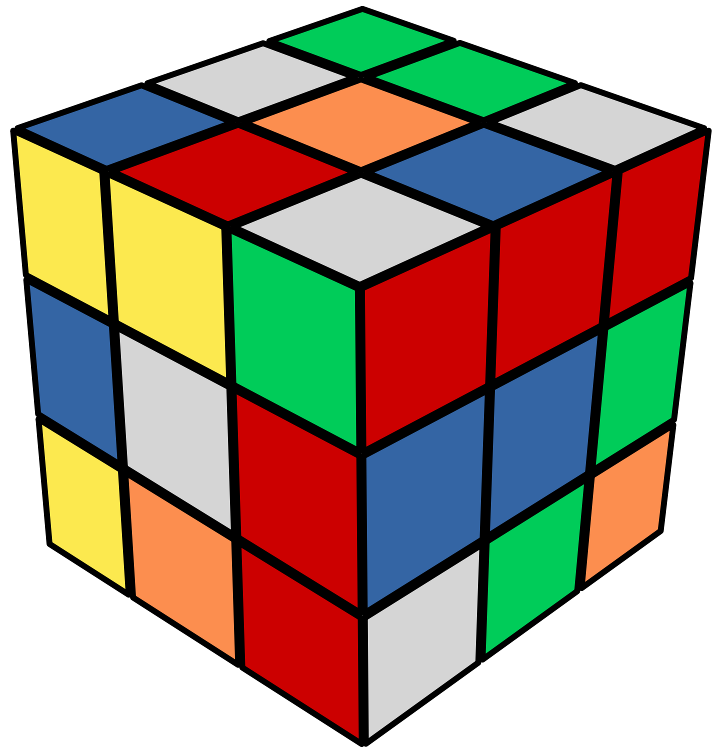
COMPSCI 761: ADVANCED TOPICS IN ARTIFICIAL INTELLIGENCE

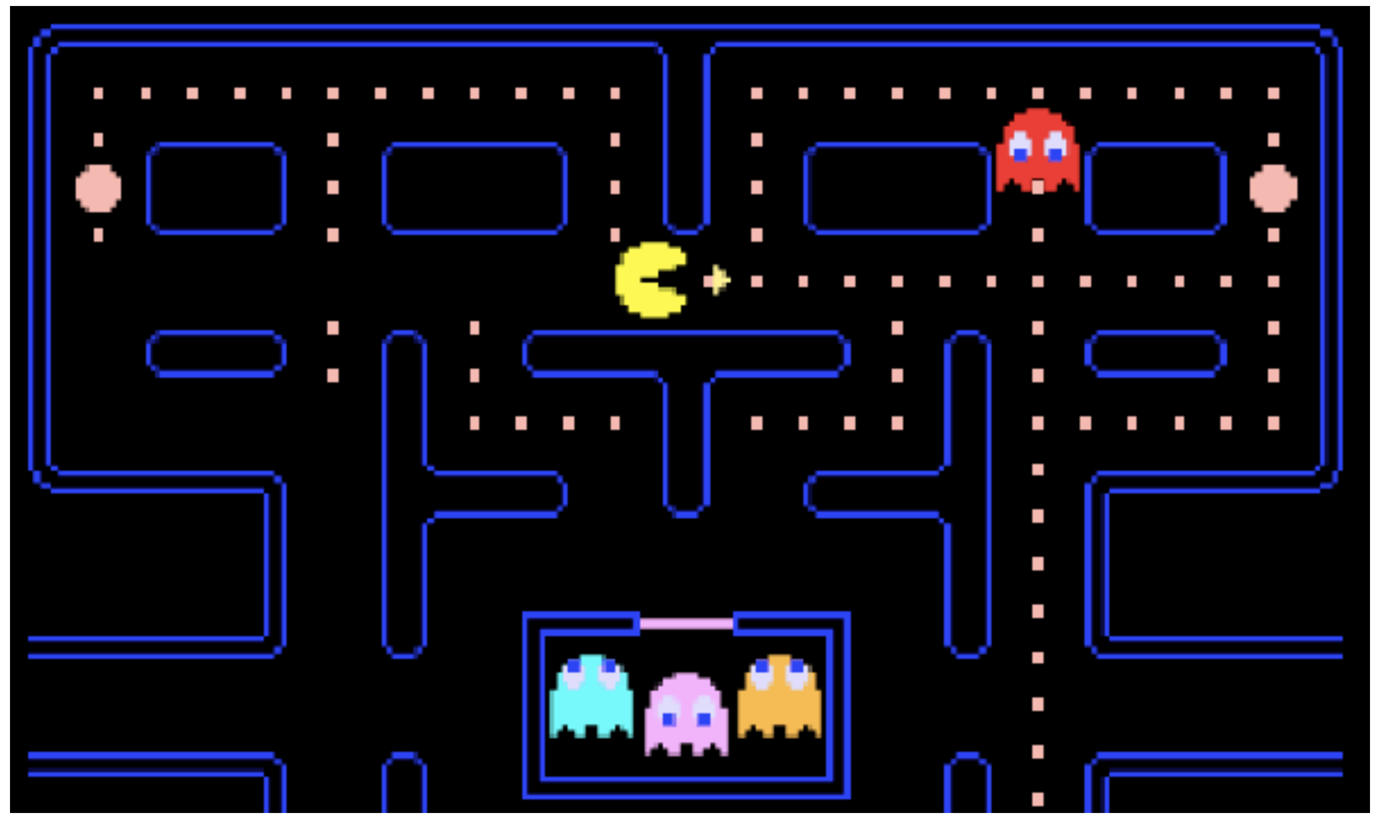
SUMMARY II

Anna Trofimova, July 2022

SEARCH PROBLEM VS CSP VS GAME



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



CONSTRAINT SATISFACTION PROBLEMS (CSPS)

