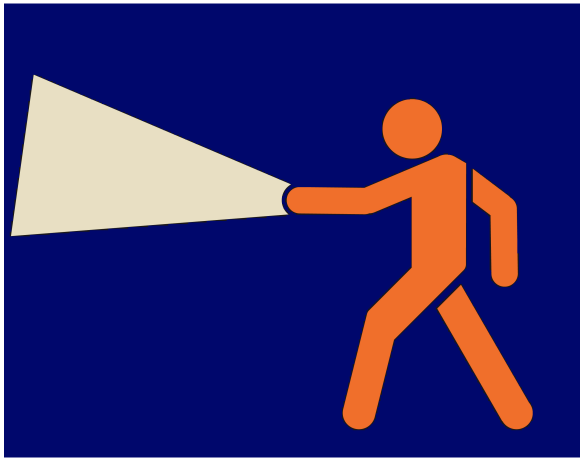
Search Agents

aiphabet

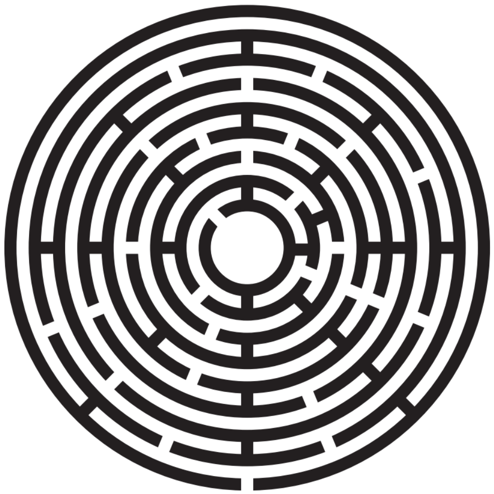
Your AI Journey Starts Here

Search Agents

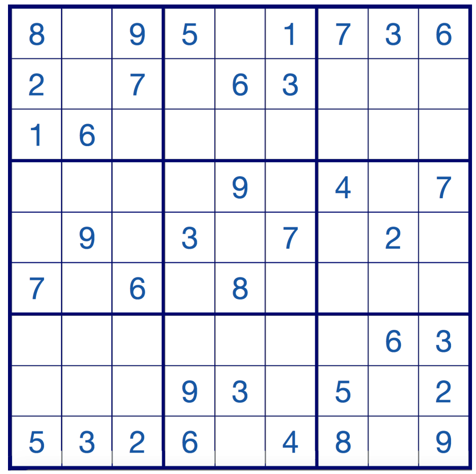
- Agents that work towards a **goal**
- Agents consider the impact of **actions** on future **states**
- Agent's job is to identify the action or series of actions that lead to the goal
- Formalized as a **search** through possible **solutions**



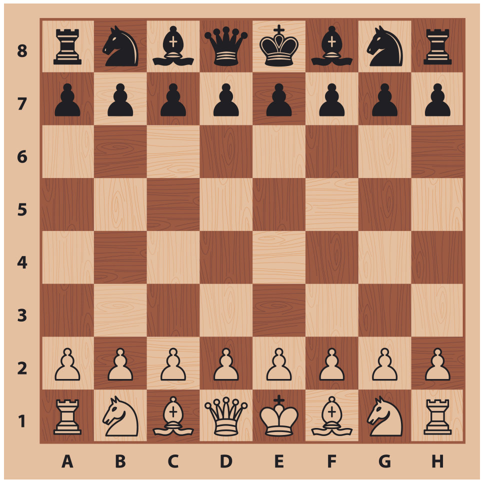
Three types of search...



Simple Search



Search with Constraints

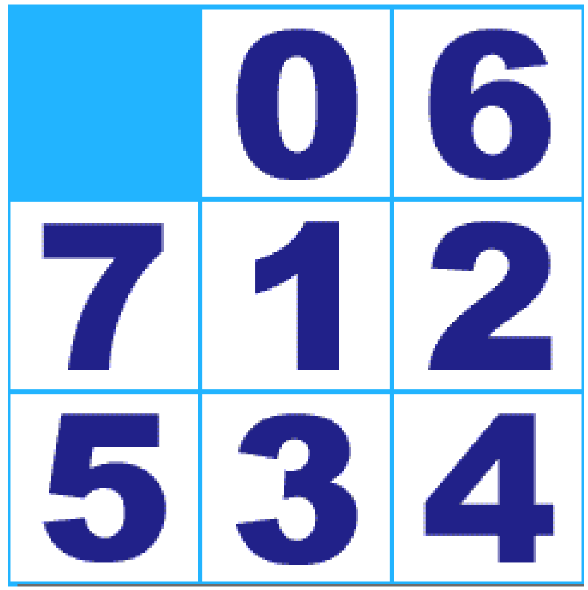


Adversarial Search

Three types of search...

