上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

Lecture 7: Constraint Satisfaction Problems

Tao Huang

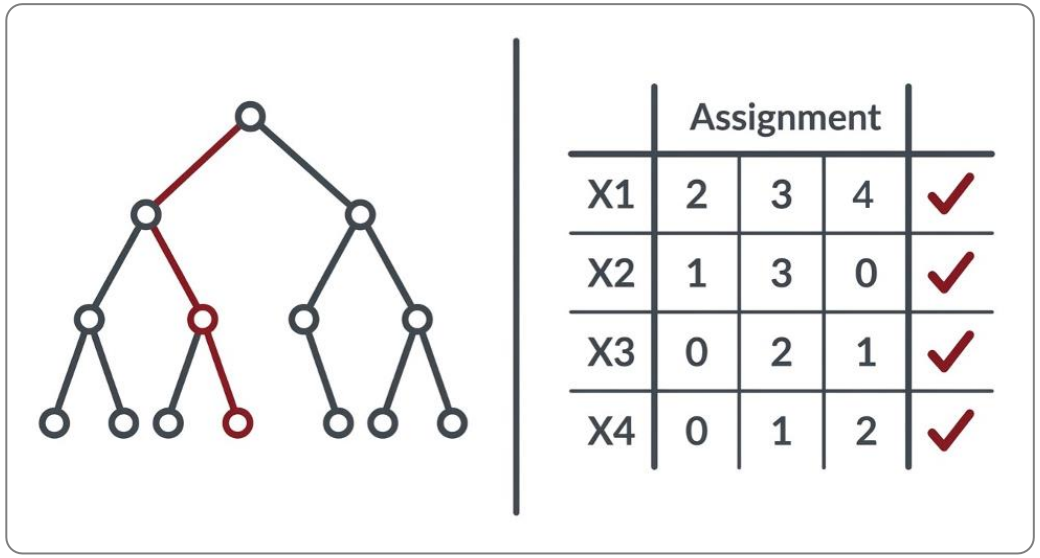
John Hopcroft Center, School of Computer Science, Shanghai Jiao Tong University

https://taohuang.info/cs3317

https://oc.sjtu.edu.cn/courses/89538

Why Another Search Paradigm?

- Many problems are not naturally about paths.
- We do not care about how to reach a state.
- We care about feasibility: satisfying all constraints.



Path search tree vs. feasible assignment table

- *Examples: Sudoku, Map coloring, Scheduling, Allocation.*

Definition of CSP

Variables: what we need to decide

$X = \{X_1, X_2, \ldots, X_n\}$

Domains: set of possible values

D_i for each X_i

Constraints: rules limiting combinations

$C = \{C_1, C_2, \ldots, C_m\}$

A Solution is a complete assignment that satisfies all C_i.

Example 1: Map Coloring

- **Variables:**

  Regions {WA, NT, Q, NSW, V, SA, T}

- **Domains:**

  {red, green, blue}

- **Constraints:**

  Adjacent regions must have different colors

Map Coloring: Standard Formulation

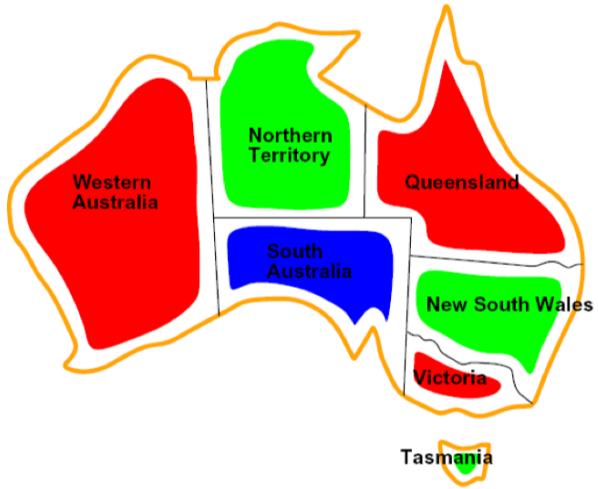
- Initial State: the empty assignment, {}
- Action: assign a value to an unassigned variable
- Goal test: the current assignment is complete and satisfies all constraints

Search Method: BFS

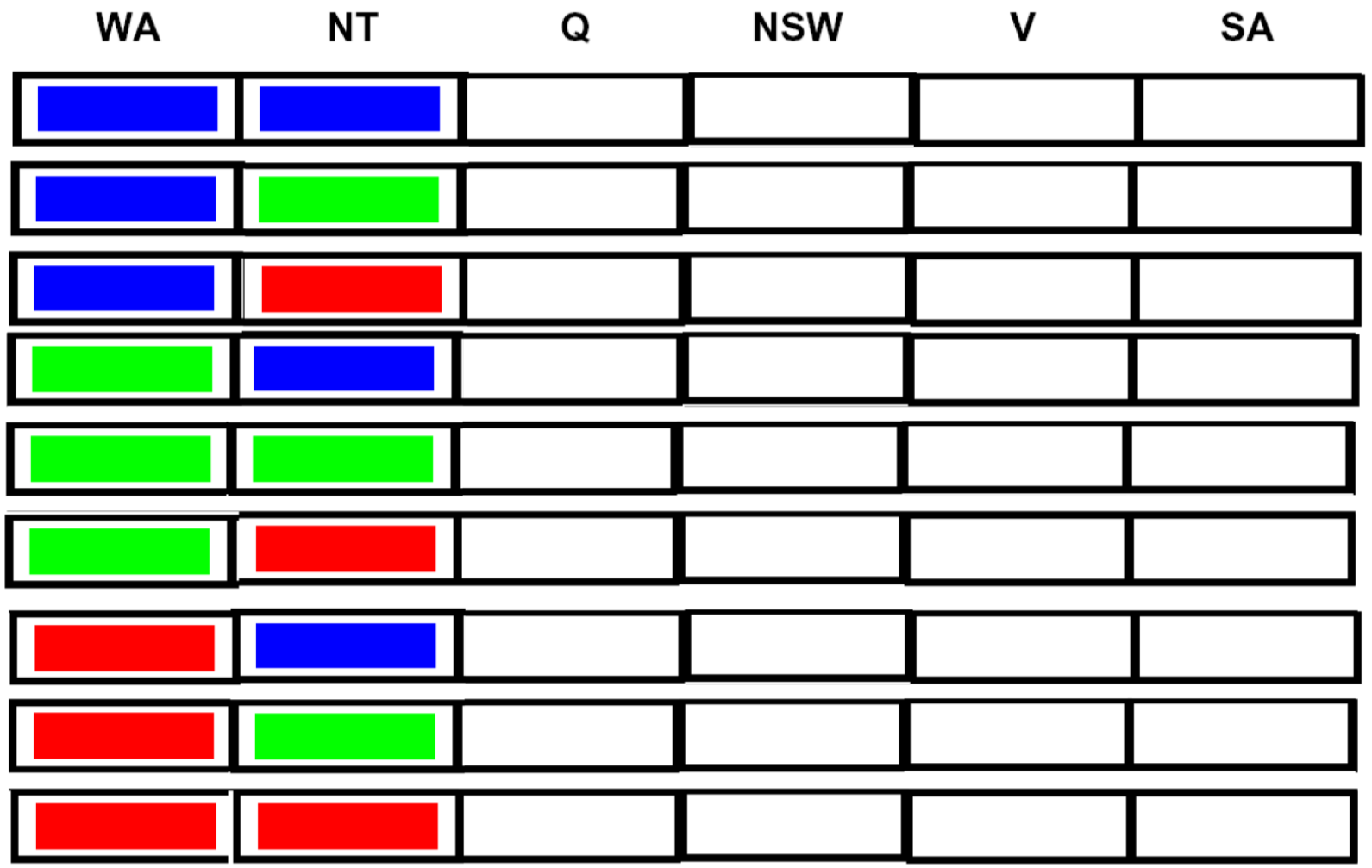
- What would BFS do?

