

Constraint Satisfaction Problem

Departemen Teknik Informatika



Capaian Pembelajaran

Mahasiswa mampu menjelaskan, mengidentifikasi, merancang, dan menerapkan *intelligent agent* untuk *problem* yang sesuai dengan memanfaatkan algoritma pencarian yang meliputi *uninformed search, informed search, heuristic search, adversarial search*, serta algoritma *search* untuk *Constraint Satisfaction Problem*

- Teknik pencarian sebelumnya *uninformed, informed, local search*
 - Fokus memecahkan problem dengan mencari *state* yang bisa menjadi solusi
- Standard search problems:
 - *State is a “black box”: arbitrary data structure*
 - *Goal test can be any function over states*
 - *Successor function can also be anything*
- Struktur internal *state* setiap problem tidak sama
- *Constraint Satisfaction Problems* (CSP) memiliki representasi states dan goal yang standard, terstruktur, dan simpel



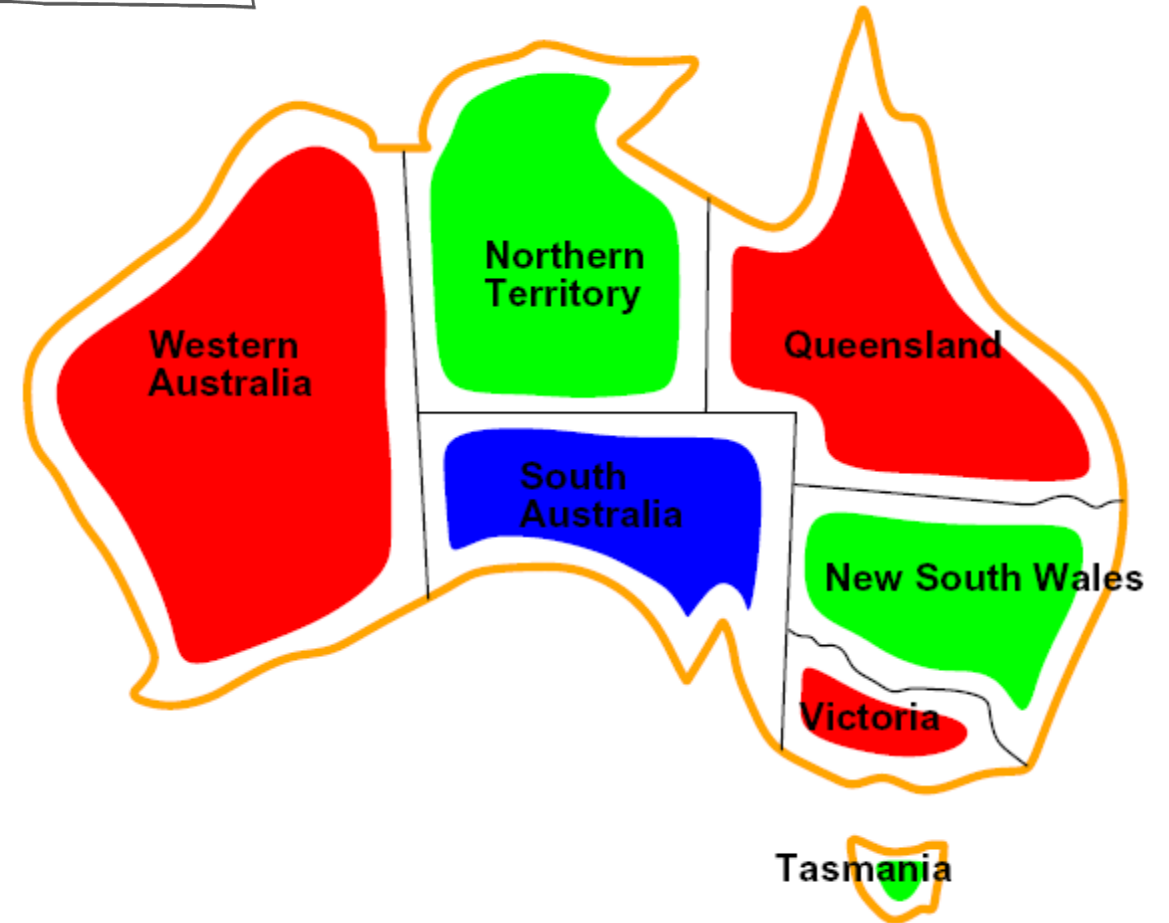
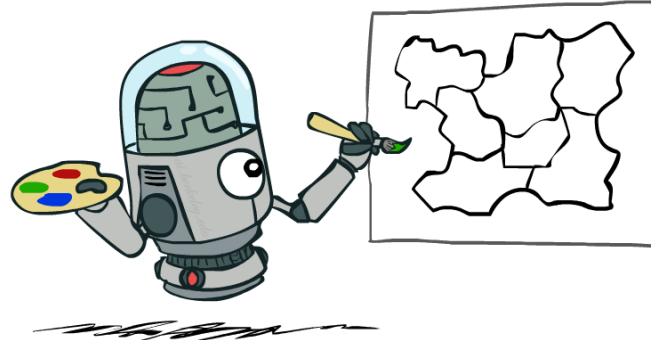
Constraint Satisfaction Problems (CSP)

- Constraint satisfaction problems (CSP):
 - State didefinisikan sebagai **variabel** X_i dengan value dari **domain** D
 - Goal merupakan himpunan Batasan (**set of constraints**) hanya mengijinkan kombinasi nilai untuk subsets of variables
- Sebuah himpunan **variables** X_1, X_2, \dots, X_n , dan sebuah himpunan **constraints** C_1, C_2, \dots, C_m
- Setiap variable X_i mempunyai nonempty **domain** D_i dari semua variasi **values**
- Setiap constraint C_i
 - Sebuah state pada problem yang didefinisikan melalui **penugasan** (**assignment**) dari nilai ke beberapa atau semua $\{X_i = v_i; X_j = v_j; \dots\}$
- Sebuah penugasan yang tidak melanggar semua Batasan (**constraint**) disebut **consistent** atau *legal assignment*
- **Solusi** sebuah CSP adalah sebuah *complete assignment* yang memenuhi (*satisfies*) semua *constraint*



IF

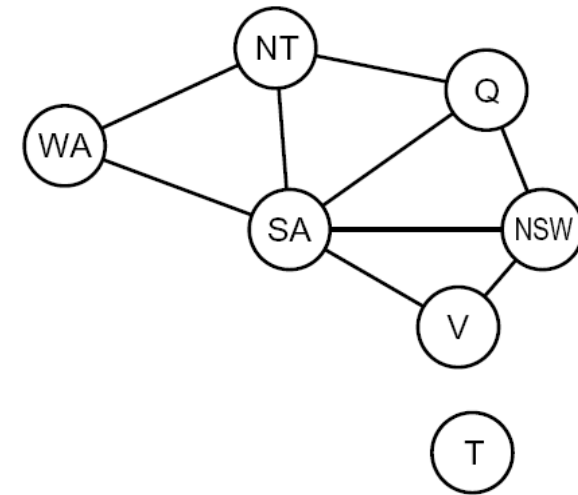
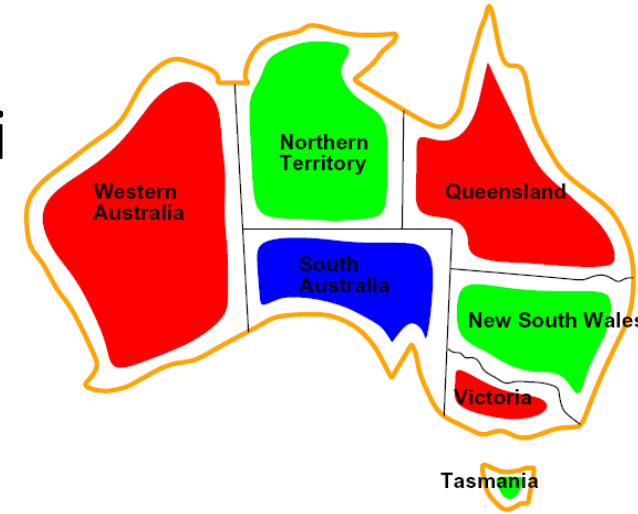
Contoh CSP: *Map Coloring*





Contoh CSP: *Map Coloring*

- Pewarnaan tiap region bisa merah, hijau, atau biru dimana region yang bertetangga **tidak boleh** mempunyai warna yang sama
- *Variables*: WA, NT, Q, NSW, V, SA, dan T
- *Domains*: $D = \{\text{red}; \text{green}; \text{blue}\}$
- *Constraints*: region yang bertetangga harus berbeda warna
 - $WA \neq NT, WA \neq SA, NT \neq Q, \dots$
- Solusi adalah semua penugasan yang memenuhi semua batasan, contoh:
 - $\{WA=\text{red}; NT=\text{green}; Q=\text{red}; NSW=\text{green}; V=\text{red}; SA=\text{blue}; T=\text{red}\}$
- Visualisasi CSP bisa menggunakan ***constraint graph***

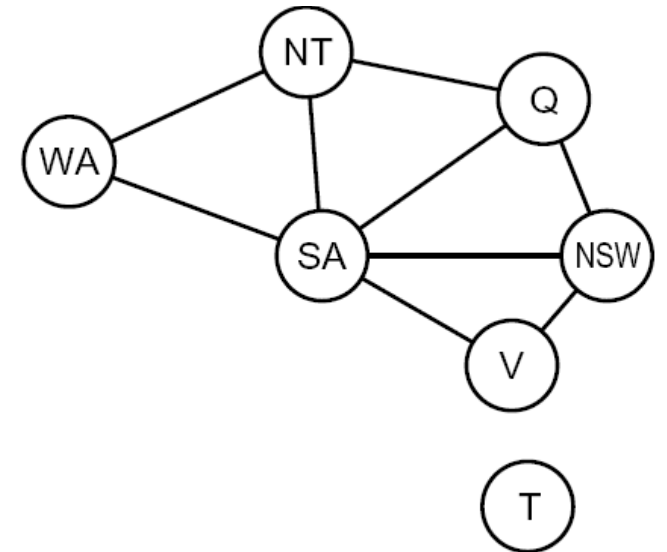
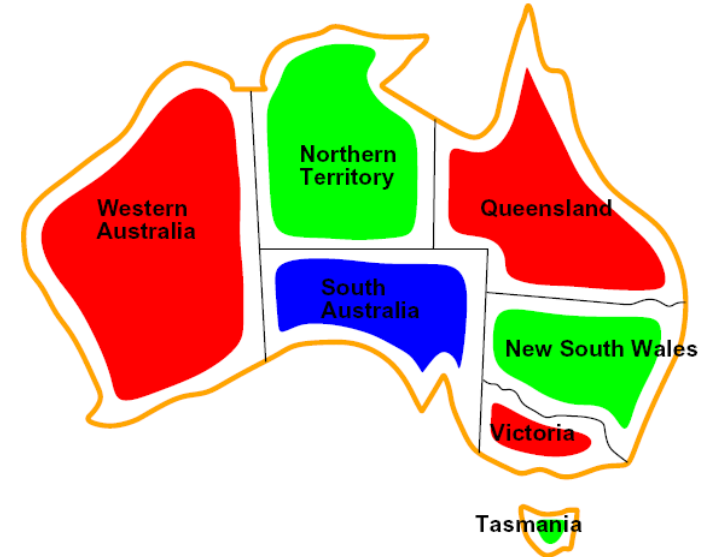




IF

Representasi CSP sebagai *Constraint Graph*

- Binary CSP: setiap constraint berelasi paling banyak dua variabel
- Binary constraint graph: setiap node adalah variabel, garis penghubung (*arcs*) menunjukkan *constraints*
- Tujuan utama algoritma CSP menggunakan struktur graf untuk mempersingkat waktu pencarian, contoh Tasmania adalah sebuah *independent subproblem*.



