Artificial Intelligence (CS6380)

Constraint satisfaction

Search techniques considered till now

- **BFS, DFS, Uniform cost, Greedy Best First, A*, IDA***
  - Systematic search methods
  - All are complete on finite spaces
  - Some have guarantees of optimality
- **Local search**
  - Not systematic and hence incomplete
  - Memory efficient, useful in infinite state spaces

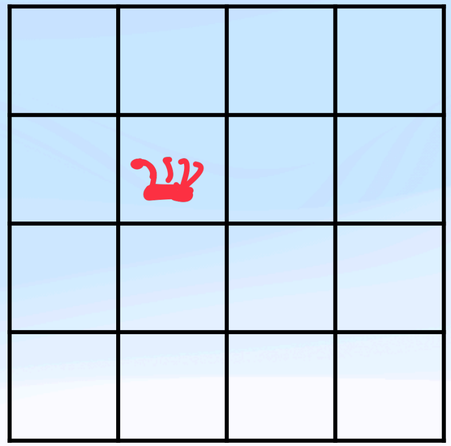
Till now we assumed that state is atomic.

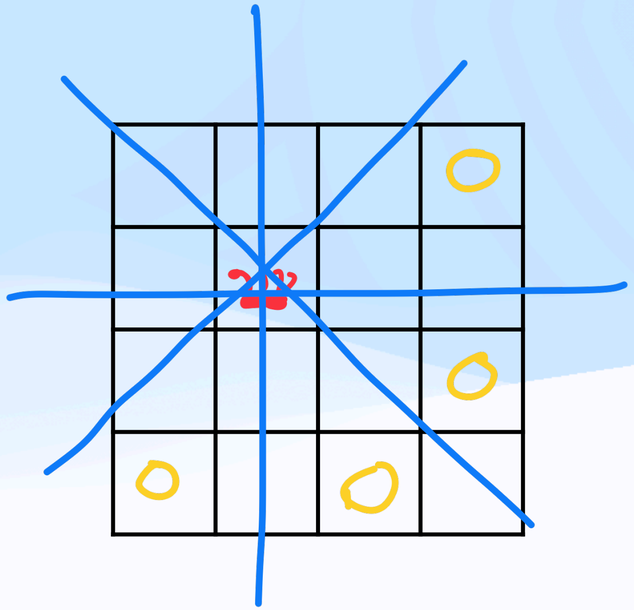
Can we have more details about the state? Use a factored representation

N Queens problem

N x N empty chess board

- Place N queens such that no queen is under attack by any other queen





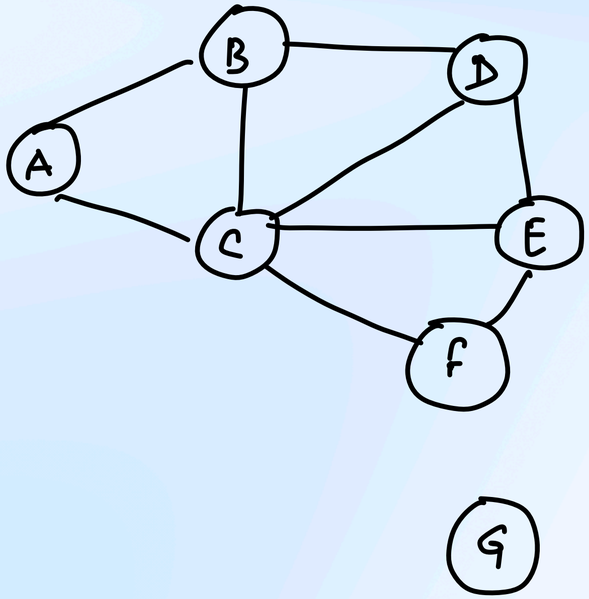
Possible approach:

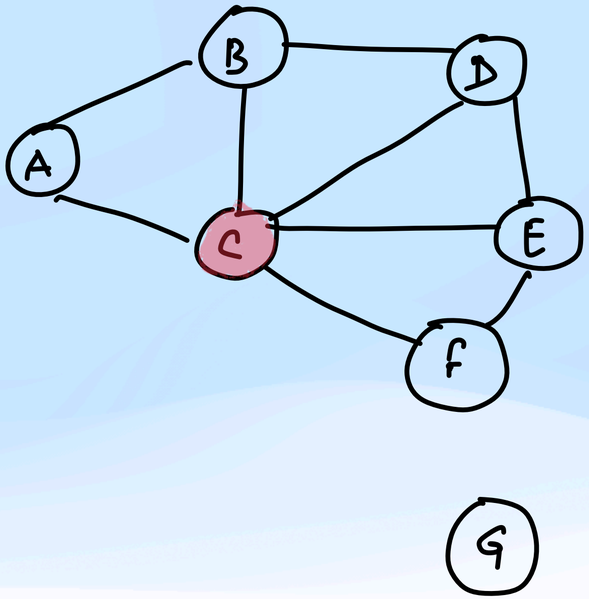
- Start one queen at a time, place the next queen
- If some queen is not placed, try placing it
- Check goal state

Map colouring problem

A map of a country

- Assign colours to regions so that adjacent regions are having different colours
- Equivalently vertex colouring of a graph





Factored representation and CSP

State

- Set of variables each of which has a value
- Variables have constraints

Goal

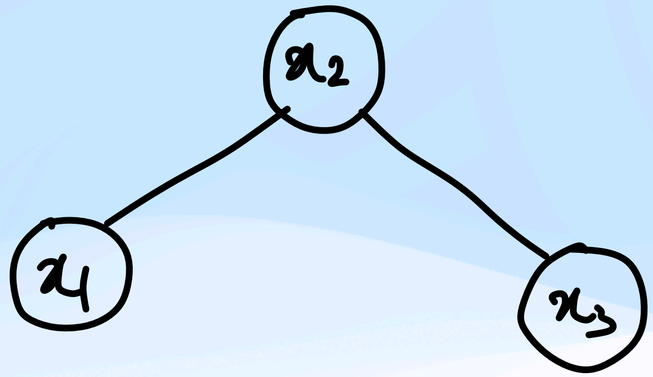
- Assignment of values to all variables such that constraints are satisfied

Constraint satisfaction problem

- A set of variables { x1, x2, …, x_n }
- A set of domains { D1, D2, …, D_n }
- A set of constraints that specify allowable combinations of values

CSP : first example

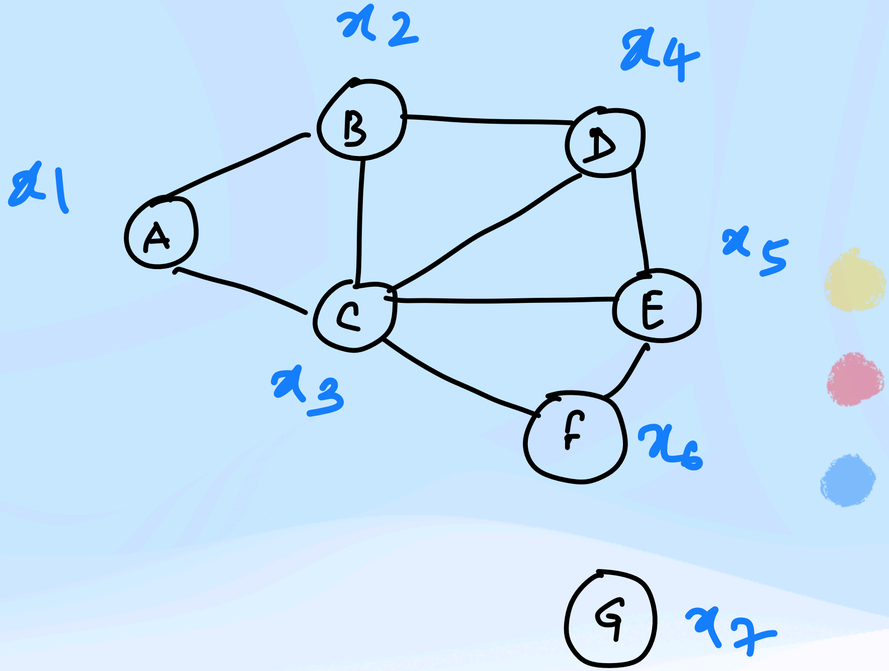
- {x1, x2, x3}
- D1 = D2 = D3 = { 1, 2, 3}
- C12, C23