{}

{WA=g} {WA=r} {WA=b} {NT=r} …

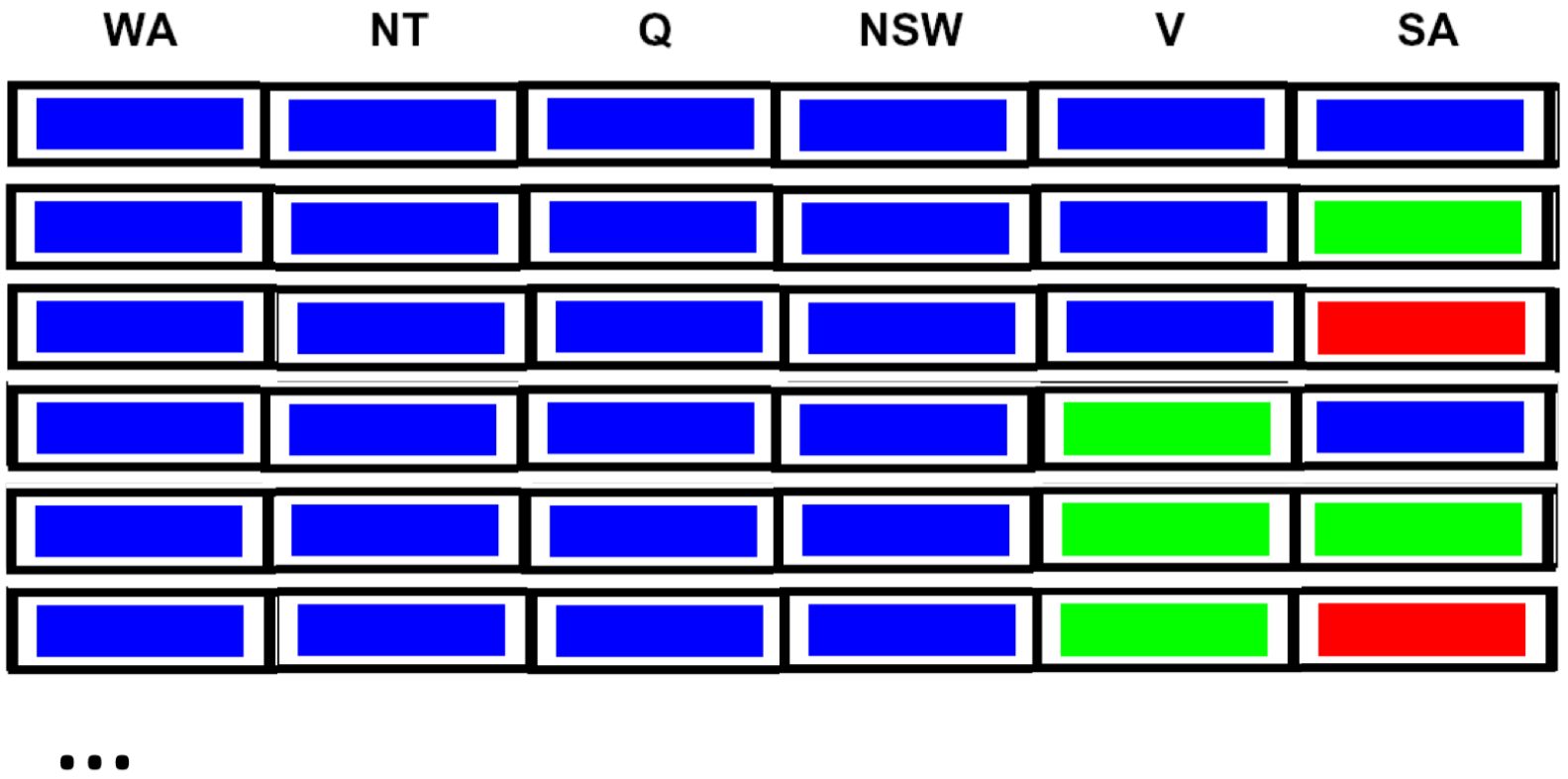


Search Method: BFS



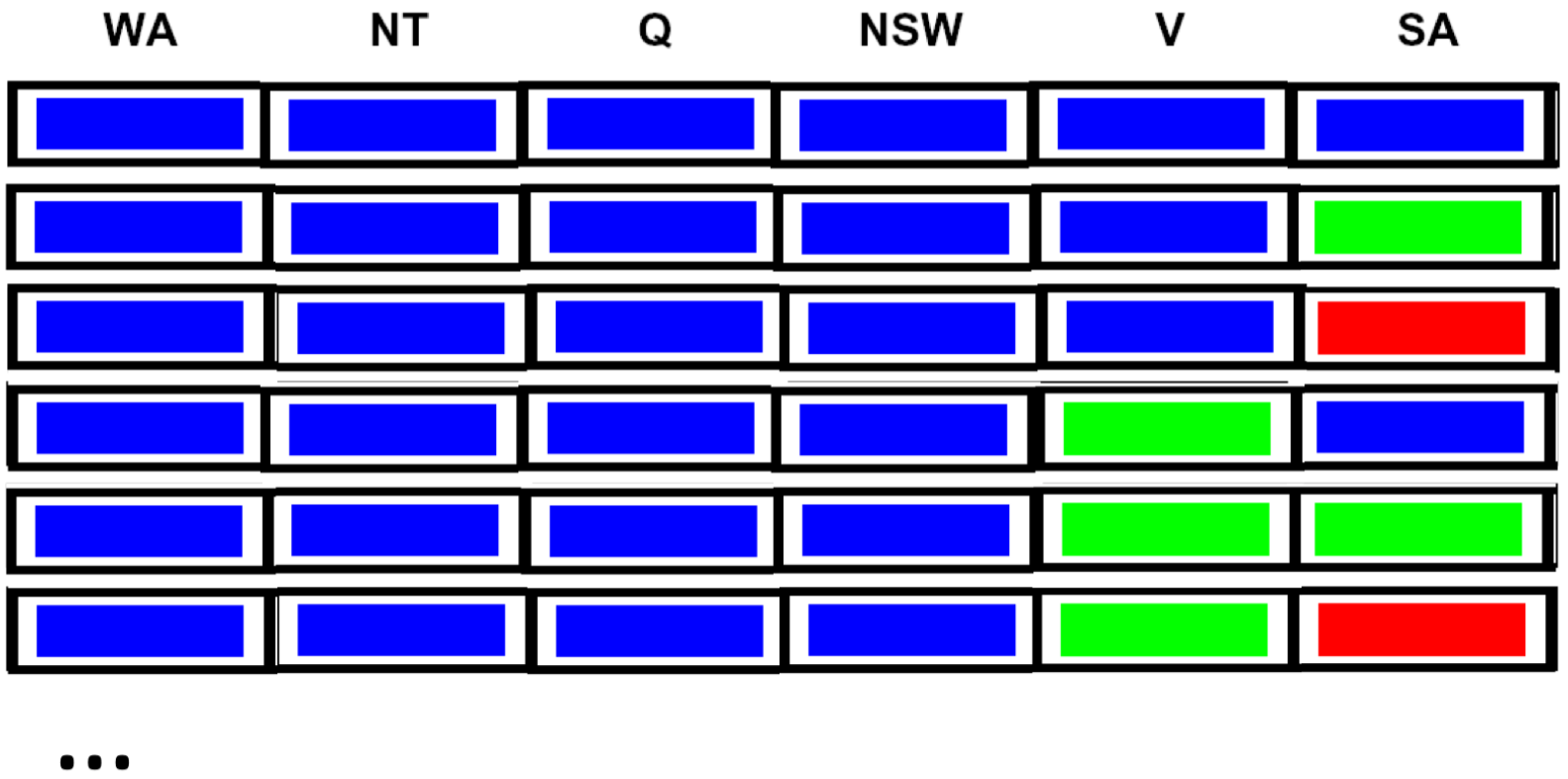
Any two variables

Search Method: BFS



Any assignment for all variables

Search Method: DFS



Any assignment for all variables

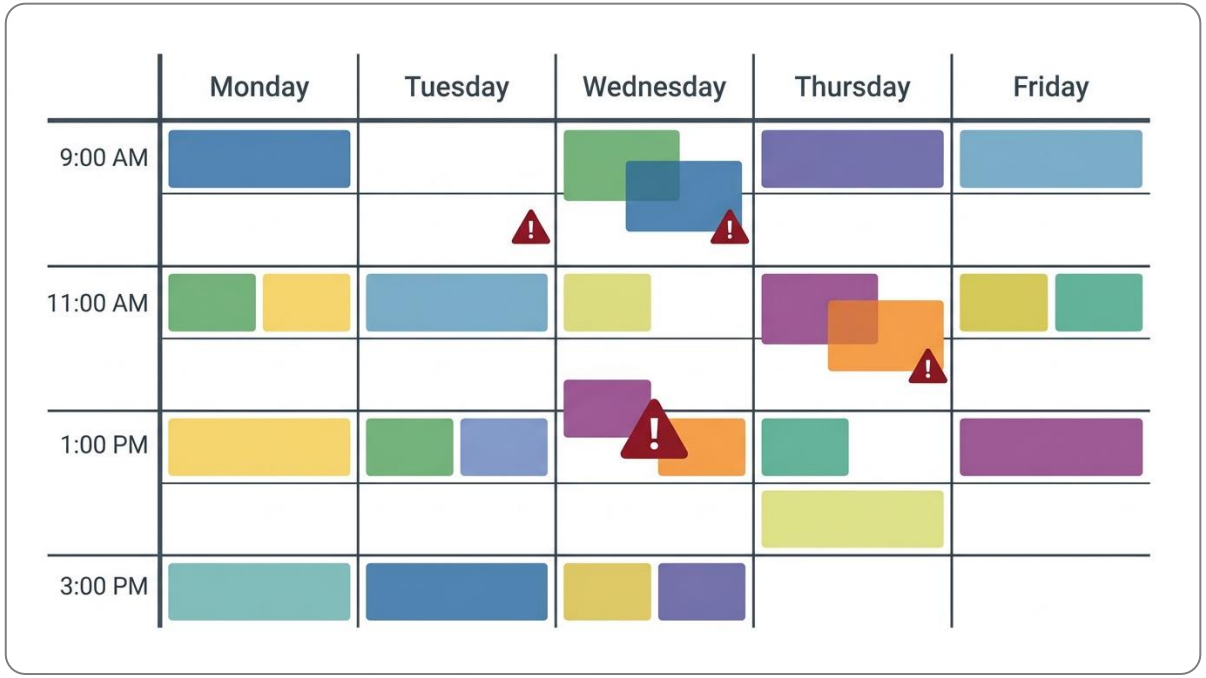
Example 2: Sudoku

- **Variables:**
  81 cells (X_{11} to X_{99})
- **Domains:**
  {1, 2, …, 9}
- **Constraints:**
  - Alldiff(row)
  - Alldiff(column)
  - Alldiff(3x3 box)