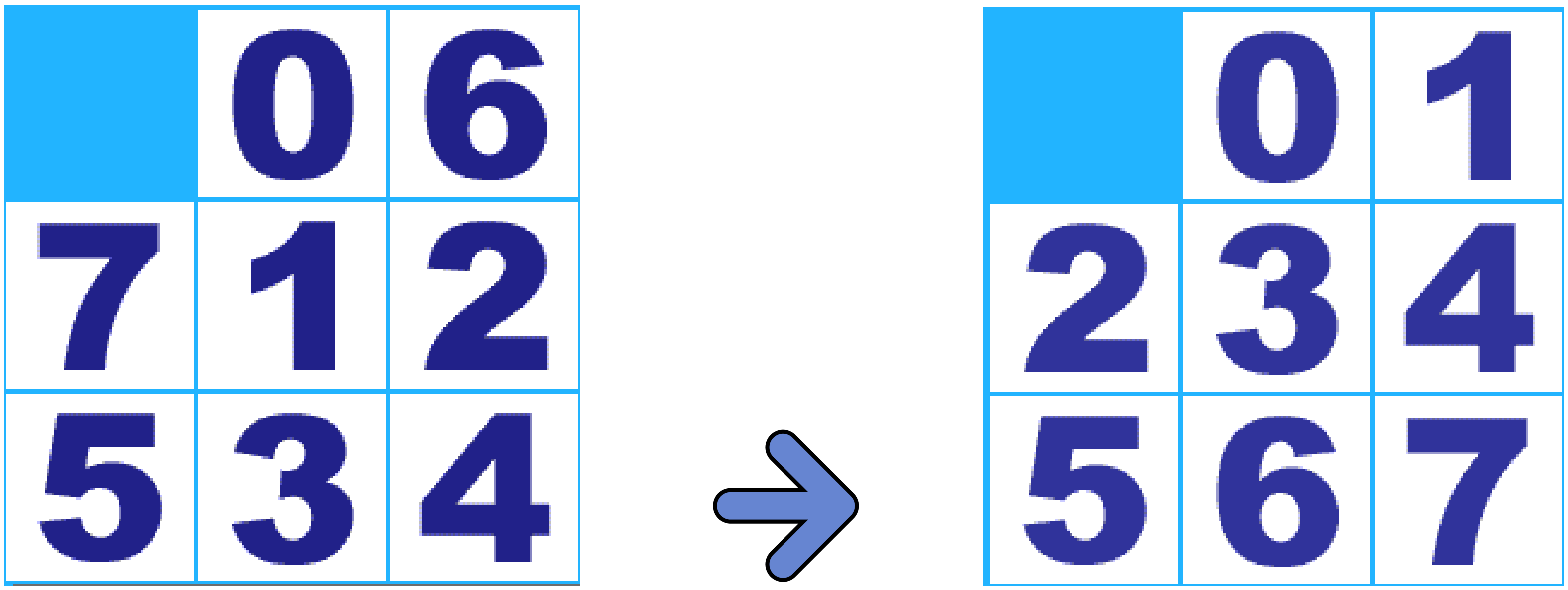
Simple Search

Examples



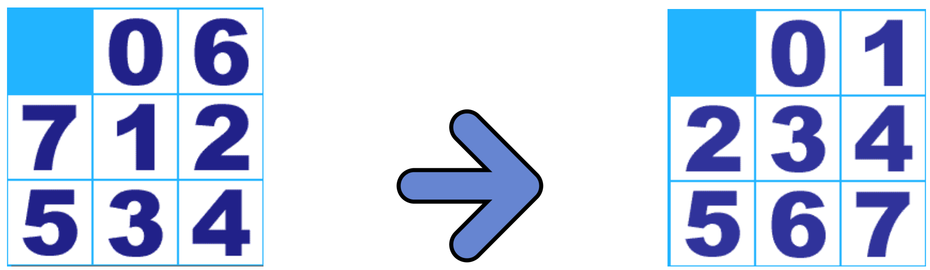
Eight puzzle

Examples



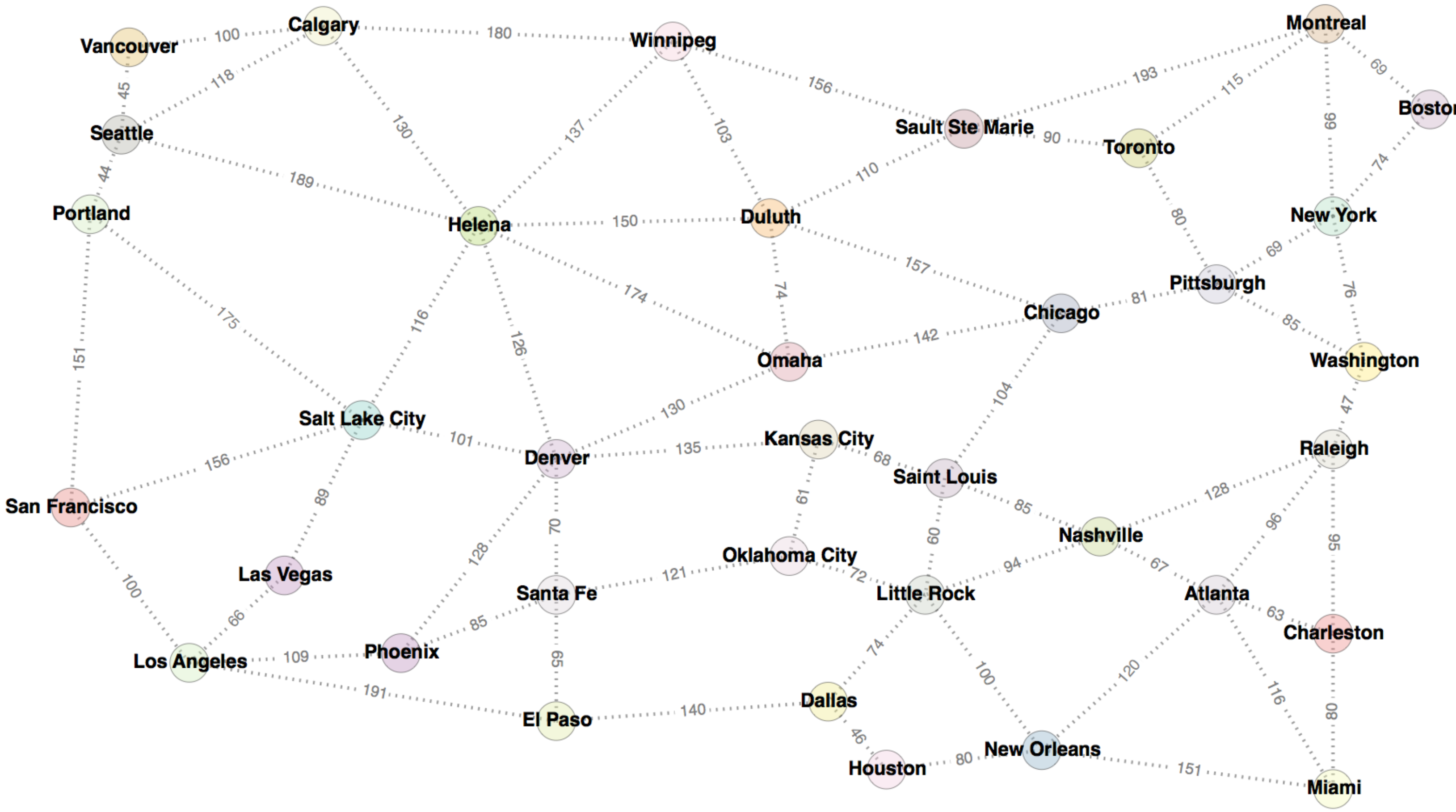
Start state Goal state

Examples



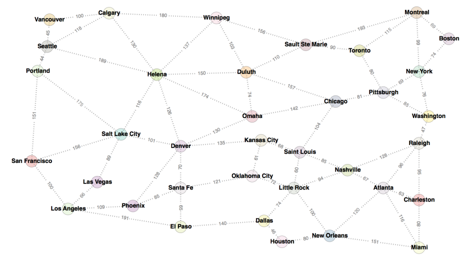
- **States:** location of each of the 8 tiles in the 3x3 grid
- **Initial state:** any state
- **Actions:** move left, right, up or down
- **Transition model:** given a state and an action, returns resulting state
- **Goal test:** state matches the goal state?
- **Path cost:** total moves, each move costs one

Examples



Vancouver
Calgary
Winnipeg
Montreal
Boston
Seattle
Sault Ste Marie
Toronto
Portland
Helena
Duluth
New York
Pittsburgh
Chicago
Omaha
Washington
Salt Lake City
Denver
Kansas City
Raleigh
Saint Louis
San Francisco
Nashville
Oklahoma City
Las Vegas
Santa Fe
Little Rock
Atlanta
Charleston
Los Angeles
Phoenix
Dallas
El Paso
New Orleans
Houston
Miami
100
180
156
193
115
69
45
118
130
137
103
90
66
74
44
189
150
110
80
69
151
175
116
126
174
74
157
81
142
76
85
47
104
130
101
135
68
156
89
128
70
61
60
85
128
96
95
121
72
94
67
63
100
66
109
85
65
74
100
120
116
80
191
140
46
80
151