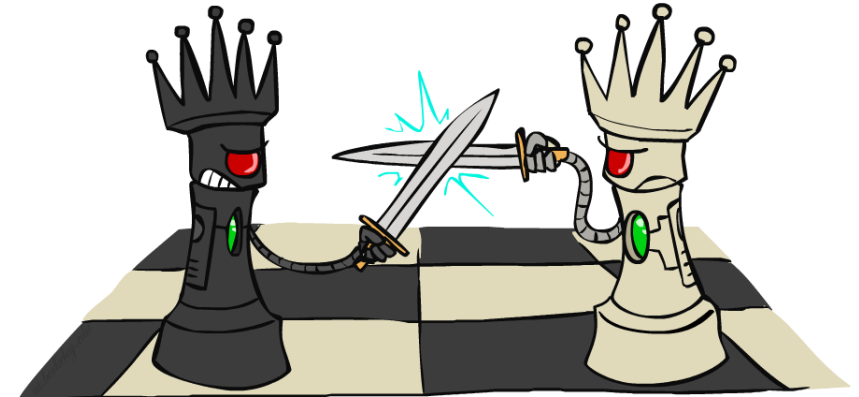
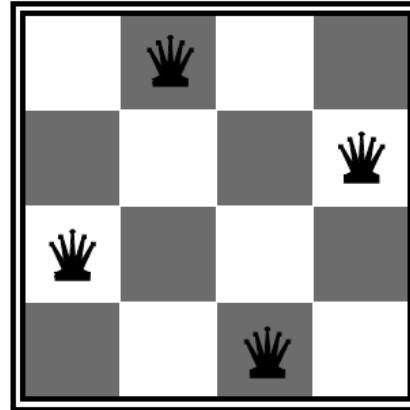
Contoh CSP: N-Queens

Formulasi 1:

Variables: X_{ij}

Domains: $\{0, 1\}$

Constraints



$$\forall i, j, k \quad (X_{ij}, X_{ik}) \in \{(0, 0), (0, 1), (1, 0)\}$$

$$\forall i, j, k \quad (X_{ij}, X_{kj}) \in \{(0, 0), (0, 1), (1, 0)\}$$

$$\forall i, j, k \quad (X_{ij}, X_{i+k, j+k}) \in \{(0, 0), (0, 1), (1, 0)\}$$

$$\forall i, j, k \quad (X_{ij}, X_{i+k, j-k}) \in \{(0, 0), (0, 1), (1, 0)\}$$

$$\sum_{i,j} X_{ij} = N$$



Contoh CSP: N-Queens

Formulasi 2:

Variables: Q_k

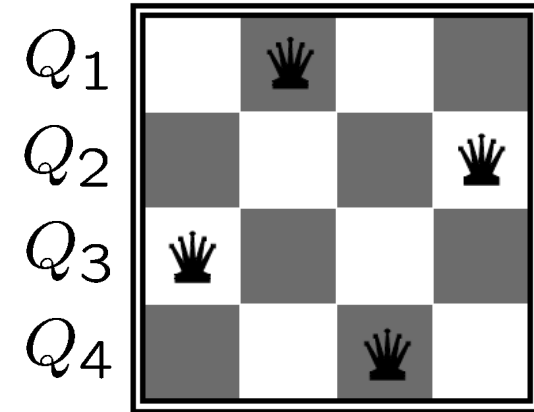
Domains: $\{1, 2, 3, \dots, N\}$

Constraints

Implicit: $\forall i, j \text{ non-threatening}(Q_i, Q_j)$

Explicit: $(Q_1, Q_2) \in \{(1, 3), (1, 4), \dots\}$

...

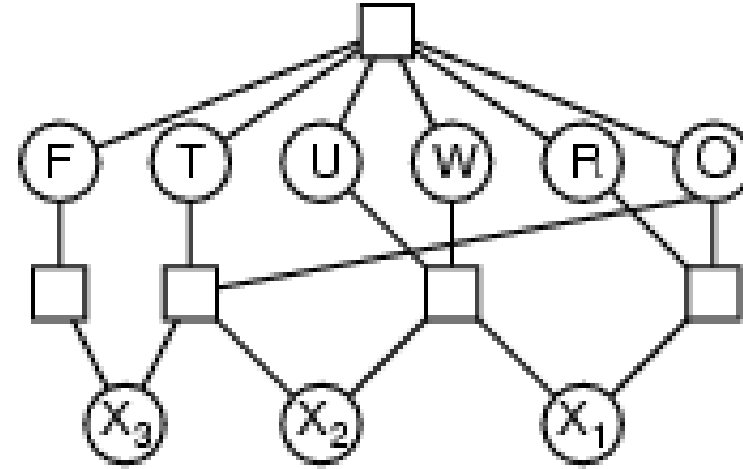




IF

Contoh CSP: Cryptarithmic

$$\begin{array}{r} \text{ T W O} \\ + \text{ T W O} \\ \hline \text{ F O U R} \end{array}$$



Variables: $F T U W R O X_1 X_2 X_3$

Domains: $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

Constraints: $AllDiff(F, T, U, W, R, O)$

$$O + O = R + 10 \cdot X_1$$

$$X_1 + W + W = U + 10 \cdot X_2$$

$$X_2 + T + T = O + 10 \cdot X_3$$

$$X_3 = F, T \neq 0, F \neq 0$$

- Discrete Variables
 - Finite domains
 - Size d means $O(d^n)$ complete assignments
 - E.g., Boolean CSPs, including Boolean satisfiability (NP-complete)
 - Infinite domains (integers, strings, etc.)
 - E.g., job scheduling, variables are start/end times for each job
- Continuous variables
 - E.g., start/end times for Hubble Telescope observations

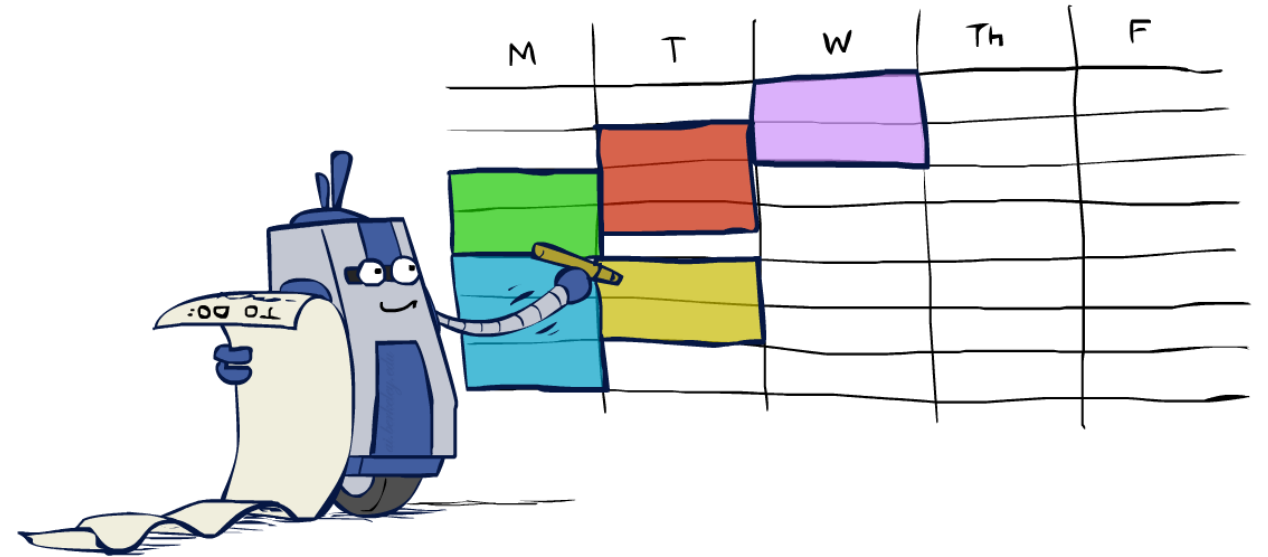


Jenis Constraints

- *Unary constraint* untuk satu variabel
=> SA \neq green
- *Binary constraint* untuk pasangan variabel
=> SA \neq WA
- *Higher-order constraint* untuk >2 variabel
- *Preference constraint (soft constraint)*
Representasi *red is better than green* adalah memberikan nilai bobot yang berbeda
=> *constrained optimization problem*

Real Worlds CSP

- Assignment problems
- Timetabling problems
- Transportation scheduling
- Factory scheduling
- Notice that many real-world problems involve real-valued variables



- State ditentukan oleh nilai yang diberikan (*partial assignments*)
 - Initial state: {}
 - Successor function: memberikan nilai pada sebuah variabel yang belum ada nilai (*unassigned variable*)
 - Goal test: semua tugas selesai (*complete*) dan tidak melanggar constraints (*satisfies all constraints*)



CSP vs Search Space

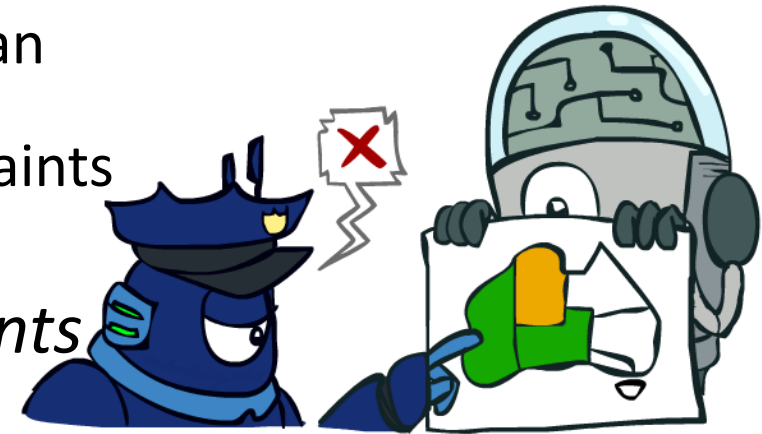
- Jika problem diselesaikan dengan pencarian
 - Ada $3^5 = 243$ kombinasi nilai warna setiap variabel
 - Jumlah anggota pada state-space = 243
- Jika problem diselesaikan dengan CSP
 - Hanya $2^5 = 32$ kombinasi karena satu warna sudah terpilih
 - Jumlah anggota pada state-space = 32



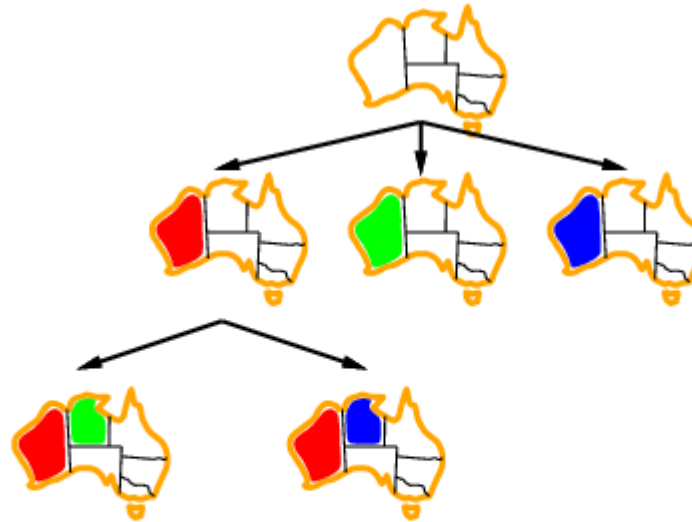
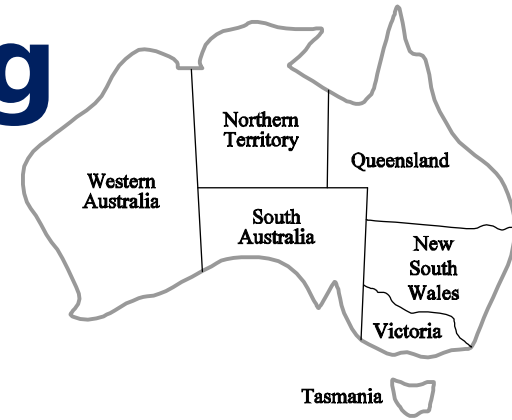
IF