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

Example 3: Scheduling

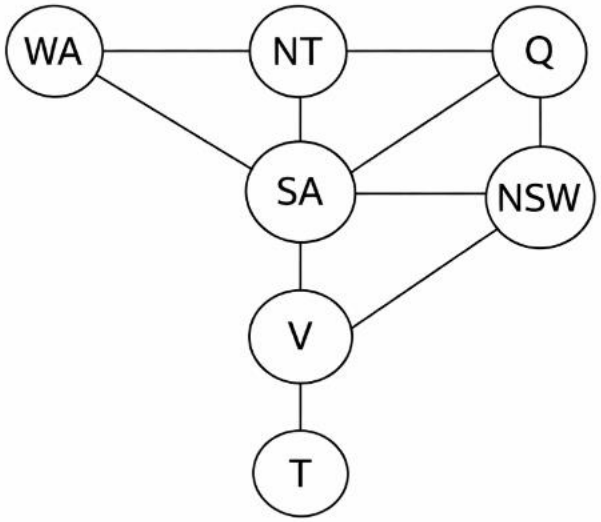
- **Variables:**
  Exams / Courses
- **Domains:**
  Time slots / Rooms
- **Constraints:**
  - No student has 2 exams at once
  - Room capacity limits
  - Resource availability



Constraint Graph

A CSP can often be represented as a graph

- Nodes = Variables
- Edges = Binary Constraints
- Explicitly exposes the structure of the problem
- Graph properties (connectivity, degree) drive solving efficiency



Constraint Graph for Australia map coloring

Naive Generate-and-Test

1. Generate all possible assignments
2. Test constraints for each assignment afterward

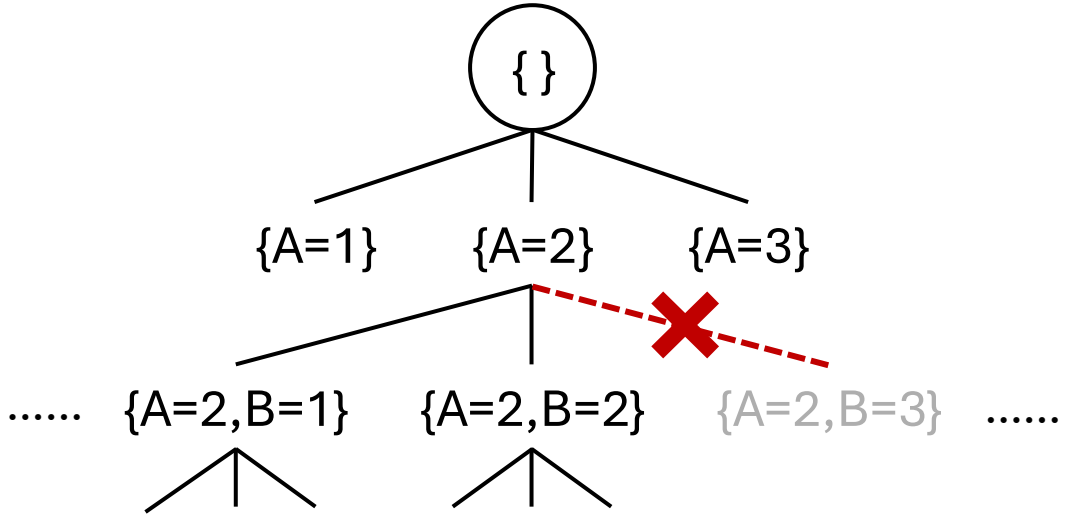
Combinatorial Explosion!

$$d^n$$

n variables, domain size d

Backtracking Search

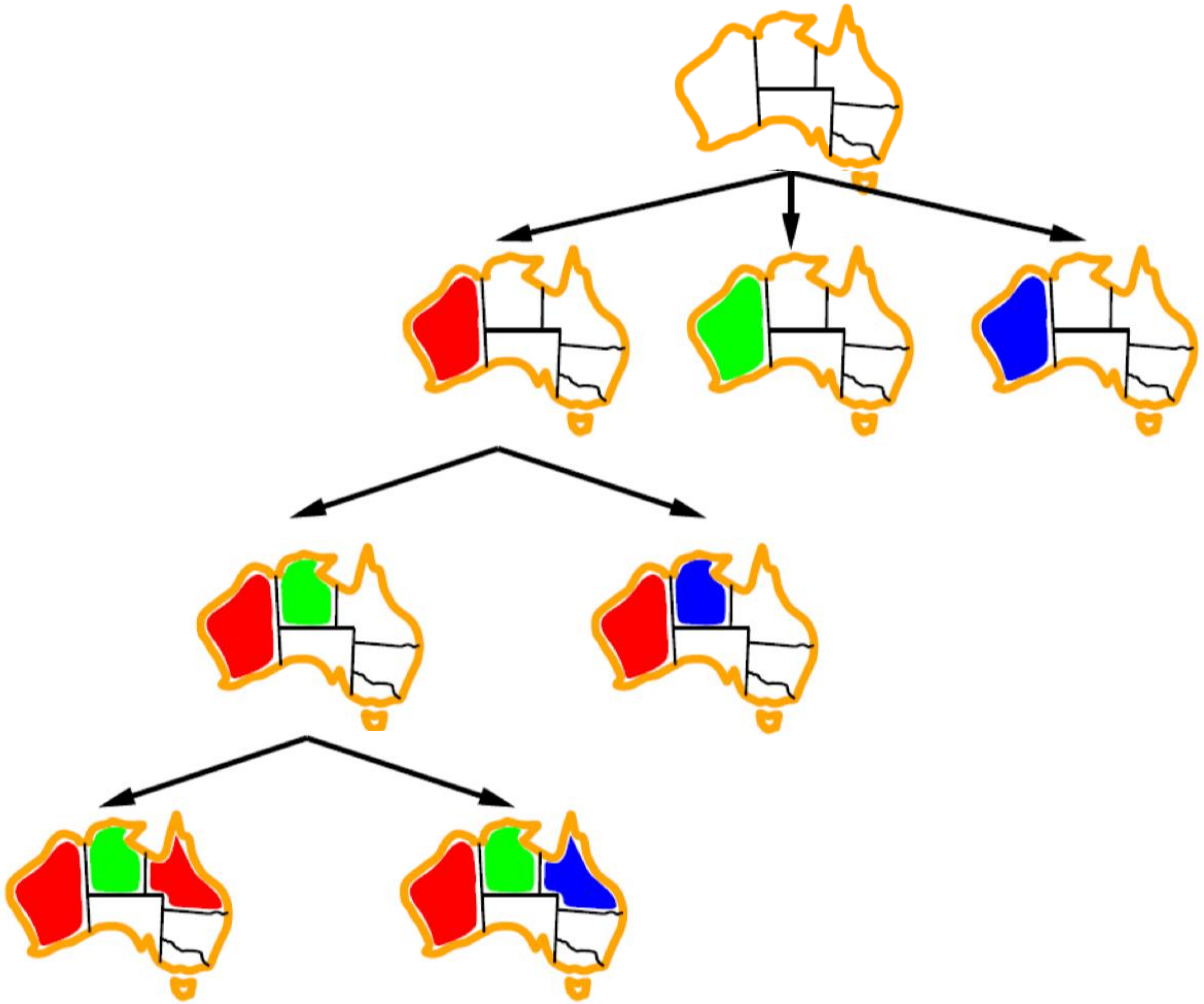
- Assign variables one by one.
- Check constraints after each assignment.
- If inconsistent: Backtrack immediately.
- Key Intuition: **Fail Early**.

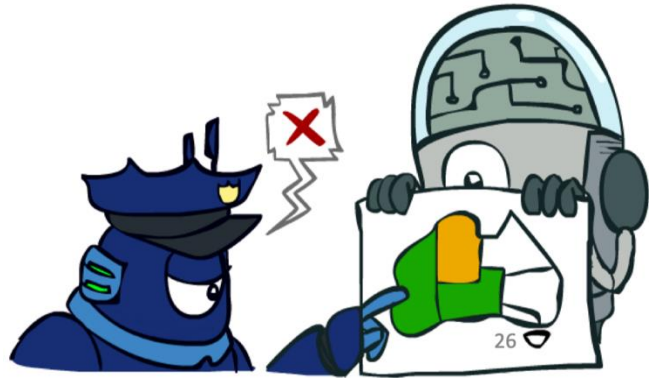


Pseudo Code of Backtracking

```
function Backtrack(assignment, csp):
    if assignment is complete:
        return assignment
    var <= Select-Unassigned-Variable(csp)
    for each value v in Order-Domain-Values(var):
        if v is consistent with assignment:
            add {var = v} to assignment
            result <= Backtrack(assignment, csp)
            if result != failure:
                return result
            remove {var = v} from assignment // undo
    return failure
```

Example





Why Backtracking Works Better

- Standard path search represents the same assignment in many different orders.
- Commutativity: The order of assigning variables does not affect the final solution.
- *Example: {A=1, B=2} is the same as {B=2, A=1}.*
- Branching factor is significantly reduced as we only consider one variable at each depth.

