Week 2: Search and Constraint Satisfaction Problems

Dr. Miqing Li

What is Search?

- Search is not about prediction but about choice and decision-making, such as planning and assigning.
- Search techniques are universal problem-solving methods.
- Examples of search: path-finding (Google map), taking next move in games (AlphaGo), and task assigning (Uber taxi).

> Search is the process of navigating from a start state to a goal state by transitioning through intermediate states.

Structure of Week2-Week4

- Week 2: Search and Constraint Satisfaction Problems (CSPs)
- Week 3: Solving CSPs
- Week 4: Games

Office hours: 15:00-17:00 Thursday, drop by in person: room 212 in the School of Computer Science; online: https://bham-ac-uk.zoom.us/j/2066382427

What we learn today

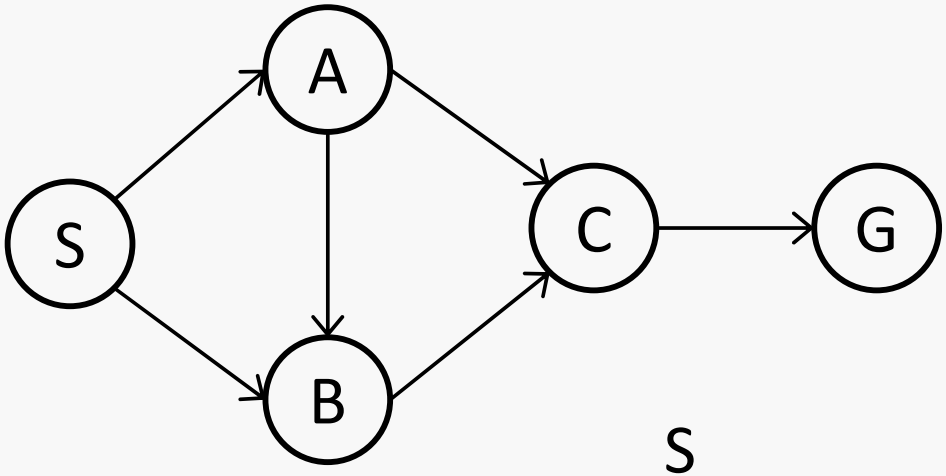
- Search recap
- Constraint Satisfaction Problems:
  A formal representation language
  of a large class of search problems

Search Problems

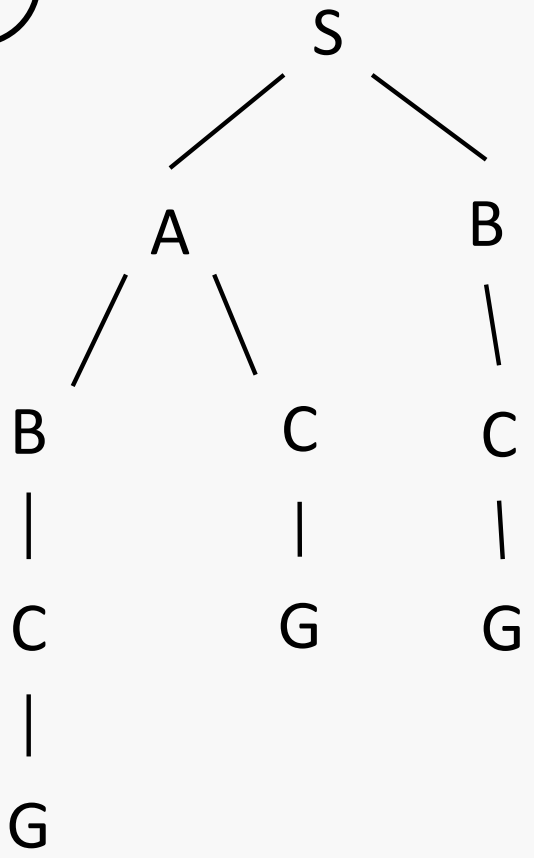
- Search is the process of navigating from the start state to a goal state by transitioning through intermediate states.
- A search problem consists of
  - State space: all possible states.
  - Start state: where the agent begins the search.
  - Goal state: the desired state that the agent is looking for.
  - Goal test: whether the goal state is achieved or not.
  - A successor function (also called transition function, often associated with a cost): given a certain state, what actions are available and for those actions what are the next stage the agent will land into.
- A solution is a sequence of actions which transforms the start state to a goal state.

State Space Graph and Search Tree

- State space graph: a mathematical representation of a search problem
  - Nodes represent states.
  - Arcs represent successor functions.

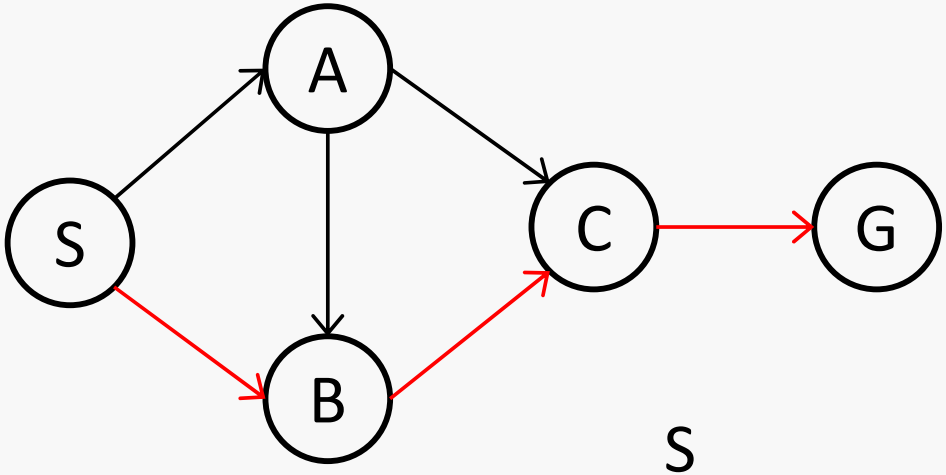


- A search tree: the process of solving the search problem can be abstracted as a search tree.
  - The start state is the root node.
  - Children correspond to successors.
  - Nodes show states, but correspond to plans (i.e., paths) that achieve those states.