Backtracking Search for CSPs

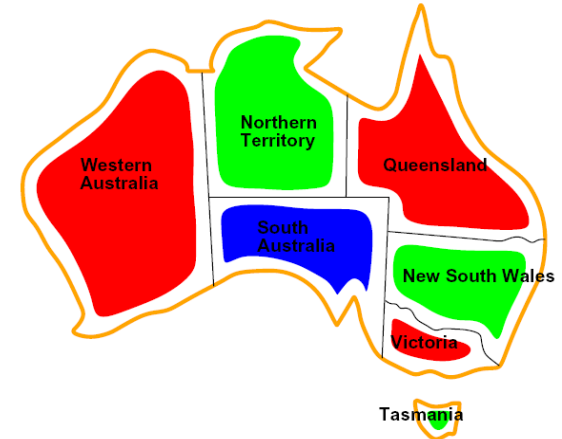
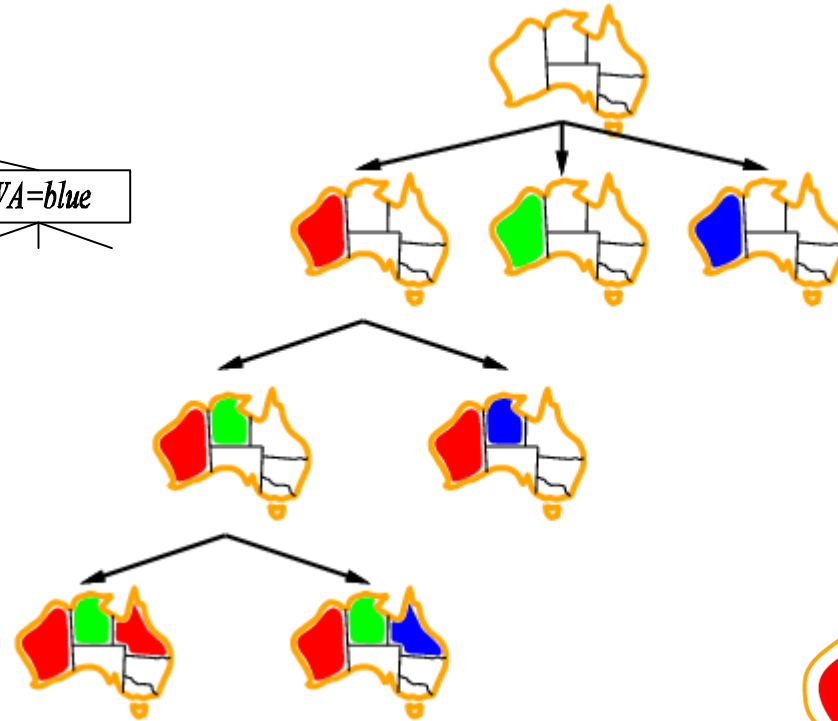
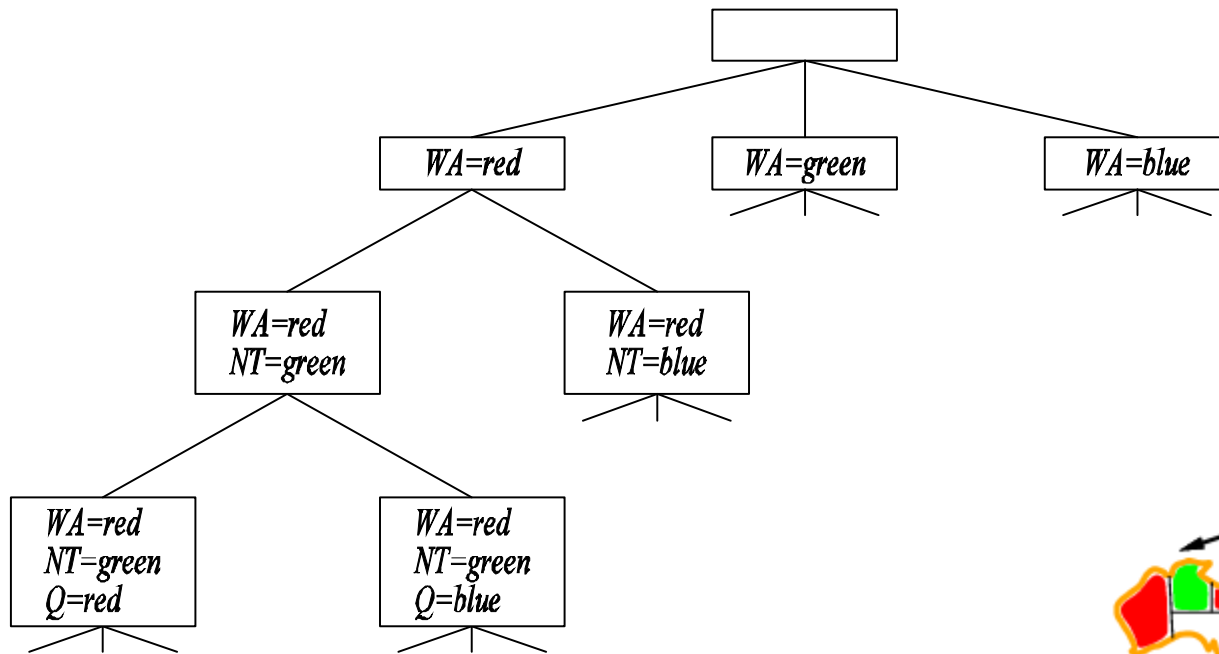
- Backtracking search adalah dasar algoritma *uninformed* untuk menyelesaikan CSP
- Idea 1: *One variable at a time*
 - Penugasan variabel bersifat *commutative*
 - Contoh: [WA = red then NT = green] sama dengan [NT = green then WA = red]
 - Hanya perlu mempertimbangkan penugasan pada suatu variabel setiap step
- Idea 2: *Check constraints as you go*
 - Mempertimbangkan hanya nilai yang tidak konflik dengan penugasan sebelumnya
 - Harus melakukan beberapa Langkah untuk check constraints
 - “Incremental goal test”
- Depth-first search dengan *single-variable assignments* disebut *backtracking search*



Contoh: Backtracking



Contoh: Backtracking





Algoritma *Backtracking*

```
function BACKTRACKING-SEARCH(csp) returns a solution, or failure
    return RECURSIVE-BACKTRACKING(f g, 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 fvar = valueg to assignment
            result ← RECURSIVE-BACKTRACKING(assignment, csp)
            if result <> failure then return result
        remove {var = value} from assignment
    return failure
```

Terdapat 2 Fungsi yaitu:

1. SELECT-UNASSIGNED-VARIABLE (urutan pemilihan variabel)
2. ORDER-DOMAIN-VALUES (urutan pemilihan nilai dari variabel)

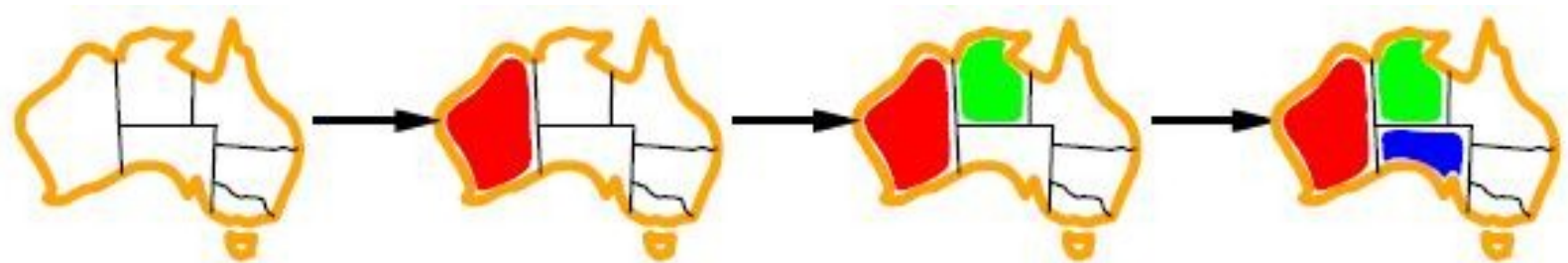


Variable and value ordering

- SELECT-UNASSIGNED-VARIABLE (urutan pemilihan variabel)
 - *Most Constrain^{ed} Variable (Minimum Remaining Values, **MRV**)*
 - Pilih variabel dengan variasi kemungkinan nilai paling sedikit
 - Jika >1 variabel, maka digunakan *Most Constraining Variable*
 - *Most Constraining^{ing} Variable (**MCV**)*
 - Pilih variabel yang memiliki jumlah *constraint* lebih banyak

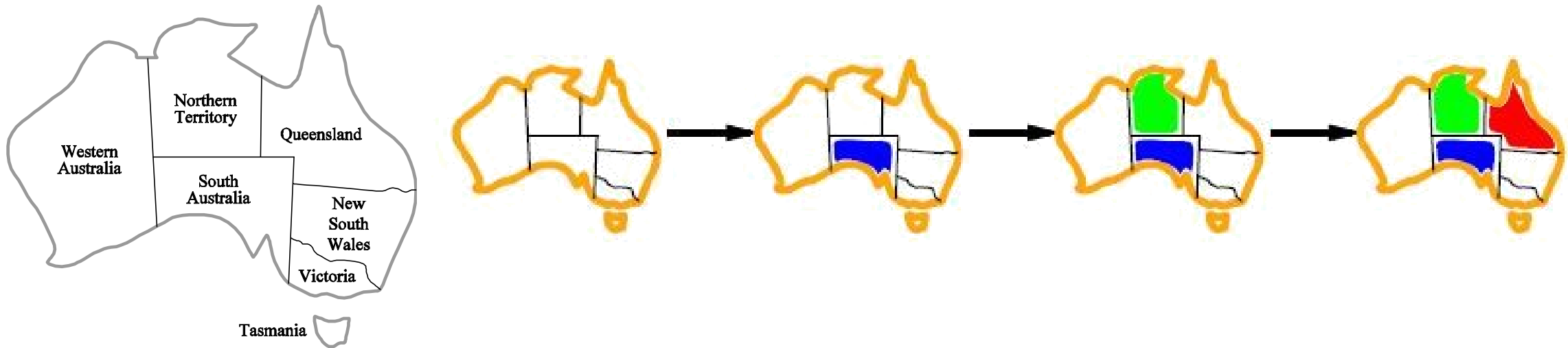
IF Variable and value ordering

- **Minimum remaining values (MRV)** or “most constrained variable” :
 - Memilih **variabel** dengan variasi kemungkinan nilai paling sedikit
 - **WA=red** dan **NT=green**, ada hanya satu kemungkinan nilai untuk **SA**, maka MRV akan memilih variable **SA** yaitu **SA=blue** berikutnya baru memilih variabel **Q**



IF Variable and value ordering

- **Most Constraining Variable (MCV)**
 - Memilih **variabel** yang memiliki jumlah *constraint* lebih banyak
 - MCV memilih variabel SA karena mempunyai jumlah *constraint* paling banyak yaitu 5 *constraints*