- Constraint Satisfaction Problems are defined by a set of variables X_i, each with a domain D_i of possible values, and a set of constraints C that specify allowable combinations of values.

- The aim is to find an assignment of the variables X_i from the domains D_i in such a way that none of the constraints C are violated.
  - i.e. all of the constraints C are satisfied

TYPES OF CONSTRAINTS

- Unary constraints involve a single variable
  - M ≠ 0
- Binary constraints involve pairs of variables
  - SA ≠ WA
- Higher-order constraints involve 3 or more variables
  - $Y = D + E$ or $Y = D + E - 10$
- Inequality constraints on Continuous variables
  - EndJob1 + 5 ≤ StartJob3
- Soft constraints (Preferences)
  - 11am lecture is better than 8am lecture!

EXAMPLE: MAP-COLOURING ★

Variables: WA, NT, Q, NSW, V, SA, T

Domains: Di = {red, green, blue}

Constraints: adjacent regions must have different colours **e.g. WA≠ NT, etc.**

BACKTRACKING SEARCH SPACE PROPERTIES

The search space for this Depth First Search has certain very specific properties:

- If there are n variables, every solution will occur at exactly depth n.
- Variable assignments are commutative

  [WA = red then NT = green] same as [NT = green then WA = red]

EXAMPLE: BACKTRACKING ★

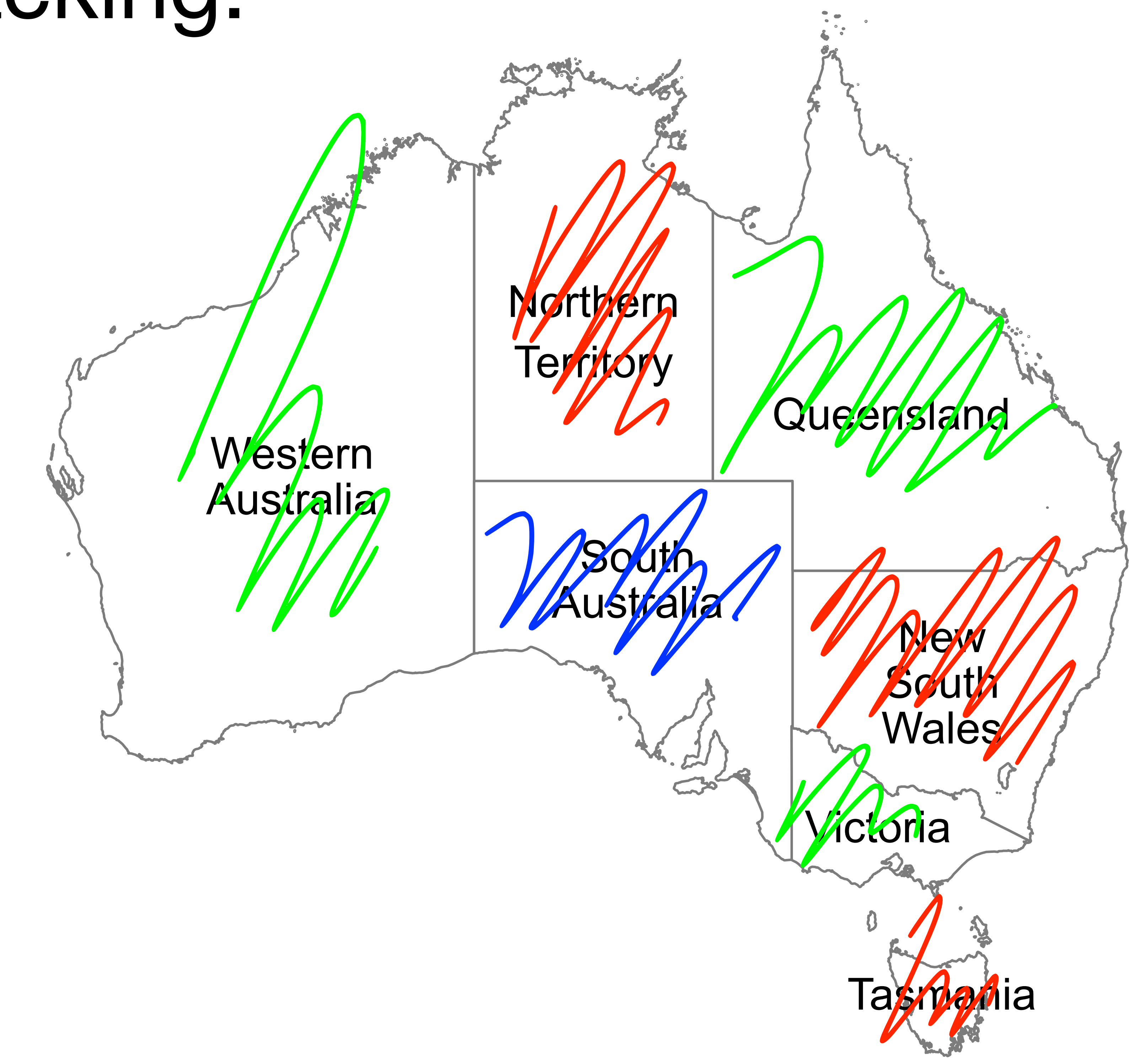
- Given that WA is green, NT is red, apply backpacking choosing the remaining states in the alphabetical order and assigning colours in the following order {r, g, b}. Select the correct outcome of the backtracking:
  - a. No solution
  - b. NSW = r, Q = g, SA = b, WA = g, T = r
  - c. NSW = r, Q = b, SA = r, WA = g, T = r
  - c. NSW = r, Q = b, SA = r, WA = b, T = r