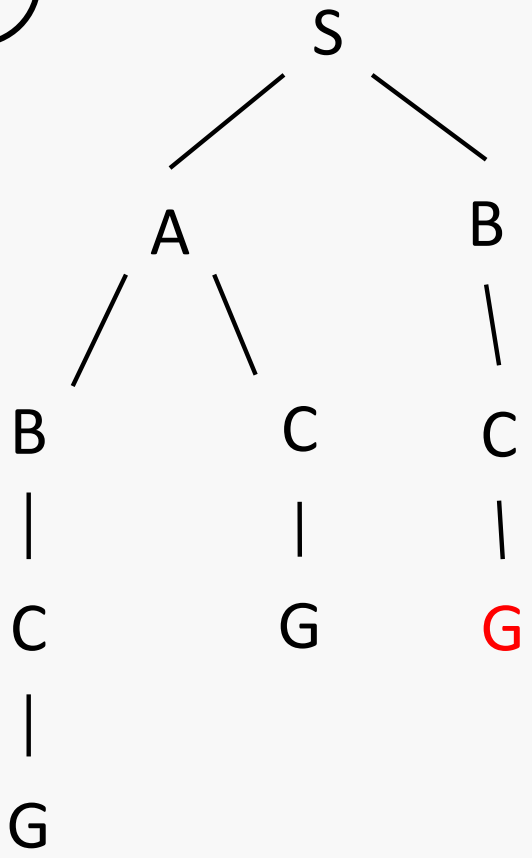


State Space Graph and Search Tree

- State space graph: a mathematical representation of a search problem
  - Nodes represent states
  - Arcs represent successor functions



- A search tree: the process of solving the search problem can be abstracted as a search tree.
  - The start state is the root node
  - Children correspond to successors
  - Nodes show states, but correspond to plans (i.e., paths) that achieve those states.

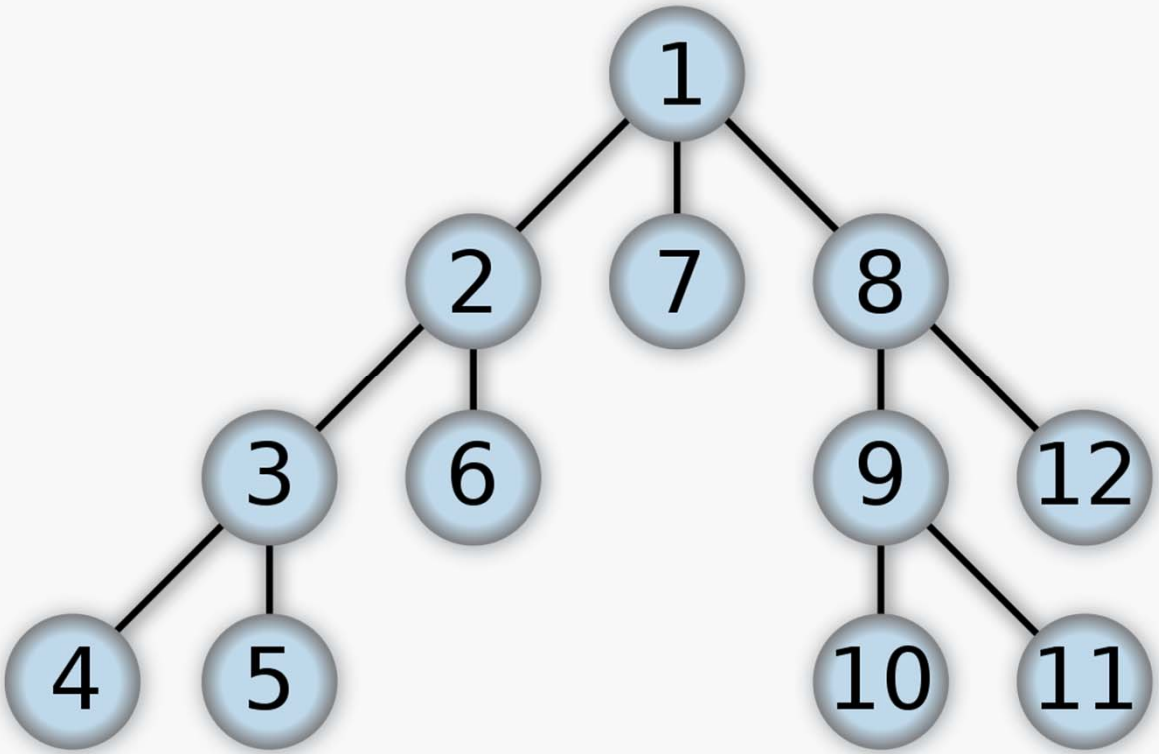


Generic tree search

```
function Tree-Search (problem, strategy) return a solution, or failure
    initialise the search tree via setting Frontier to be the start state of problem
    loop do
        if there are no nodes in Frontier for expansion then return failure
        else choose a node in Frontier for expansion according to strategy and
         remove it from Frontier
        if the chosen node is a goal state then return the corresponding solution
        else expand the node based on problem and add the resulting nodes to Frontier
    end
```

- Important things:
  - Frontier: to store the nodes waiting for expansion.
  - Expansion: to find and display the children of the node.
  - Expansion strategy: to decide which node in Frontier to expand, also called to explore.