IF

Variable and value ordering

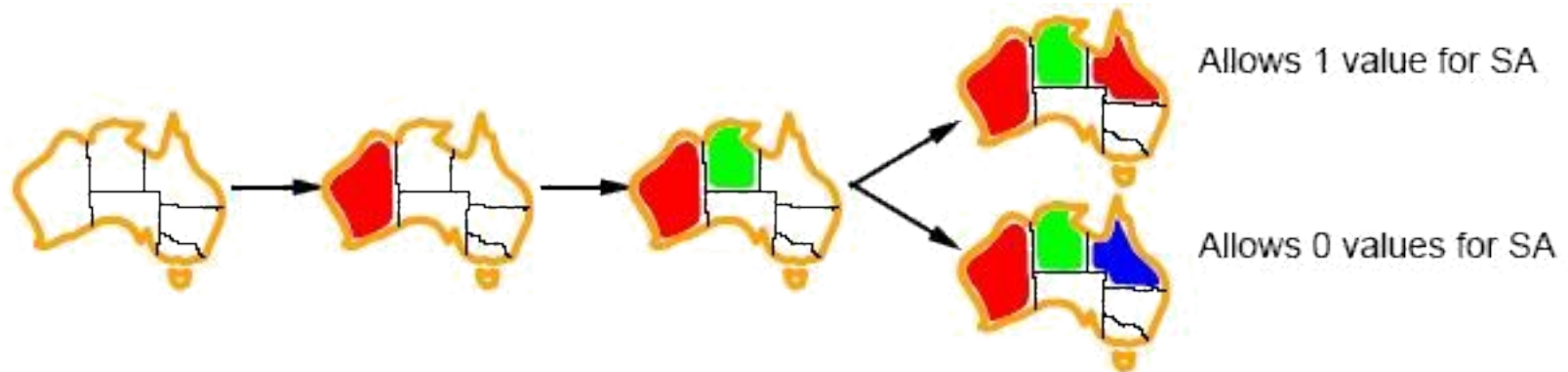
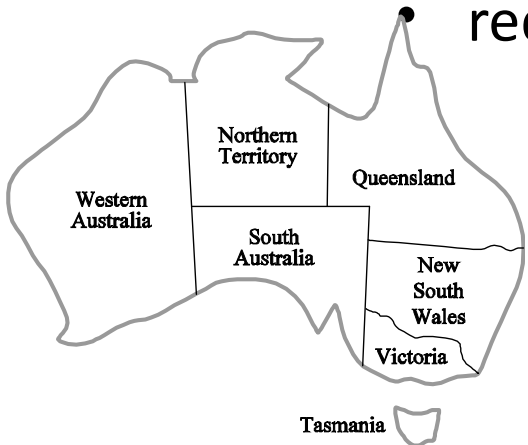
- ORDER-DOMAIN-VALUES (urutan pemilihan nilai dari variabel)
- **Least Constraining Value (LCV)**
 - Pilih **nilai** variabel yang memiliki *constraint* lebih sedikit untuk variabel lain
 - Atau pilih **nilai** variabel yang membuat variabel lain memiliki kemungkinan pilihan nilai lebih banyak



IF

Variable and value ordering

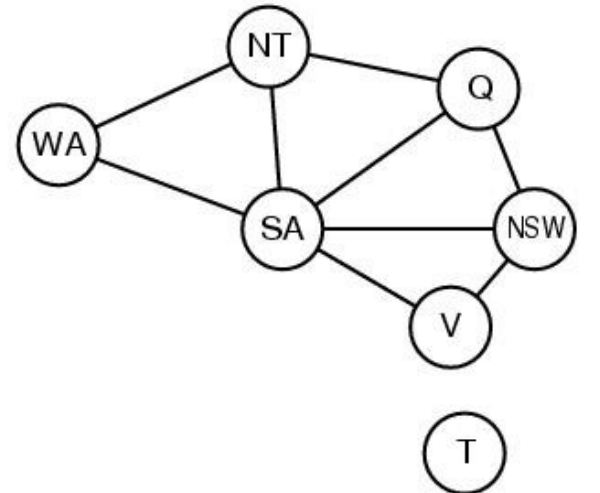
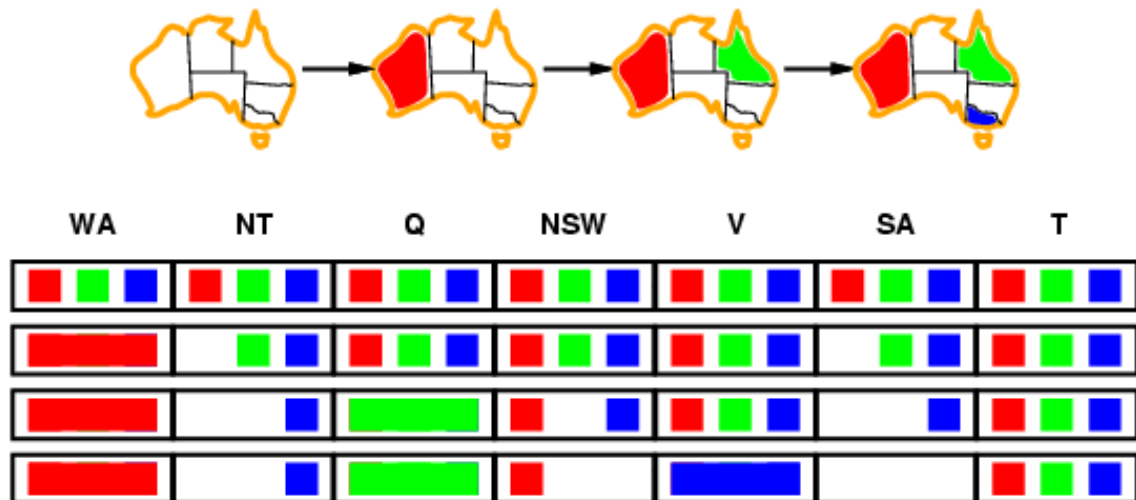
- **Least Constraining Value (LCV)**
 - Jika WA=red dan NT=green maka pilihan untuk Q={red, blue}
 - Jika Q = red maka variabel SA ada 1 pilihan nilai
 - (2 constraint, $SA \neq \text{red}$ & $SA \neq \text{green}$)
 - Jika Q = blue maka variabel SA tidak ada pilihan nilai
 - (3 constraint, $SA \neq \text{red}$ & $SA \neq \text{green}$ & $SA \neq \text{blue}$)
 - LCV akan memilih Q=red karena:
 - red membuat pilihan *constraint* lebih sedikit untuk variabel SA
 - red membuat pilihan nilai variabel SA lebih banyak ($1 > 0$)





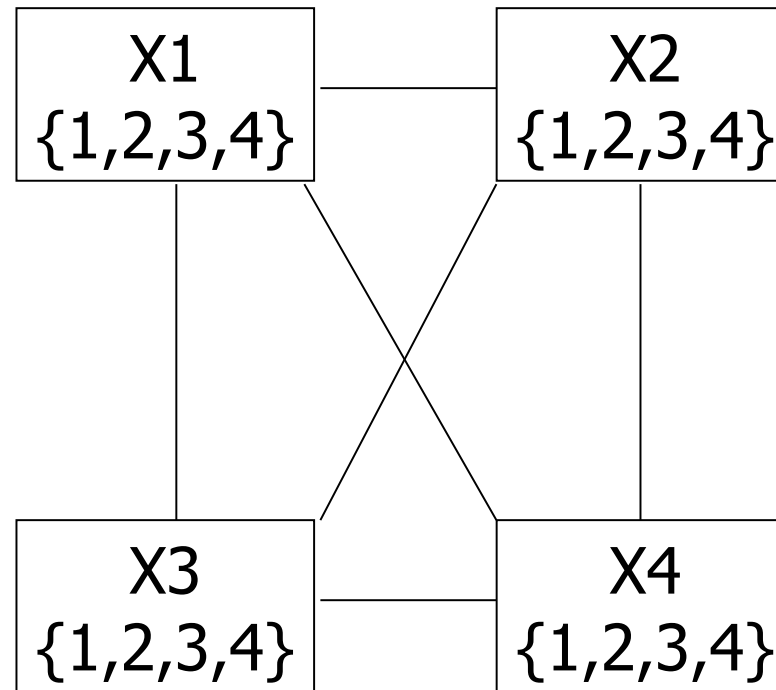
Constraint Propagation: Forward Checking

- Mencatat (*keep track*) kemungkinan nilai yang konsisten dengan *constraint* untuk semua variabel
- Pencarian dihentikan jika salah satu variabel sudah tidak memiliki kemungkinan nilai
 - backtrack dengan pilihan nilai lain



Contoh Forward Checking: 4-Queens Problem

	1	2	3	4
1				
2				
3				
4				




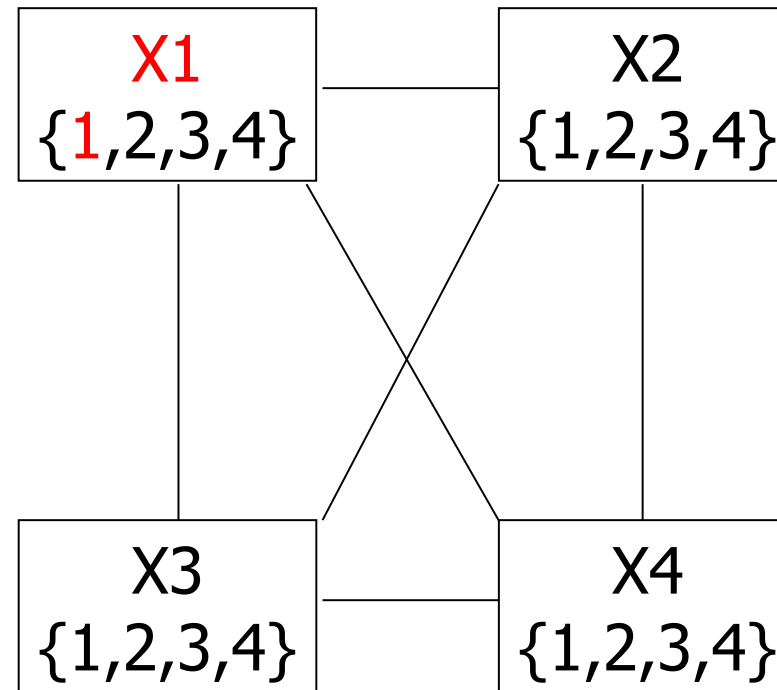
Representasi *complete graph* untuk 4 variabel problem 4-Queens saat inisialisasi. Variabel menunjukkan indeks kolom dan domain value adalah baris