Examples



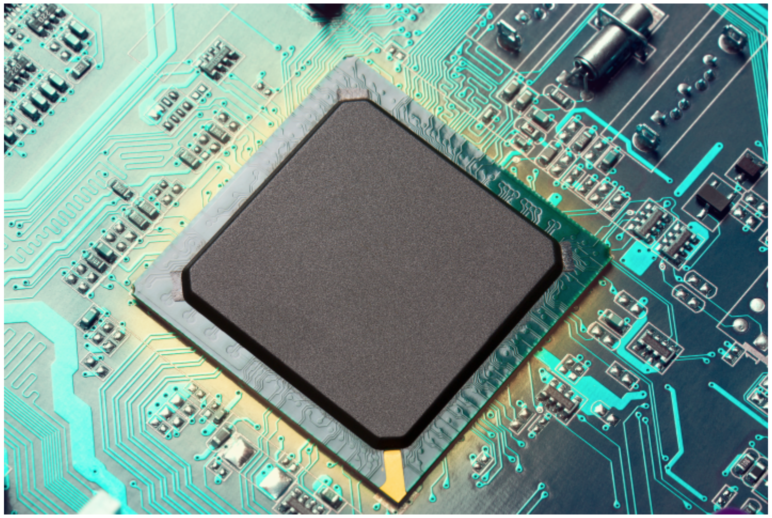
- **States:** In(city) where city $\in$ {Los Angeles, San Francisco, Denver,...}
- **Initial state:** In(Boston)
- **Actions:** Go(New York), etc.
- **Transition model:**
  Results (In(Boston), Go(New York)) = In(New York)
- **Goal test:** In(Denver)
- **Path cost:** path length in kilometers

Real-World Examples

- **Route finding problem:** typically our example of map search, where we need to go from location to location using links or transitions; examples of applications include tools for driving directions in websites, in-car systems, etc.

Real-World Examples

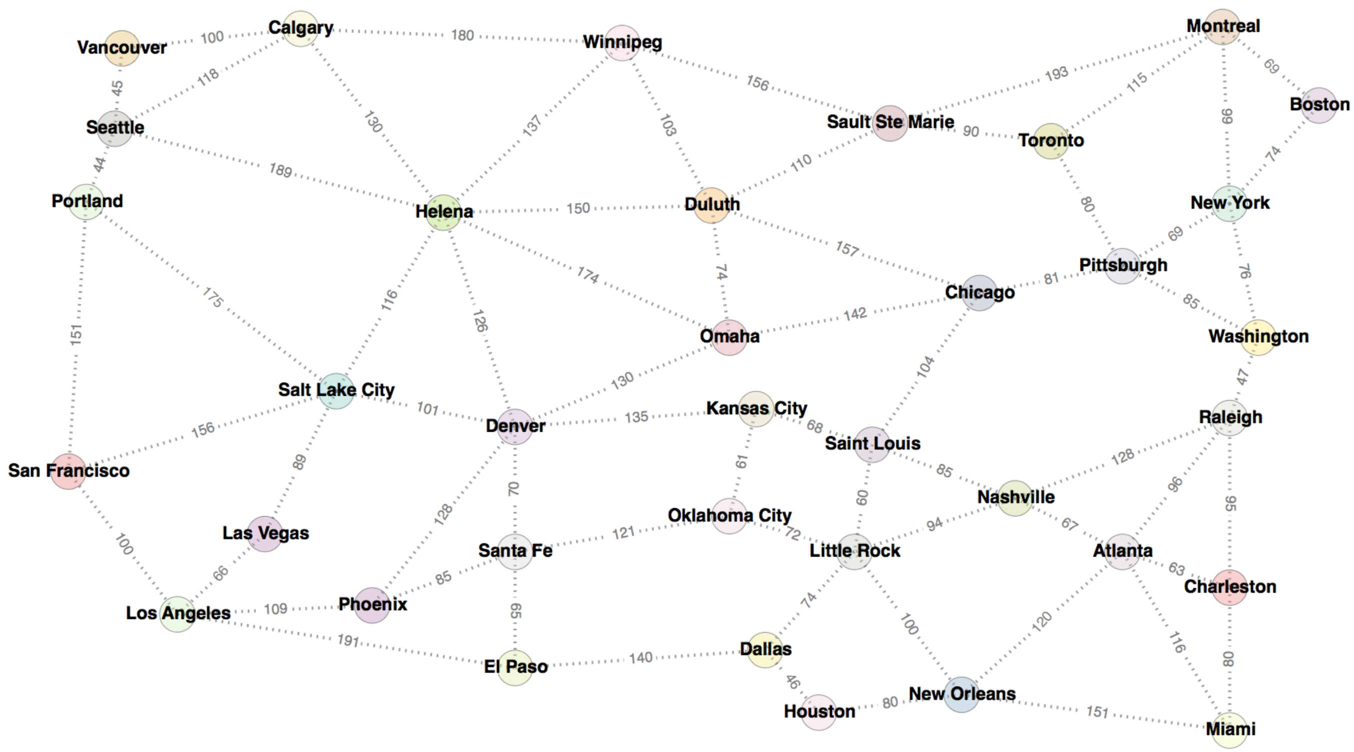
- **VLSI layout:** position millions of components and connections on a chip to minimize area, shorten delays; aim: put circuit components on a chip, so they don't overlap and leave space to wiring, which is a complex problem



Real-World Examples

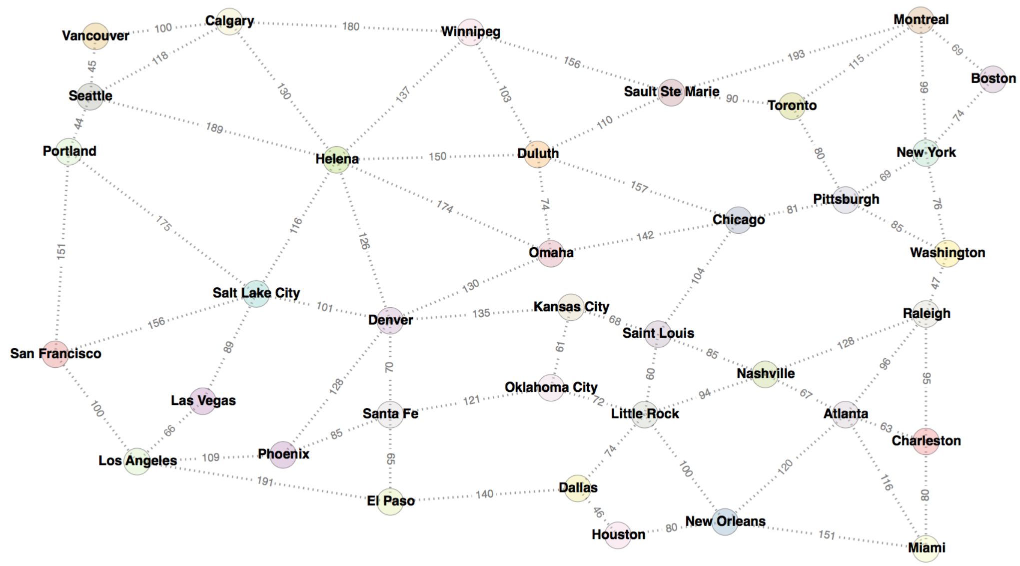
- **Robot navigation:** special case of route finding for robots with no specific routes or connections; the robot navigates in 2D or 3D space where the state space and action space are potentially infinite.