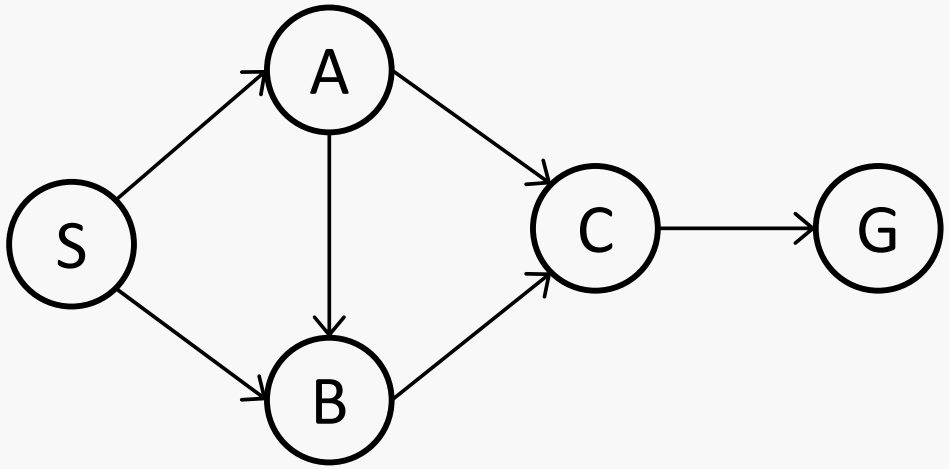
Depth-First Search (DFS)

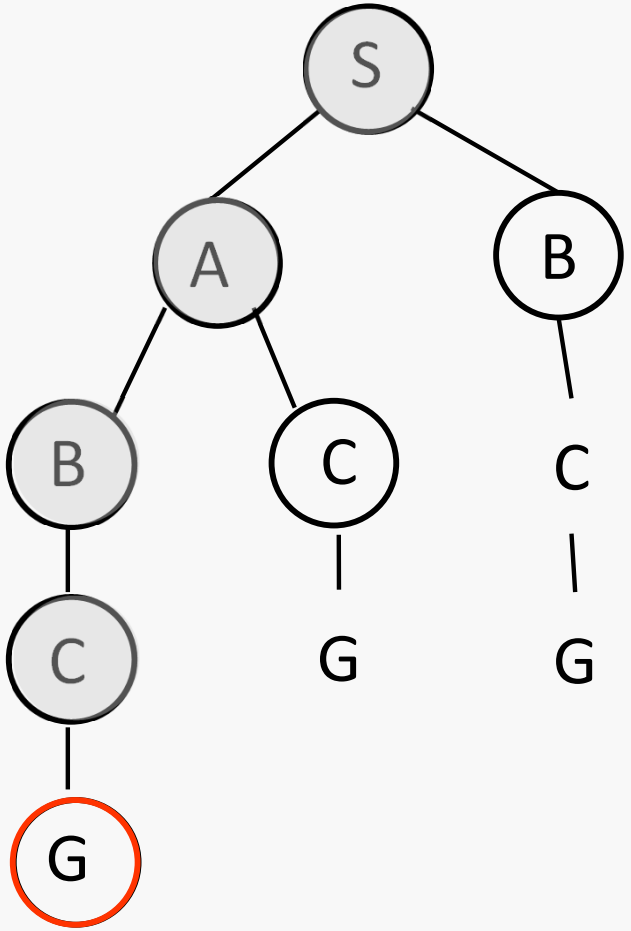


https://en.wikipedia.org/wiki/Depth-first_search

- The number inside a node represent the order in which the node is expanded (tie is broken from left to right)

Example: DFS



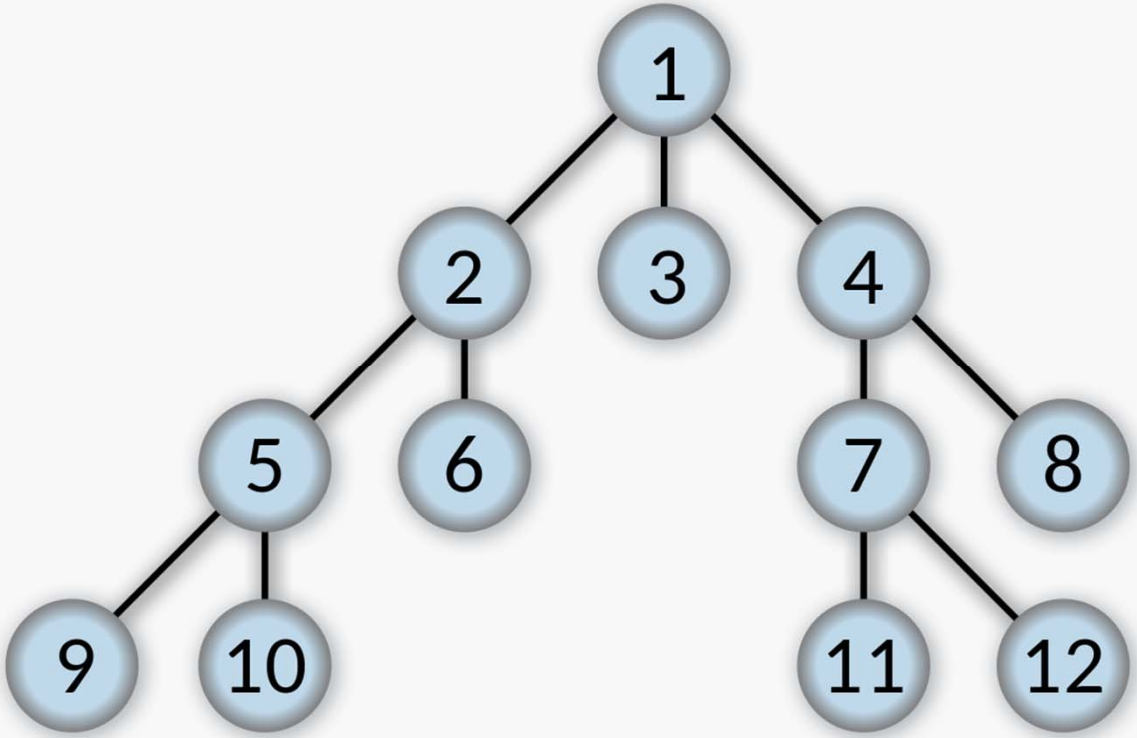


- What is the order of the nodes to be expanded by DFS in this example (tie will be broken alphabetically)?

