

- Heuristic Static Ordering
  - Sort variables before search based on heuristic
  - Most important decisions
  - Smallest initial domain
- Dynamic variable ordering
  - Use information from constraint propagation
  - Different orders in different parts of search tree
  - Use all information available

# First Fail strategy

- Dynamic variable ordering
- At each step, select variable with smallest domain
- Idea: If there is a solution, better chance of finding it
- Idea: If there is no solution, smaller number of alternatives
- Needs tie-breaking method

# Search Strategy Choices

- Minizinc [▶ Show](#)
- Choco-solver [▶ Show](#)

# Modified MiniZinc Program

```
int: n=8;
array[1..n] of var 1..n: queens;
constraint
  forall(i, j in 1..n where i < j) (
    queens[i] != queens[j] /\ 
    queens[i] + i != queens[j] + j /\ 
    queens[i] - i != queens[j] - j
  )
;
solve :: int_search(
  queens,
  first_fail,
  indomain_min)
  satisfy;
```

## Variable Choice (MiniZinc)

- Determines the order in which variables are assigned
- `input_order` assign variables in static order given
- `smallest` assign variable with smallest value in domain first
- `first_fail` select variable with smallest domain first
- `dom_w_deg` consider ratio of domain size and failure count
- Others, including programmed selection for specific solvers

## Value Choice (MiniZinc)

- Determines the order in which values are tested for selected variables
- `indomain_min` Start with smallest value, on backtracking try next larger value
- `indomain_median` Start with value closest to middle of domain
- `indomain_random` Choose values in random order
- `indomain_split` Split domain into two intervals

▶ Continue

# Modified Choco-solver Model

```
int n = 8;

Model model = new Model(n + "-queens problem");
IntVar[] vars = model.intVarArray("Q", n, 1, n, false);
IntVar[] diag1 = IntStream.range(0, n).
    mapToInt(i -> vars[i].sub(i).intVar()).
    toArray(IntVar[]::new);
IntVar[] diag2 = IntStream.range(0, n).
    mapToInt(i -> vars[i].add(i).intVar()).
    toArray(IntVar[]::new);

model.post(
    model.allDifferent(vars),
    model.allDifferent(diag1),
    model.allDifferent(diag2)
);

Solver solver = model.getSolver();
solver.showStatistics();
solver.setSearch(Search.domOverWDegSearch(vars));
Solution solution = solver.findSolution();

if (solution != null) {
    System.out.println(solution.toString());
}
```

## VariableSelector Choice (Choco-solver)

- Determines the order in which variables are assigned
- InputOrder assign variables in static order given
- Smallest assign variable with smallest value in domain first
- FirstFail select variable with smallest domain first
- DomOverWDeg consider ratio of domain size and failure count
- ActivityBased dynamic, based on dynamic observed behaviour
- ImpactBased dynamic, based on dynamic observed behaviour

## IntValueSelector Choice (Choco-solver)

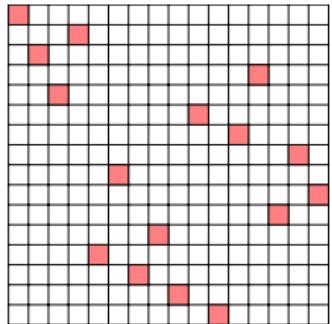
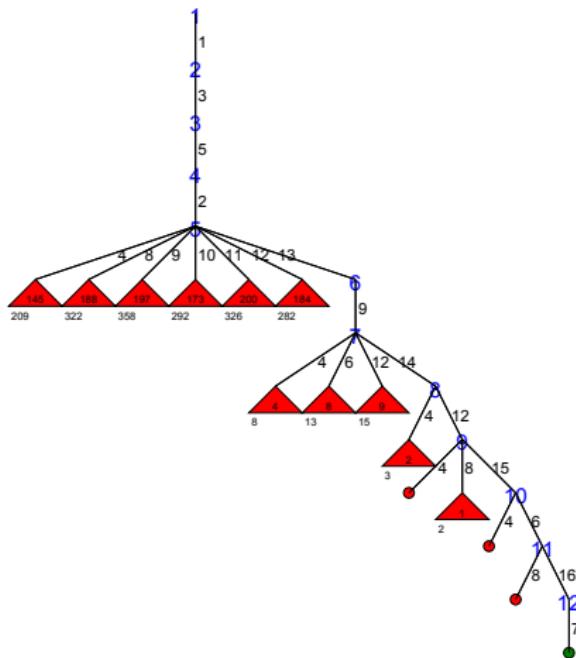
- Determines the order in which values are tested for selected variables
- IntDomainMin Start with smallest value, on backtracking try next larger value
- IntDomainMiddle Start with value closest to middle of domain
- IntDomainRandom Choose values in random order
- IntDomainRandomBound Randomly choose between smallest and largest value