IF

Contoh Forward Checking: 4-Queens

	1	2	3	4
1				
2				
3				
4				



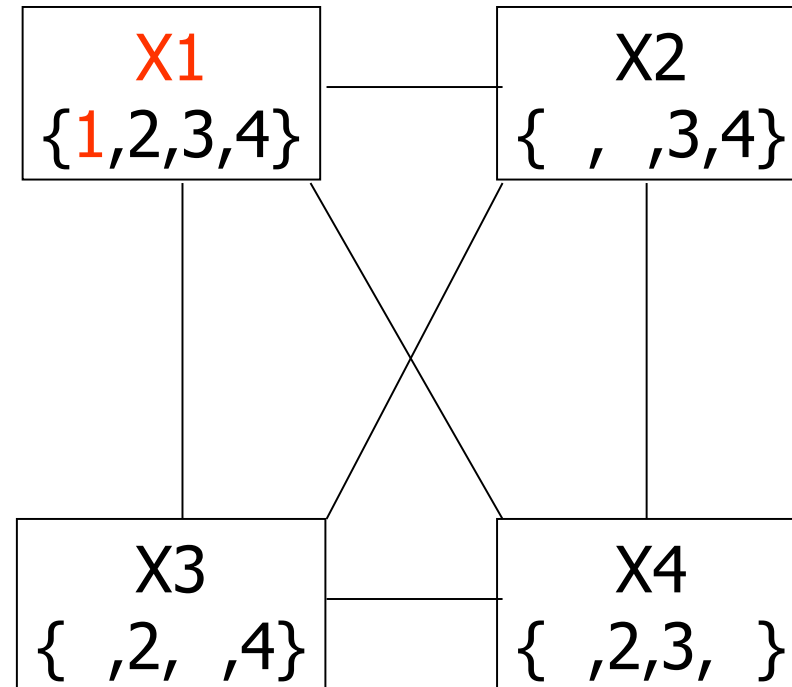
Pilih variabel $X1=1$



IF

Contoh Forward Checking: 4-Queens

	1	2	3	4
1	★	●	●	●
2		●		
3			●	
4				●



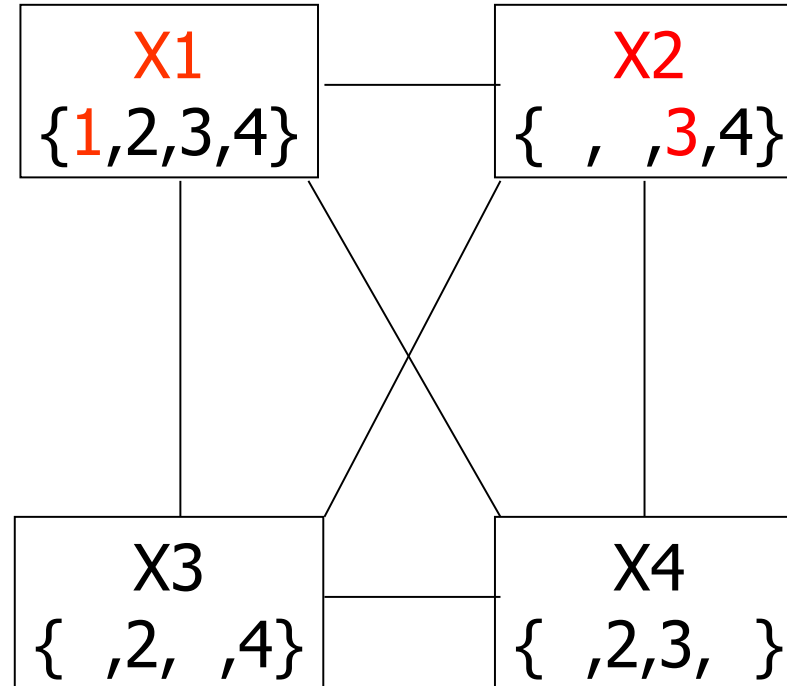
Pilih variabel $X1=1$, sisa nilai variabel $X2$, $X3$, $X4$ dengan forward checking



IF

Contoh Forward Checking: 4-Queens

	1	2	3	4
1	★	●	●	●
2		●		
3		★	●	
4				●



Pilih variabel $X1=1$ dan $X2=3$

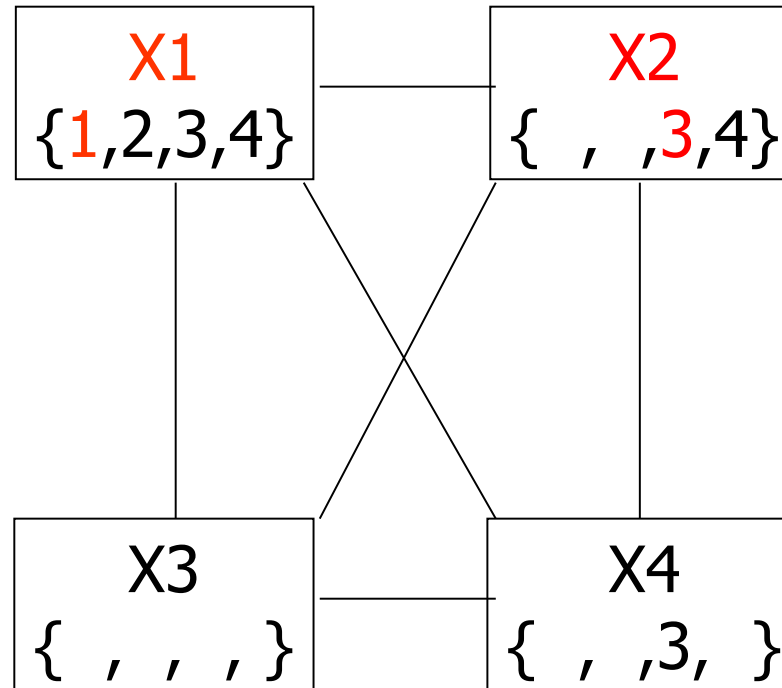


IF

Contoh Forward Checking: 4-Queens

	1	2	3	4
1	★	●	●	●
2		●	●	
3		★	●	●
4			●	●

Pilih variabel $X1=1$ dan $X2=3$,
membuat variabel $X3$ tidak memiliki
nilai sehingga perlu backtrack untuk
 $X2=4$

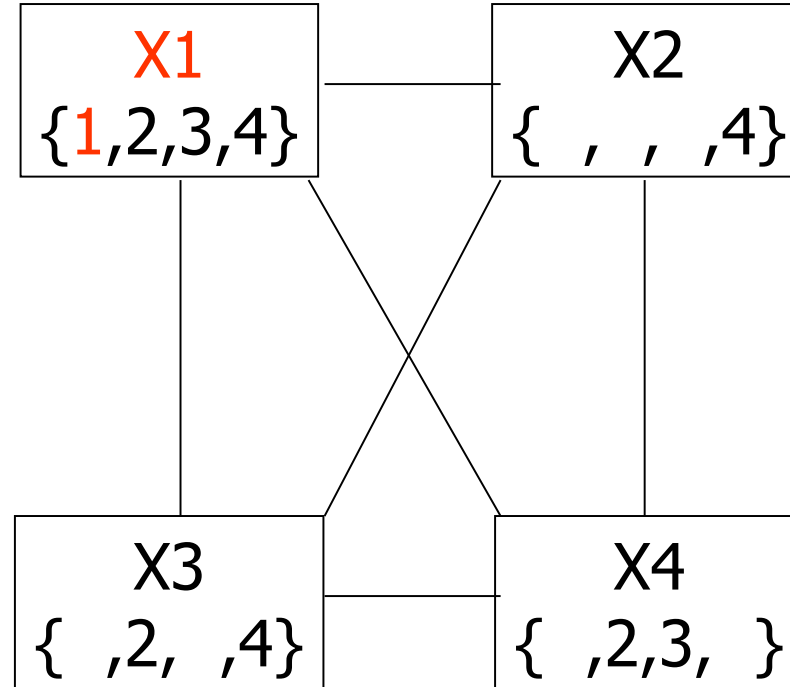
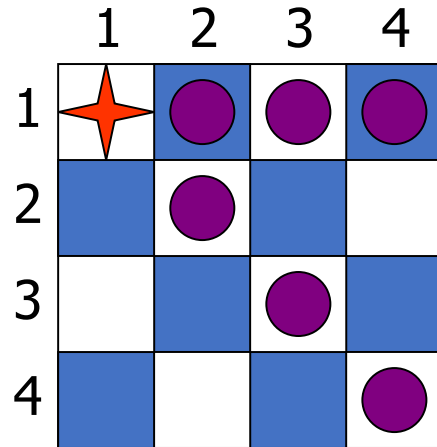


Pilih variabel $X1=1$ dan $X2=3$, sisa nilai variabel
 $X3, X4$ dengan forward checking



IF

Contoh Forward Checking: 4-Queens



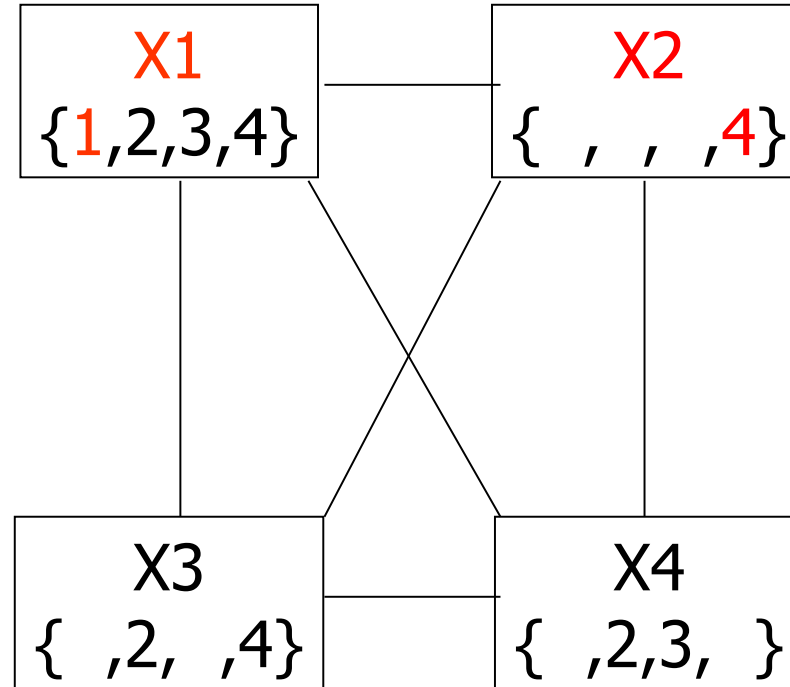
Kembali ke state setelah variabel $X1=1$,
Lalu nilai $X2=3$ dibuang



IF

Contoh Forward Checking: 4-Queens

	1	2	3	4
1	★	●	●	●
2		●		
3			●	
4		★		●



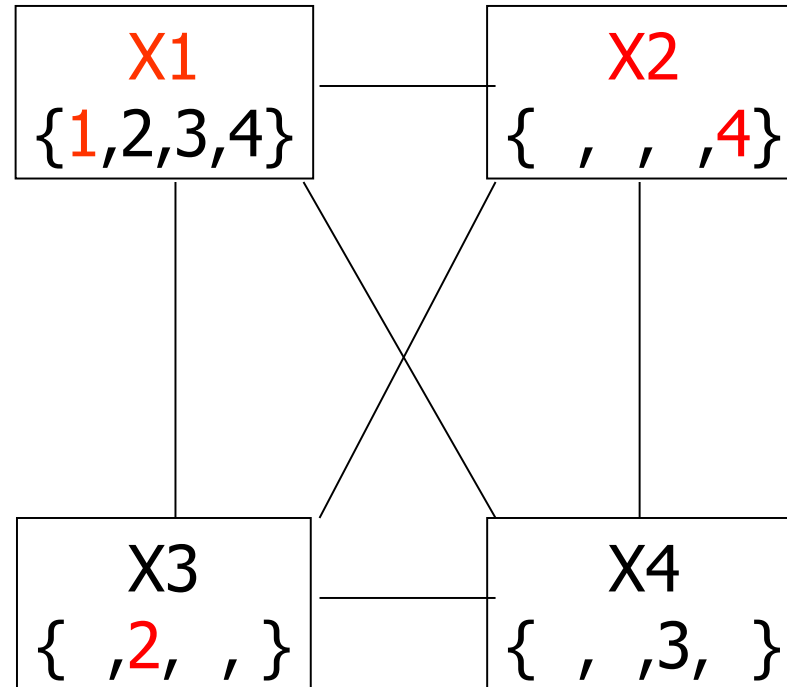
Pilih variabel $X1=1$ dan $X2=4$, sisa nilai variabel $X3$, $X4$ dengan forward checking