EXAMPLE: BACKTRACKING ★

- Given that WA is green, NT is red, apply backpacking choosing the remaining states in the alphabetical order and assigning colours in the following order {r, g, b}. Select the correct outcome of the backtracking:

NSW, NT , Q , SA, T , V , WA
rgb , R , rgb, rgb, rgb, rgb, G
R , R , rgb, rgb, rgb, rgb, G
R , R , G , rgb, rgb, rgb, G
R. , R , G, B, rgb, rgb, G
R , R , G, B, R, rgb, G
R, R, G, B, R, G, G

=> R, G, B, R, G



EXAMPLE: BACKTRACKING ★

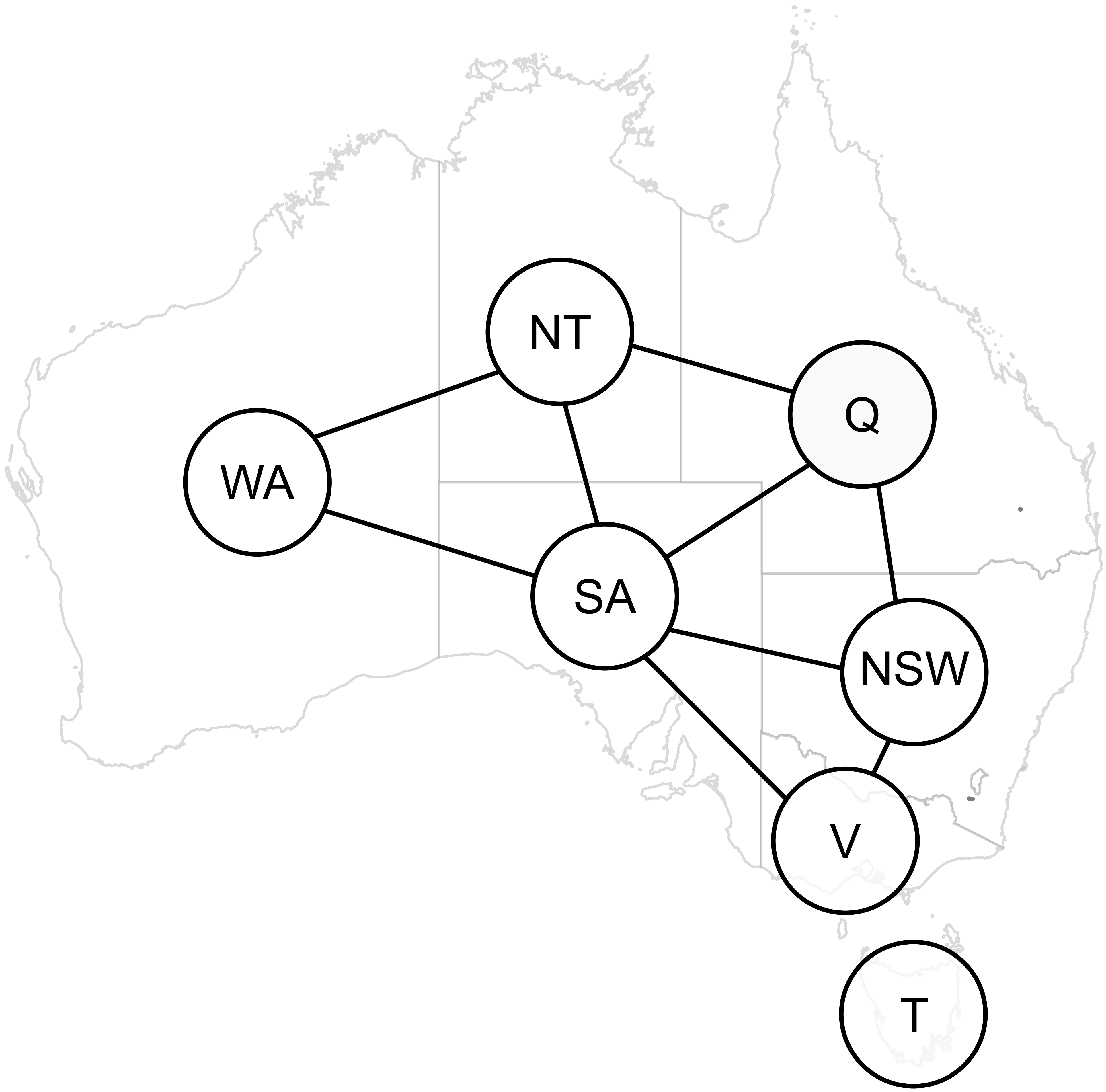
- Given that WA is green, NT is red, apply backpacking
  - With arc consistency checking
  - Using Minimum Remaining Value
  - Degree heuristic
  - Using least constrained value
- … Select the correct outcome:
  - a. No solution
  - b. NSW = r, Q = g, SA = b, WA = g, T = r
  - c. NSW = r, Q = b, SA = r, WA = g, T = r
  - c. NSW = r, Q = b, SA = r, WA = b, T = r