▶ Continue

# Comparison

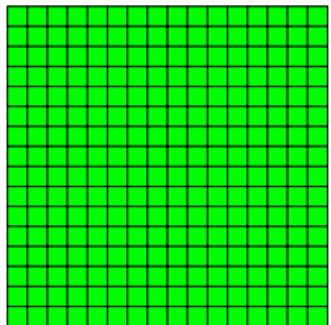
- Board size 16x16
- Naive (Input Order) Strategy
- First Fail variable selection

# Naive (Input Order) Strategy (Size 16)



# FirstFail Strategy (Size 16)

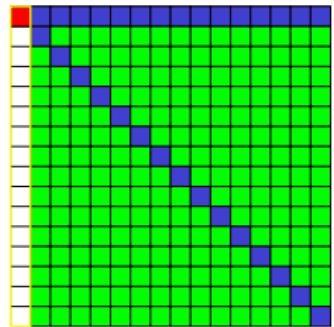
1



▶ Skip Animation

# FirstFail Strategy (Size 16)

1  
1  
2

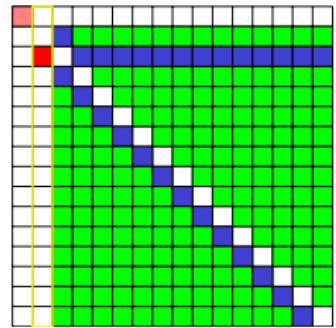


◀ Back to Start

► Skip Animation

# FirstFail Strategy (Size 16)

1  
1  
2  
3  
3

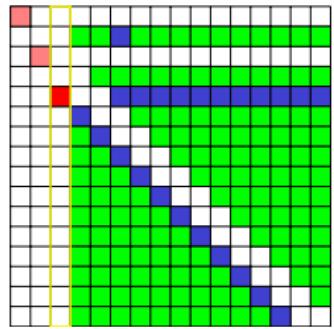


◀ Back to Start

► Skip Animation

# FirstFail Strategy (Size 16)

1  
1  
2  
3  
3  
5  
6

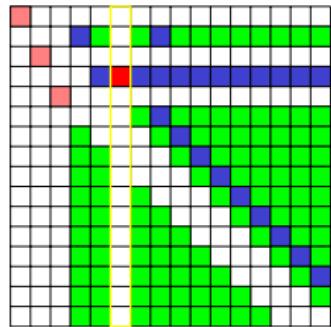


◀ Back to Start

► Skip Animation

# FirstFail Strategy (Size 16)

1  
1  
2  
3  
3  
5  
6  
4  
8

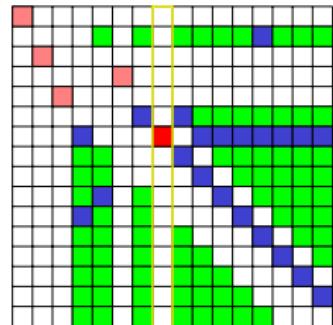


◀ Back to Start

► Skip Animation

## FirstFail Strategy (Size 16)

1  
1  
2  
3  
3  
5  
6  
4  
8  
7  
13

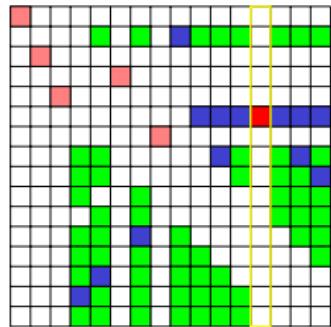


Back to Start

► Skip Animation

# FirstFail Strategy (Size 16)

1  
1  
2  
3  
3  
5  
6  
4  
8  
7  
13  
6  
11

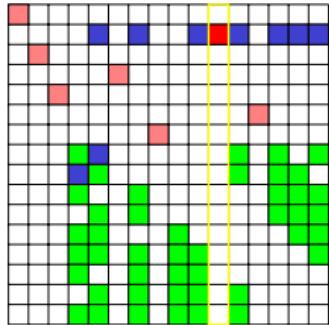


◀ Back to Start

► Skip Animation

# FirstFail Strategy (Size 16)

1  
1  
2  
3  
3  
5  
6  
4  
8  
7  
13  
6  
11  
2  
10