Can we detect inevitable failure early?

Variable Ordering Matters

Which variable should we assign next?

MRV

Minimum Remaining Values

Degree Heuristic

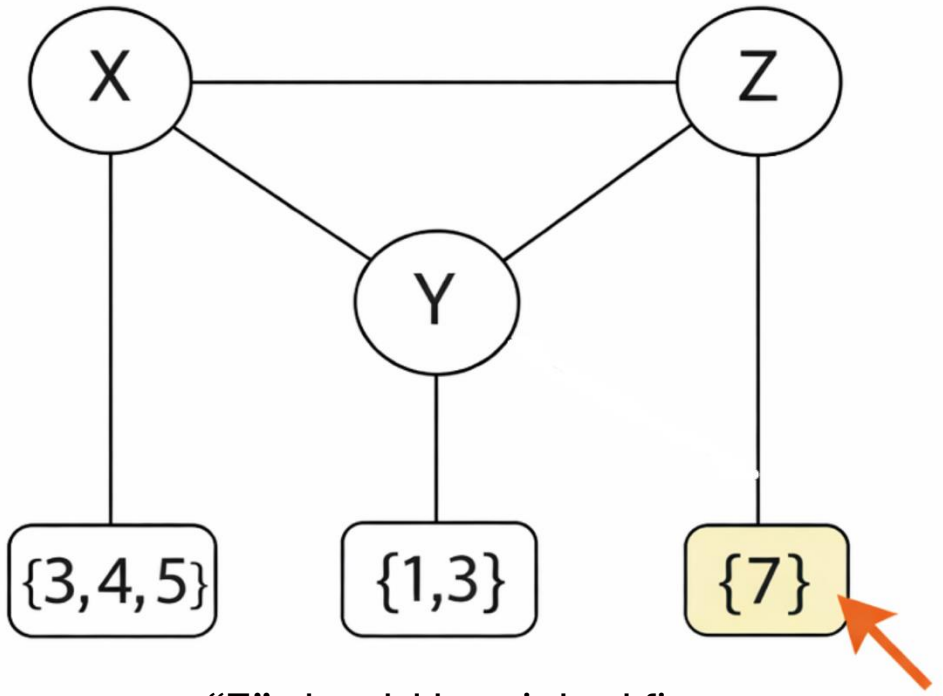
Most Constraining Variable

A good ordering can make search dramatically faster.

MRV Heuristic

Minimum Remaining Values

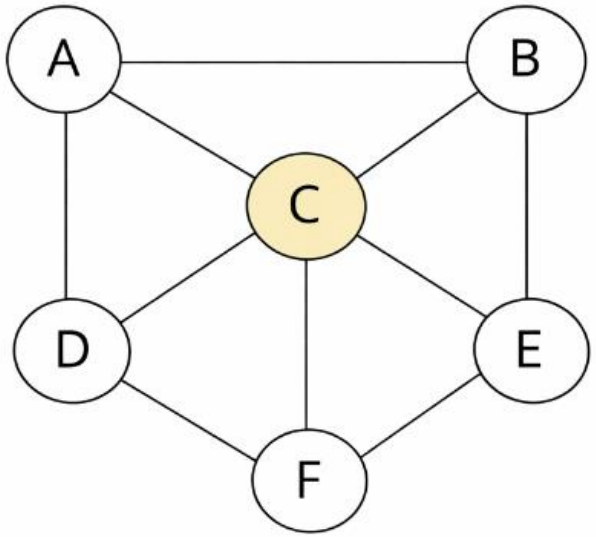
- Choose the variable with the fewest legal values left in its domain.
- Also known as the "Most Constrained Variable" or "Fail-First" heuristic.
- **Intuition:** Tackle the hardest part of the problem first.



"Z" should be picked first

Degree Heuristic

- Often used as a tie-breaker for MRV.
- Choose the variable that constrains the largest number of unassigned variables.
- **Intuition:** Pick the variable that has the most impact on future choices.

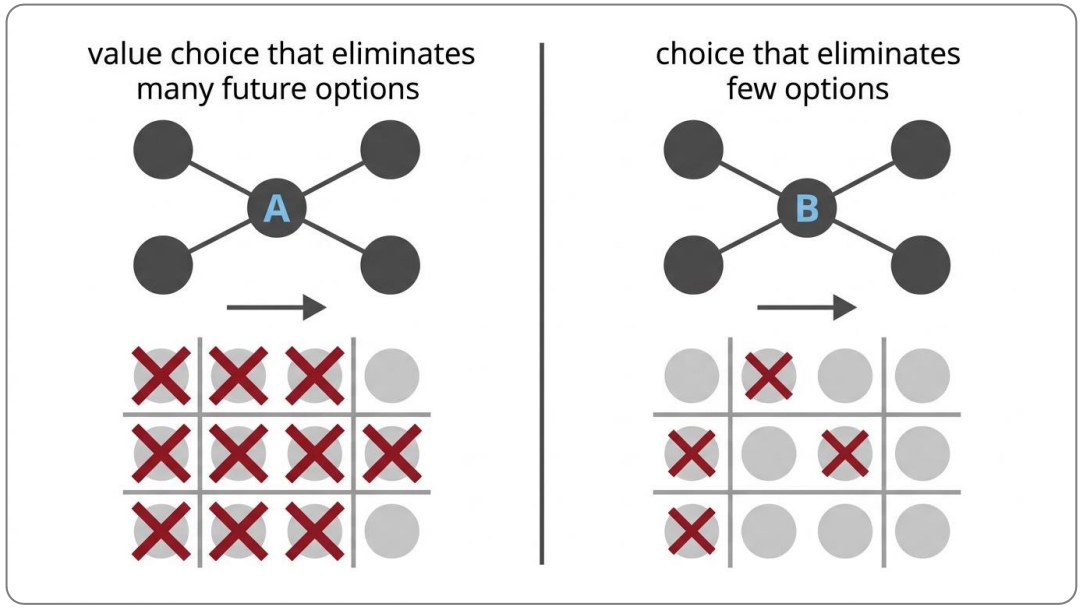


"C" should be picked first

Value Ordering

Least Constraining Value (LCV)

- Given a variable, choose the value that eliminates the fewest options for its neighbors.
- **Goal:** Preserve as much flexibility as possible for future assignments.



Beyond Backtracking: Propagation

Search alone is blind to future consequences.

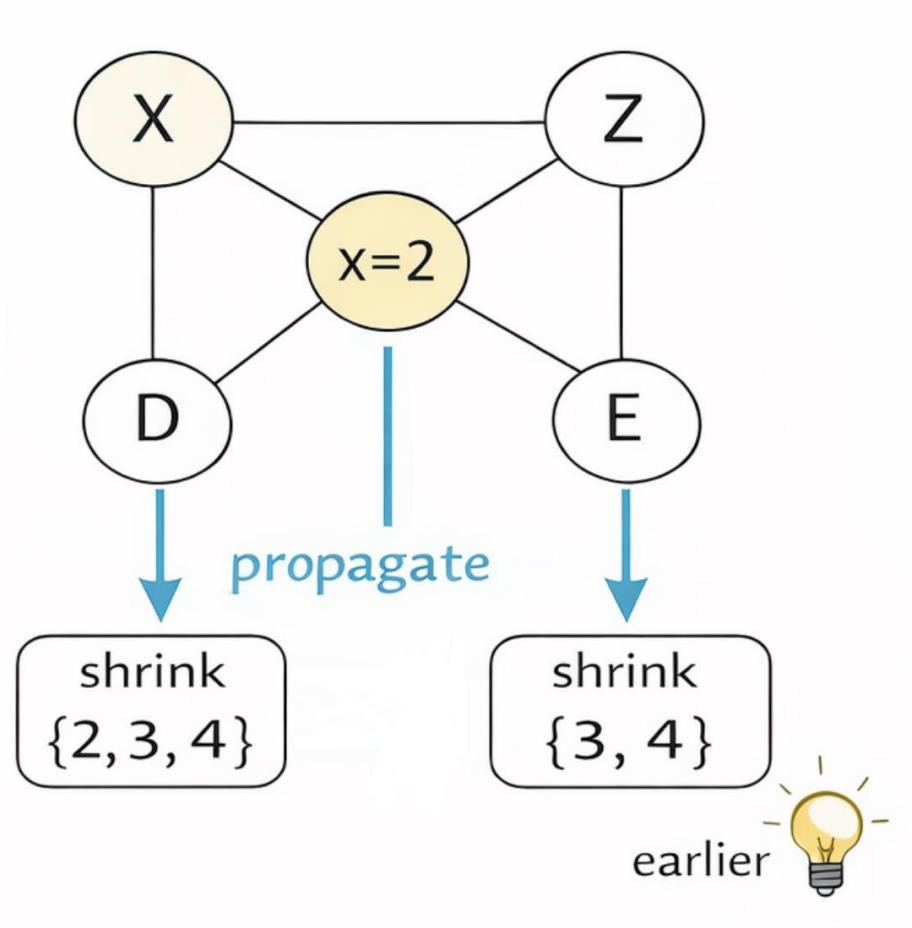
Constraint Propagation:

- After assigning a variable, propagate the results to neighbors.
- Shrink domains of related variables.
- Identify dead-ends early before searching deeper.