VARIABLE ELIMINATION

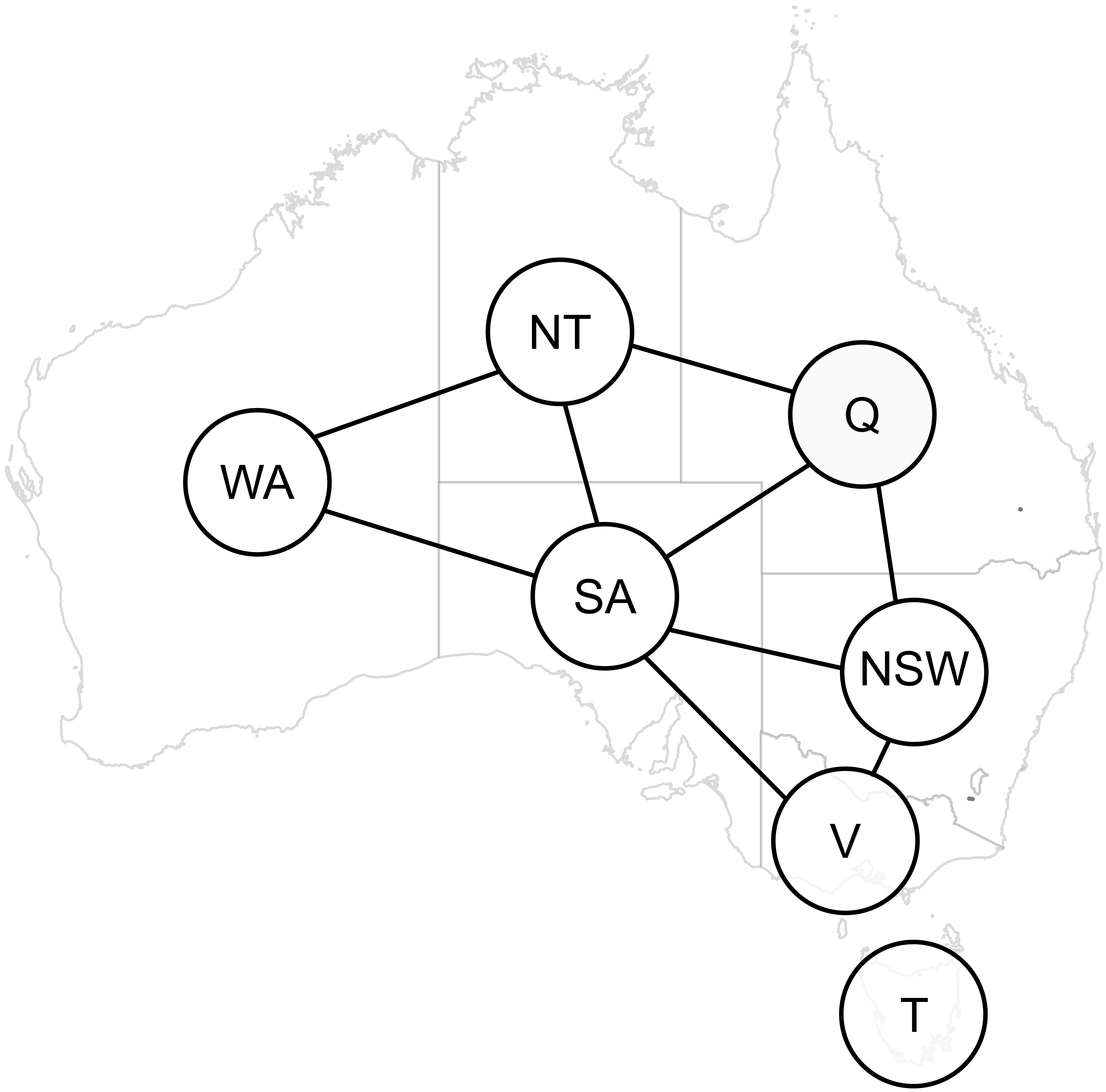
- If there is only one variable, return the intersection of its (unary) constraints
- Otherwise
  - Select a variable X
  - Join the constraints in which X appears, forming constraint R1
  - Project R1 onto its variables other than X, forming R2
  - Replace all of the constraints in which X appears by R2
  - Recursively solve the simplified problem, forming R3
  - Return R1 joined with R3