IF

Contoh Forward Checking: 4-Queens

	1	2	3	4
1	★	●	●	●
2		●		●
3			●	
4		★	●	●

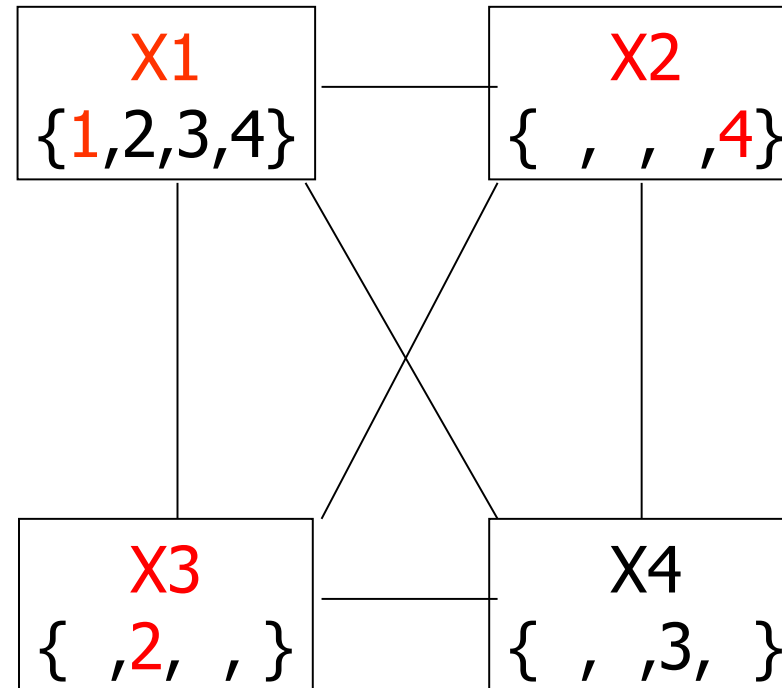




IF

Contoh Forward Checking: 4-Queens

	1	2	3	4
1	★	●	●	●
2		●	★	●
3			●	
4		★	●	●



Pilih variabel $X1=1$ dan $X2=4$ dan $X3=2$

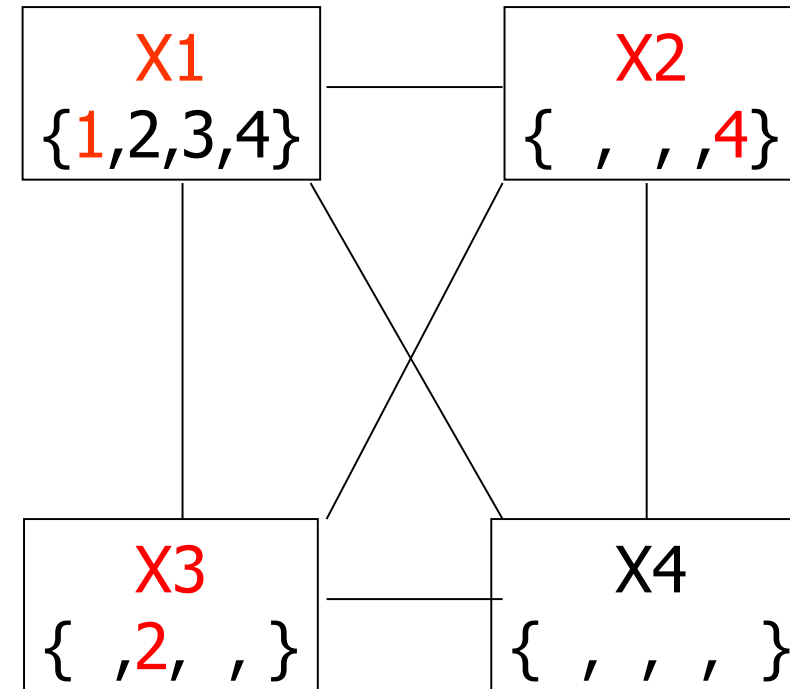


IF

Contoh Forward Checking: 4-Queens

	1	2	3	4
1	★	●	●	●
2		●	★	●
3			●	●
4		★	●	●

Pilih variabel $X1=1$ dan $X2=4$ dan $X3=2$, membuat variabel $X4$ tidak memiliki nilai sehingga Backtrack $X3$ tidak bisa karena saat $X2=4$, nilai $X3=2$; Backtrack $X2$ tidak bisa karena sudah tdk ada nilainya; **Jadi backtrack $X1=2$**



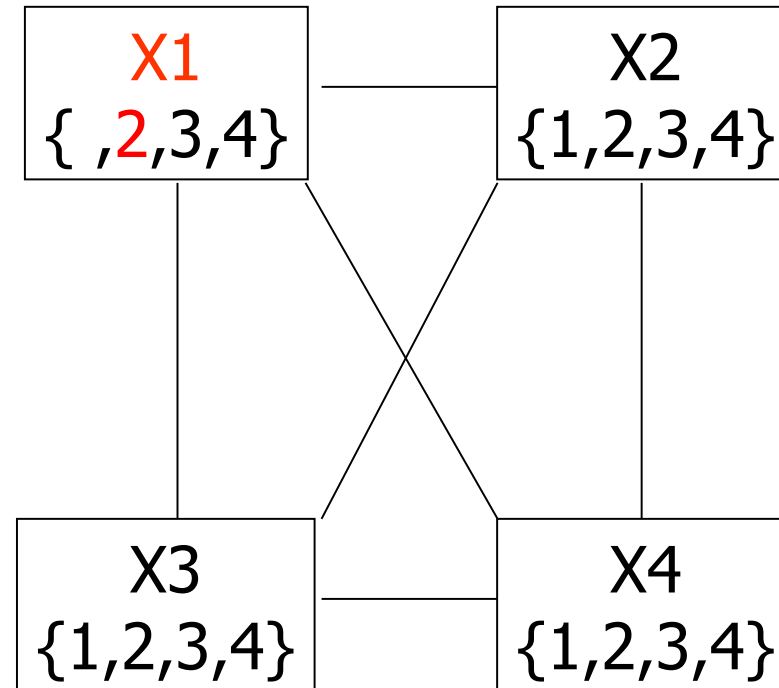
Pilih variabel $X1=1$ dan $X2=4$ dan $X3=2$, sisa nilai variabel $X4$ dengan forward checking



IF

Contoh Forward Checking: 4-Queens

	1	2	3	4
1		●		
2	★	●	●	●
3		●		
4			●	



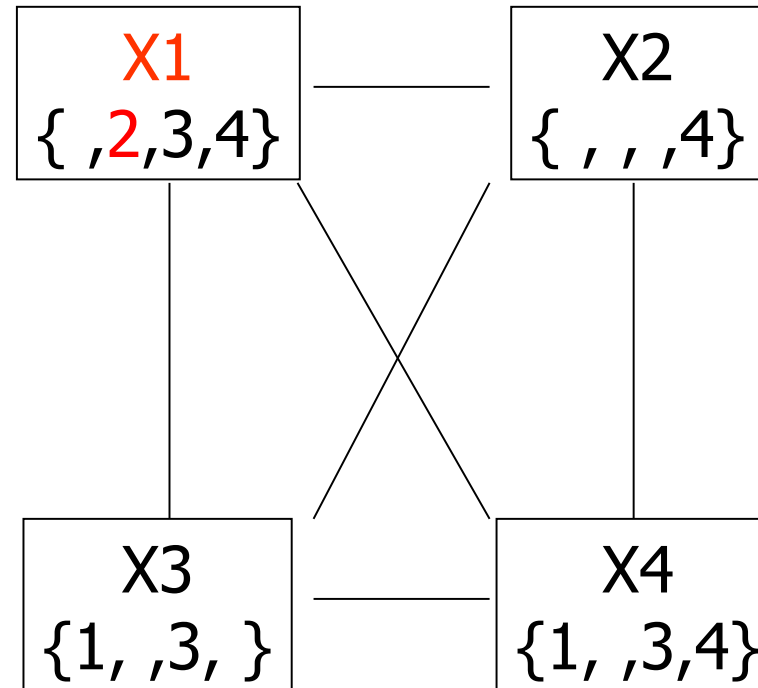
Pilih variabel $X1=2$, nilai $X1=1$ sudah dibuang karena tidak dapat menghasilkan solusi



IF

Contoh Forward Checking: 4-Queens

	1	2	3	4
1		●		
2	★	●	●	●
3		●		
4			●	

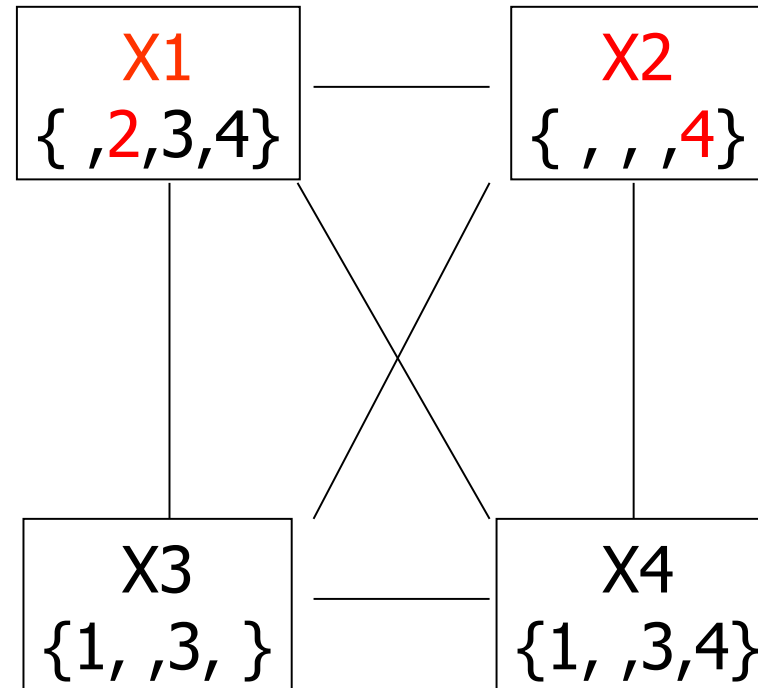
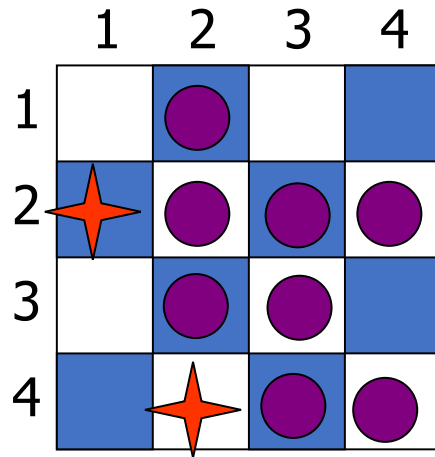


Pilih variabel $X1=2$, sisa nilai variabel $X2, X3, X4$ dengan forward checking



IF

Contoh Forward Checking: 4-Queens

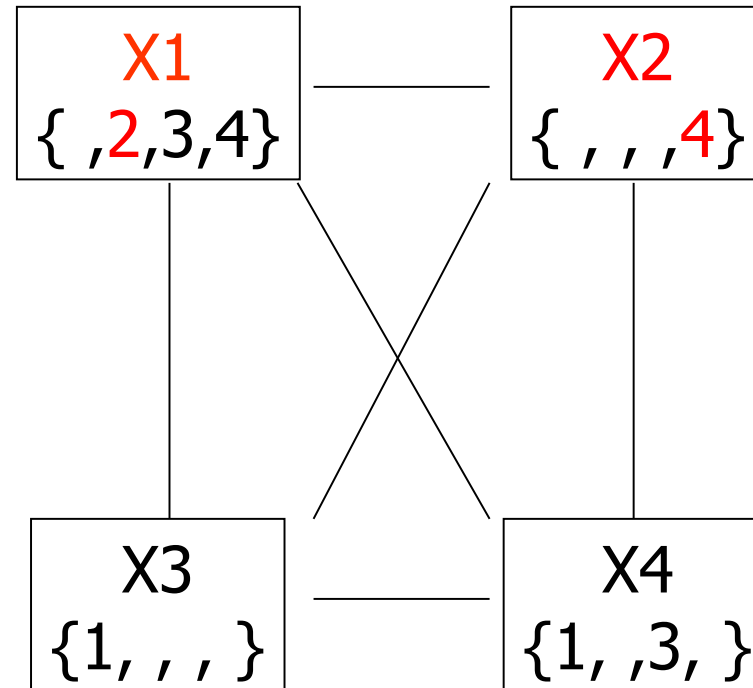
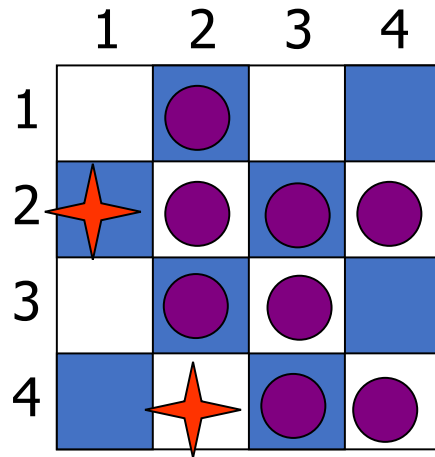


