Artificial Intelligence

Ch-6

[Constraint Satisfaction Problems]

Ass. Prof. Taher Hamza
Dr. Sara Elmetwally

Contents

1 Constraint Satisfaction Problems

2 Backtracking for CSP

3 Improving backtracking efficiency

CSP

- CSP is a type of problems which is defined by:
  - Set of variables X_1, X_2,.....X_n
  - The domains (values) for each variables D_1,D_n,....D_m
  - Set of constraint C_i to define the relation between variables and values

- The goal is the complete assignment of variables where no constraint is violated

CSP

- Assignment A is defined as, assign value D_i from domain set to variable X_i from variable set

  EX A: $X_1=4$, $X_3=7$,……

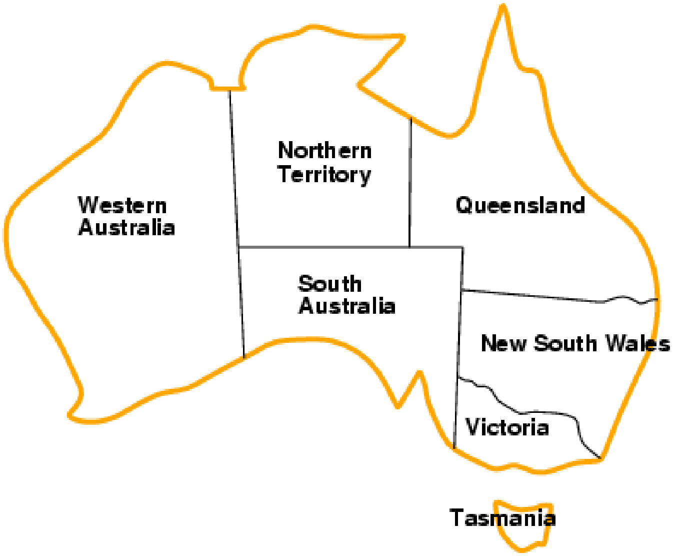
- A is called legal assignment if it does not violate any constraints C_i

- A is called complete assignment if all problems variables have assigned values

- If the complete assignment is legal, it is called solution

CSP, example

Map coloring problem

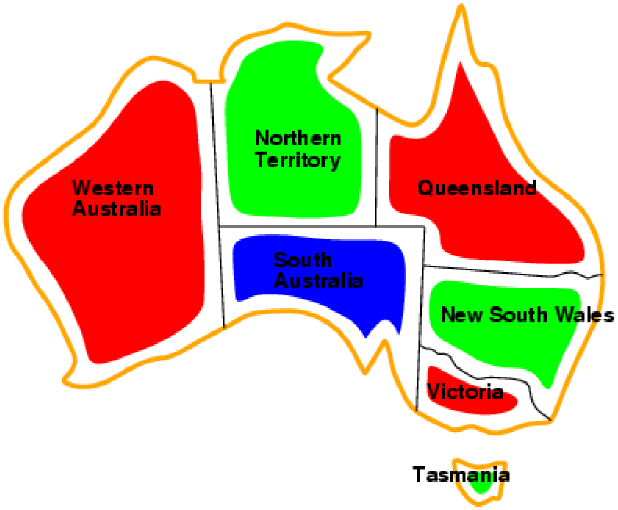


- ✓ Variables:{WA, NT, SA, Q, NSW, V, T}
- ✓ Domains:{red, green, blue}
- ✓ Constraints: no adjacent countries have the same color

  WA ≠ NT, WA ≠ SA, NT ≠ SA,.....

CSP, example

Map coloring problem



- ✓ Variables:{WA, NT, SA, Q, NSW, V, T}
- ✓ Domains:{red, green, blue}
- ✓ Constraints: no adjacent countries have the same color

  WA ≠ NT, WA ≠ SA, NT ≠ SA,.....

Solution : **WA** = red, **NT** = green, **Q** = red, **NSW** = green,
V = red, **SA** = blue, **T** = green

CSP, example

Cryptarithmetic Puzzle

$$
\begin{array}{cccc}
 & T & W & O \\
+ & T & W & O \\
\hline
F & O & U & R
\end{array}
$$

- ✓ Variables: $\{F,T,W,O,U,R,X_1,X_2,X_3\}$
- ✓ Domains: $\{0,1,2,3,4,5,6,7,8,9\}$
- ✓ Constraints: $O+O=R+10.X_1$

  $X_1 + W+W=U+10.X_2$

  $X_2 + T+T=O+10.X_3$

  $X_3 = F$

  $T,F \neq 0$

  All variables are different($T\neq W, T\neq U, T\neq O$,………)

CSP, example

Cryptarithmetic Puzzle

- ✓ Variables: $\{F,T,W,O,U,R,X_1,X_2,X_3\}$
- ✓ Domains: $\{0,1,2,3,4,5,6,7,8,9\}$
- ✓ Constraints: $O+O=R+10.X_1$

  $X_1 + W + W = U + 10.X_2$

  $X_2 + T + T = O + 10.X_3$

  $X_3 = F$

  $T,F \neq 0$

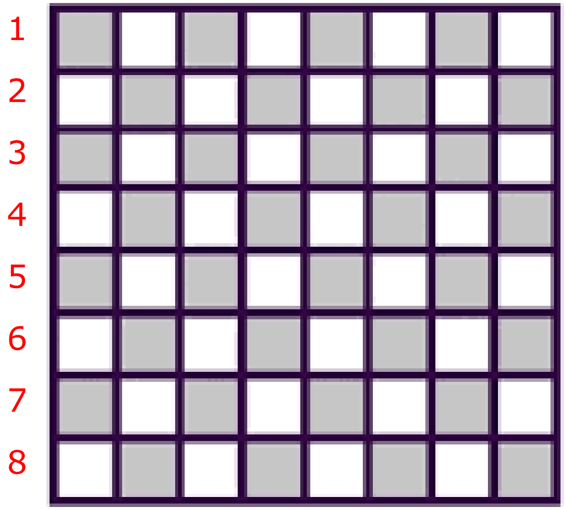
  All variables are different($T \neq W, T \neq U, T \neq O$,………)

$$\begin{array}{cccc} & T & W & O \\ + & T & W & O \\ \hline F & O & U & R \end{array}$$

Solution : **T** = 9, **W** = 3, **O** = 8,
F = 1, **U** = 7, **R** = 6

CSP, example

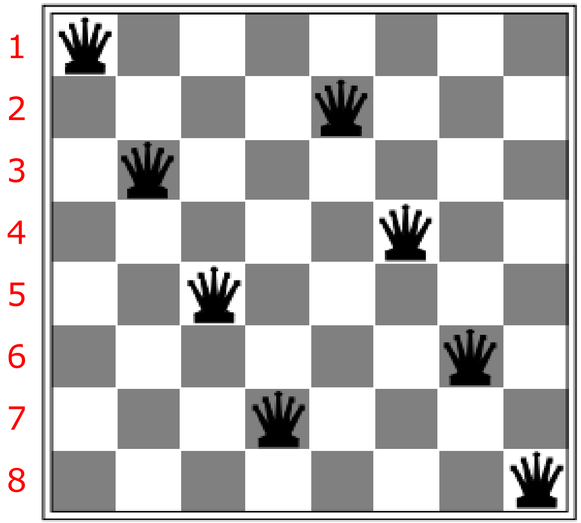
N-Queens



- Variables: $\{Q_1, Q_2, Q_3, Q_4, Q_5, Q_6, Q_7, Q_8\}$
- Domains: $\{1,2,3,4,5,6,7,8\}$
- Constraints: $Q_i \neq Q_j$, (row)
  $|Q_i - Q_j| \neq |i-j|$ (diagonal)

CSP, example

N-Queens



- ✓ Variables: $\{Q_1, Q_2, Q_3, Q_4, Q_5, Q_6, Q_7, Q_8\}$
- ✓ Domains: $\{1,2,3,4,5,6,7,8\}$
- ✓ Constraints: $Q_i \neq Q_j$, (row)

  $|Q_i - Q_j| \neq |i-j|$ (diagonal)
- ✓ **Solution** : $Q_1 = 1$, $Q_2 = 3$, $Q_3 = 5$, $Q_4 = 7$,

  $Q_5 = 2$, $Q_6 = 4$, $Q_7 = 6$, $Q_8 = 8$

CSP

- Types of constraints:

  - Unary constraints involve a single variable,

    e.g., $SA \neq \text{green}$

  - Binary constraints involve pairs of variables,

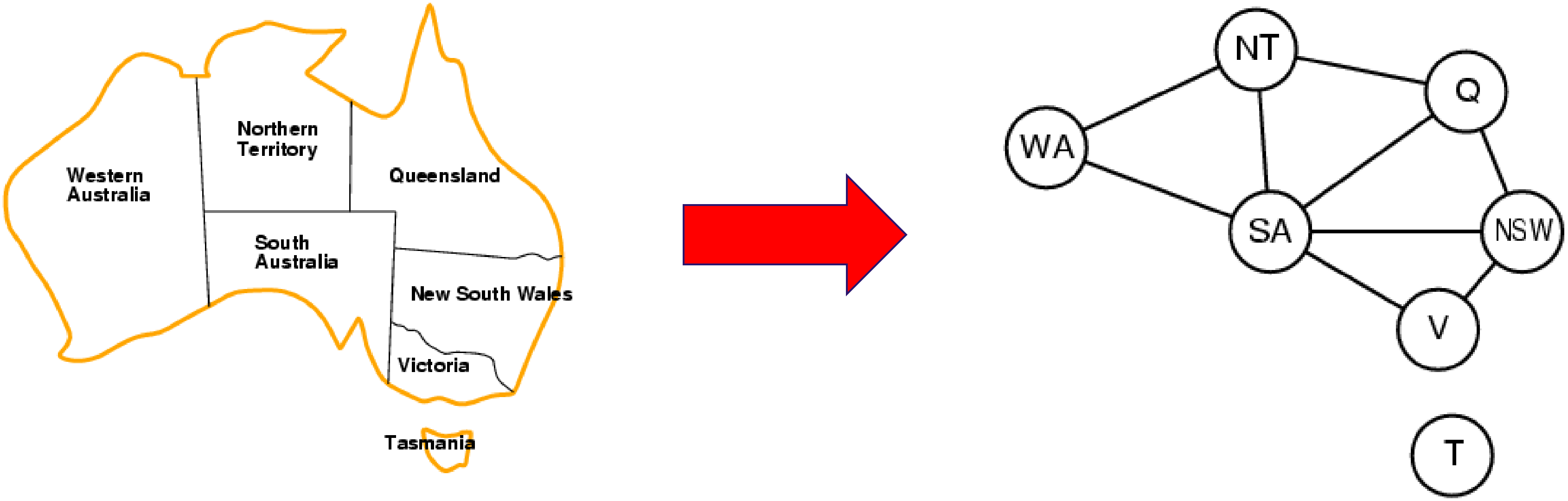
    e.g., $SA \neq WA$

  - Higher-order constraints involve 3 or more variables

    e.g., $O+O=R+10.X1$

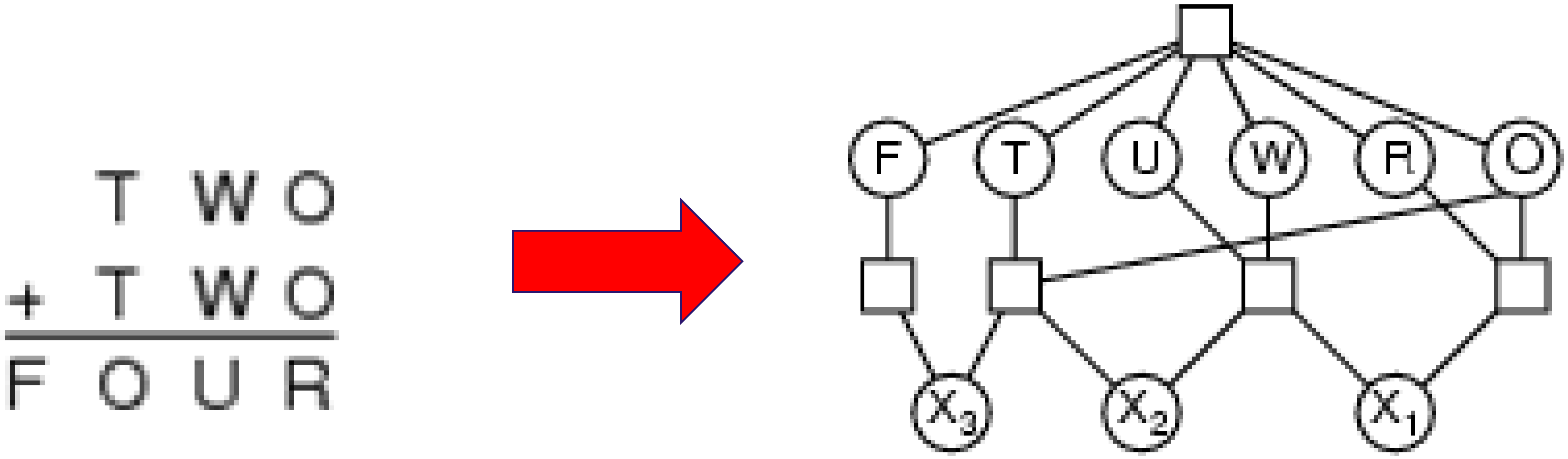
CSP

- Constraint graph:
  - Visualization for problem constraints
  - Constraints represented by arcs
  - Variables represented by nodes



CSP

- Constraint graph:
  - Visualization for problem constraints
  - Constraints represented by arcs
  - Variables represented by nodes



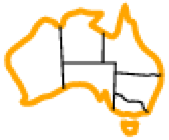
Backtracking for CSP

- Backtracking search:

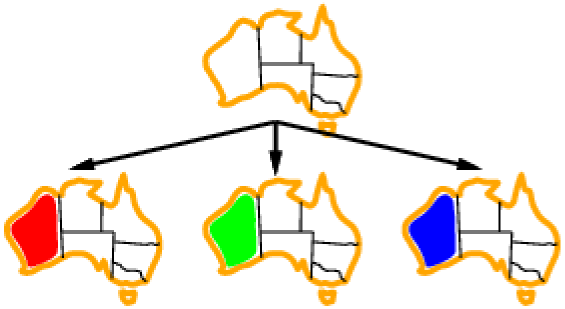
- The order of assignment doesn’t matter

- Do not generate violating assignment
  - Use depth-first search
  - Backtrack if constraints can not be satisfied