EXAMPLE: VARIABLE ELIMINATION ★

- Eliminate variable V, and then join the new constraint with the constraint SA ≠ NSW
- V ≠ NSW, V ≠ SA



EXAMPLE: VARIABLE ELIMINATION ★



LOCAL SEARCH?

- In Systematic Search a solution can be a path or a state!
- In Local Search a state is a solution!
    - Since it doesn't maintain a path ("only a current state") it can't return a path
    - Can use a goal test or partial goal state
        - (but not a complete goal state – why?)

LOCAL SEARCH ALGORITHMS

Hill Climbing – chose best child

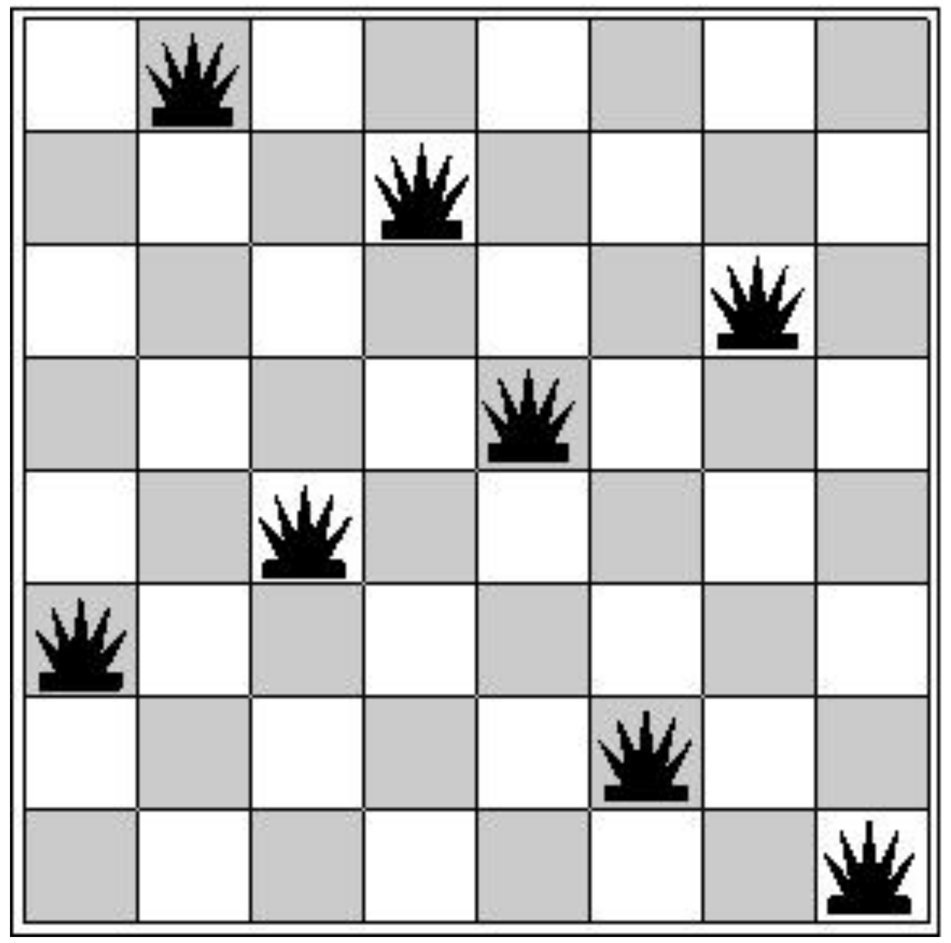
Hill Climbing with Sideways Moves – chose best child or equal child