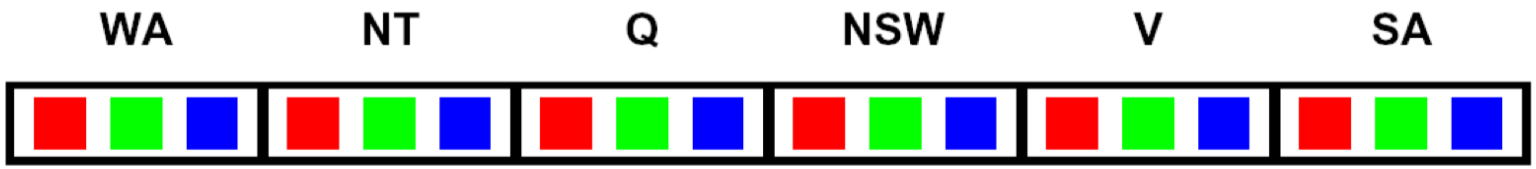
Forward Checking

When variable X is assigned:

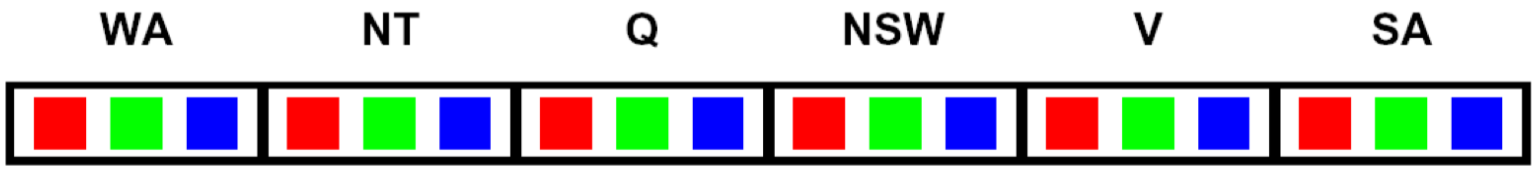
- Look at each **unassigned** variable Y connected to X.
- Delete any value from Y's domain that is inconsistent with X's value.
- If any domain becomes empty, failure.