Examples of Search Agents

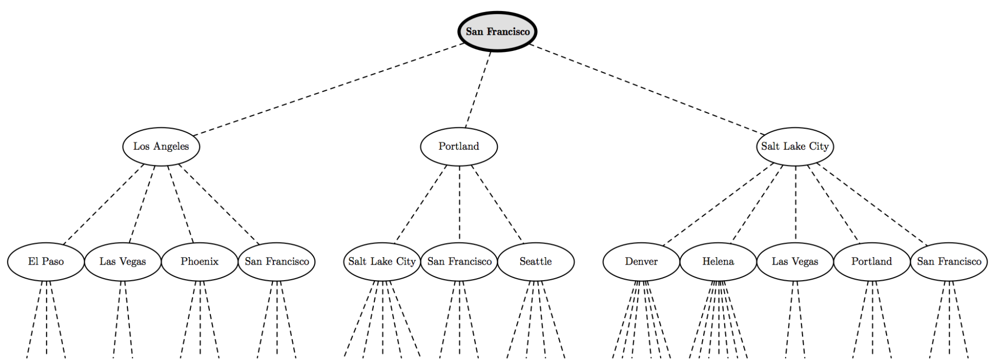


Let's show the first steps in growing the search tree to find a route from San Francisco to another city.

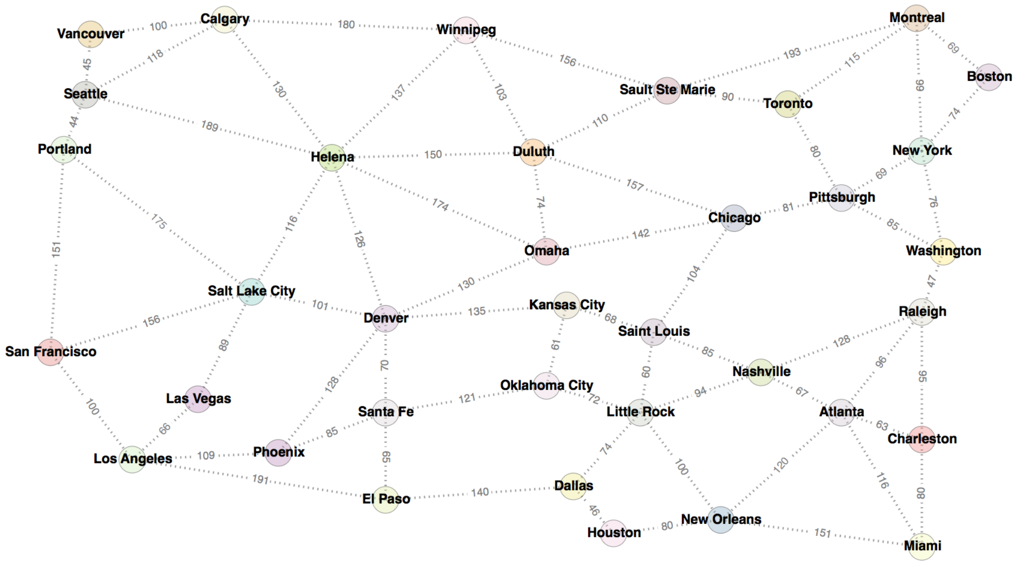
Examples of Search Agents



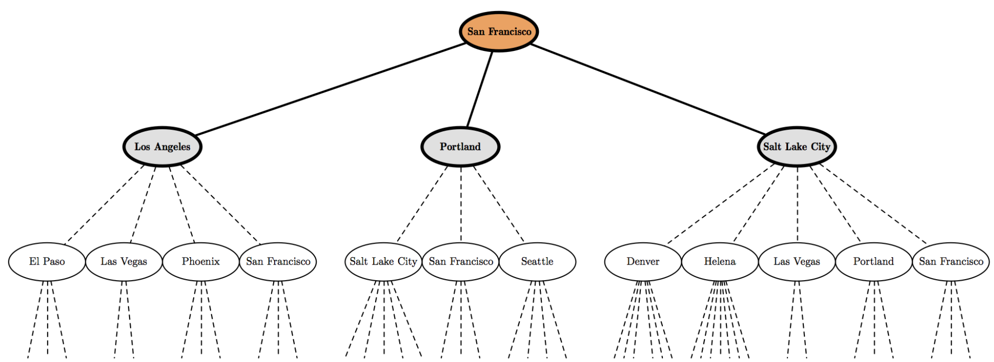

```
function TREE-SEARCH(initialState, goalTest)
      returns SUCCESS or FAILURE :

      initialize frontier with initialState

      while not frontier.isEmpty():
            state = frontier.remove()

            if goalTest(state):
                  return SUCCESS(state)

            for neighbor in state.neighbors():
                  frontier.add(neighbor)

      return FAILURE
```



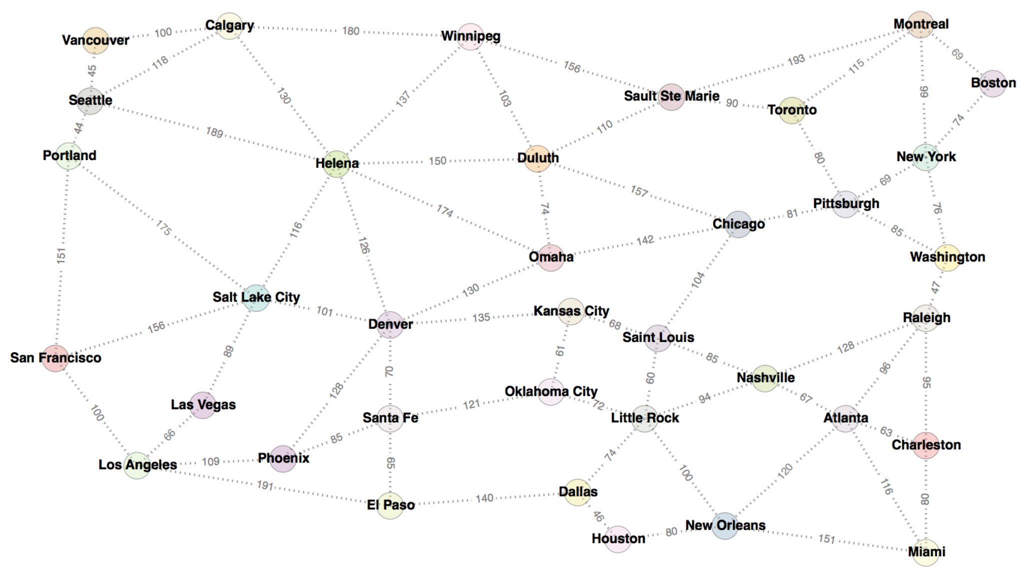
Examples of Search Agents



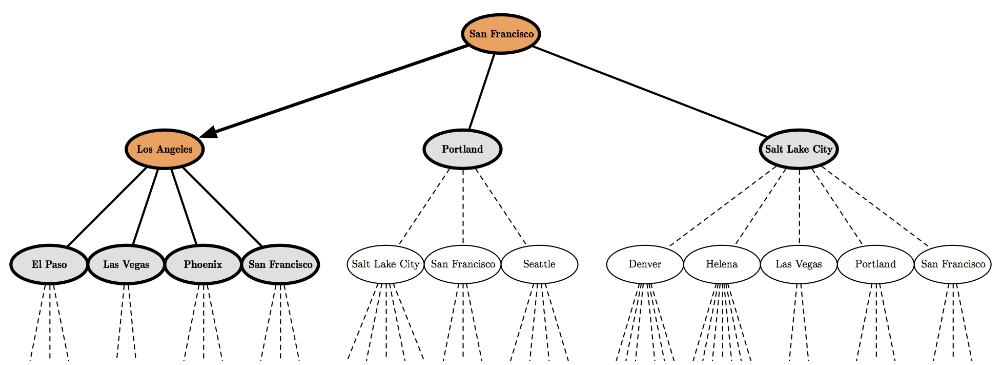

```
function TREE-SEARCH(initialState, goalTest)
    returns SUCCESS or FAILURE :

    initialize frontier with initialState

    while not frontier.isEmpty():
        state = frontier.remove()

        if goalTest(state):
            return SUCCESS(state)

        for neighbor in state.neighbors():
            frontier.add(neighbor)

    return FAILURE
```