Tabu Search – keep list of nodes you have been to and don't go back

Enforced Hill Climbing – use breadth first search until you find a "better" node

EXAMPLE: HILL CLIMBING ★

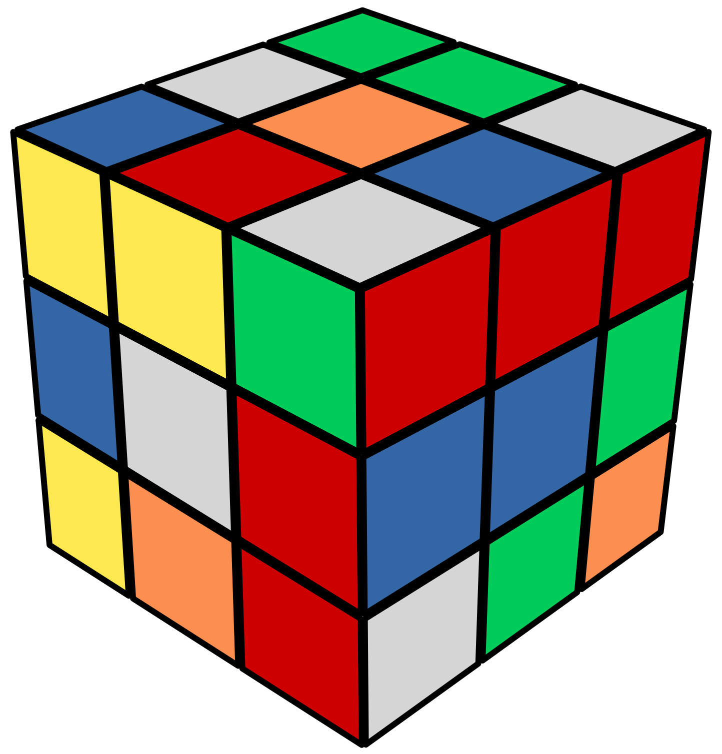
- How many violations are there?
- Label the queens according to the column number. Run Hill-climbing with side way moves and Min Conflicts and in case of a tie prioritise the queen on the left. With respect to the solution, select the position for the 2nd queen:
    - No solution
    - Row 1
    - Row 4
    - Row 7



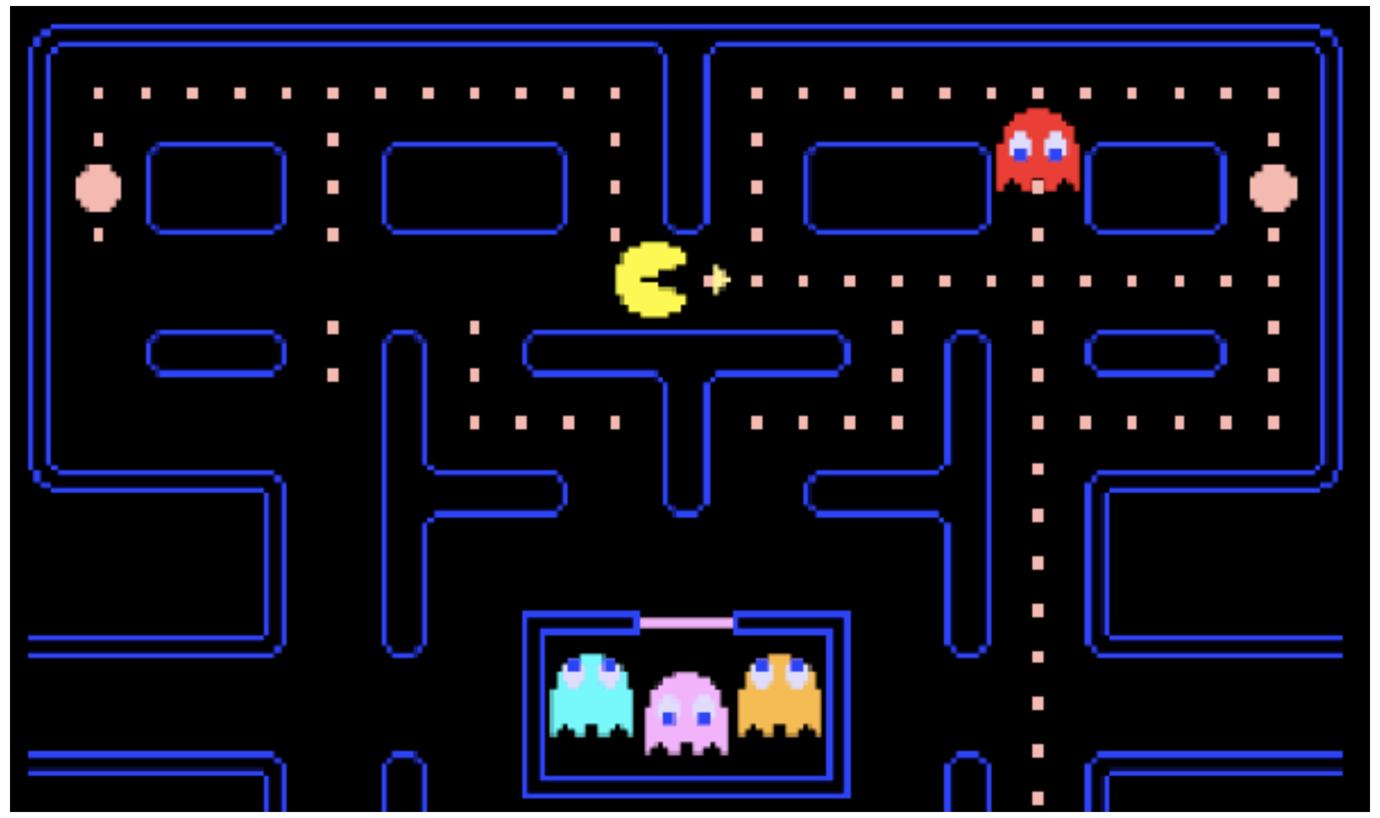
STOCHASTIC SEARCH ALGORITHMS

- Stochastic Hill Climbing – chose probabilistically from among better children based on their fitness
- First Choice Hill-Climbing - chose the first better random child
- Random Walking Hill Climbing – chose probabilistically between the best child and a random child
- Random Restart Hill Climbing – do any hill climbing method with randomly restarts to try and get a better result
- Simulated Annealing – first choice hill climbing with a stochastic chance of choosing a worse node. The chance of choosing a worse node reduces over time with the temperature schedule