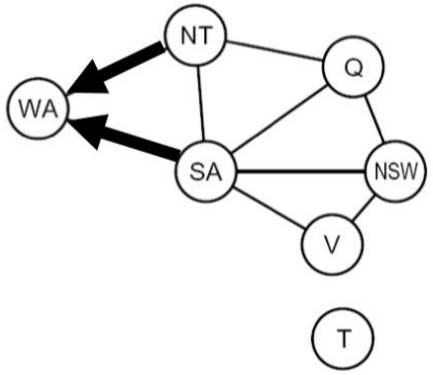
Map Coloring: Forward Checking



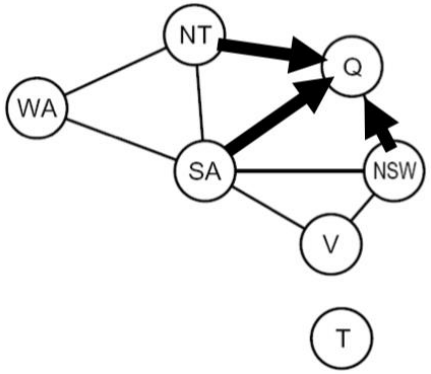
Map Coloring: Forward Checking

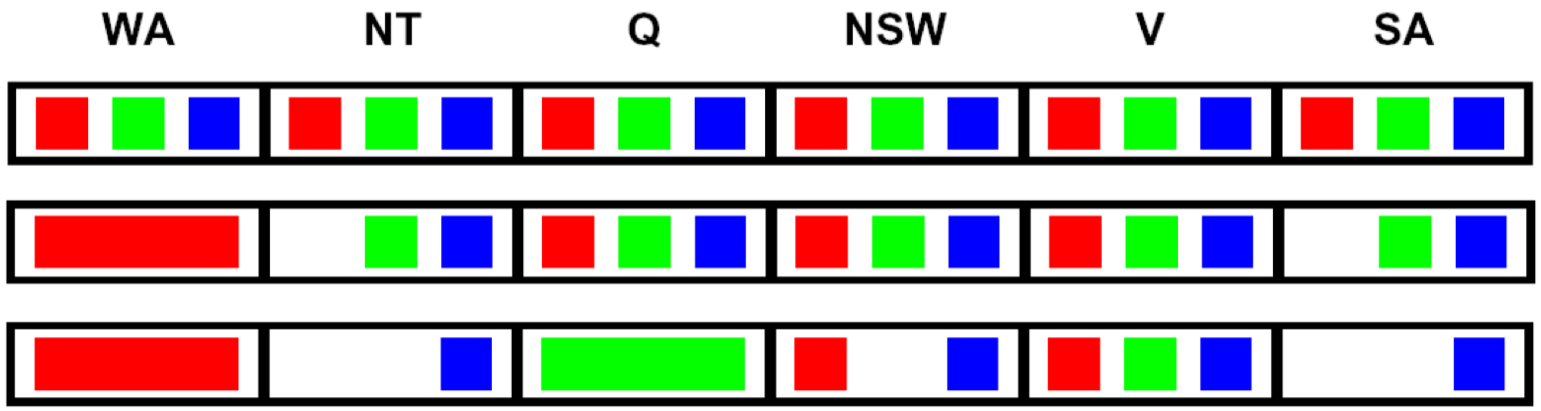


Map Coloring: Forward Checking

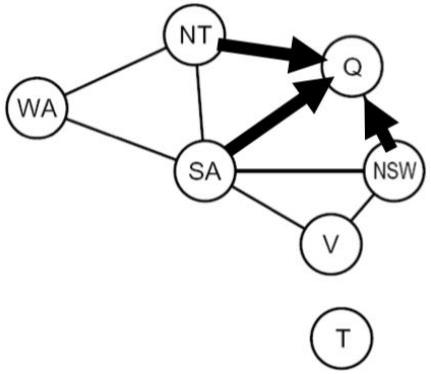


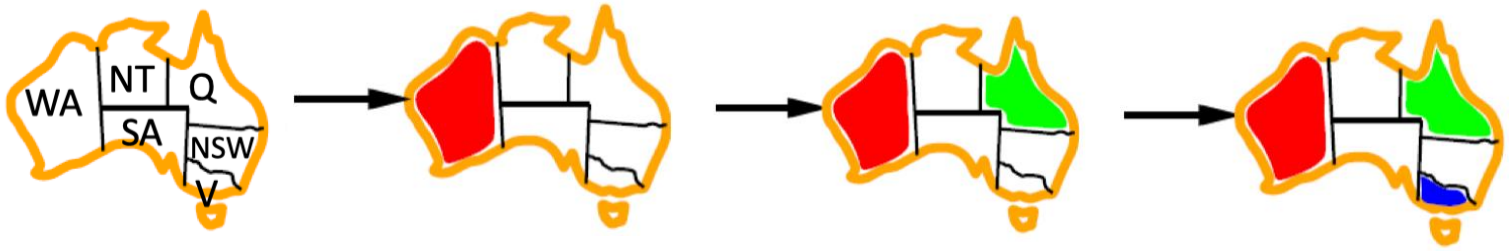
Map Coloring: Forward Checking

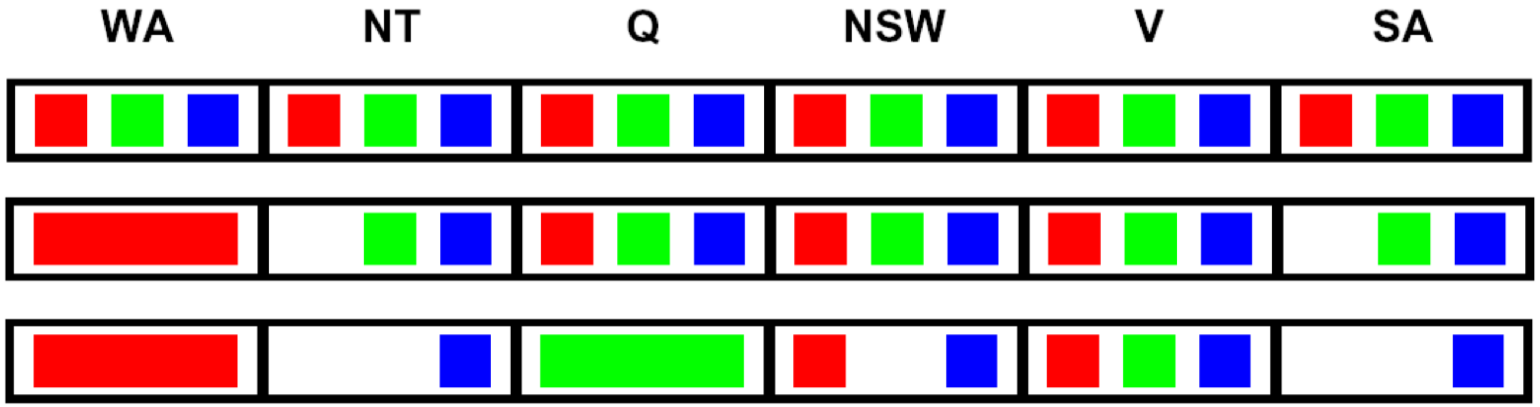




Map Coloring: Forward Checking

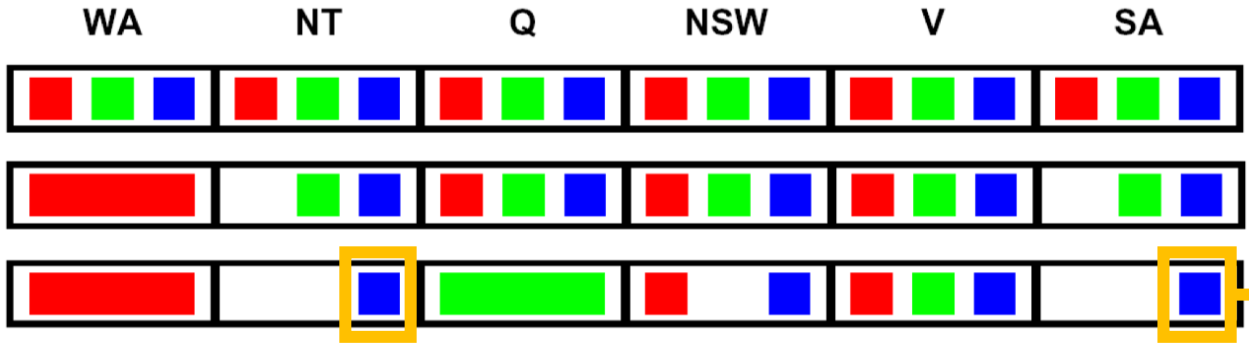






Fail: NT has no possible values

Limitation of Forward Checking



NT and SA cannot be both blue!
Why didn't we detect this yet?

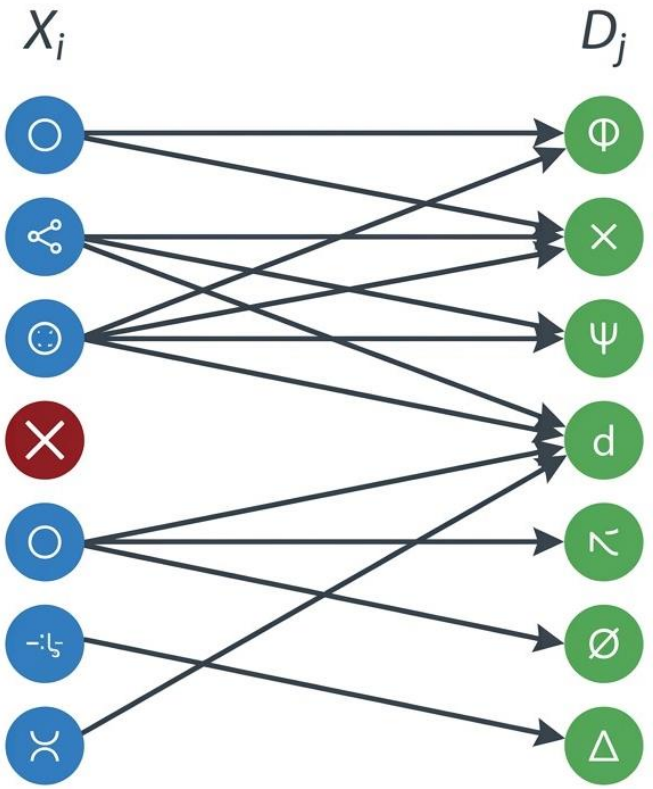
- Forward checking only propagates information from **assigned** to **unassigned** variables.
- It does not check for consistency between unassigned variables.

We need stronger propagation: Arc Consistency.

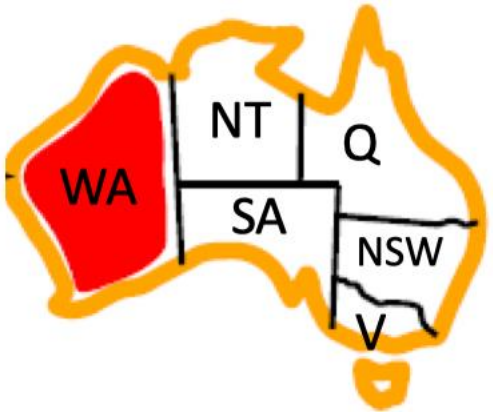
Arc Consistency

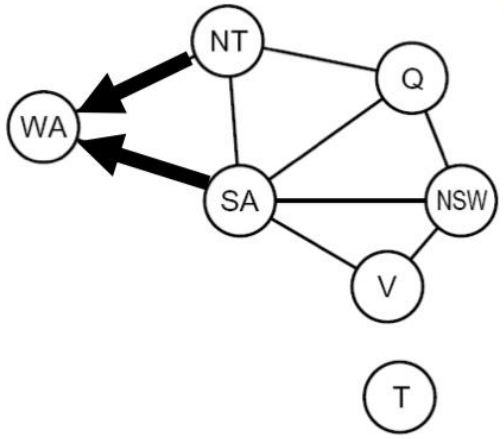
An arc X → Y is consistent if:

- For every value x in Domain(X)…
  …there is some value y in Domain(Y) that satisfies the constraint.
- Enforcing consistency = deleting unsupported values.

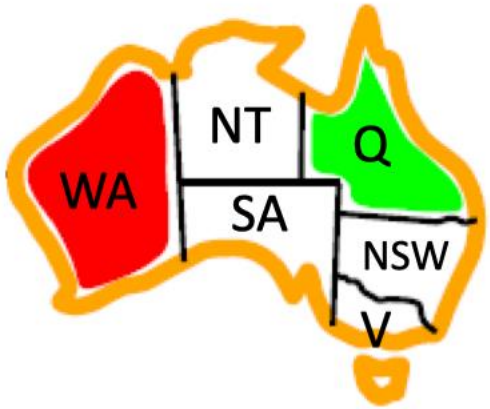


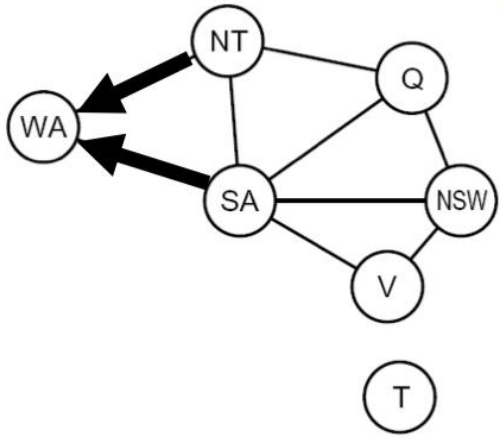
Map Coloring: Single Arc Consistency





Map Coloring: Single Arc Consistency



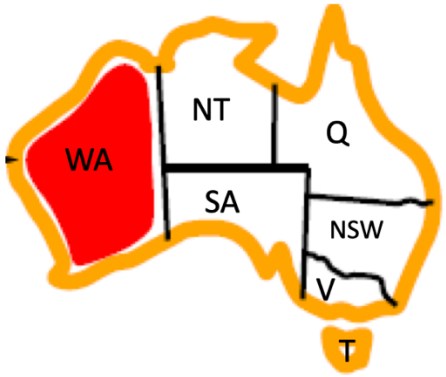


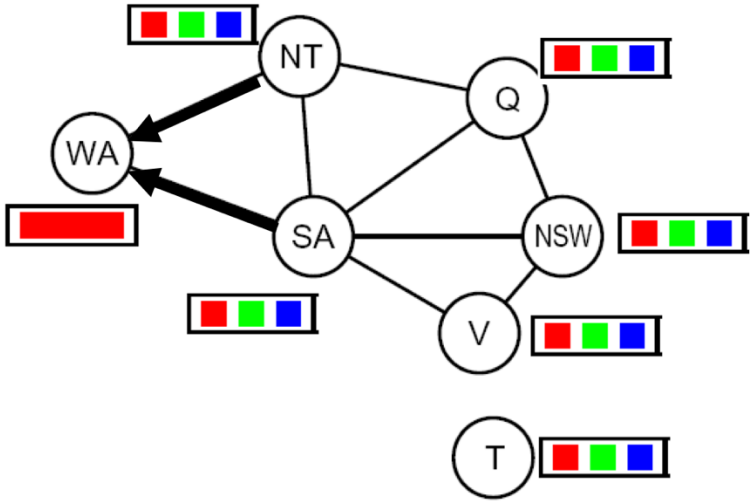
AC-3 Algorithm

Algorithm for enforcing arc consistency globally:

1. Initialize a queue with all arcs in the CSP.
2. Pop an arc (X_i, X_j) from the queue.
3. Make X_i arc-consistent with respect to X_j.
   (remove inconsistent values in X_i)
4. If X_i's domain was reduced, re-add all arcs $(\{X_k, X_i\})$ pointing to X_i to the queue.
5. Repeat until the queue is empty.