  S -> A -> B -> C -> G

Nodes in circle mean those in the search tree, including the expanded ones (shaded) or to be expanded (i.e. in the frontier).

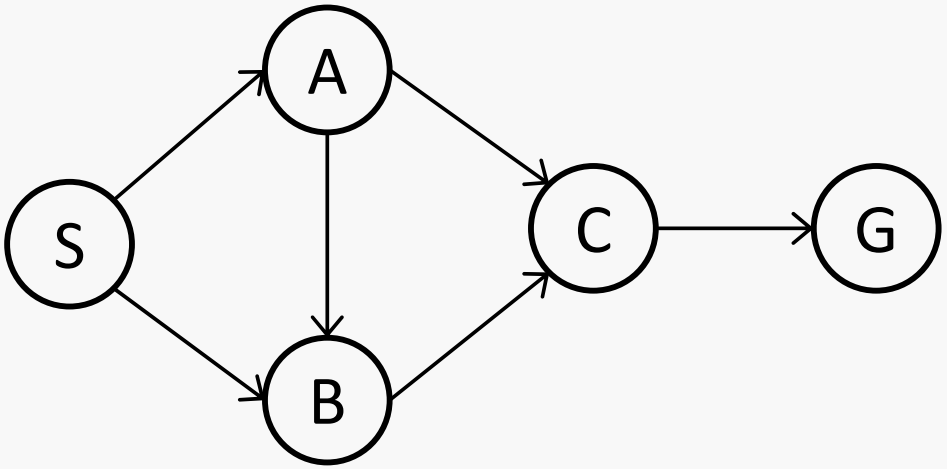
Breadth-First Search (BFS)

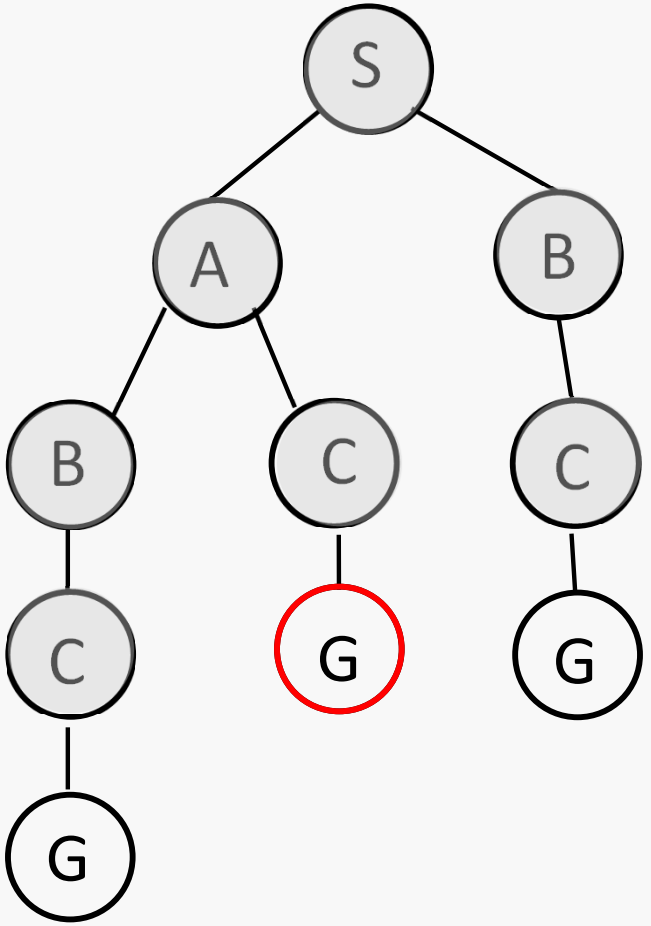


https://en.wikipedia.org/wiki/Breadth-first_search

- The number inside a node represent the order in which the node is expanded (tie is broken from left to right)

Example: BFS



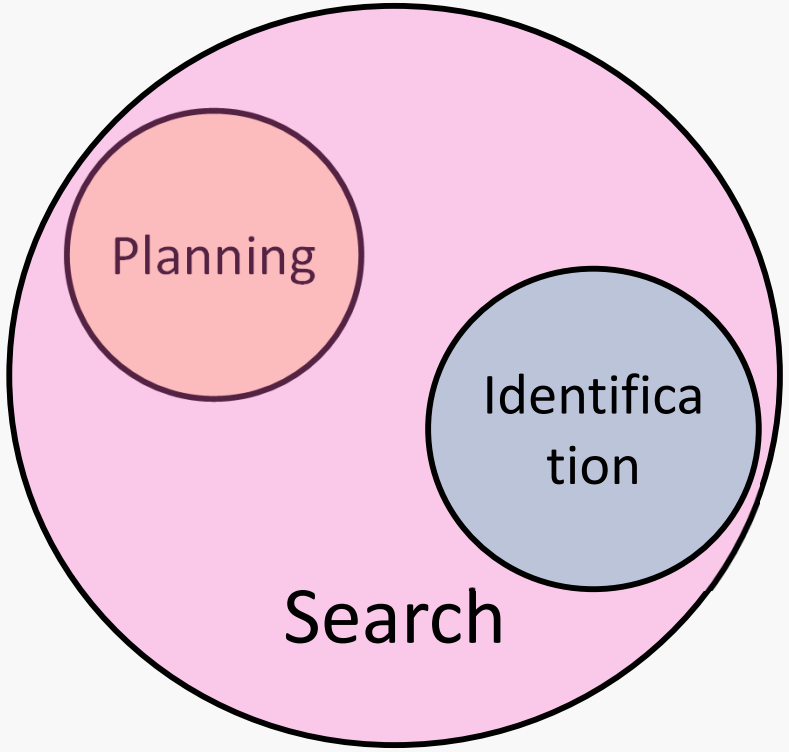


- What is the order of the nodes to be expanded by BFS in this example (tie will be broken alphabetically)?