- **C12 : { <a, b> | a in D1, b in D2, a < b}**
- **C23 : { <a, b> | a in D2, b in D3, a < b}**

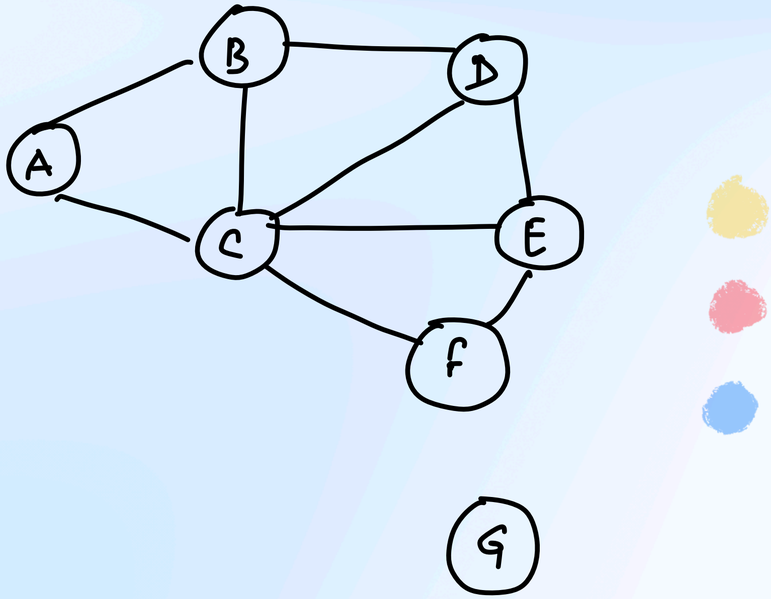
- **C12 : { <1, 2> <1, 3> <2, 3>}**
- **C23 : { <1, 2> <1, 3> <2, 3>}**

Map colouring as CSP



A map of a country

- Assign colours to regions so that adjacent regions are having different colours
- Equivalently vertex colouring of a graph



- **C12 : { <a, b> | a in D1, b in D2, a != b}**
- **C23 : { <a, b> | a in D2, b in D3, a != b}**
- **…**

N Queens as CSP

Variables : Q_a Q_b Q_c Q_d

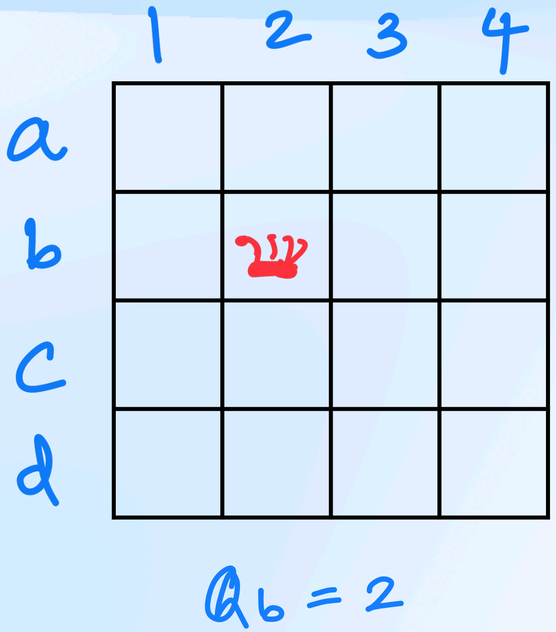
Q_a is for row a

Q_b is for row b

⋮

N x N empty chess board

- Place N queens such that no queen is under attack by any other queen



- **C12 : { <x, y> | x in D1, y in D2, x != y}**
- **C13 : { <x, y> | x in D2, y in D3, x != y}**
- **…**

if $Q_b = 2$ can $Q_c = 3$?

$Q_c = 1$?