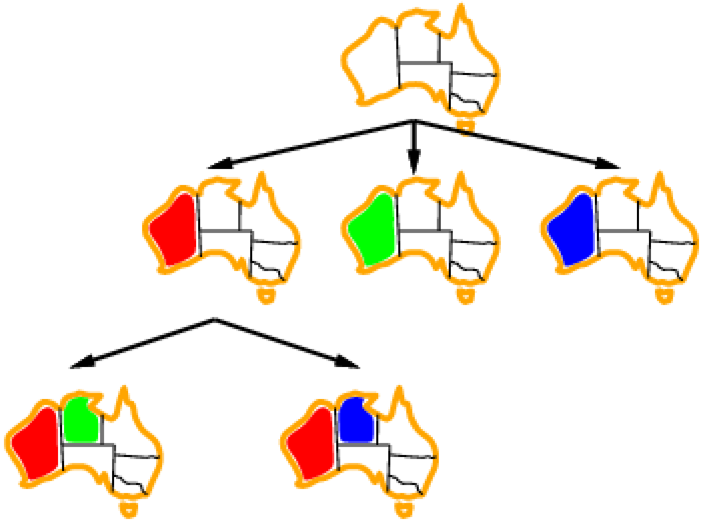
Backtracking, example



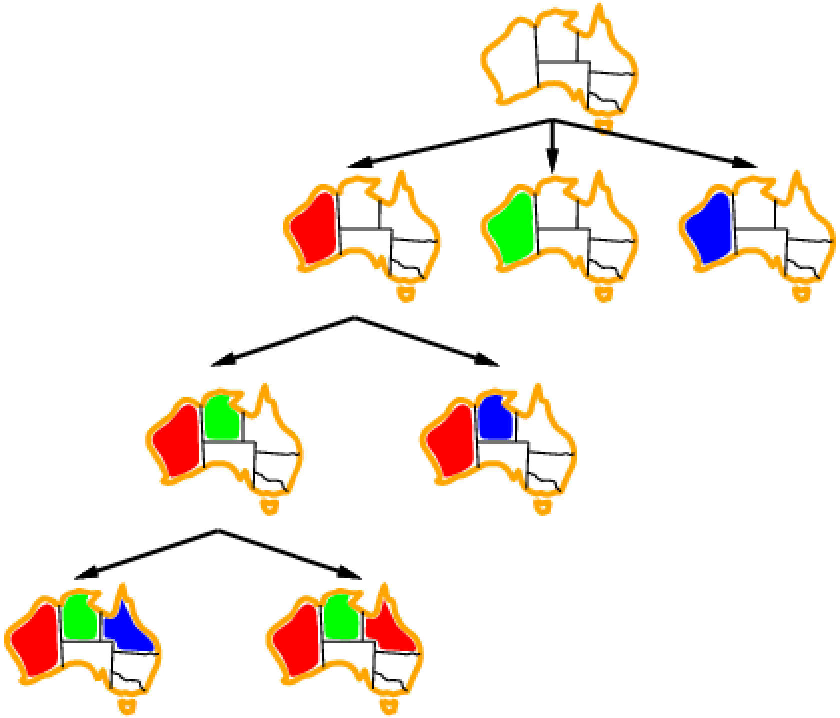
Backtracking, example



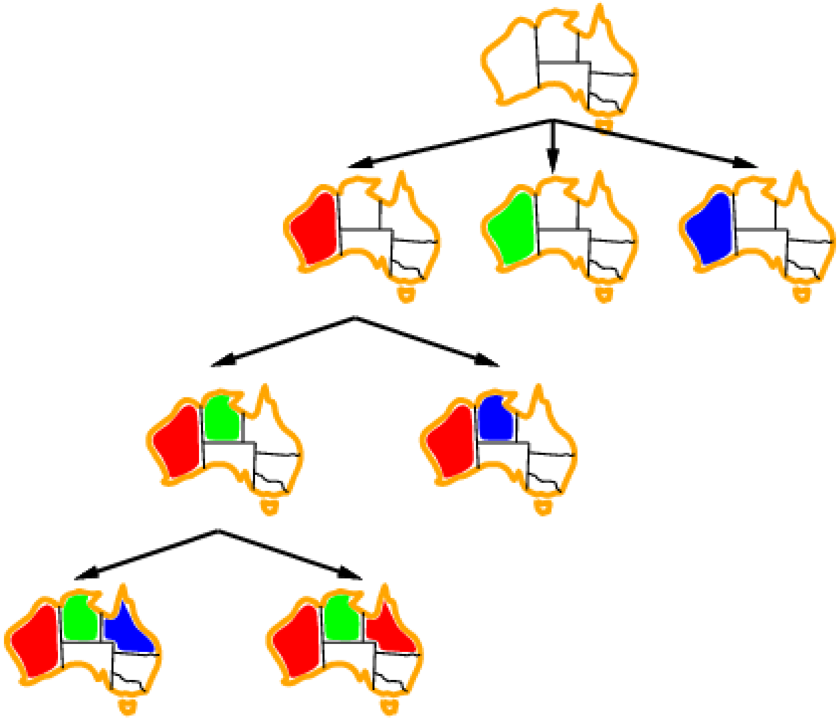
Backtracking, example



Backtracking, example

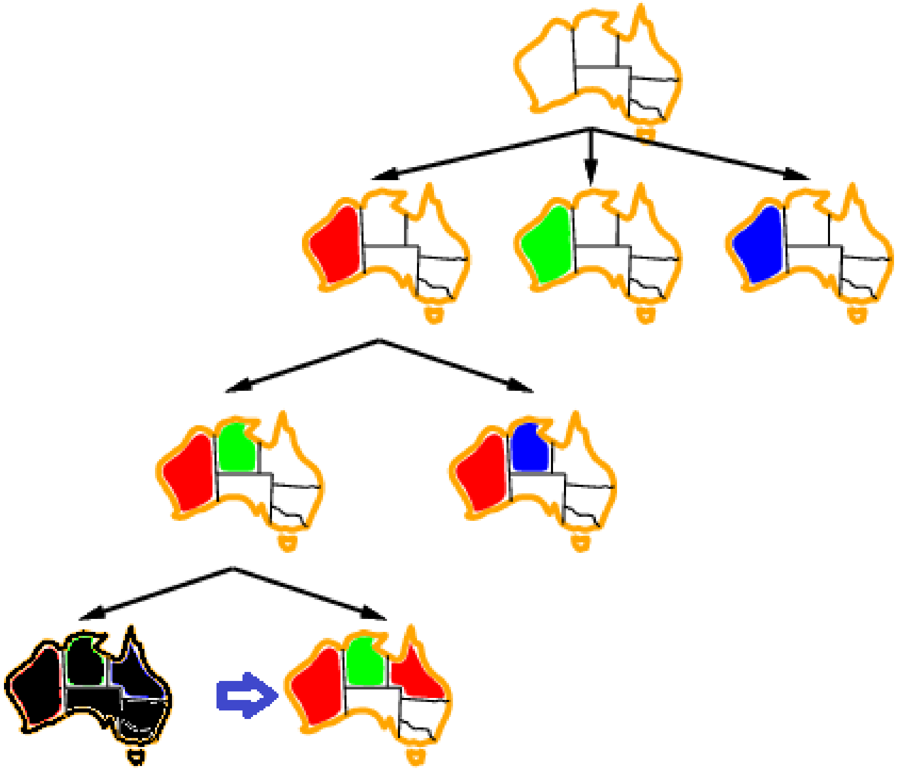


Backtracking, example



Backtrack!