Note that we claim success when we first choose a goal state to expand rather than when we first find a goal state.

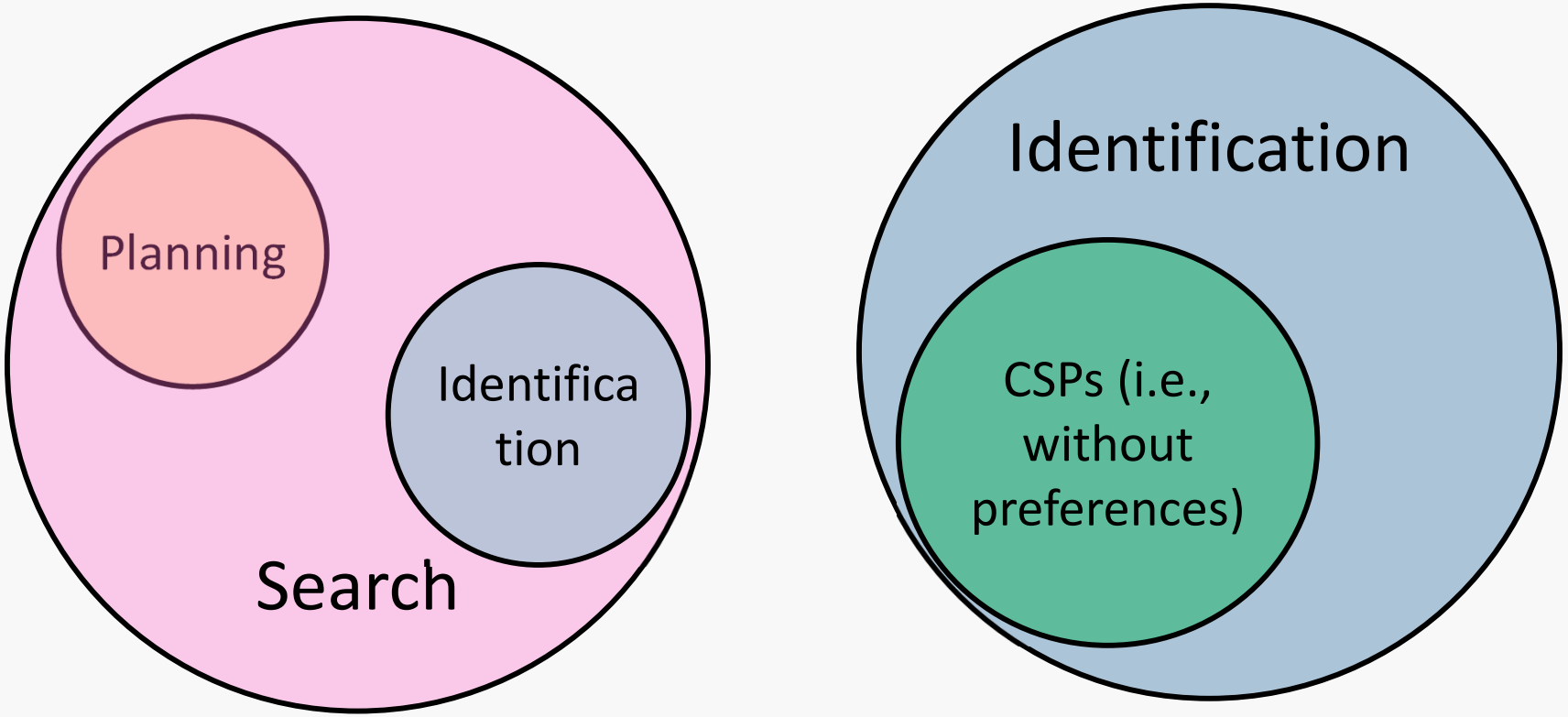
Another type of search problems: Identification

- All the above cases are about planning, which is only one type of search problems.



- Planning: a sequence of actions
  - We care about the path to the goal.
- Identification: an assignment
  - We care about the goal itself, not the path.
  - For example, a taxi firm assigns taxis a, b, c to customers x, y, z such that the cost incurred (e.g., fuel) is minimal.

Search -> Identification -> CSP



- Constraint Satisfaction Problems (CSPs): Identification problems have constraints to be satisfied; there is no preference in CSPs.
- Constraints refer to hard constraints which a legal solution cannot violate.
- Preferences sometimes are referred to as soft constraints (or objectives), where we need to optimise, e.g., to minimise cost.

CSPs

- A constraint satisfaction problem consists of
  - A set of variables
  - A domain for each variable
  - A set of constraints
- In a CSP, an assignment is complete if every variable has a value, otherwise it is partial.
- Solutions are complete assignments satisfying all the constraints.
- Example: Module scheduling problem
  - Variables – Modules: AI1, Data Structure, Software Engineering, OOP, AI2, Neural Computation…
  - Domain - year-term: {1-1, 1-2, 2-1, 2-2, 3-1, 3-2}
  - Constraints - pre-requisites: {AI1 < AI2, OOP < SE…}
  - Solutions - e.g.: (AI1=1-2, OOP=1-1, AI2=2-2, SE=2-1…)

Standard Search Problems vs CSPs

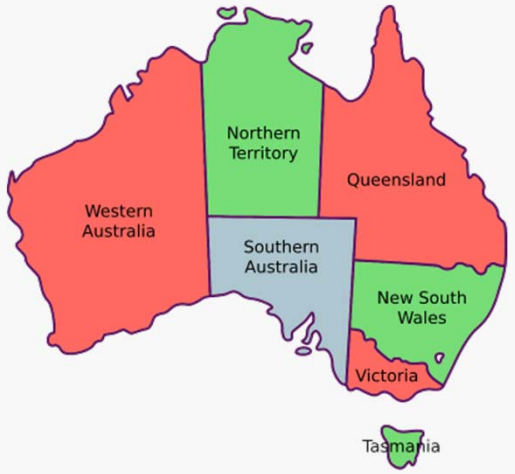
- Standard Search problems
  - State is a “black-box”: arbitrary data structure
  - Goal test can be any function over states

- Constraint Satisfaction Problems (CSPs)
  - State is defined by variables X_1, X_2,… with values from domains D_1, D_2, …
  - Goal test is a set of constraints specifying allowable combinations of values of variables.
  - An example of a formal representation language, in which many search problems can be abstracted.
  - This allows useful general-purpose algorithms with more power than standard search algorithms.