Examples of Search Agents




```
function TREE-SEARCH(initialState, goalTest)
    returns SUCCESS or FAILURE :

    initialize frontier with initialState

    while not frontier.isEmpty():
        state = frontier.remove()

        if goalTest(state):
            return SUCCESS(state)

        for neighbor in state.neighbors():
            frontier.add(neighbor)

    return FAILURE
```



There are different strategies!

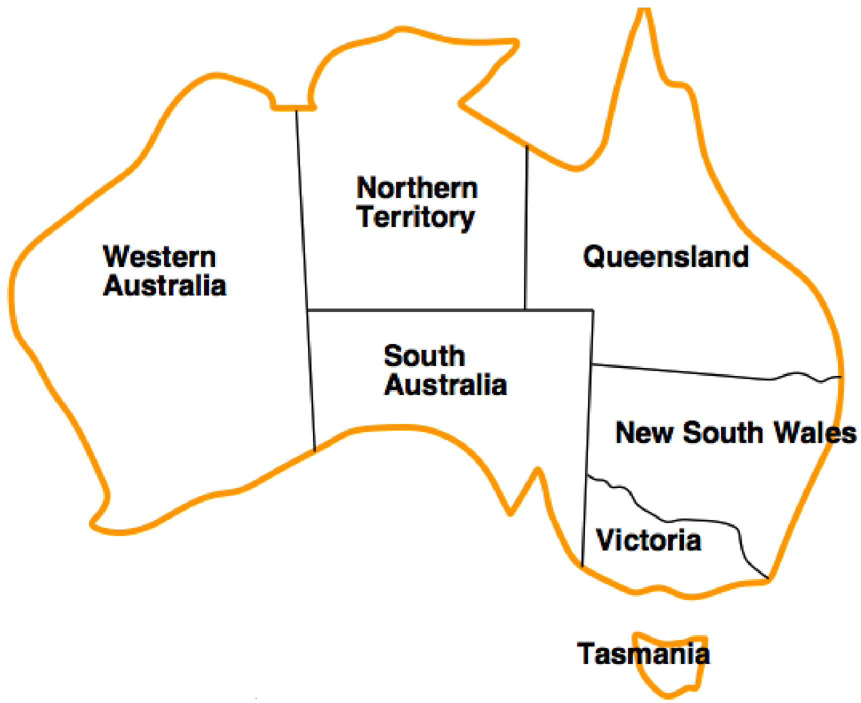
Three types of search...

Search with Constraints

CSPs

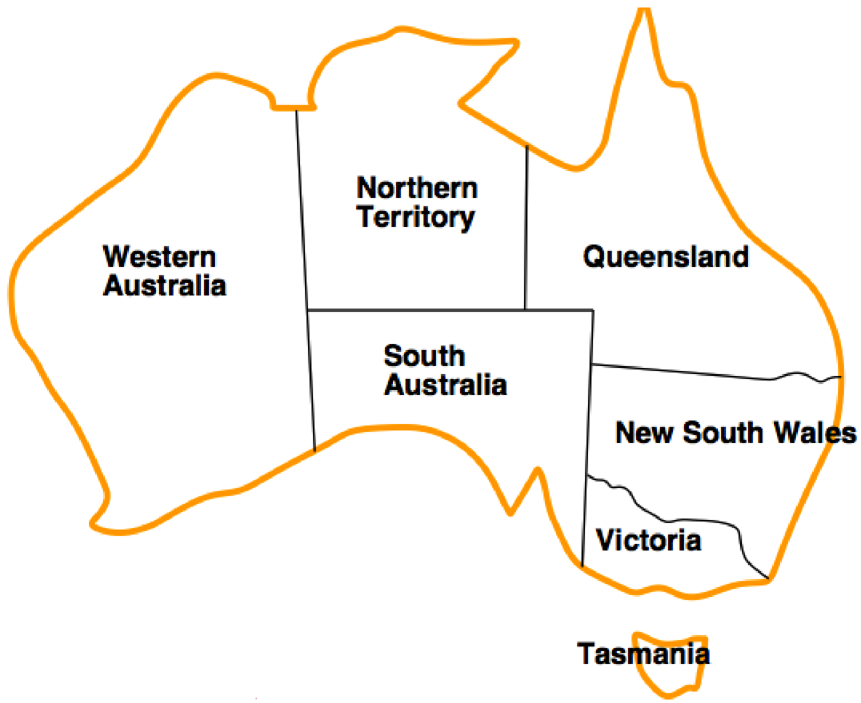
- **Search problems**
  - Find the **sequence of actions** that leads to the goal.
  - Sequence of actions means a **path** in the search space.
  - Paths come with different costs.
- **Constraint satisfaction problems**
  - A search problem too!
  - We need to incorporate constraints.
  - We care about the **goal itself**.