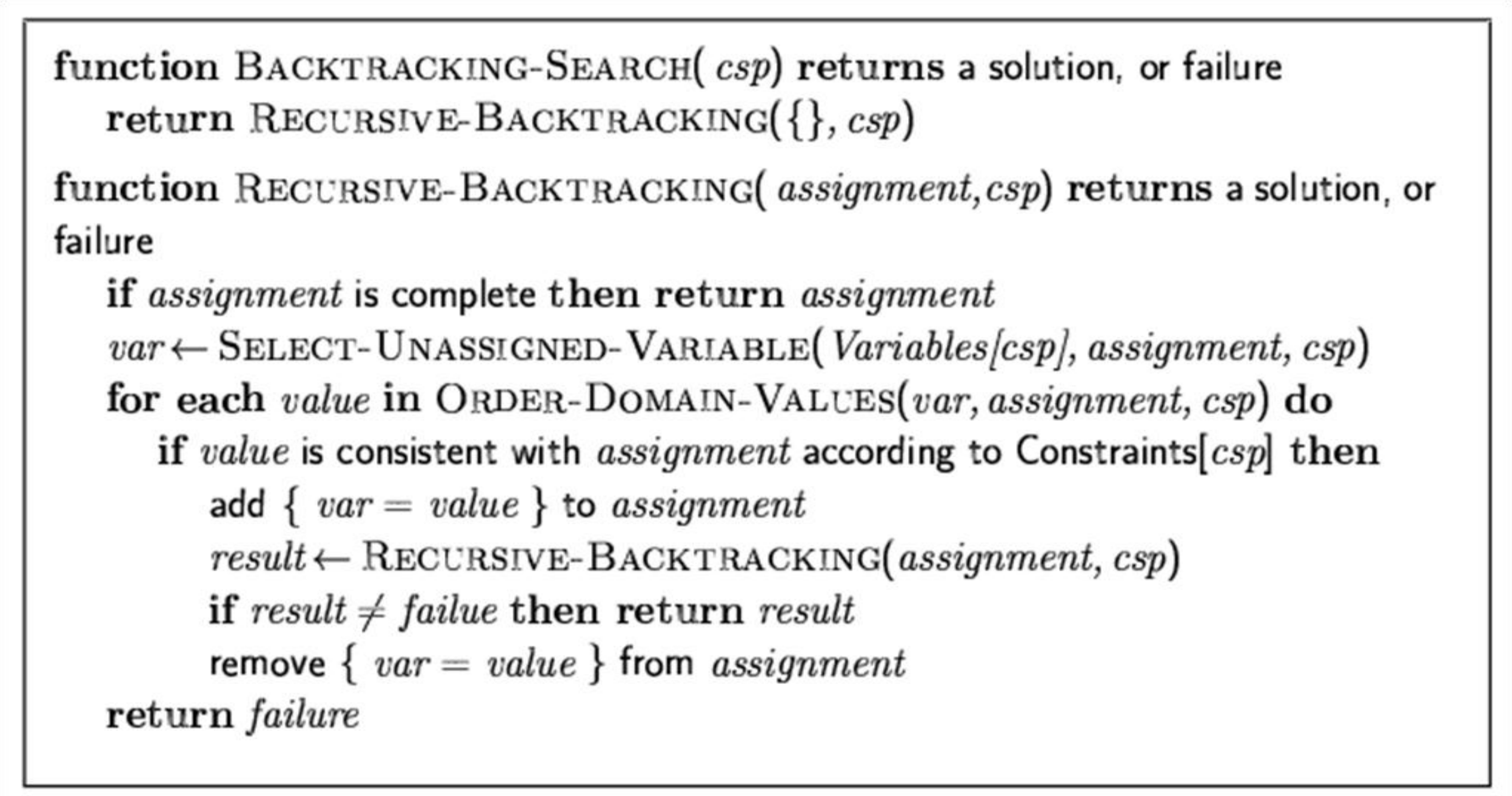
Backtracking, example



Go on to reach your goal

Backtracking for CSP

- Backtracking search:



```
function BACKTRACKING-SEARCH( csp) returns a solution, or failure
    return RECURSIVE-BACKTRACKING({}, csp)

function RECURSIVE-BACKTRACKING( assignment, csp) returns a solution, or
failure
    if assignment is complete then return assignment
    var ← SELECT-UNASSIGNED-VARIABLE( Variables[csp], assignment, csp)
    for each value in ORDER-DOMAIN-VALUES(var, assignment, csp) do
        if value is consistent with assignment according to Constraints[csp] then
            add { var = value } to assignment
            result ← RECURSIVE-BACKTRACKING(assignment, csp)
            if result ≠ failue then return result
            remove { var = value } from assignment
    return failure
```

Can solve N-queens for n ≈ 25

Improving backtracking efficiency

- General-purpose methods can give **huge gains in speed**:

1. Which variable should be assigned next?
2. In what order should its values be tried?
3. Can we detect inevitable failure early?

Improving backtracking efficiency

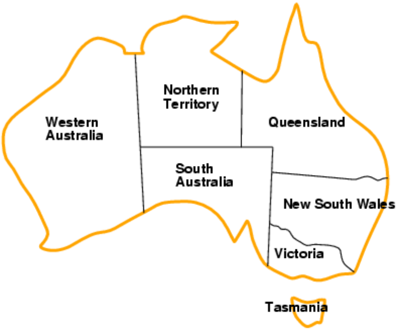
Western Australia
Northern Territory
Queensland
South Australia
New South Wales
Victoria
Tasmania

1. Which variable should be assigned next?

- **Minimum remaining values (MRV):**
  - Choose the variable with the fewest legal values

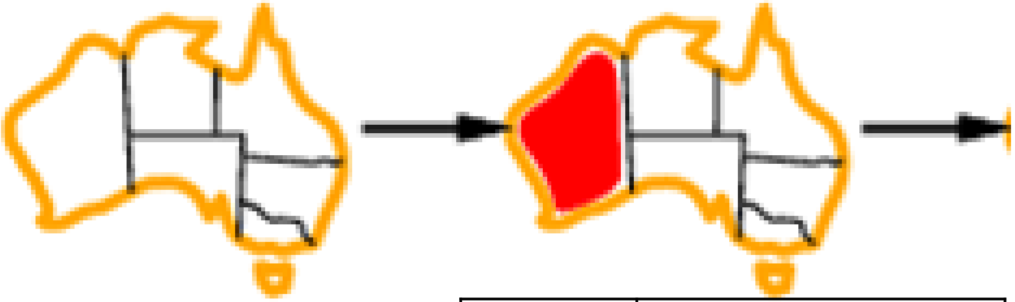
WA	3
NT	3
SA	3
Q	3
NSW	3
V	3
T	3

Improving backtracking efficiency



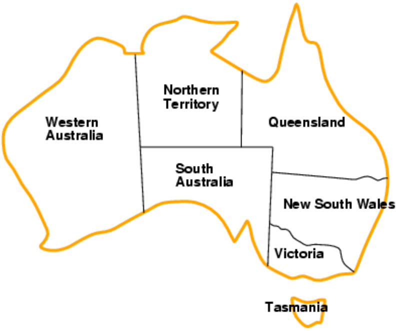
1. Which variable should be assigned next?

- Minimum remaining values (MRV):
  - Choose the variable with the fewest legal values



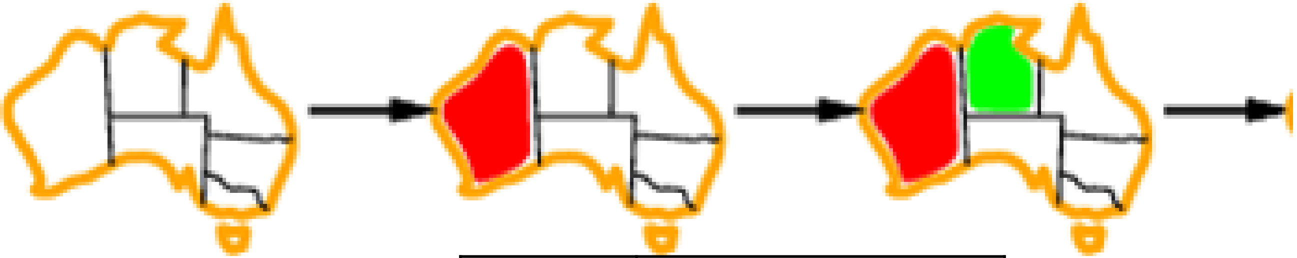
WA	Done..
NT	2
SA	2
Q	3
NSW	3
V	3
T	3

Improving backtracking efficiency



1. Which variable should be assigned next?

- Minimum remaining values (MRV):
  - Choose the variable with the fewest legal values



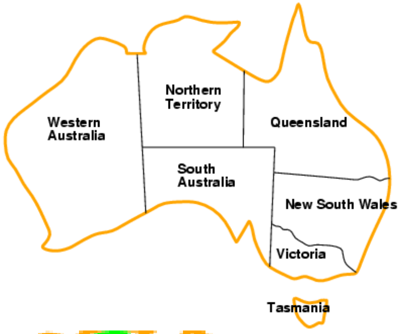
WA	Done..
NT	Done ..
SA	1
Q	2
NSW	3
V	3
T	3

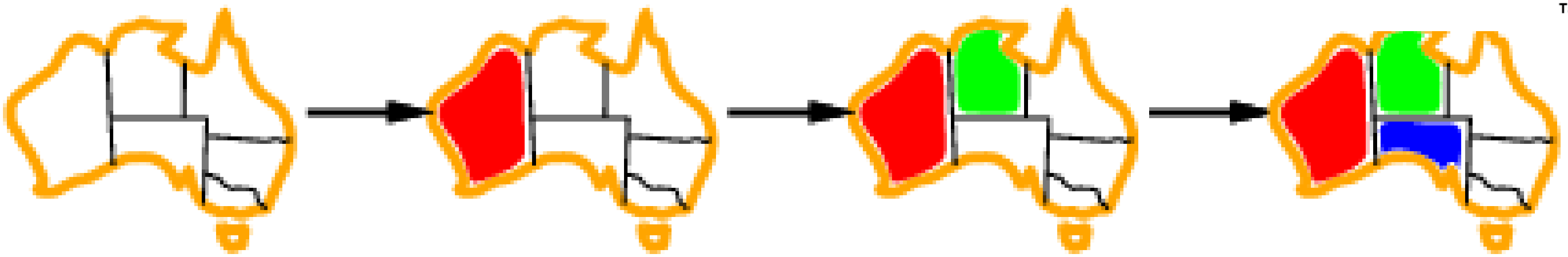
Improving backtracking efficiency

1. Which variable should be assigned next?

❖ Minimum remaining values (MRV):