SEARCH PROBLEM VS CSP VS GAME



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



- "Unpredictable" opponent → specifying a move for every possible opponent reply
- Time limits → unlikely to find optimal solution, must approximate

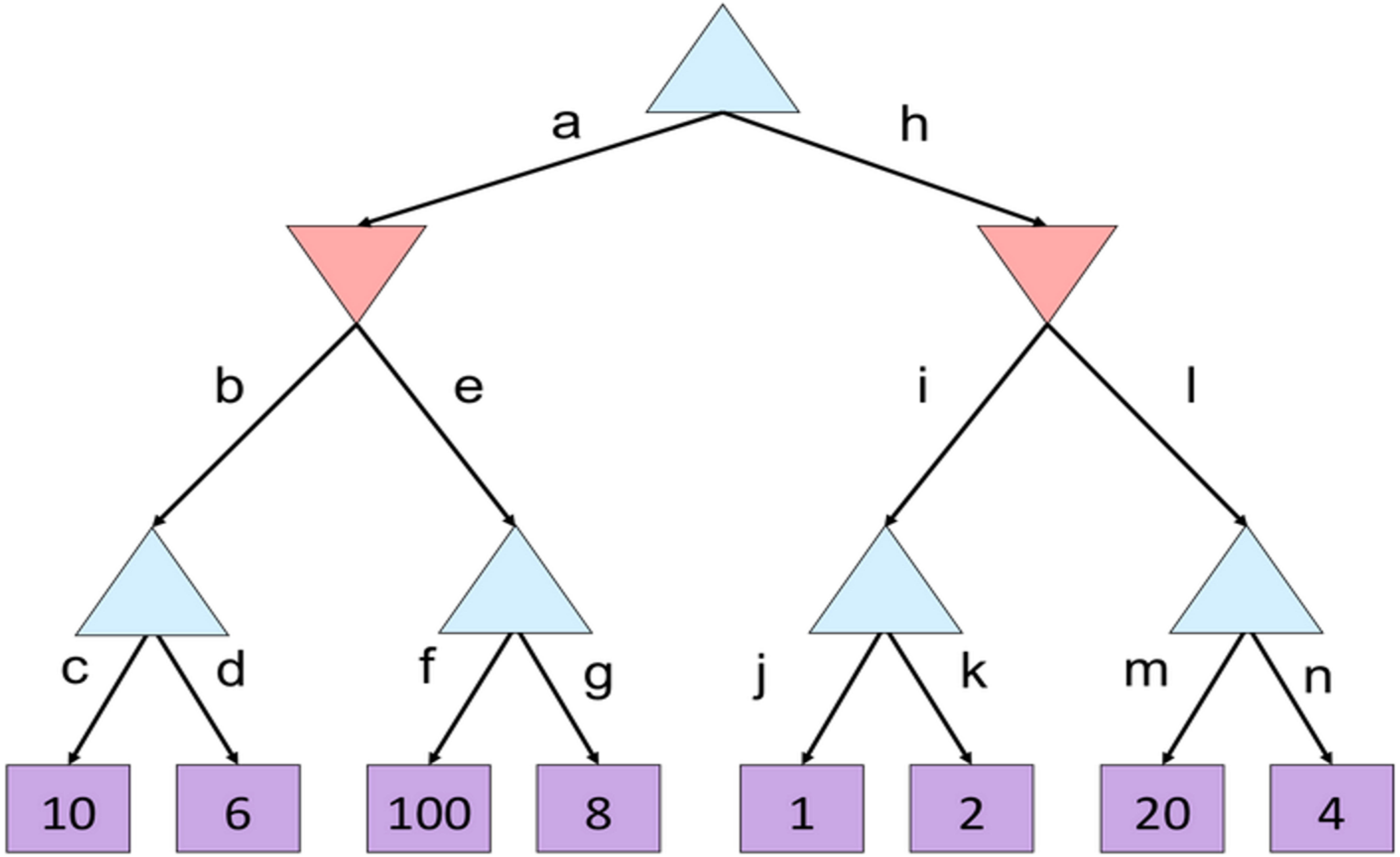
MINIMAX SEARCH

```
function minimax-decision(s) returns an action
    return the action a in Actions(s) with the highest
     minimax_value(Result(s,a))
```

⇕

```
function minimax_value(s) returns a value
    if Terminal-Test(s) then return Utility(s)
    if Player(s) = MAX then return max_{a in Actions(s)} minimax_value(Result(s,a))
    if Player(s) = MIN then return min_{a in Actions(s)} minimax_value(Result(s,a))
```