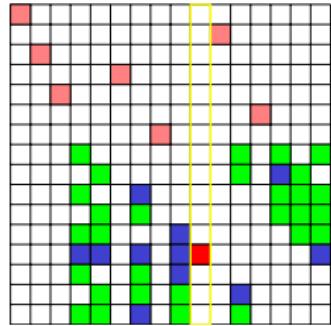


◀ Back to Start

► Skip Animation

# FirstFail Strategy (Size 16)

1  
1  
2  
3  
3  
5  
6  
4  
8  
7  
13  
6  
11  
2  
10  
13  
9

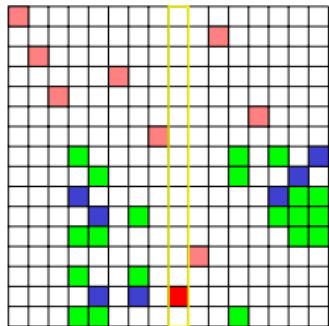


◀ Back to Start

► Skip Animation

# FirstFail Strategy (Size 16)

1  
2  
3  
3  
5  
6  
4  
8  
7  
13  
13  
11  
10  
13  
9  
15  
7

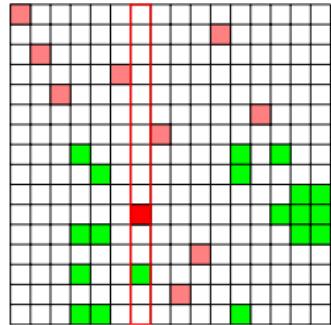


◀ Back to Start

► Skip Animation

# FirstFail Strategy (Size 16)

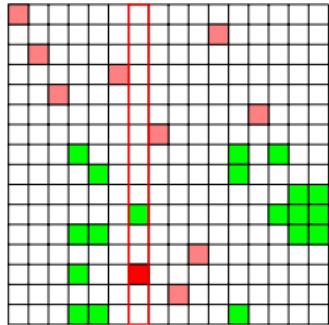
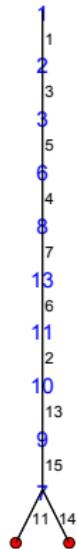
1  
2  
3  
3  
5  
6  
4  
8  
7  
13  
6  
11  
2  
10  
13  
9  
15  
7  
11



◀ Back to Start

▶ Skip Animation

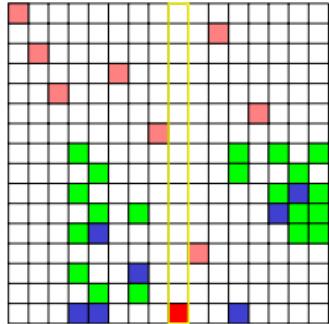
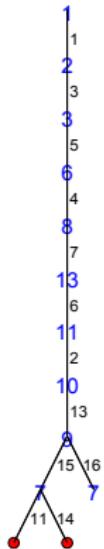
# FirstFail Strategy (Size 16)



◀ Back to Start

► Skip Animation

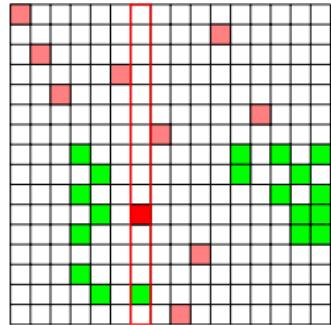
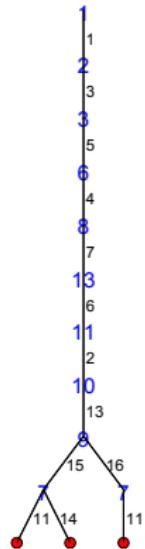
# FirstFail Strategy (Size 16)



◀ Back to Start

► Skip Animation

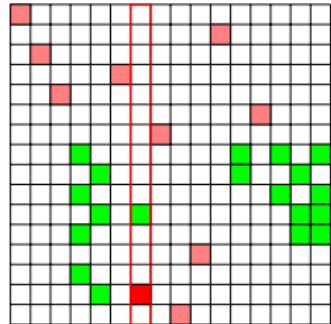
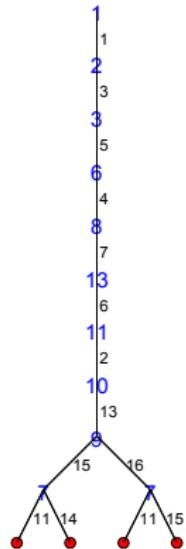
# FirstFail Strategy (Size 16)



◀ Back to Start

► Skip Animation

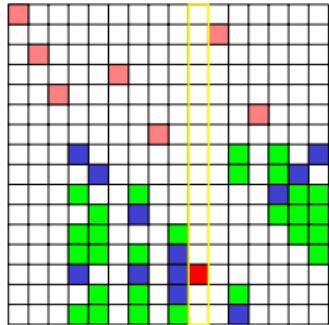
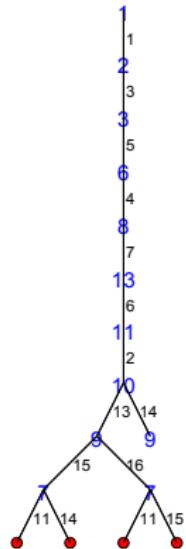
# FirstFail Strategy (Size 16)