Example: Map Colouring

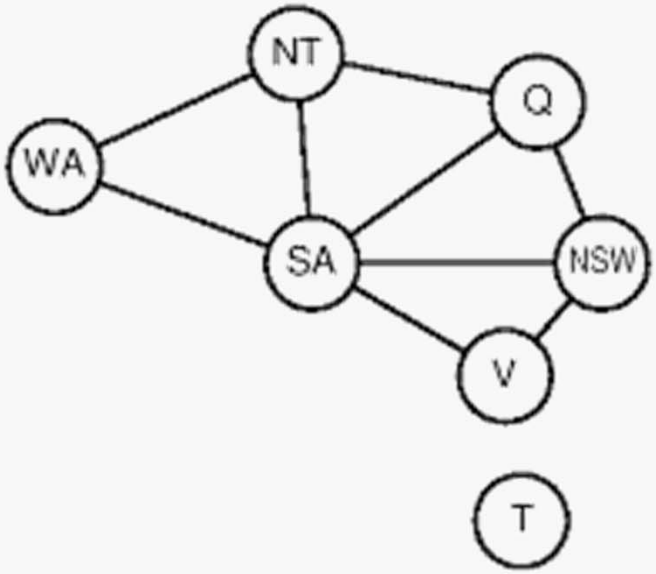
Problem: Map colouring problem is to paint a map (e.g., via three colours red, green and blue) in such a way that none of adjacent regions can have the same colour.

- Variables: WA, NT, Q, NSW, V, SA, T
- Domain: D = {red, green, blue}
- Constraints: adjacent regions must have different colours
  - WA ≠ NT, WA ≠ SA, NT ≠ SA, NT ≠ Q,…
- Solutions: e.g. {WA=red, NT=green, Q=red, NSW=green, V=red, SA=blue, T=green}



Constraint Graphs

- Constraint graphs are used to represent relations among constraints in CSPs, where nodes correspond to the variables and arcs reflect the constraints.



Example: Sudoku

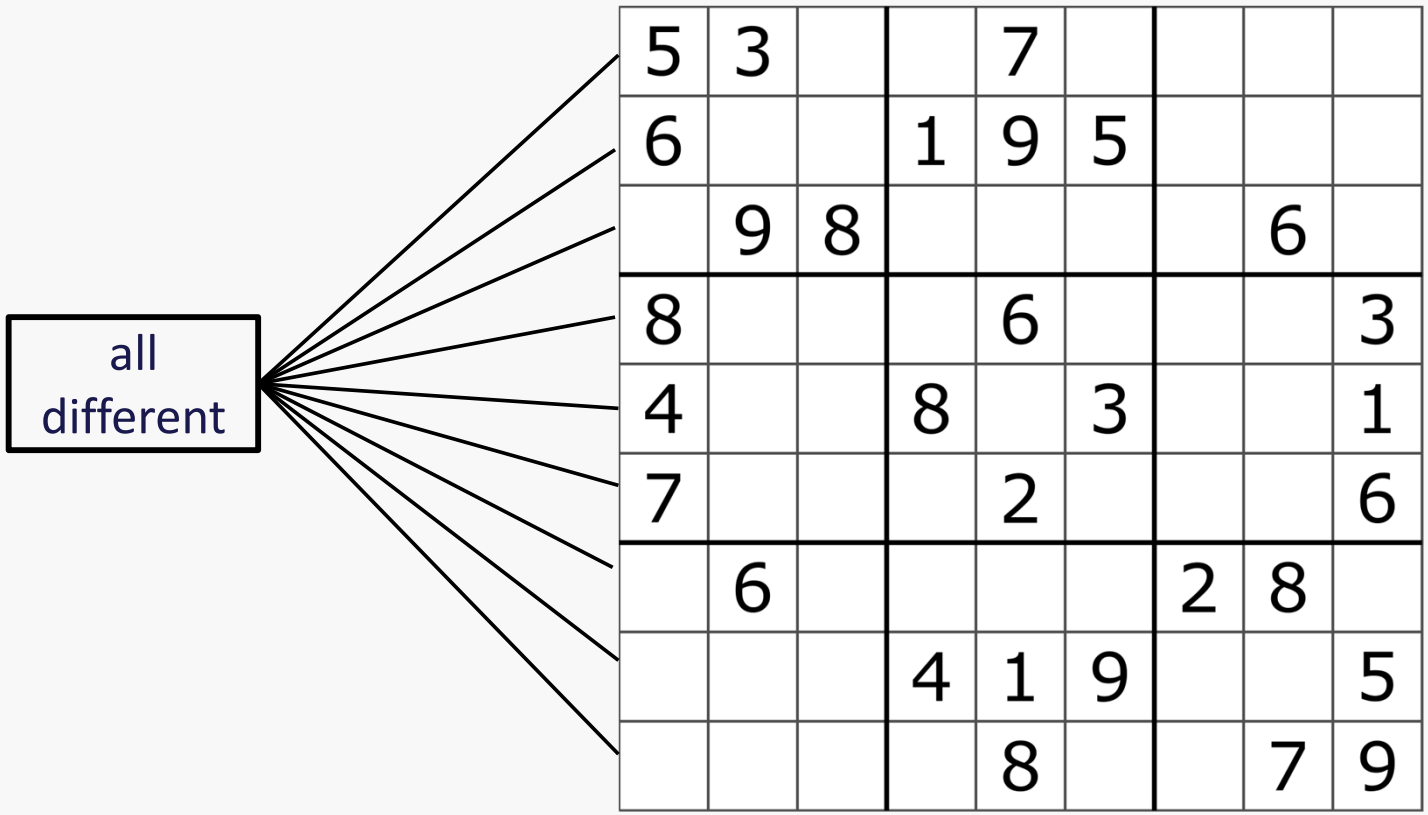
Problem: Sudoku is to fill a 9×9 grid with digits so that each column, each row, and each of the regions contain all of the digits from 1 to 9.