- Choose the variable with the fewest legal values





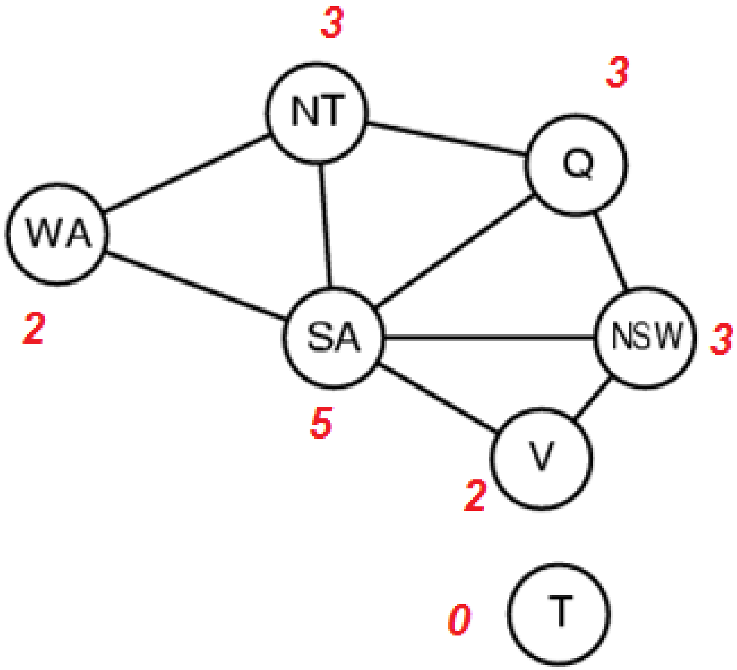
WA	Done..
NT	Done ..
SA	Done..
Q	1
NSW	2
V	2
T	3

Improving backtracking efficiency

1. Which variable should be assigned next?

❖ Most constraints variable (MCV):

- Choose the variable with the most constraints on

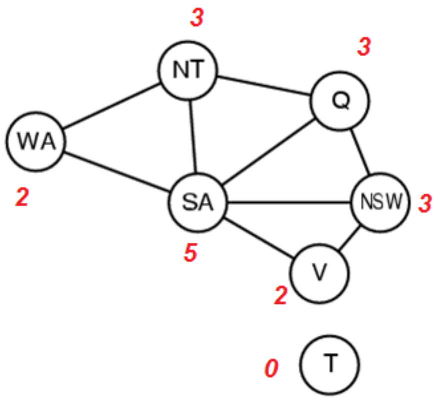


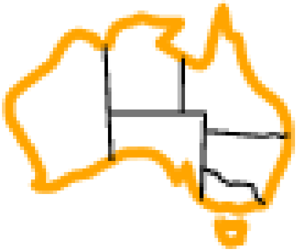
Improving backtracking efficiency

1. Which variable should be assigned next?

❖ Most constraints variable (MCV):

- Choose the variable with the most constraints on



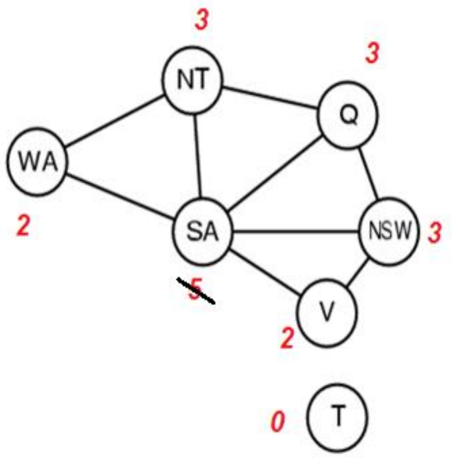


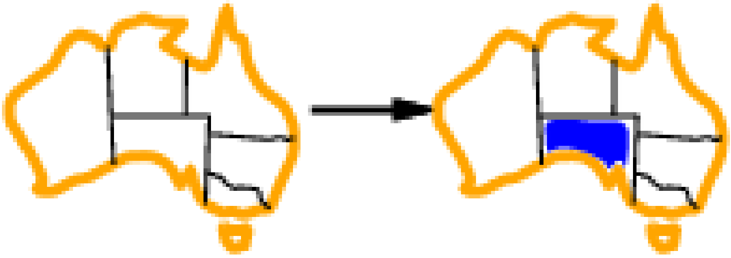
Improving backtracking efficiency

1. Which variable should be assigned next?

❖ Most constraints variable (MCV):

- Choose the variable with the most constraints on



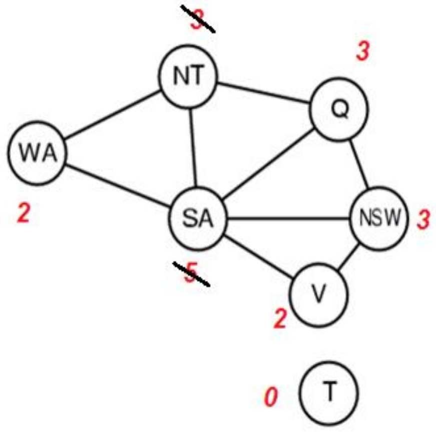


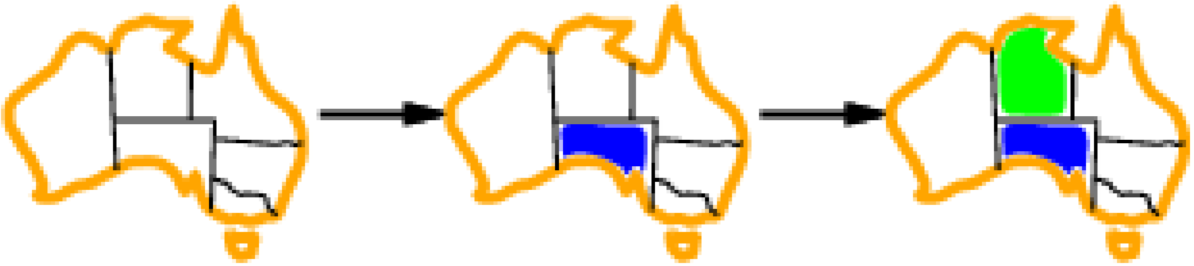
Improving backtracking efficiency

1. Which variable should be assigned next?

❖ Most constraints variable (MCV):

- Choose the variable with the most constraints on

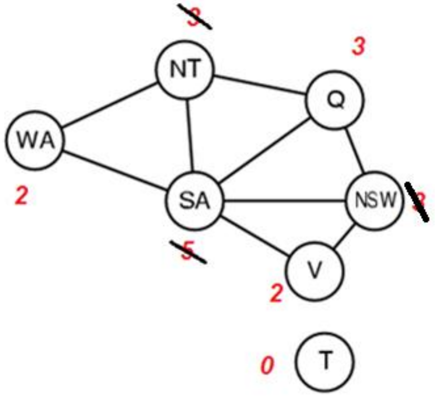


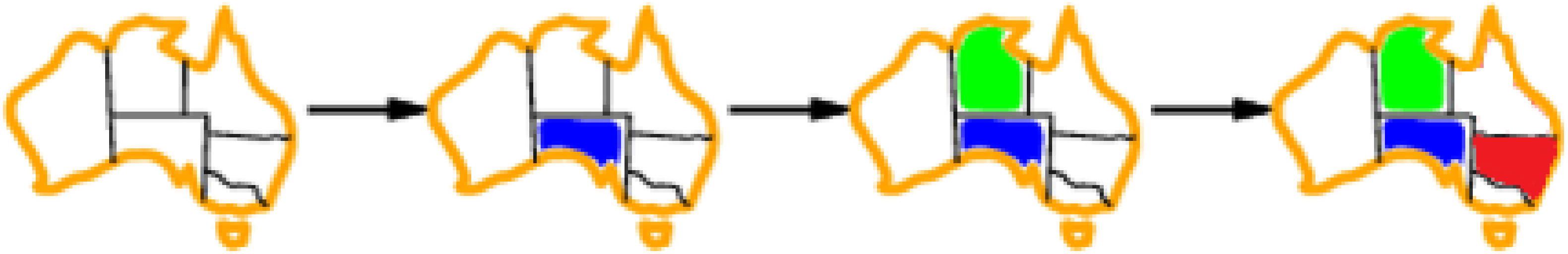


Improving backtracking efficiency

1. Which variable should be assigned next?

- Most constraints variable (MCV):
  - Choose the variable with the most constraints on