◀ Back to Start

► Skip Animation

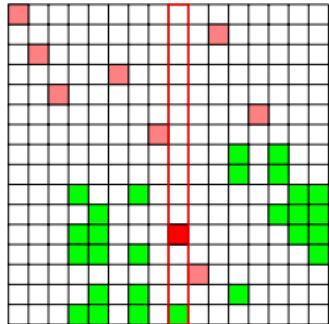
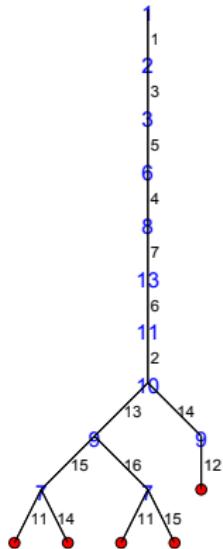
# FirstFail Strategy (Size 16)



◀ Back to Start

► Skip Animation

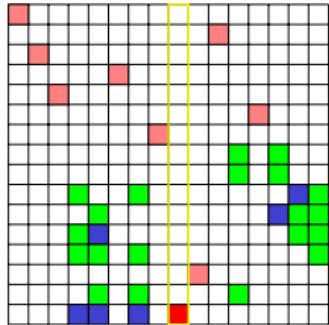
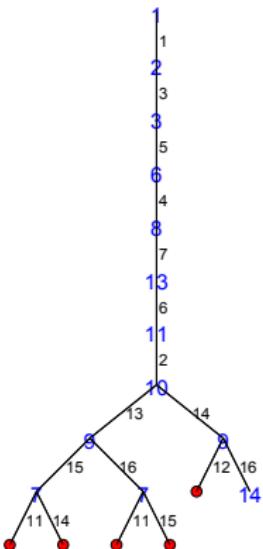
# FirstFail Strategy (Size 16)



◀ Back to Start

► Skip Animation

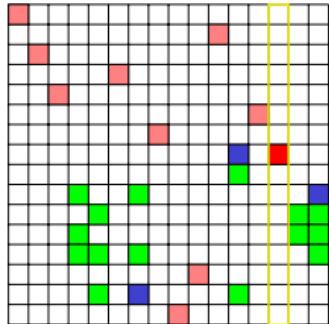
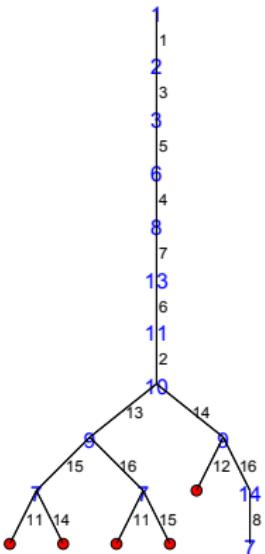
# FirstFail Strategy (Size 16)



◀ Back to Start

► Skip Animation

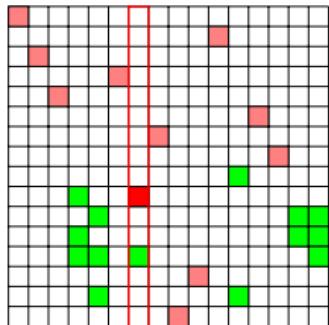
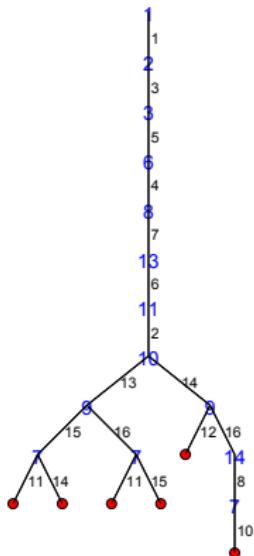
# FirstFail Strategy (Size 16)



◀ Back to Start

► Skip Animation

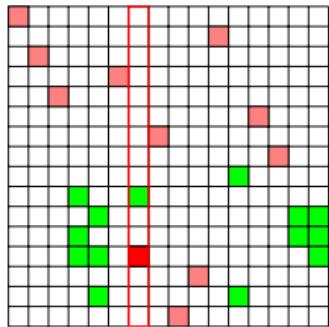
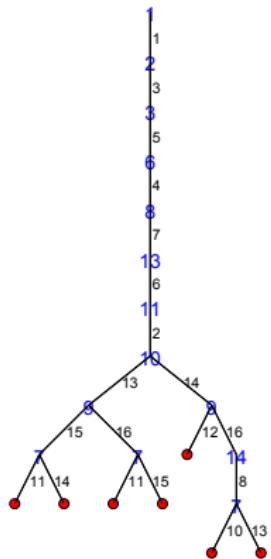
# FirstFail Strategy (Size 16)