- Variables: each open cell
- Domain: D = {1,2,3,…,9}
- Constraints:
  - Each row contains different numbers
  - Each column contains different numbers
  - Each region contains different numbers

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

When a constraint relates to more than two variables

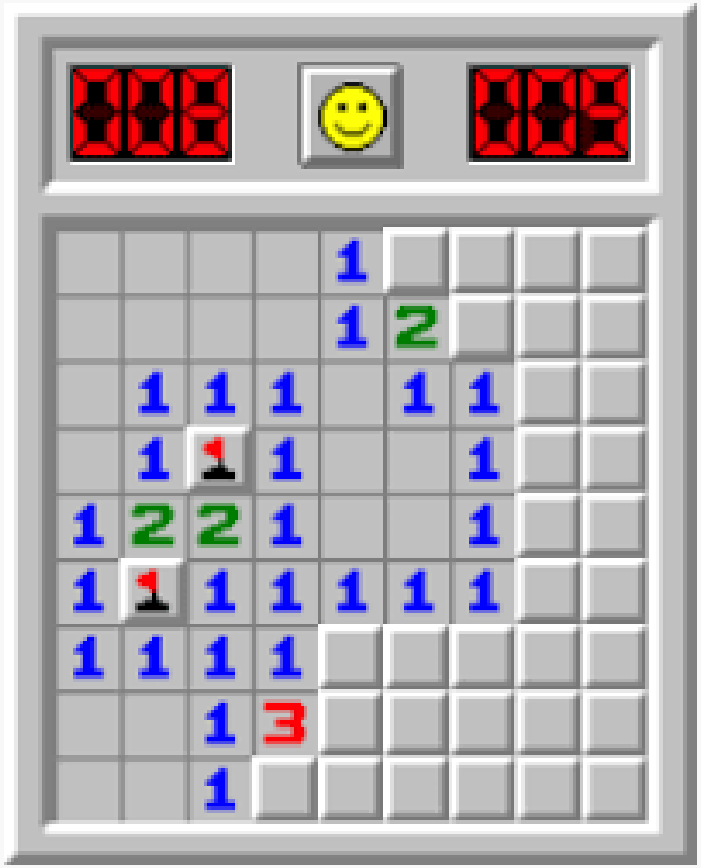
- Use a square to represent a constraint.
- The square connects all the variables involved in that constraint.



Minesweeper

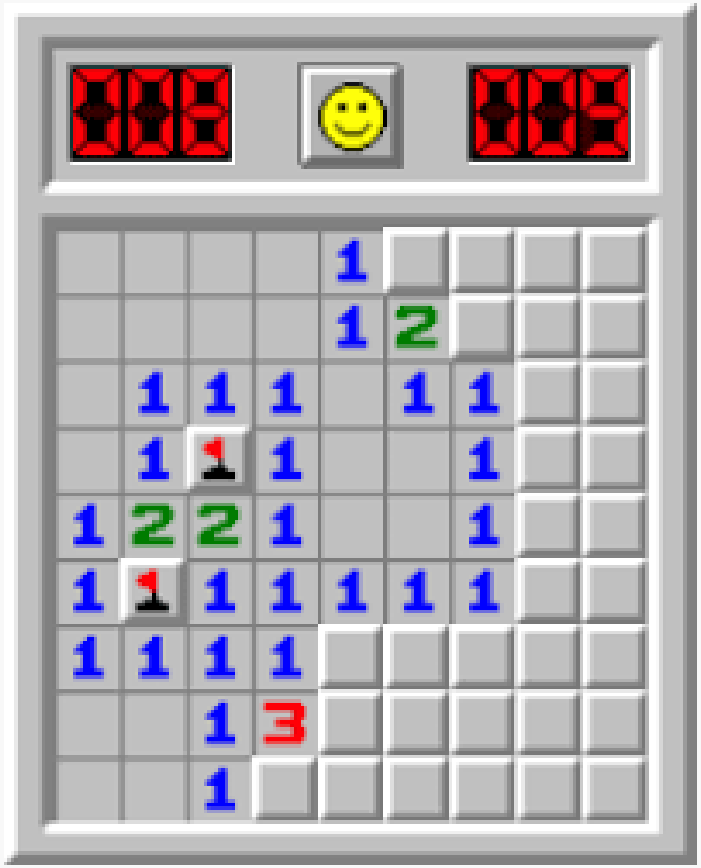
- Minesweeper is a single-player puzzle game. The goal is to not uncover a square that contains a mine; if you've identified a square that you think it is a mine, then you flag it.

- When you uncover a square, if the square is a mine, the game ends and you lost; otherwise it is a numbered square indicating how many mines are around it (between 0 and 8). If you uncover all of the squares except for any mines, you win the game.