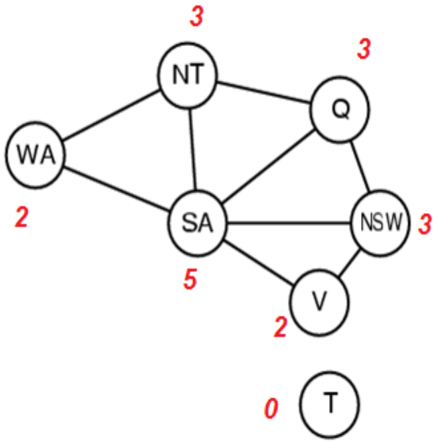




Improving backtracking efficiency

1. Which variable should be assigned next?

- Usually first applies MRV and breaks ties by MCV

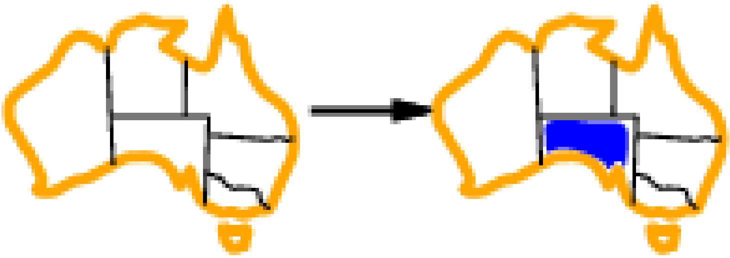


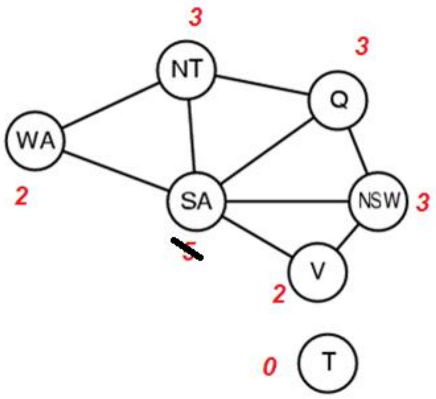
WA	3
NT	3
SA	3
Q	3
NSW	3
V	3
T	3

Improving backtracking efficiency

1. Which variable should be assigned next?

❖ Usually first applies MRV and breaks ties by MCV



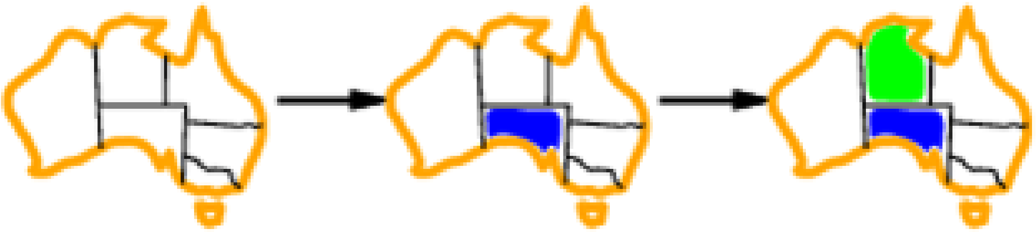


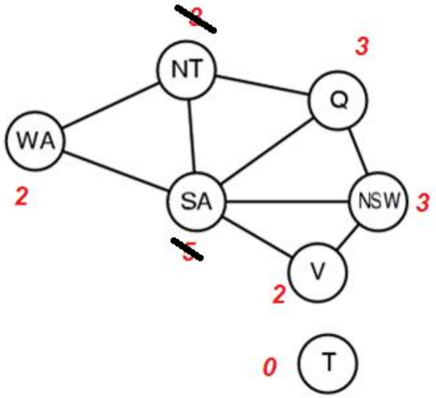
WA	2
NT	2
SA	Done..
Q	2
NSW	2
V	2
T	3

Improving backtracking efficiency

1. Which variable should be assigned next?

- Usually first applies MRV and breaks ties by MCV





WA	1
NT	Done..
SA	Done..
Q	1
NSW	2
V	2
T	3

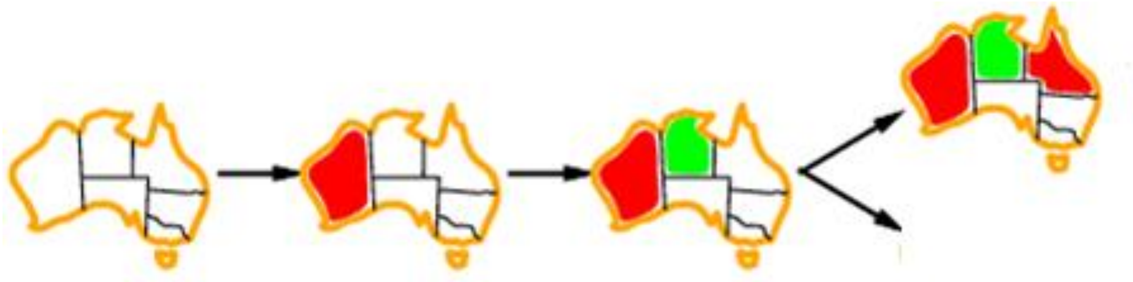
Next will be Q

Improving backtracking efficiency

2. Which order should its values be tried?

❖ Least constraining value (LCV):

- Given a variable, choose the least constraining value



Q=Red

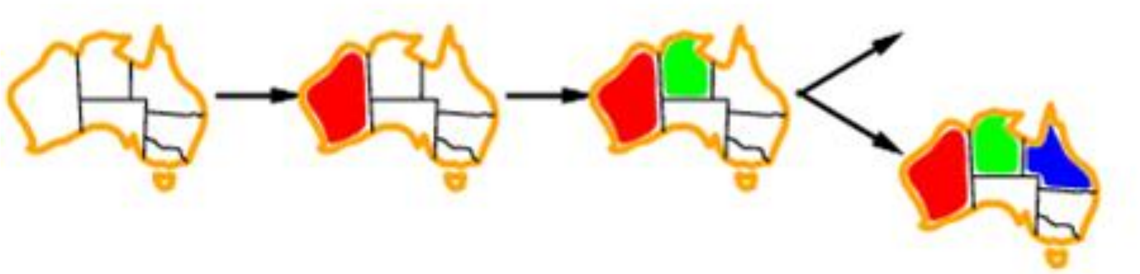
SA	1
NSW	2
V	*
T	*

Improving backtracking efficiency

2. Which order should its values be tried?

❖ Least constraining value (LCV):

- Given a variable, choose the least constraining value



Q=Red

SA	1
NSW	2
V	*
T	*

Q=blue

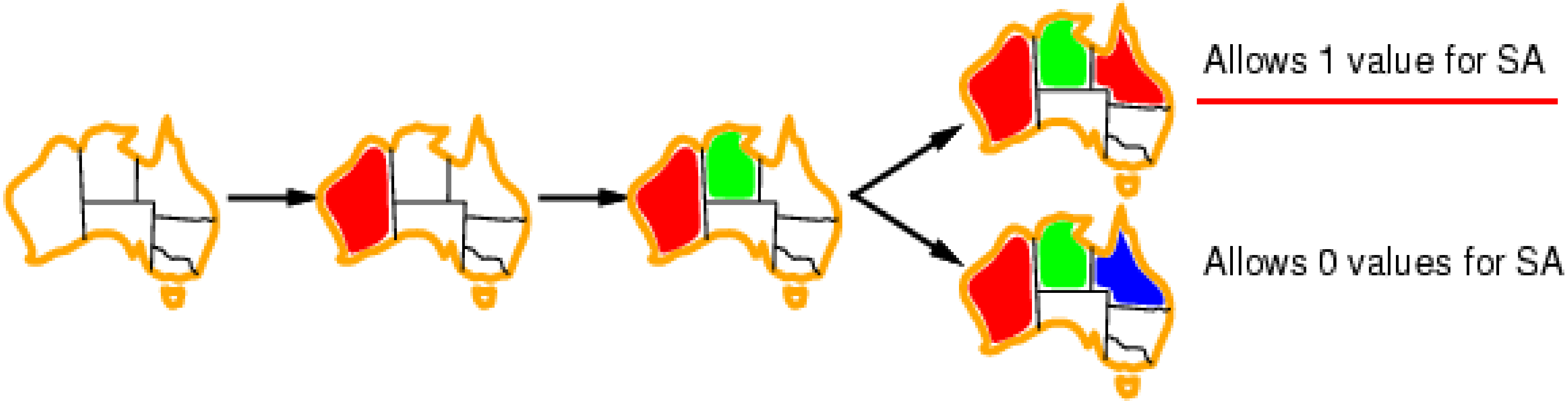
SA	0
NSW	2
V	*
T	*

Improving backtracking efficiency

2. Which order should its values be tried?

❖ Least constraining value (LCV):

- Given a variable, choose the least constraining value
- Order the values by descending number of choices for the remaining variables



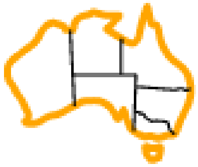
Can solve n-queens for n ≈ 1000

Improving backtracking efficiency

3. Can we detect inevitable failure early?

❖ Forward Checking:

- Keep track of remaining legal values for unassigned variables
- Terminate search when any variable has no legal values