◀ Back to Start

► Skip Animation

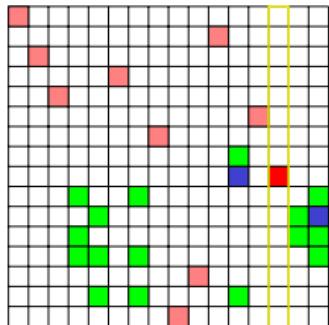
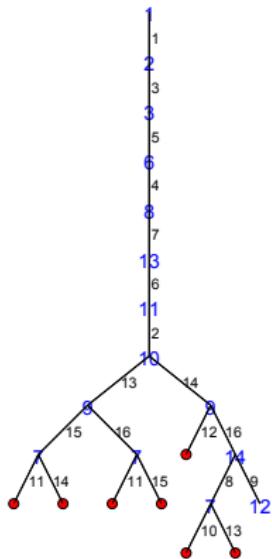
# FirstFail Strategy (Size 16)



◀ Back to Start

► Skip Animation

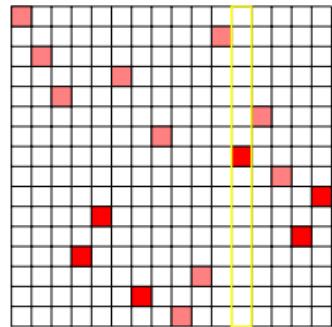
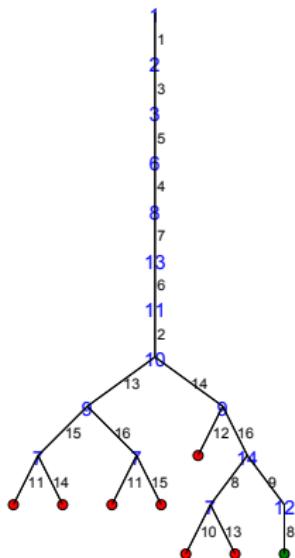
# FirstFail Strategy (Size 16)



◀ Back to Start

► Skip Animation

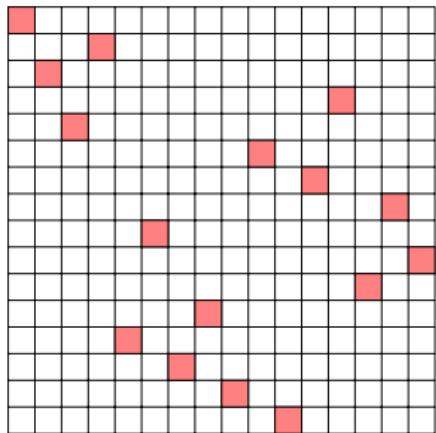
## FirstFail Strategy (Size 16)



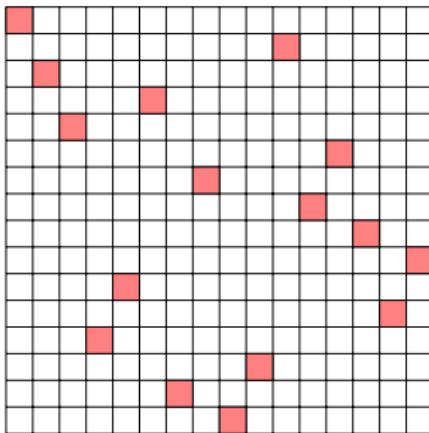
[◀ Back to Start](#)