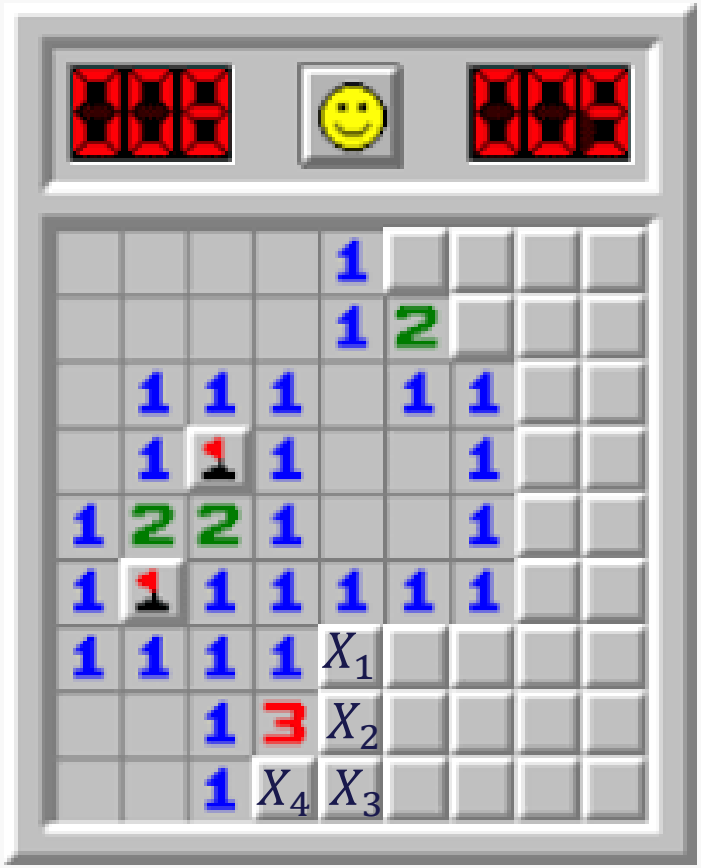
Formalise Minesweeper as a CSP

- Variables:
  - All squares to be uncovered X_1, X_2,…
- Domain:
  - $D = \{0, 1\}$, where 0 denotes not a mine and 1 denotes a mine
- Constraint description:
  - The number on a square is the sum of its neighbour's values.



Minesweeper - example

- Variables:
  - X_1, X_2, X_3, X_4
- Domain:
  - $D = \{0, 1\}$, where 0 denotes not a mine and 1 denotes a mine.
- Constraints:
  - $X_1 = 1$
  - $X_1 + X_2 = 1$
  - $X_1 + X_2 + X_3 + X_4 = 3$
  - $X_4 = 1$
  - ...



N-Queens

Problem: N-queens puzzle is the problem of placing N chess queens on an N×N chessboard so that no two queens threaten each other.

- Variables: X_{ij}, where *i* is the *i*th row and *j* is the *j*th column
- Domain: D = {0, 1}, where 1 means having a queen
- Constraints:
  - One queen each row:

$$\forall\, i, j, k \in \{1,2,\dots,N\}, j \neq k\colon \left(X_{ij}, X_{ik}\right) \in \{(0,0),(0,1),(1,0)\}$$

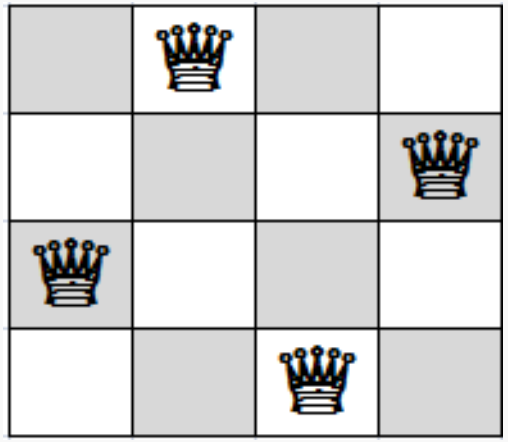
  - One queen each column:

$$\forall\, i, j, k \in \{1,2,\dots,N\}, j \neq k\colon \left(X_{ji}, X_{ki}\right) \in \{(0,0),(0,1),(1,0)\}$$

  - One queen each diagonal:

$$\forall\, i, j, k \in \{1,2,\dots,N\}, i + k \leq N, j + k \leq N\colon$$
$$\left(X_{ij}, X_{i+k,j+k}\right) \in \{(0,0),(0,1),(1,0)\}$$
$$\forall\, i, j, k \in \{1,2,\dots,N\}, i + k \leq N, j - k \geq 1\colon$$
$$\left(X_{ij}, X_{i+k,j-k}\right) \in \{(0,0),(0,1),(1,0)\}$$

  - Must have *N* queens in total: $\sum_{i,j\in\{1,2,\dots,N\}} X_{ij} = N$



Variety of CSPs

- Variables
  - Finite domains (discrete), e.g. all the preceding examples.
  - Infinite domains (discrete or continuous), e.g., variables involving time.
- Constraints
  - Unary, binary and high-order constraints.
- CSPs are difficult search problems
  - If a CSP has n variables, the size of each domain is d, then there are $O(d^n)$ complete assignments.
  - For the preceding representation of the 4 × 4 queens puzzle, there are 2^{16} complete assignments.

Real-world CSPs

- Assignment problems, e.g. who teaches which class
- Timetabling problems, e.g. which class is offered when and where
- Hardware configuration
- Transportation scheduling
- Factory scheduling
- Circuit layout
- Fault diagnosis
- …

Many of CSP problems can also consider the preferences (i.e., objectives), in which case they turn into constrained optimisation problems.

Homework: Cryptarithmetic

Cryptarithmetic is a puzzle where the digits of numbers are represented by letters. Each letter represents a unique digit. The goal is to find the digits such that a given equation is verified.

Question: Formalise the cryptarithmetic problem as a CSP, i.e., formally give its variables, domain and constraints.

```
    T W O          8 3 6
+   T W O      +   8 3 6
-----------    -----------
  F O U R        1 6 7 2
```