Example: Map Coloring



Variables: $X = \{WA, NT, Q, NSW, V, SA, T\}$

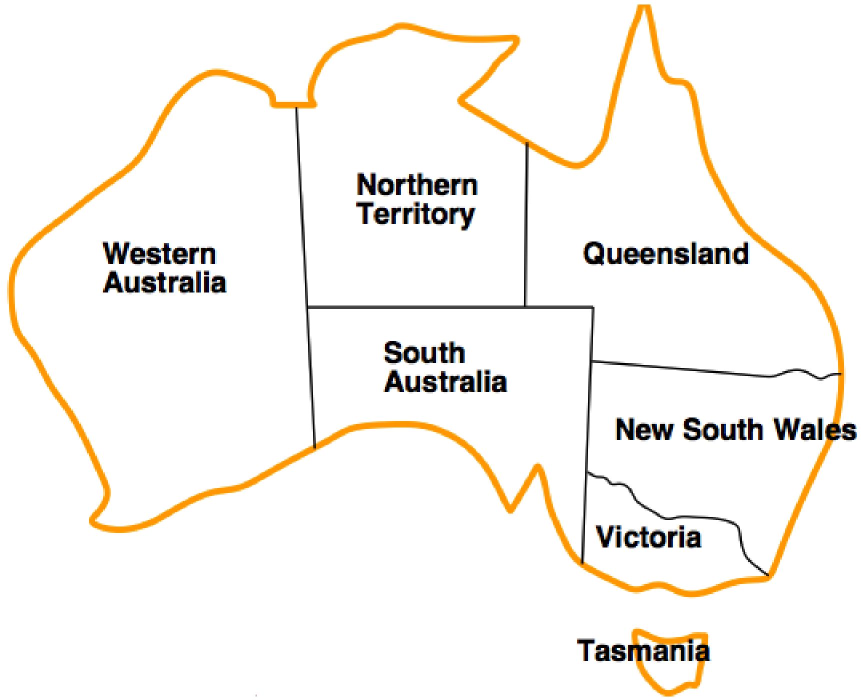
Example: Map Coloring



Variables: $X = \{WA, NT, Q, NSW, V, SA, T\}$

Domains: $D_i = \{\text{red}, \text{green}, \text{blue}\}$

Example: Map Coloring

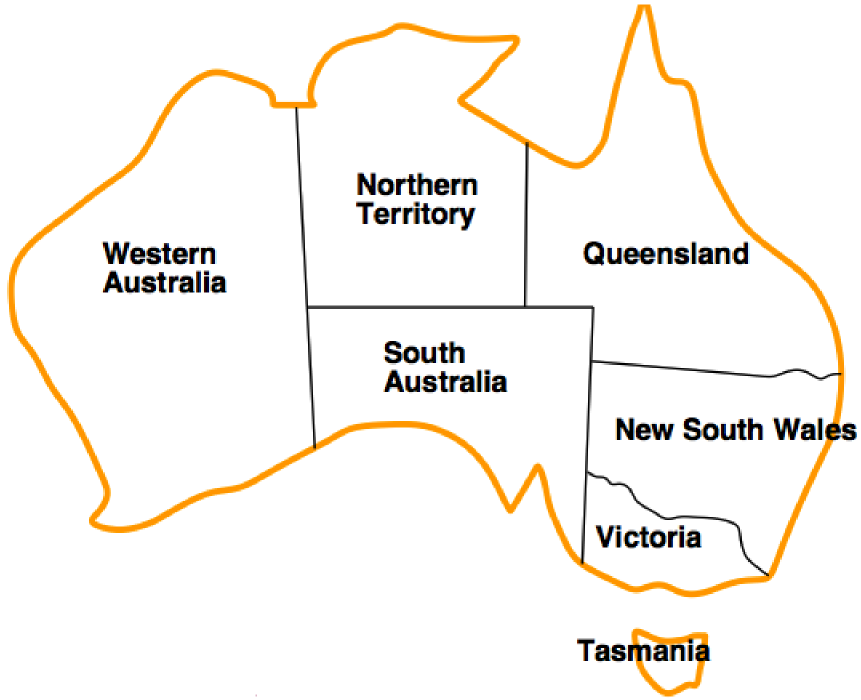


Variables: $X = \{WA, NT, Q, NSW, V, SA, T\}$

Domains: $D_i = \{\text{red}, \text{green}, \text{blue}\}$

Constraints: adjacent regions must have different colors;

Example: Map Coloring



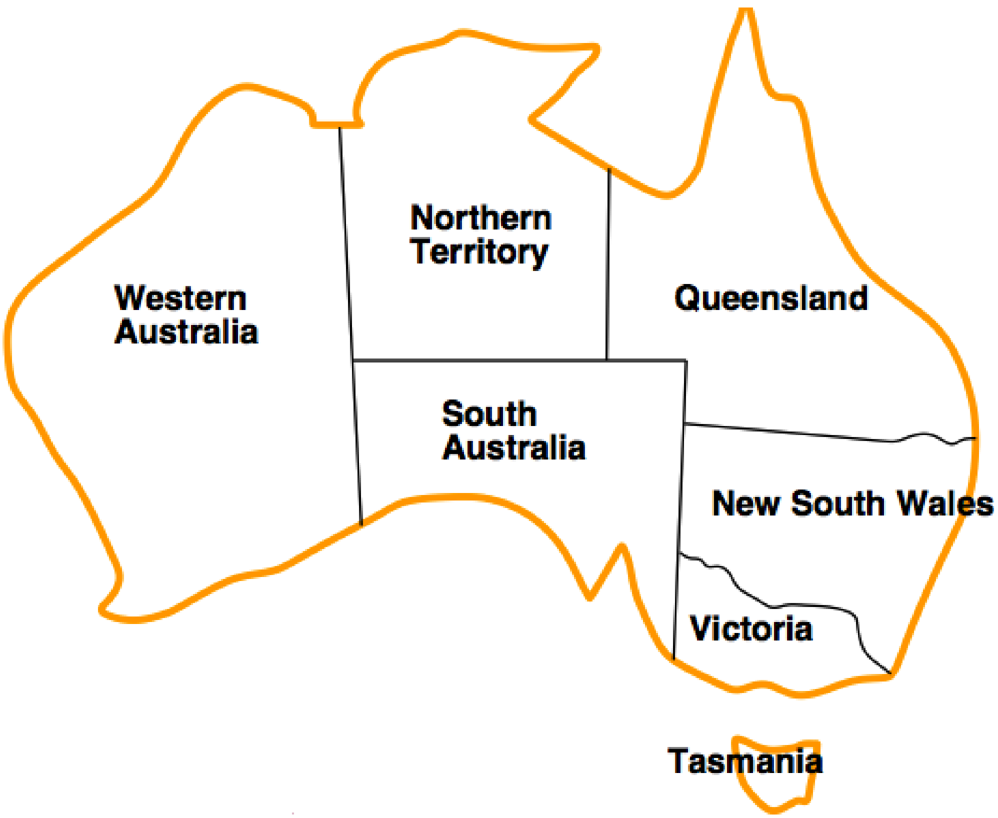
Variables: $X = \{WA, NT, Q, NSW, V, SA, T\}$

Domains: $D_i = \{\text{red}, \text{green}, \text{blue}\}$

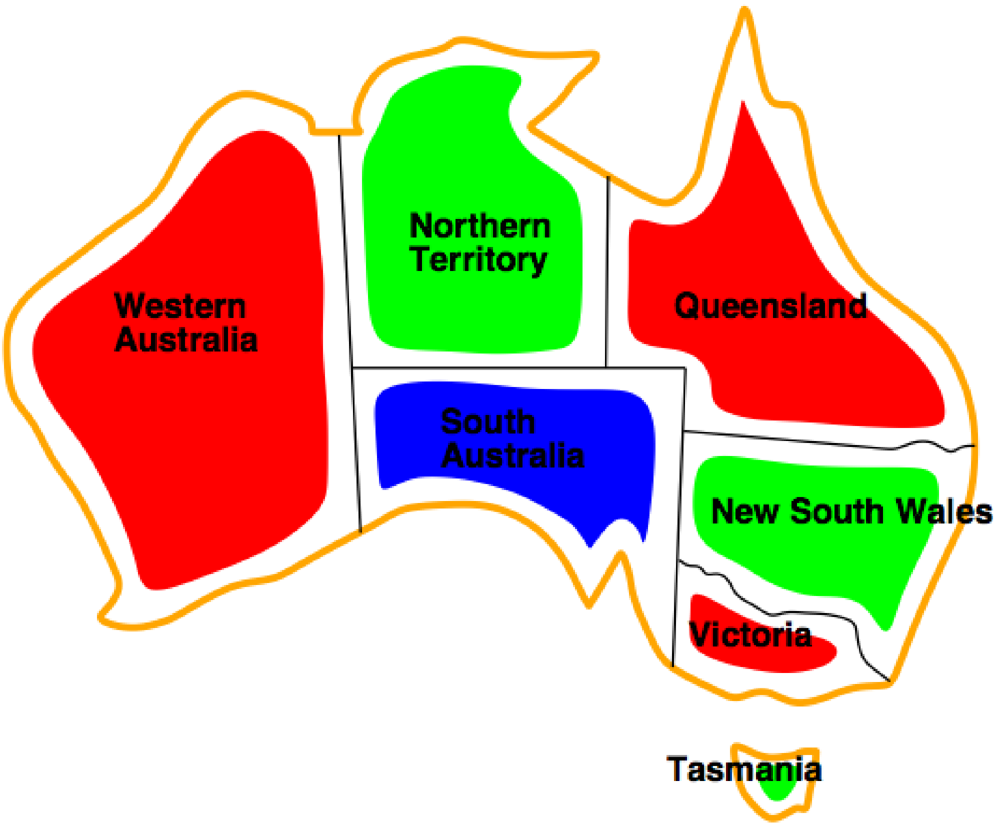
Constraints: adjacent regions must have different colors;