# Comparing Solutions

Naive

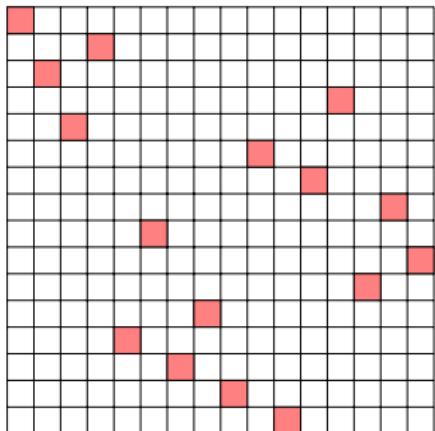


First Fail

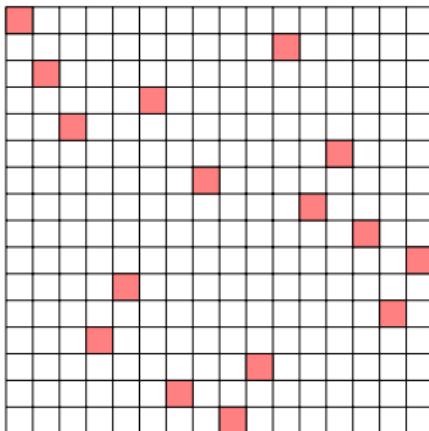


# Comparing Solutions

Naive

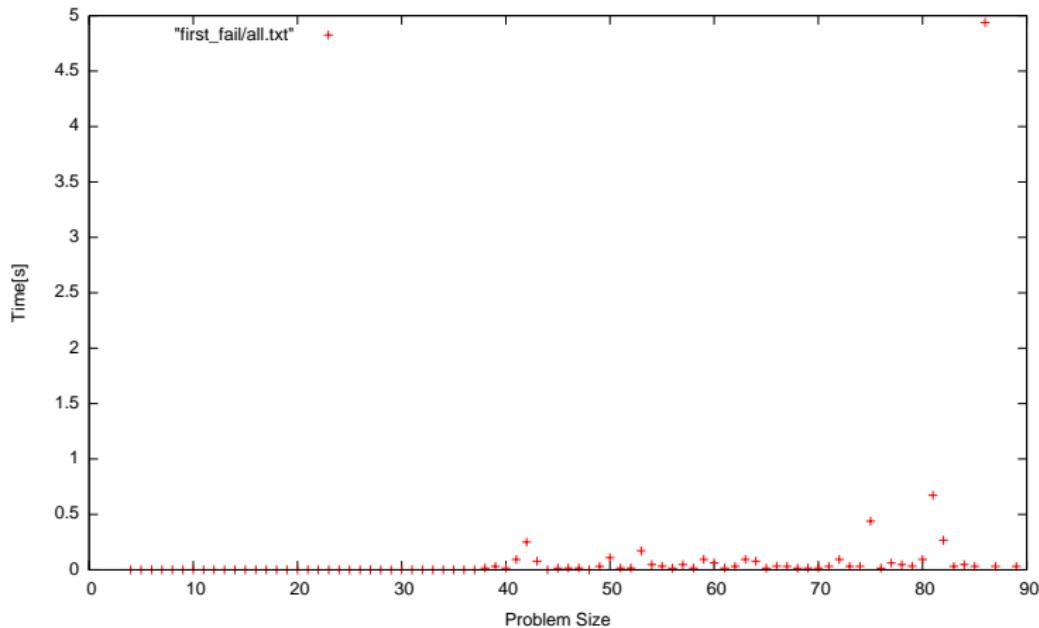


First Fail



Solutions are different!

# FirstFail, Problem Sizes 4-100



# Observations

- This is much better
- But some sizes are much harder
- Timeout for sizes 88, 91, 93, 97, 98, 99

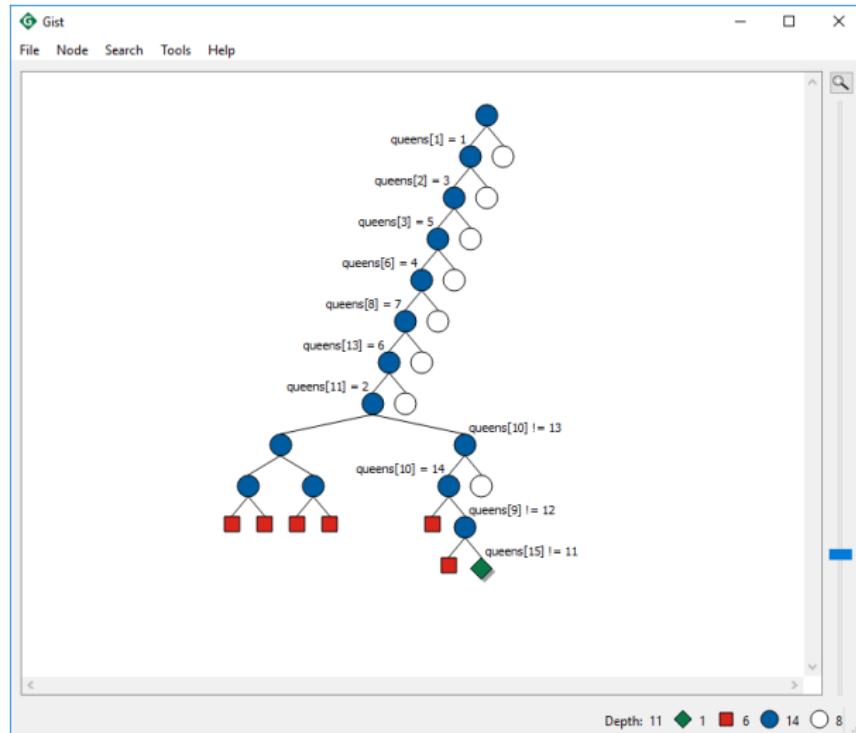
## More Reactive Variable Selection

- Domain size is important, but other information is useful as well
- Dom/Weighted Degree: better results in many situations
- Weight Degree: count how often variable has been involved in failure
- Focus on more complicated part of problem
- Changes during search, learns from past performance
- Option **dom\_w\_deg**