Pilih variabel X1=2 dan X2=4



IF

Contoh Forward Checking: 4-Queens

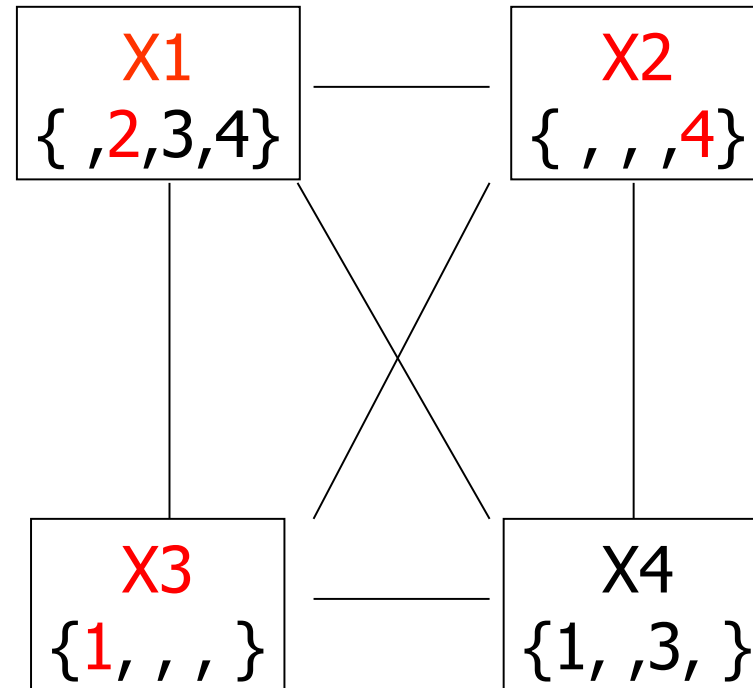
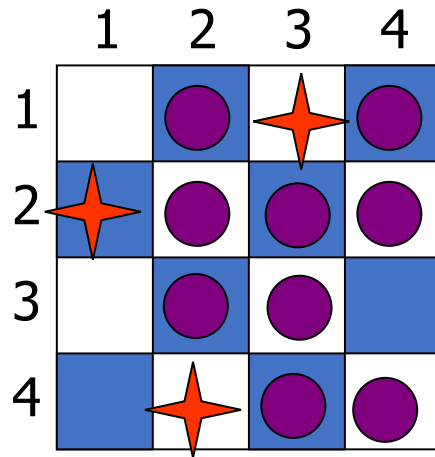


Pilih variabel $X1=2$ dan $X2=4$,
sisa nilai variabel $X3$, $X4$ dengan forward
checking



IF

Contoh Forward Checking: 4-Queens

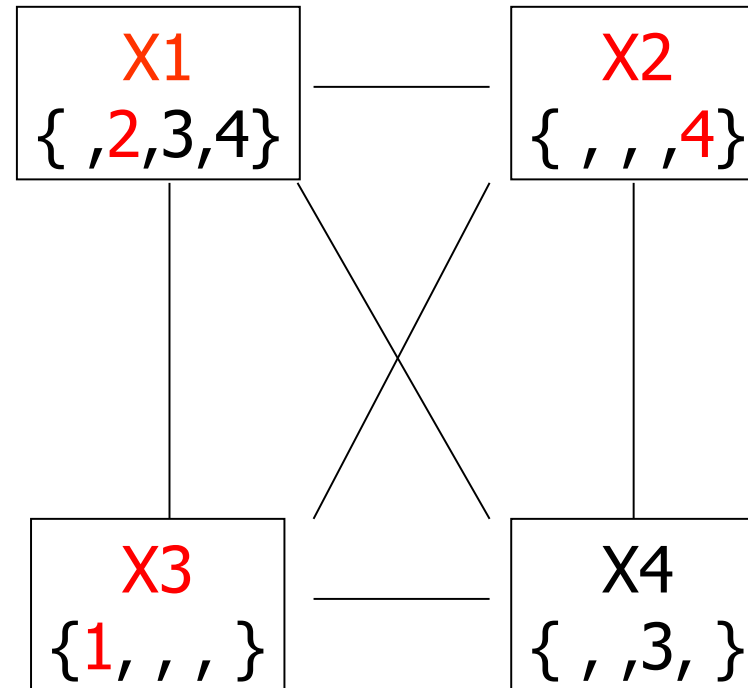
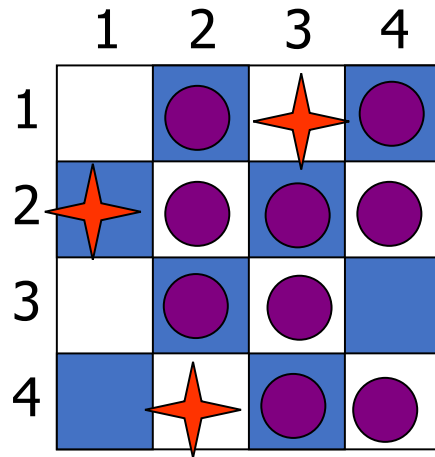


Pilih variabel $X1=2$ dan $X2=4$ dan $X3=1$



IF

Contoh Forward Checking: 4-Queens

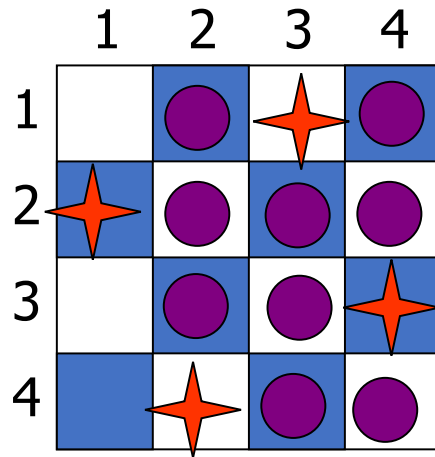


Pilih variabel $X1=2$ dan $X2=4$ dan $X3=1$,
sisa nilai variabel $X4$ dengan forward
checking

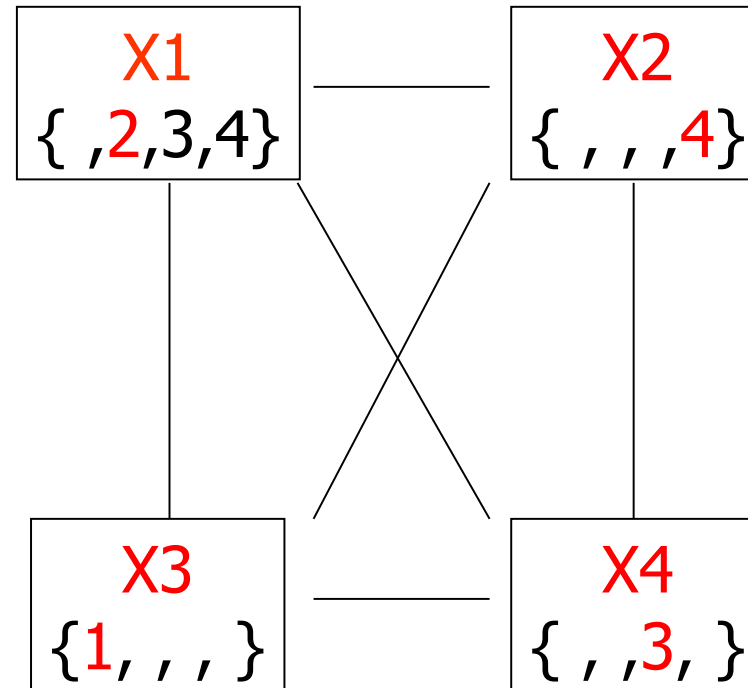


IF

Contoh Forward Checking: 4-Queens



Pilihan solusi lain dapat dilakukan dengan mencoba $X1=\{3, 4\}$



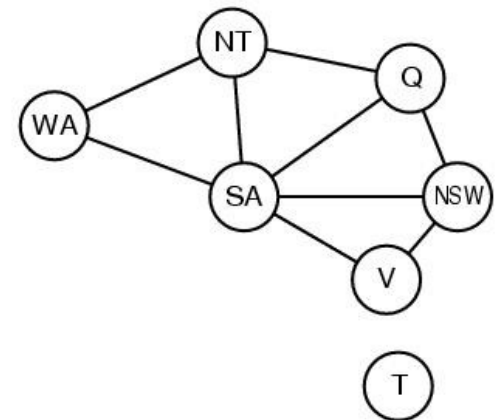
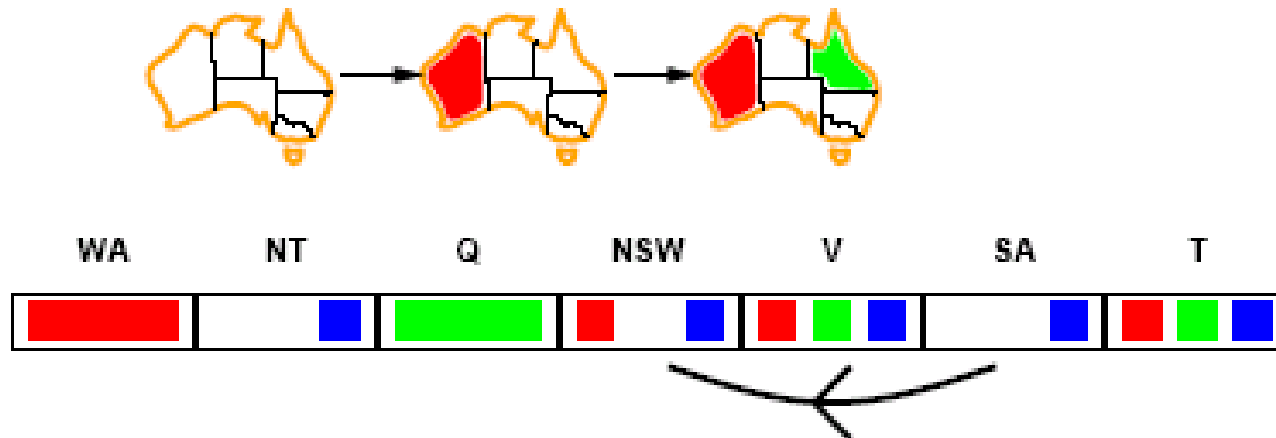
Pilih variabel $X1=2$ dan $X2=4$ dan $X3=1$ dan $X4=3$; karena sudah tidak ada nilai lain $X4$ maka **solusi ditemukan**



Constraint Propagation: Arc consistency

- $Arc X \rightarrow Y$ adalah konsisten jika untuk setiap nilai x pada X , beberapa nilai y konsisten dengan x
- Contoh setelah WA dan Q sudah ditentukan warnanya:

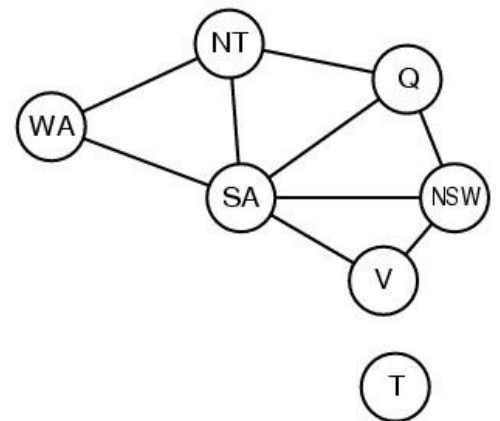
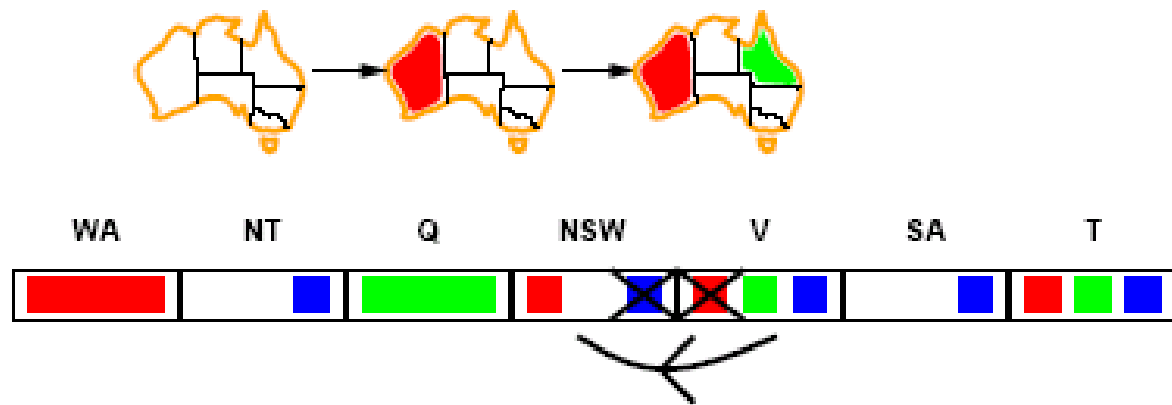
$SA \rightarrow NSW$ is consistent if
 $SA=blue$ and $NSW=red$





Constraint Propagation: Arc consistency

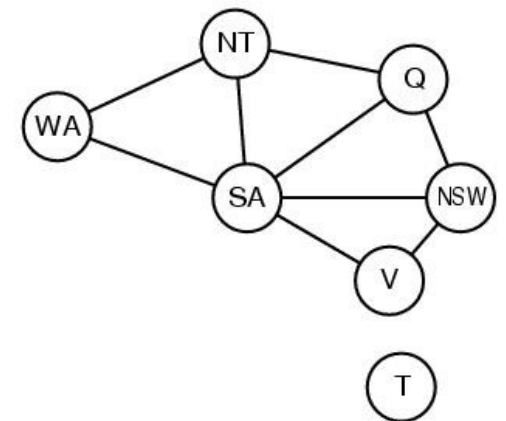
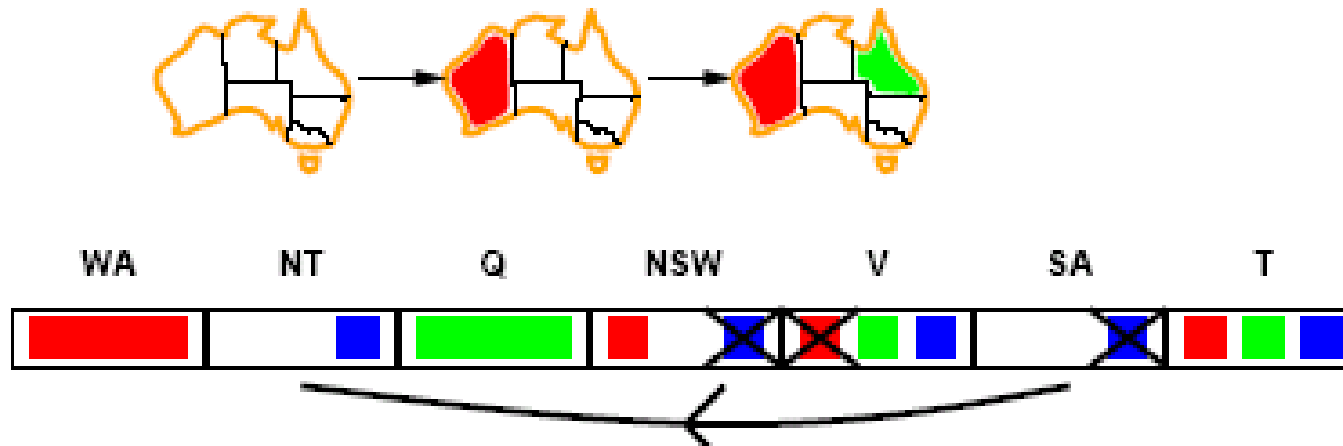
- Dapat mempertahankan *arc-consistency*:
Arc dapat konsisten dengan menghapus value *blue* dari variabel *NSW*
- Selanjutnya *propagate constraints....*
 - Check $V \rightarrow NSW$
 - Not consistent for $V = \text{red}$
 - Remove red from V





Constraint Propagation: Arc consistency

- Selanjutnya *propagate constraints*
 - $SA \rightarrow NT$ tidak *consistent*
 - Arc consistency* mendeteksi kesalahan lebih awal dari pada *Forward checking* (FC)





Arc consistency algorithm

function AC-3(*csp*) **returns** the CSP, possibly with reduced domains

inputs: *csp*, a binary CSP with variables $\{X_1, X_2, \dots, X_n\}$

local variables: *queue*, a queue of arcs, initially all the arcs in *csp*

while *queue* is not empty **do**

$(X_i, X_j) \leftarrow \text{REMOVE-FIRST}(\textit{queue})$

if REMOVE-INCONSISTENT-VALUES(X_i, X_j) **then**

for each X_k **in** NEIGHBORS[X_i] **do**

 add (X_k, X_i) to *queue*

function REMOVE-INCONSISTENT-VALUES(X_i, X_j) **returns** true iff succeeds

removed \leftarrow false

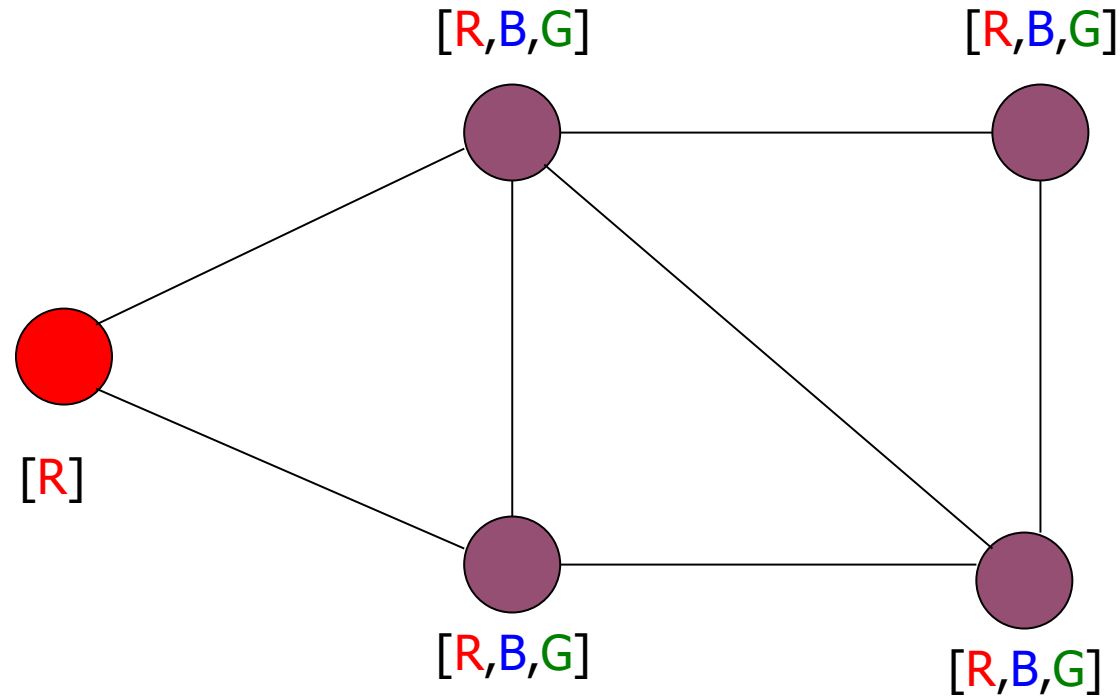
for each x **in** DOMAIN[X_i] **do**

if no value y in DOMAIN[X_j] allows (x, y) to satisfy the constraint $X_i \leftrightarrow X_j$

then delete x from DOMAIN[X_i]; *removed* \leftarrow true

return *removed*

Contoh Problem CSP



Gunakan *arc-propagation* untuk menyelesaikan problem CSP diatas!



Constraint Propagation.

Forward checking (FC) vs Arc consistency (AC)

- FC memberikan solusi dengan waktu yang lebih cepat dibanding AC
- AC membutuhkan waktu lebih lama namun melakukan *pruning* lebih efektif pada *state space*
- Ada banyak pendekatan *consistency* lain yang memberikan solusi lebih baik namun waktu lebih lama
 - Node consistency, Path consistency, dll

K-consistency

- *Arc consistency* tidak dapat mendeteksi semua *inconsistencies*:
 - *Partial assignment* $\{WA=red, NSW=red\}$ is inconsistent.
- Bentuk propagation dapat didefinisikan menggunakan notasi *k-consistency*.
- Sebuah CSP adalah *k-consistent* jika untuk semua himpunan pada $k-1$ variabel dan untuk semua penugasan pada variabel tersebut, maka *consistent value* dapat selalu diberikan pada semua variabel ke k .
 - 1-consistency = node-consistency
 - 2-consistency = arc-consistency
 - 3-consistency = path-consistency
- *Strongly k-consistent*:
 - k -consistent for all values $\{k, k-1, \dots, 2, 1\}$



Tugas No.1

Anda diminta merancang menu makanan di suatu event. Menu makanan terdiri dari Appetizer, Beverage, Main_Course, dan Dessert. Domain dari setiap variabel menu makanan adalah:

Appetizer : veggies, escargot

Beverage : water, soda, milk

Main_Course : fish, beef, pasta

Dessert : apple pie, ice cream, cheese

Semua tamu akan mendapat menu yang sama dan harus mengikuti batasan (constraints) diet sebagai berikut:

- (i) Batasan vegetarian: Appetizer harus sayuran (veggies) atau Main_Course harus pasta atau fish.
- (ii) Batasan total biaya: Jika memilih escargot, maka pilihan untuk beverage hanya water.
- (iii) Batasan kebutuhan kalsium: Anda harus memilih setidaknya salah satu di antara milk, ice cream, atau keju.

Merujuk pada permasalahan tersebut di atas, jawablah pertanyaan berikut:

- (a) Gambarkan *constraint graph* dari variabel-variabel pada permasalahan di atas.
- (b) Seandainya Anda memilih untuk menghidangkan escargot sebagai Appetizer dan belum ada pilihan value untuk variabel yang lain, sebutkan value apa saja yang dieliminasi setelah dilakukan forward checking untuk menunjukkan sisa domain pada setiap variabel.
- (c) Sebutkan 3 contoh solusi dari CSP di atas, jika ada.



Tugas No.2

You are in charge of scheduling for computer science classes that meet Mondays, Wednesdays and Fridays. There are 5 classes that meet on these days and 3 professors who will be teaching these classes. You are constrained by the fact that each professor can only teach one class at a time.

The classes are:

- Class 1 - Intro to Programming: meets from 8:00-9:00am
- Class 2 - Intro to Artificial Intelligence: meets from 8:30-9:30am
- Class 3 - Natural Language Processing: meets from 9:00-10:00am
- Class 4 - Computer Vision: meets from 9:00-10:00am
- Class 5 - Machine Learning: meets from 9:30-10:30am

The professors are:

- Professor A, who is available to teach Classes 3 and 4.
- Professor B, who is available to teach Classes 2, 3, 4, and 5.
- Professor C, who is available to teach Classes 1, 2, 3, 4, 5.

- a. Formulate this problem as a CSP problem
- b. Solve this problem using CSP



- TERIMA KASIH -



Teknik Informatika
department of informatics
Fakultas Teknologi Informasi



www.its.ac.id



[its_campus](#)



[institut teknologi sepuluh nopember](#)