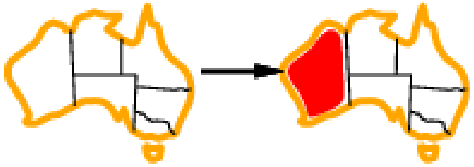
WA	NT	Q	NSW	V	SA	T
🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦

Improving backtracking efficiency

3. Can we detect inevitable failure early?

❖ Forward Checking :

- Keep track of remaining legal values for unassigned variables
- Terminate search when any variable has no legal values

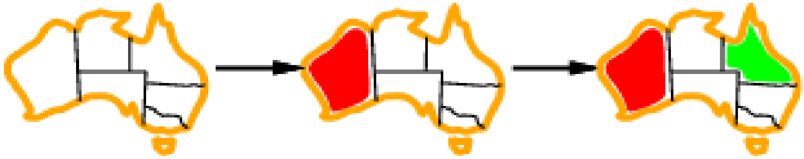


WA	NT	Q	NSW	V	SA	T
🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦
🟥	🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟩🟦	🟥🟩🟦

Improving backtracking efficiency

3. Can we detect inevitable failure early?

- Forward Checking :
  - Keep track of remaining legal values for unassigned variables
  - Terminate search when any variable has no legal values



WA	NT	Q	NSW	V	SA	T
🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦
🟥	🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟩🟦	🟥🟩🟦
🟥	🟦	🟩	🟥🟦	🟥🟩🟦	🟦	🟥🟩🟦

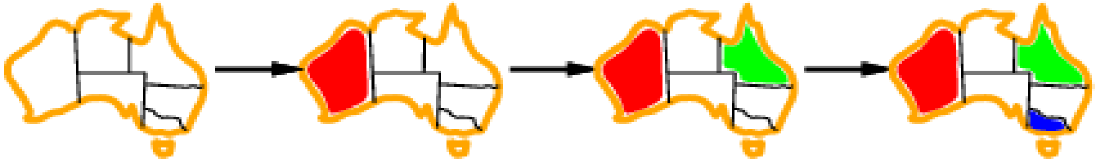
Improving backtracking efficiency

Western Australia
Northern Territory
Queensland
South Australia
New South Wales
Victoria
Tasmania

3. Can we detect inevitable failure early?

❖ Forward Checking :

- Keep track of remaining legal values for unassigned variables
- Terminate search when any variable has no legal values



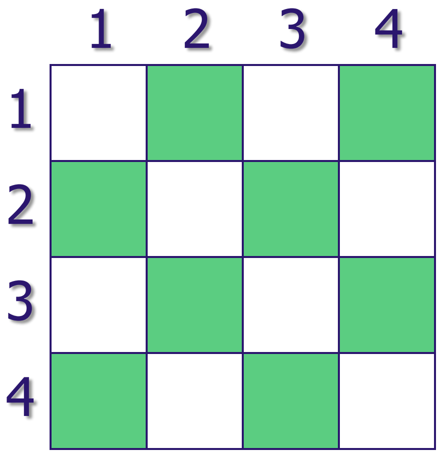
WA	NT	Q	NSW	V	SA	T
🟥 🟩 🟦	🟥 🟩 🟦	🟥 🟩 🟦	🟥 🟩 🟦	🟥 🟩 🟦	🟥 🟩 🟦	🟥 🟩 🟦
🟥	🟩 🟦	🟥 🟩 🟦	🟥 🟩 🟦	🟥 🟩 🟦	🟩 🟦	🟥 🟩 🟦
🟥	🟦	🟩	🟥 🟦	🟥 🟩 🟦	🟦	🟥 🟩 🟦
🟥	🟦	🟩	🟥	🟦		🟥 🟩 🟦

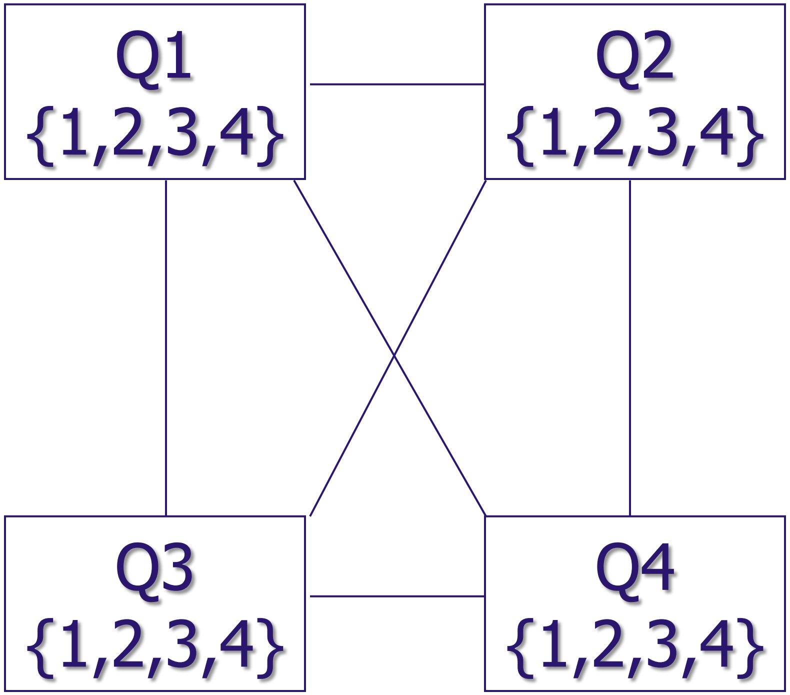
Terminate!

Improving backtracking efficiency

3. Can we detect inevitable failure early?

- Forward Checking :
  - Keep track of remaining legal values for unassigned variables
  - Terminate search when any variable has no legal values





Q1 {1,2,3,4}

Q2 {1,2,3,4}

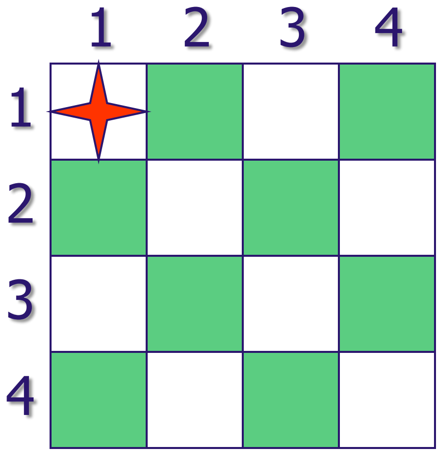
Q3 {1,2,3,4}

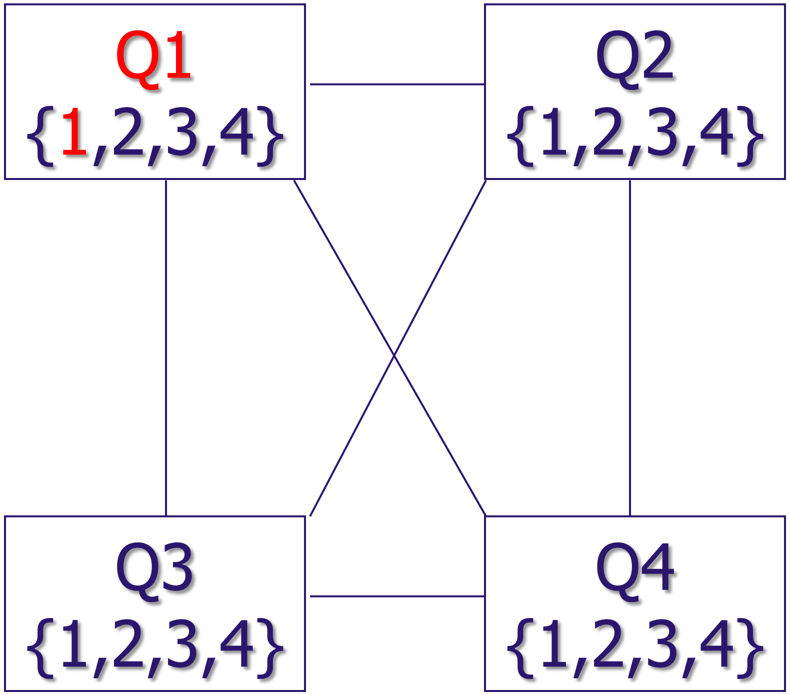
Q4 {1,2,3,4}

Improving backtracking efficiency

3. Can we detect inevitable failure early?

- Forward Checking :
  - Keep track of remaining legal values for unassigned variables
  - Terminate search when any variable has no legal values