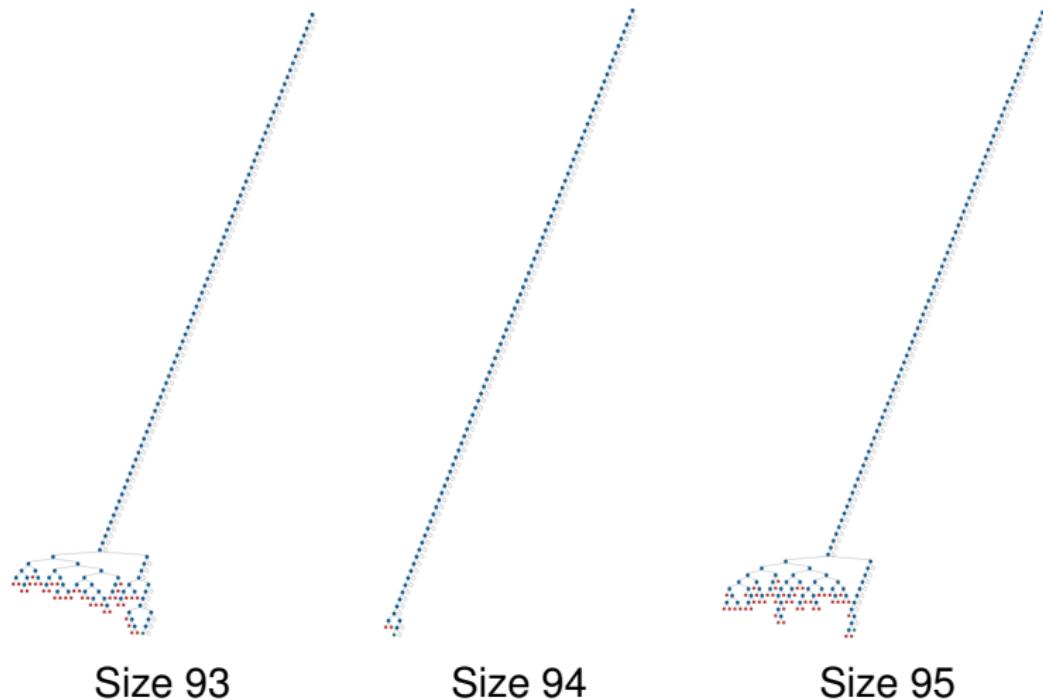
# Weighted Degree Variable Selection

```
int: n=8;
array[1..n] of var 1..n: queens;
constraint
    forall(i, j in 1..n where i < j) (
        queens[i] != queens[j] /\ 
        queens[i] + i != queens[j] + j /\ 
        queens[i] - i != queens[j] - j
    )
;
solve :: int_search(
    queens,
    dom_w_deg,
    indomain_random)
    satisfy;
```

# Result for size 16 with Gecode-Gist



## Sample Results for Larger Sizes



## Approach 1: Heuristic Portfolios

- Try multiple strategies for the same problem
- With multi-core CPUs, run them in parallel
- Only one needs to be successful for each problem

## Approach 2: Restart with Randomization

- Only spend limited number of backtracks for a search attempt
- When this limit is exceeded, restart at beginning
- Requires randomization to explore new search branch
- Randomize variable choice by random tie break
- Randomize value choice by shuffling values
- Needs strategy when to restart

# Random Variable Choice and Restarts

```
int: n=8;
array[1..n] of var 1..n: queens;
constraint
    forall(i, j in 1..n where i < j) (
        queens[i] != queens[j] /\ 
        queens[i] + i != queens[j] + j /\ 
        queens[i] - i != queens[j] - j
    )
;
solve :: int_search(
    queens,
    dom_w_deg,
    indomain_random)
    :: random_linear(100)
    satisfy;
```