Map Coloring: AC-3 (1)



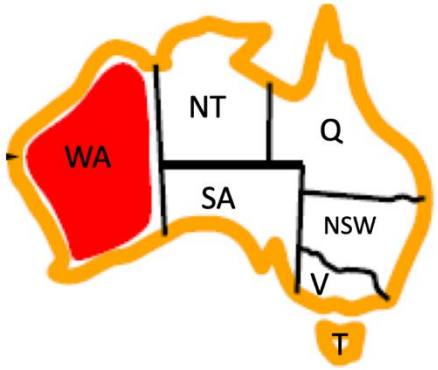


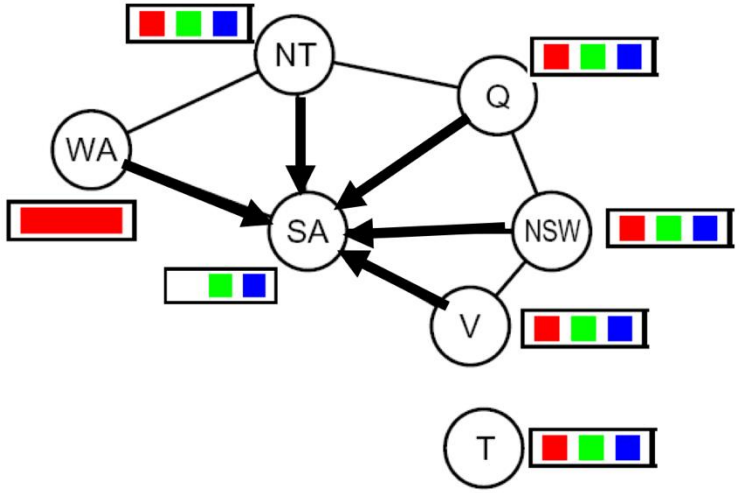
Queue:

SA->WA

NT->WA

Map Coloring: AC-3 (2)





Queue:

~~SA->WA~~

NT->WA

WA->SA

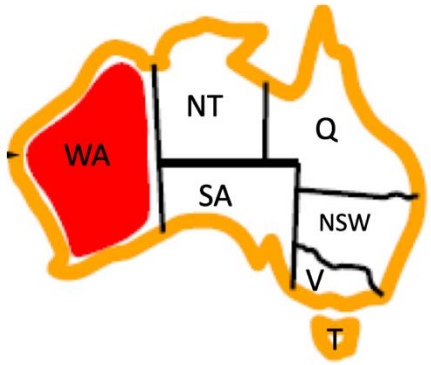
NT->SA

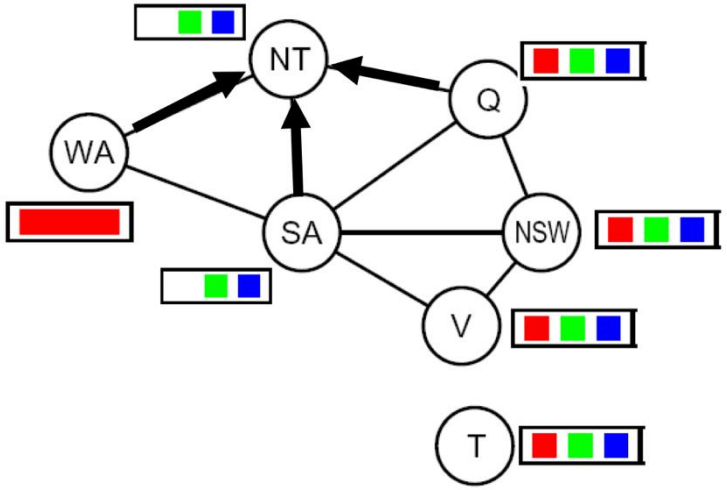
Q->SA

NSW->SA

V->SA

Map Coloring: AC-3 (3)





Queue:

~~SA->WA~~

~~NT->WA~~

WA->SA

NT->SA

Q->SA

NSW->SA

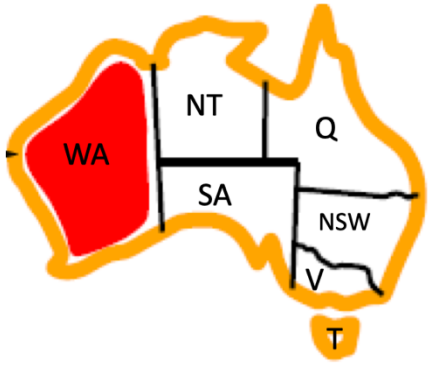
V->SA

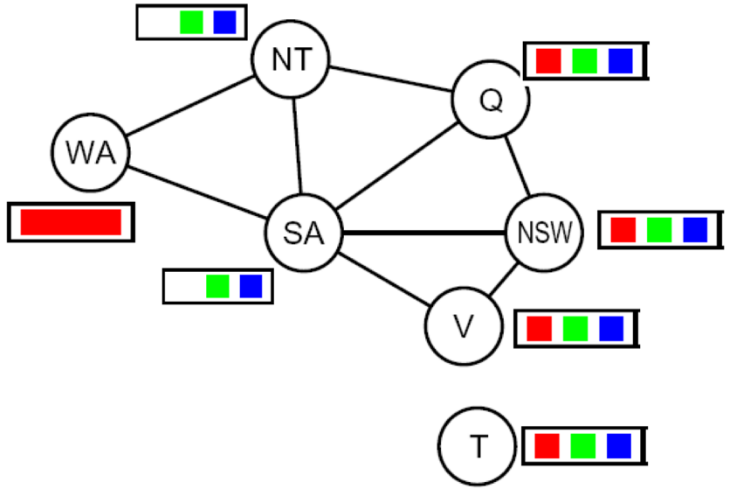
WA->NT

SA->NT

Q->NT

Map Coloring: AC-3 (4)





Queue:

~~SA->WA~~

~~NT->WA~~

~~WA->SA~~

NT->SA

Q->SA

NSW->SA

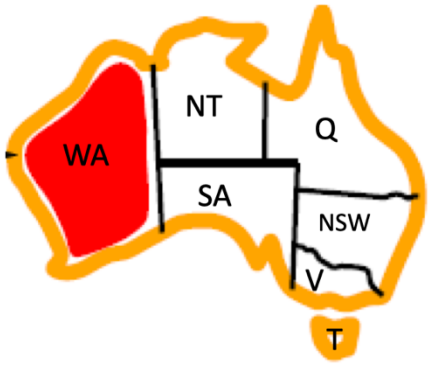
V->SA

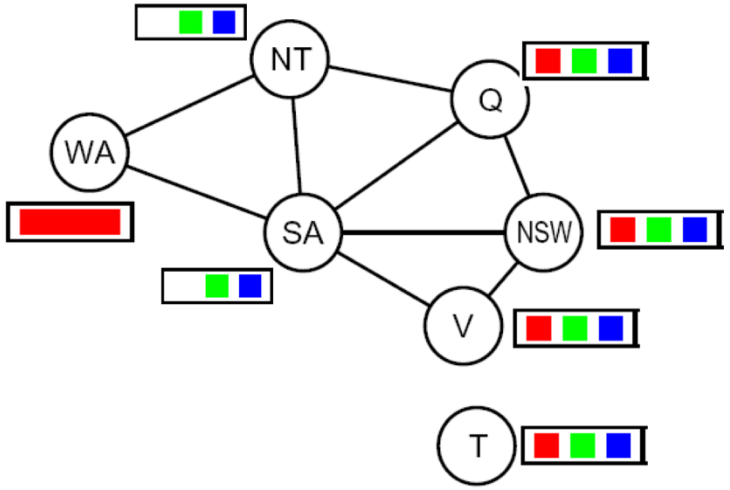
WA->NT

SA->NT

Q->NT

Map Coloring: AC-3 (5)





Queue:

~~SA->WA~~

~~NT->WA~~

~~WA->SA~~

~~NT->SA~~

~~Q->SA~~

~~NSW->SA~~

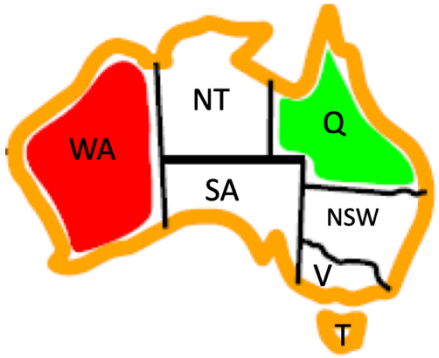
~~V->SA~~

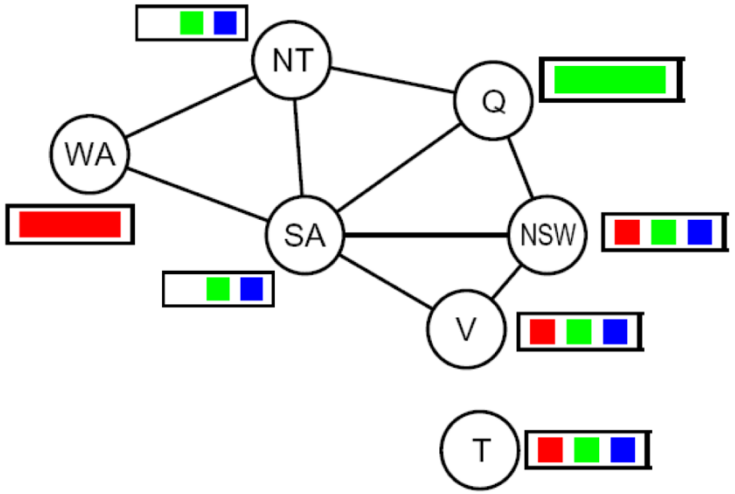
~~WA->NT~~

~~SA->NT~~

~~Q->NT~~

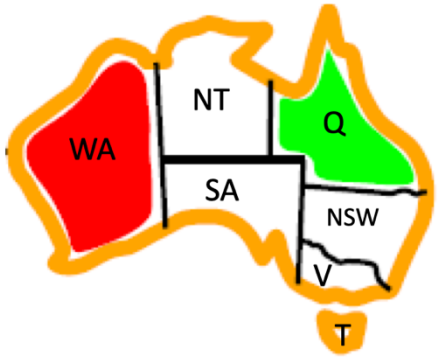
Map Coloring: AC-3 (6)

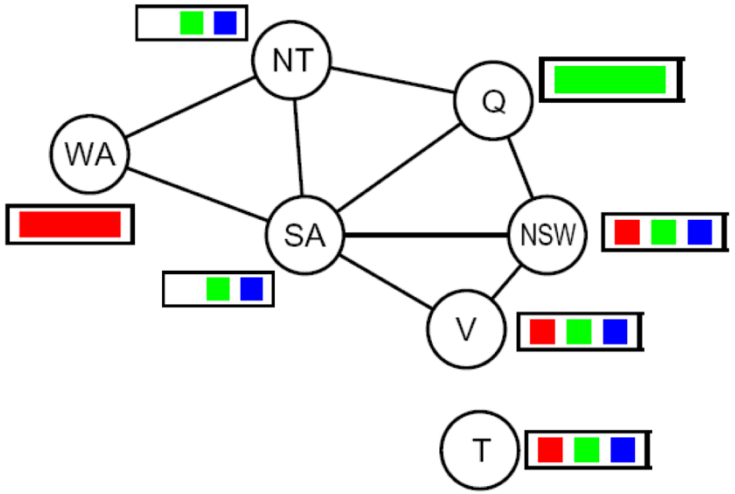




Queue:

Quiz: What Would be Added to the Queue?





Queue:

A: NSW->Q, SA->Q, NT->Q

B: Q->NSW, Q->SA, Q->NT

Limitations of Arc Consistency

- After enforcing arc consistency:
  - Can have one solution left
  - Can have multiple solutions left
  - Can have no solutions left (and not know it)
- Arc consistency still runs inside a backtracking search!
- And will be called many times

How the Pieces Fit Together

Modern CSP Solver Recipe

Backtracking Search

+

Ordering Heuristics (MRV, Degree, LCV)

+

Constraint Propagation (AC-3)

Search + Structure + Propagation

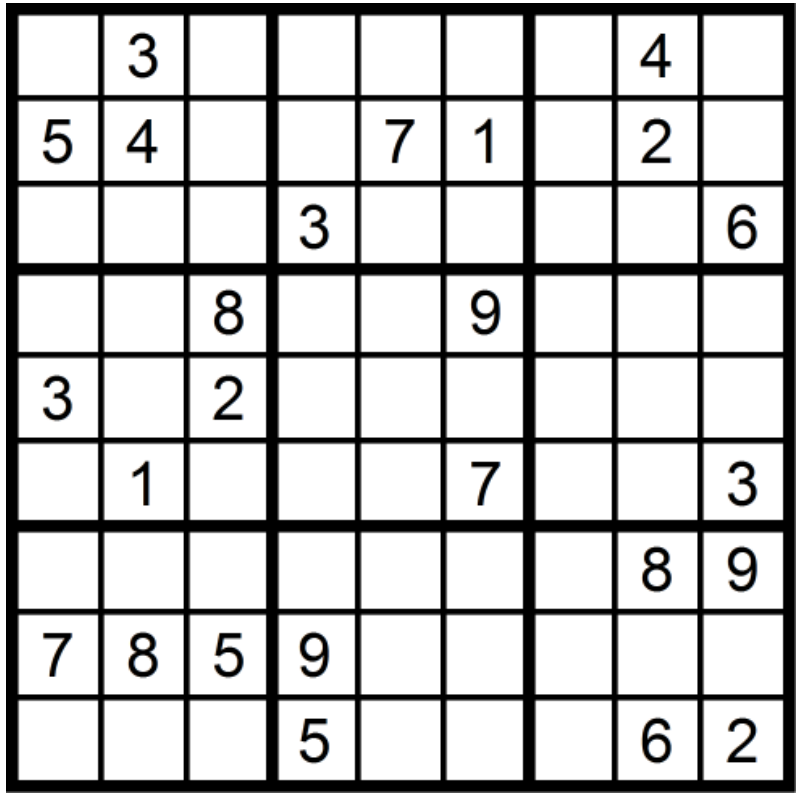
Complexity and Reality

- General CSP is NP-complete.
- However, **structure** matters immensely.
- Tree-structured CSPs can be solved in **linear time** $O(n\, d^2)$.
- Good heuristics can solve many massive instances in practice.

Sudoku as a CSP Solver Story

A human/AI solver doesn't just "guess".

- **Constraint Propagation:** Eliminating impossible numbers in a cell.
- **Ordering:** Filling the cell with the fewest candidates first (MRV).
- **Backtracking:** Making a trial assignment when propagation stops.



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

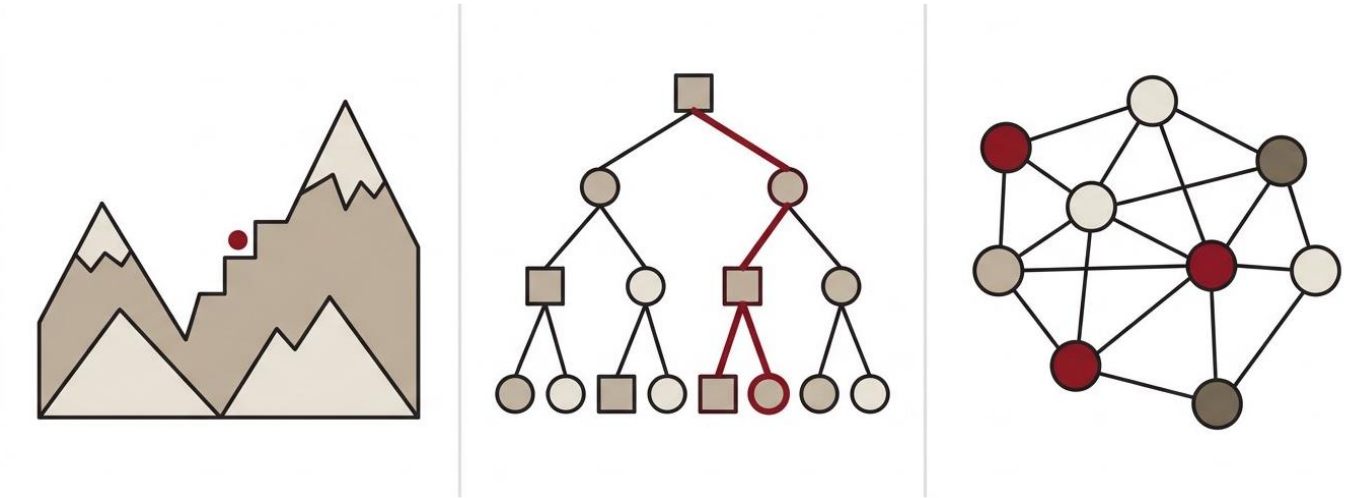
Key Takeaways

- **CSP Modeling:** Variables, Domains, Constraints.
- **Backtracking:** Depth-first search with early failure pruning.
- **Heuristics:** MRV and Degree for variable choice, LCV for value choice.
- **Propagation:** Forward checking and Arc Consistency (AC-3) exploit graph structure.

Week 3 Summary

Three paradigms of "Search" in AI:

- **Local Search:** Optimization of state (hill climbing).
- **Adversarial:** Minimax against an opponent.
- **CSP:** Reasoning over structured assignments.



Next Week

Machine Learning Foundations