Q1
{1,2,3,4}

Q2
{1,2,3,4}

Q3
{1,2,3,4}

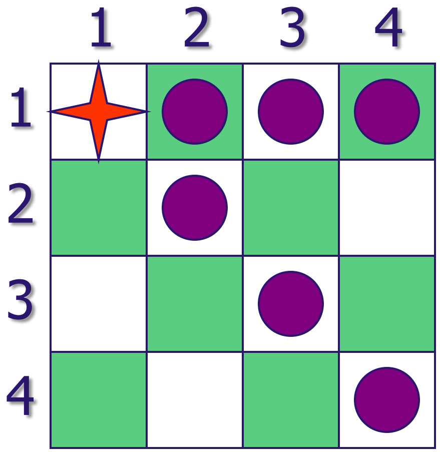
Q4
{1,2,3,4}

Improving backtracking efficiency

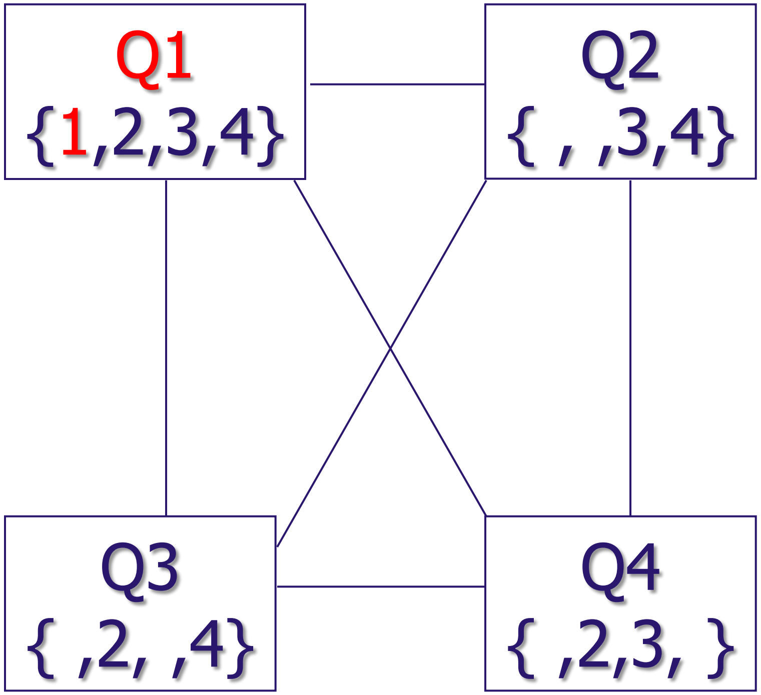
3. Can we detect inevitable failure early?

❖ Forward Checking :

- Keep track of remaining legal values for unassigned variables
- Terminate search when any variable has no legal values



	1	2	3	4
1	✦	●	●	●
2		●		
3			●	
4				●



Q1 {1,2,3,4}

Q2 { , ,3,4}

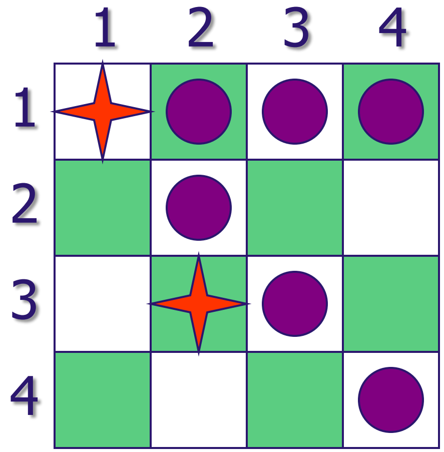
Q3 { ,2, ,4}

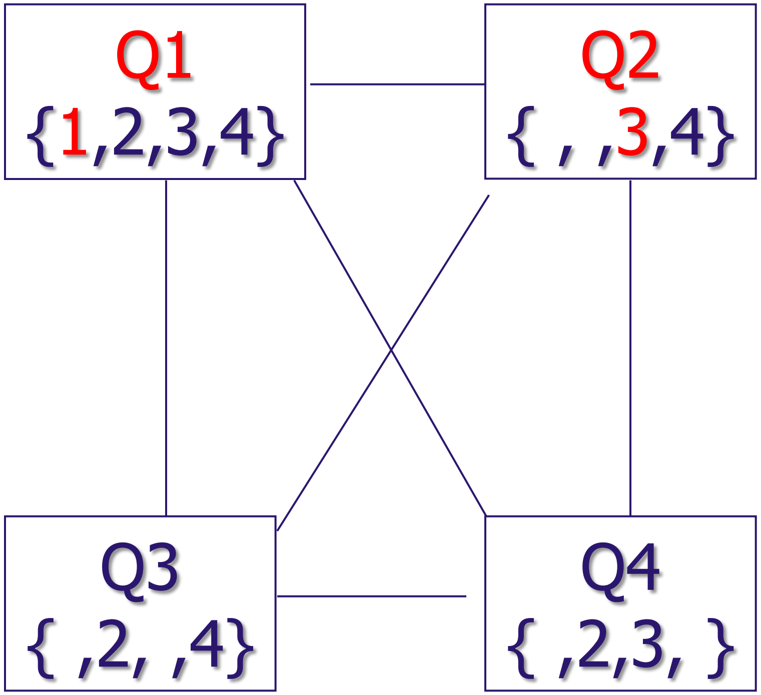
Q4 { ,2,3, }

Improving backtracking efficiency

3. Can we detect inevitable failure early?

- Forward Checking :
  - Keep track of remaining legal values for unassigned variables
  - Terminate search when any variable has no legal values





Q1
{1,2,3,4}

Q2
{ , ,3,4}

Q3
{ ,2, ,4}

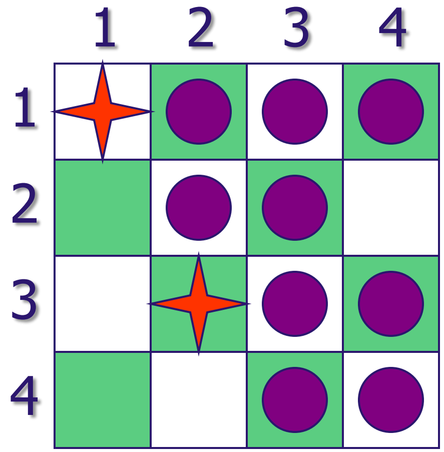
Q4
{ ,2,3, }

Improving backtracking efficiency

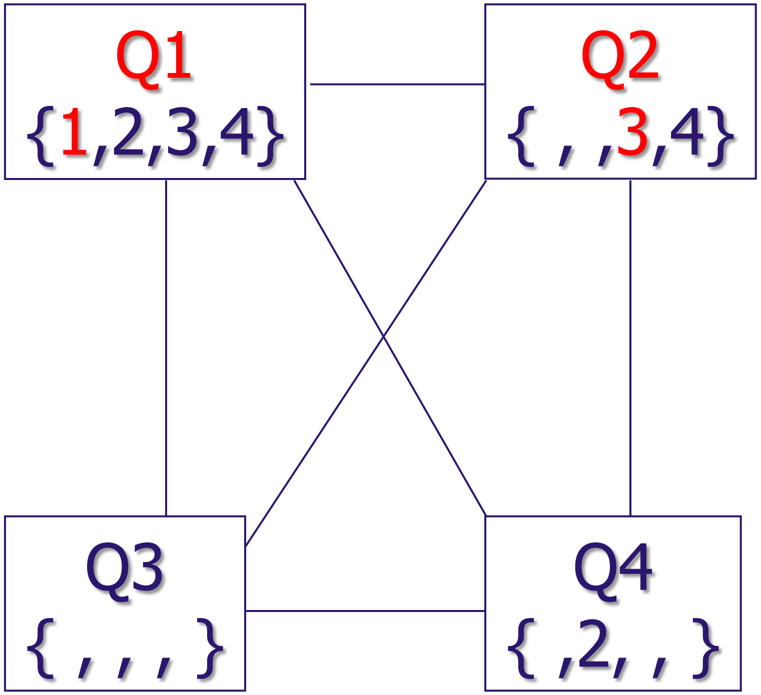
3. Can we detect inevitable failure early?

❖ Forward Checking :

- Keep track of remaining legal values for unassigned variables
- Terminate search when any variable has no legal values



Terminate!



Applications

- Assignment problems
- Timetabling problems
- Transportation scheduling
- Cryptography

Thank You !