EXAMPLE: MINIMAX ★



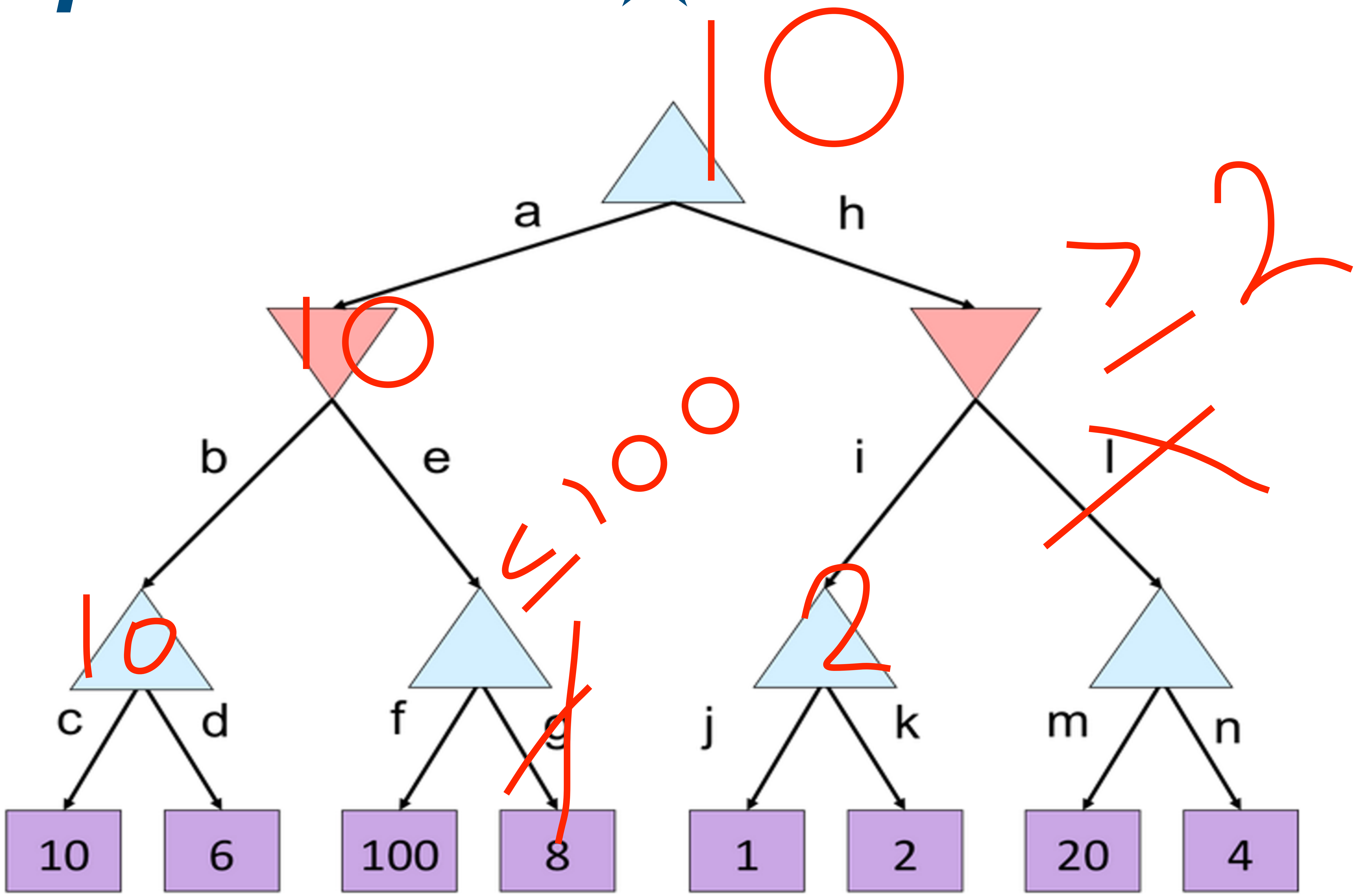
α-β PRUNING ALGORITHM

α: MAX's best option on path to root
β: MIN's best option on path to root

```
def max-value(state, α, β):
    initialize v = -∞
    for each successor of state:
        v = max(v, value(successor, α, β))
        if v ≥ β
            return v
        α = max(α, v)
    return v
```

```
def min-value(state , α, β):
    initialize v = +∞
    for each successor of state:
        v = min(v, value(successor, α, β))
        if v ≤ α
            return v
        β = min(β, v)
    return v
```

EXAMPLE: α-β PRUNING ★



EXPECTIMAX SEARCH

function decision(s) **returns** an action

  return the action a in Actions(s) with the highest value(Result(s,a))

function value(s) **returns** a value

  if Terminal-Test(s) **then return** Utility(s)

  if Player(s) = MAX **then return** $\max_{a \text{ in Actions}(s)}$ value(Result(s,a))

  if Player(s) = MIN **then return** $\min_{a \text{ in Actions}(s)}$ value(Result(s,a))

  if Player(s) = CHANCE **then return** $\text{sum}_{r \text{ in chanceEvent}(s)}$ Pr(r) * value(Result(s,r))

EXAMPLE: EXPECTIMAX ★