e.g., $WA \neq NT$ or $(WA, NT) \in \{(\text{red}, \text{green}), (\text{red}, \text{blue})\}$, etc.

Example: Map Coloring



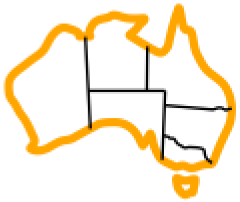
Example: Map Coloring



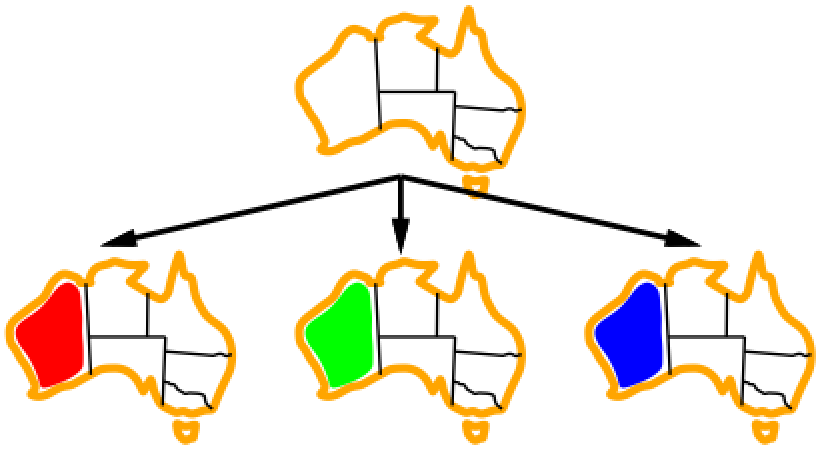
Example:

$\{WA = red, NT = green, Q = red, NSW = green, V = red, SA = blue, T = green\}$

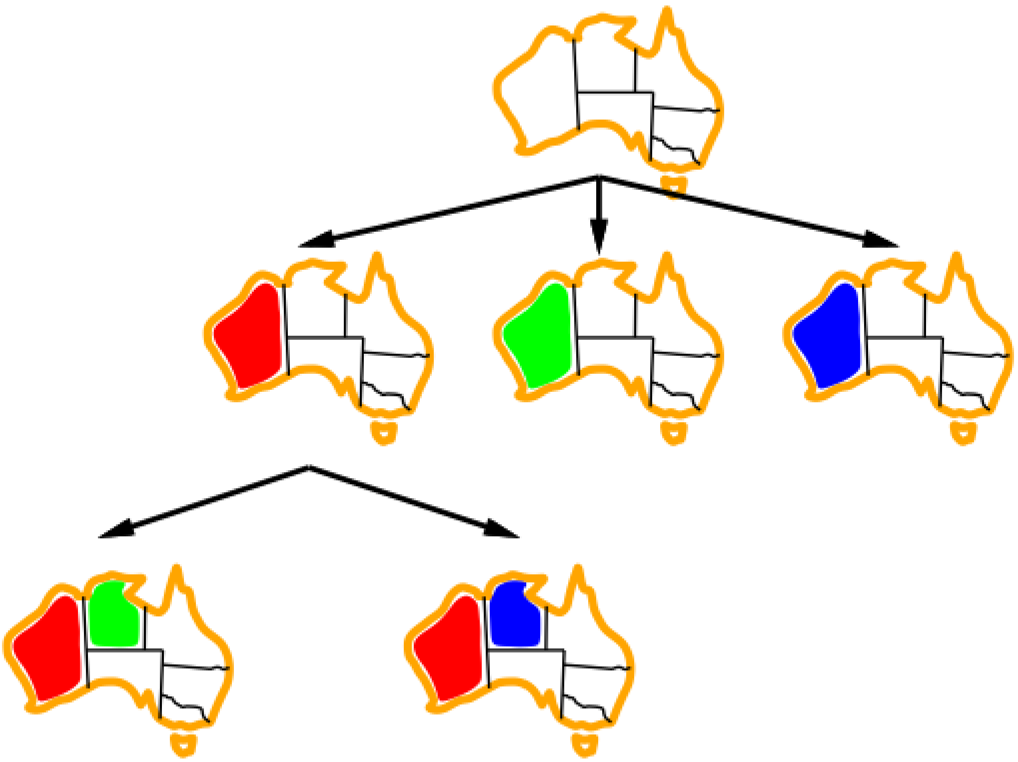
Backtracking Search



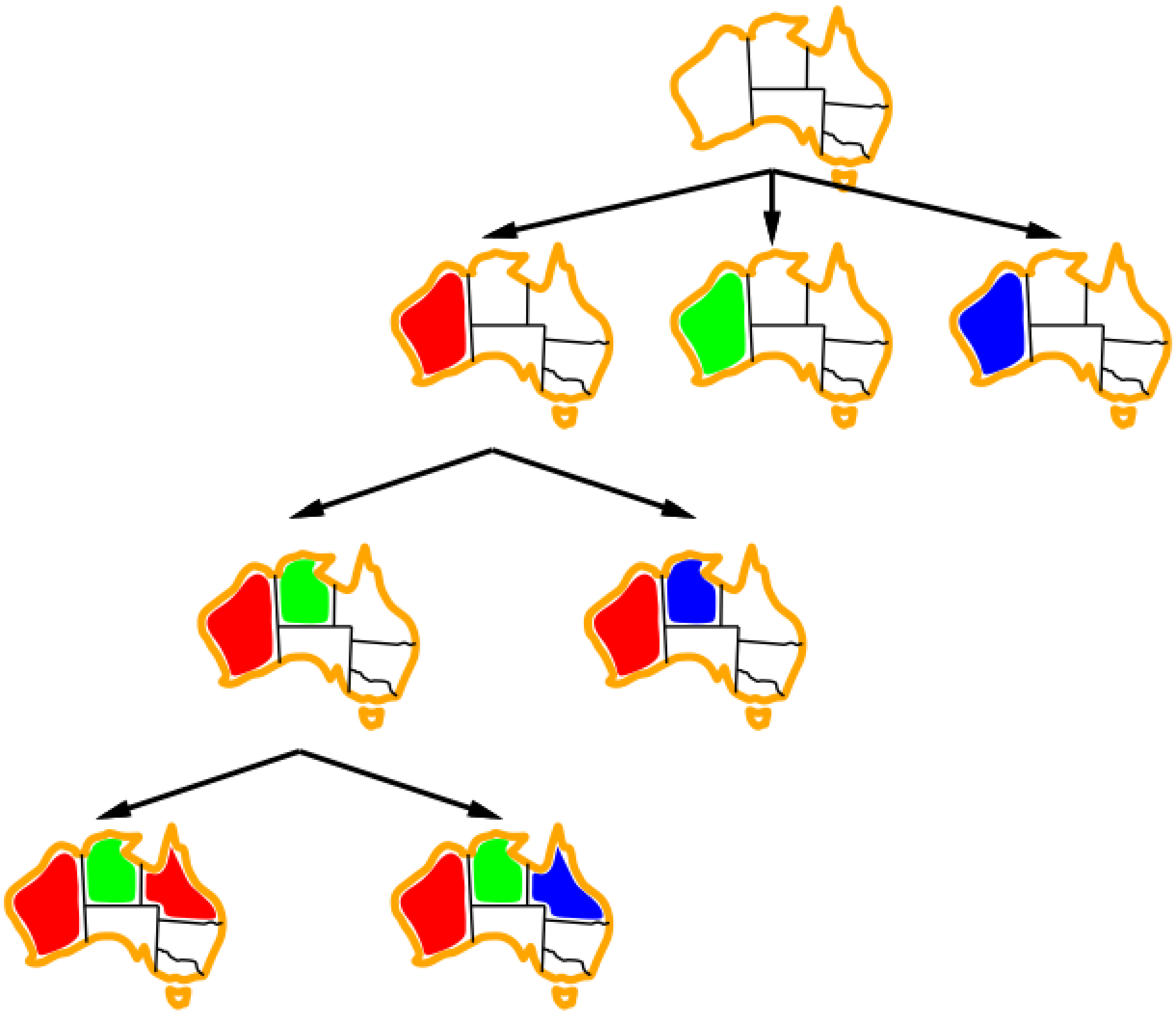
Backtracking Search



Backtracking Search



Backtracking Search



Three types of search...

Adversarial Search

Adversarial Search

- Adversarial search problems known as games
- They occur in multiagent competitive environments
- There is an **opponent** we can't control planning against us!
- Game vs. search: optimal solution is not a sequence of actions but a **strategy** (policy); if opponent does a, agent does b, else if opponent does c, agent does d, etc.
- Tedious and fragile if hard-coded (i.e., implemented with rules)
- Good news: games are modeled as **search problems** and use **heuristic evaluation** functions

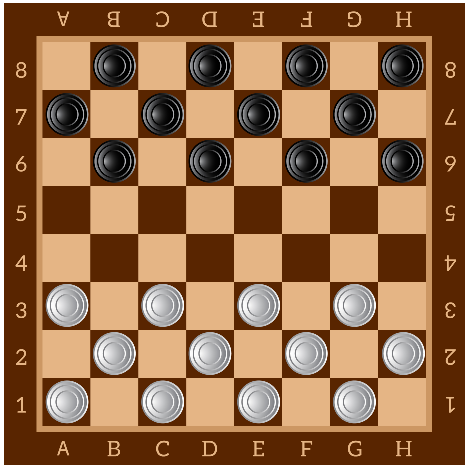
Games: Hard Topic

- Games are a big deal in AI
- Games are interesting to AI because they are too hard to solve
- Chess has a branching factor of 35, with 35^{100} nodes $\approx 10^{154}$
- Need to make some decision, even when the optimal decision is infeasible

Adversarial Search

Checkers

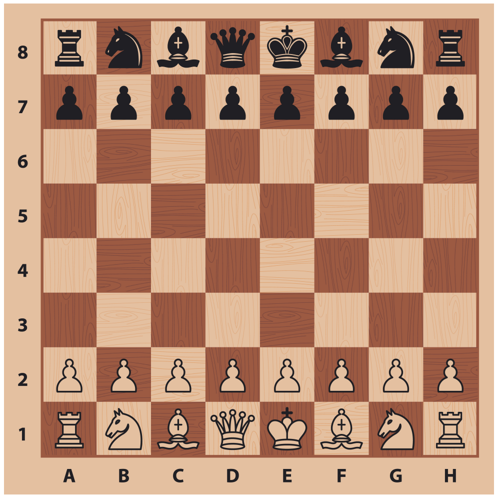
- Chinook ended 40-year reign of human world champion Marion Tinsley in 1994.
- Chinook used an endgame database defining perfect play for all positions involving eight or fewer pieces on the board, a total of 443,748,401,247 positions.



Adversarial Search

Chess

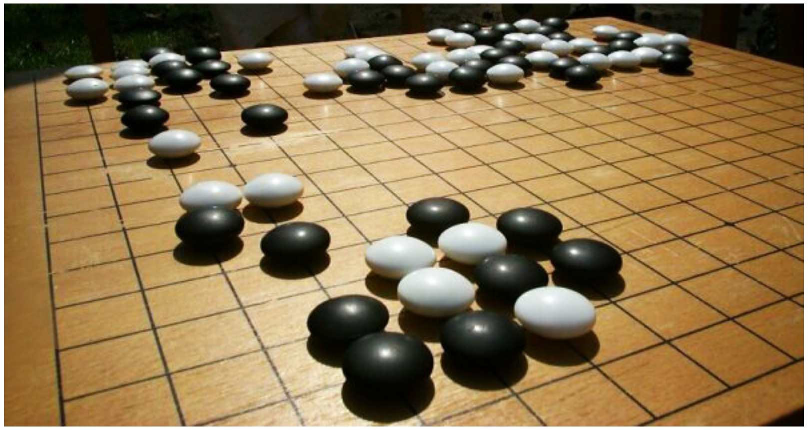
- In his 1949 paper, "Programming a Computer for Playing Chess," Claude E. Shannon suggested *chess* as an AI problem for the community.
- Deep Blue defeated human world champion Gary Kasparov in a six-game match in 1997.
- In 2006, Vladmir Kramnik, the undisputed world champion, was defeated 4–2 by Deep Fritz.

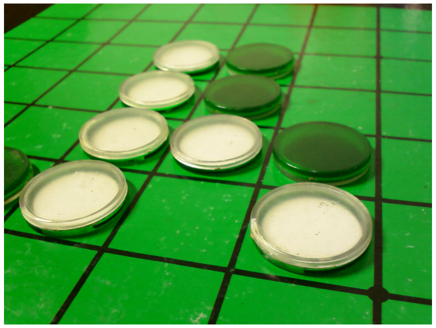


Adversarial Search

Go: $b > 300$! Google DeepMind Project AlphaGo. In 2016, AlphaGo beat both Fan Hui, the European Go champion, and Lee Sedol, the world's best player.

Othello: Several computer othello exist and human champions refuse to compete against computers that are too good.





By Donarreiskoffer By Paul_012

via Wikimedia Commons

Adversarial Search: Minimax

- Two players: Max and Min
- Players alternate turns
- Max moves first
- Max maximizes results
- Min minimizes the result
- Compute each node's minimax value's best achievable utility against an optimal adversary
- Minimax value gives best achievable payoff against best play

Minimax Example