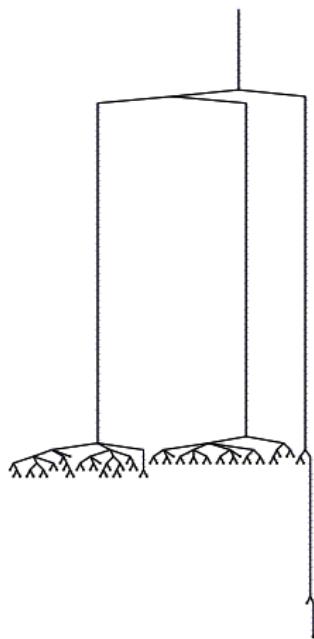
## Approach 3: Partial Search

- Abandon depth-first, chronological backtracking
- Don't get locked into a failed sub-tree
- A wrong decision at a level is not detected, and we have to explore the complete subtree below to undo that wrong choice
- Explore more of the search tree
- Spend time in promising parts of tree

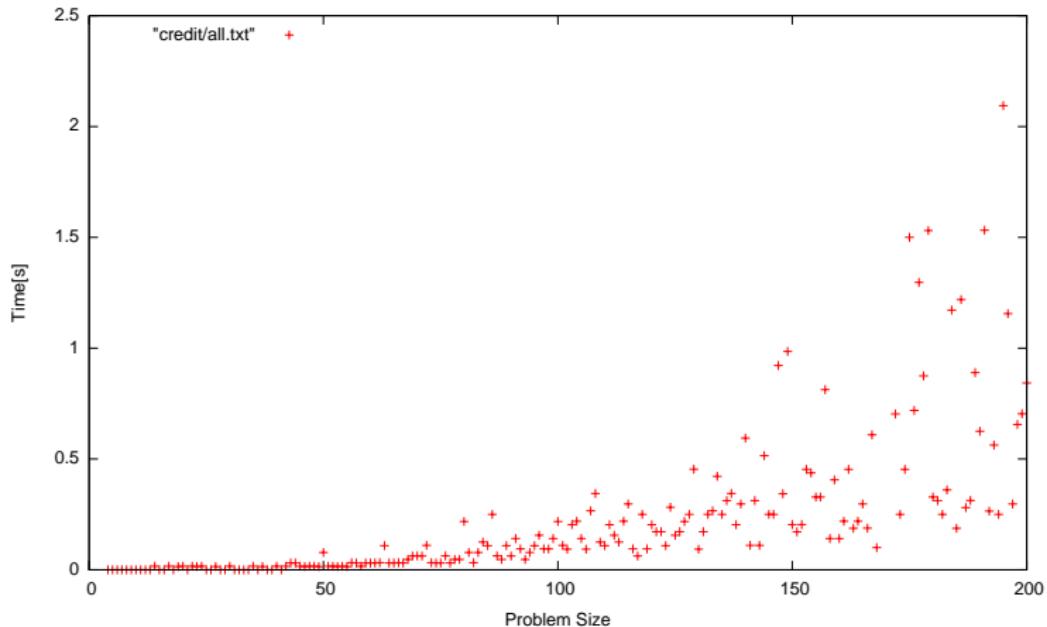
## Example: Credit Search

- Not available in all solvers
- Explore top of tree completely, based on credit
- Start with fixed amount of credit
- Each node consumes one credit unit
- Split remaining credit amongst children
- When credit runs out, start bounded backtrack search
- Each branch can use only  $K$  backtracks
- If this limit is exceeded, jump to unexplored top of tree

# Credit, Search Tree Problem Size 94



# Credit, Problem Sizes 4-200



## Points to Remember

- Choice of search can have huge impact on performance
- Dynamic variable selection can lead to large reduction of search space
- Packaged search can do a lot, but programming search adds even more
- Depth-first chronological backtracking not always best choice
- How to control this explosion of search alternatives?

## Part IV

**What is missing?**

# Many Specialized Topics

- How to design efficient core engine
- Hybrids with LP/MIP tools
- Hybrids with SAT
- Symmetry breaking
- Use of MDD/BDD to encode sets of solutions
- High level modelling tools
- Debugging/visualization

# Reformulation

- Just because the user has modelled it this way, it doesn't mean we have to solve it that way
  - Replace some constraint(s) by other, equivalent constraints
  - Because we don't have that constraint in our system
  - For performance

# Learning

- While solving the problem we can learn how to strengthen the model/search
  - Understand which constraints/method contribute to propagation and change schedule
  - Learn no-good constraints by explaining failure
  - Adapt search strategy based on search experience

# More Learning Resources

- Survey of Methods, Resources, and Formats for Teaching Constraint Programming
  - by Tejas Santanam, Helmut Simonis
  - <https://doi.org/10.48550/arXiv.2403.12717>
  - Based on survey of community for WTCP 2023
  - <https://hsimonis.github.io/WTCP2023/>