CS 156:Introduction to Artificial Intelligence

Instructor: Dr. Sayma Akther

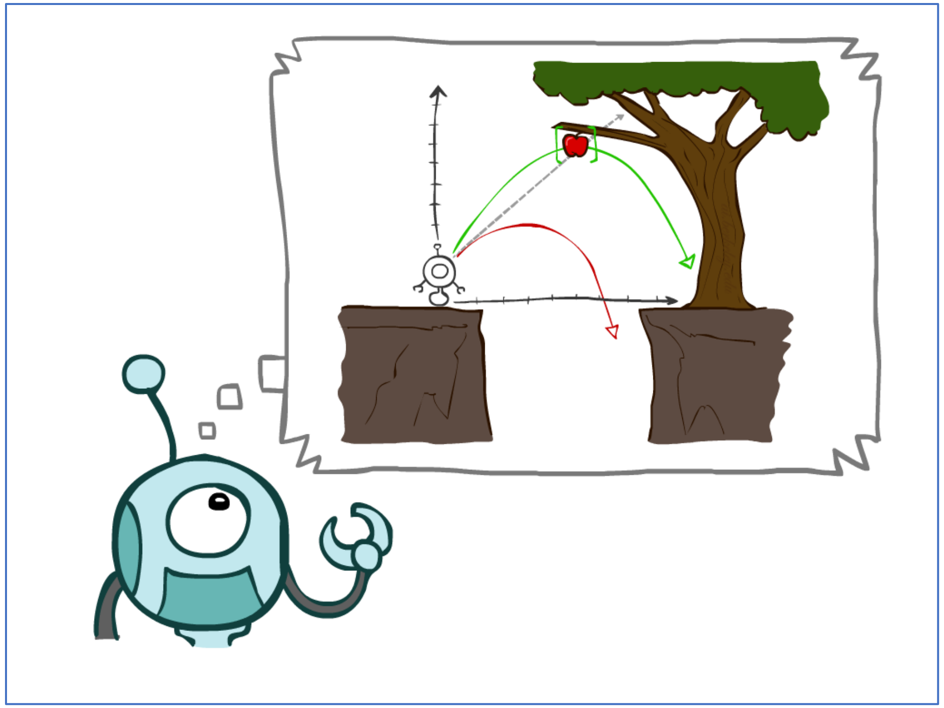
San José State University

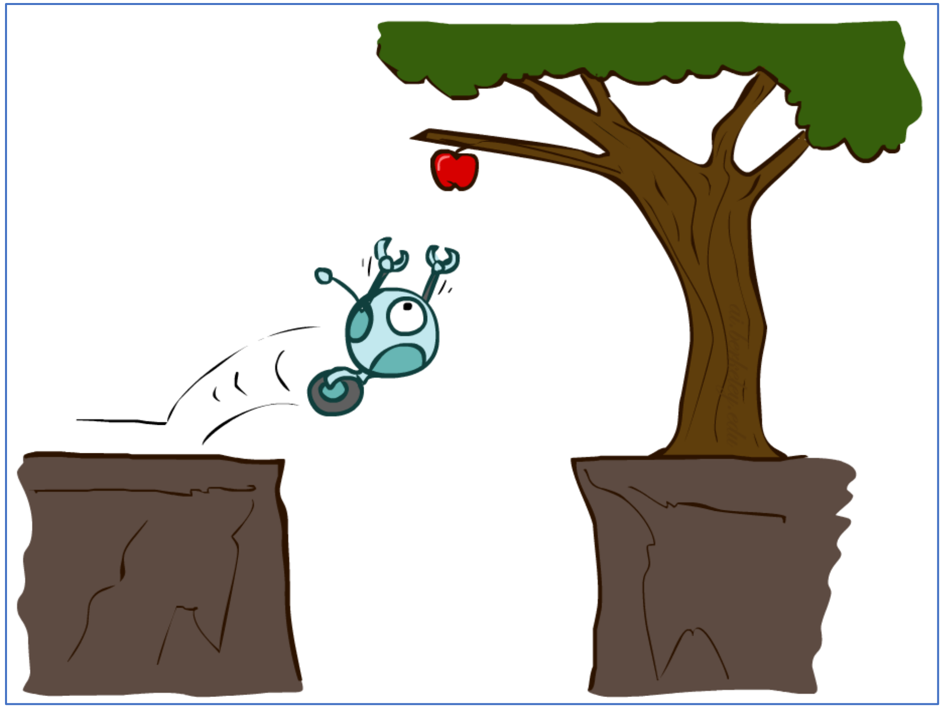
Agenda

- Agent
- Agent Types
- Environment Types and their Properties
- Search Problem

What is an Agent?

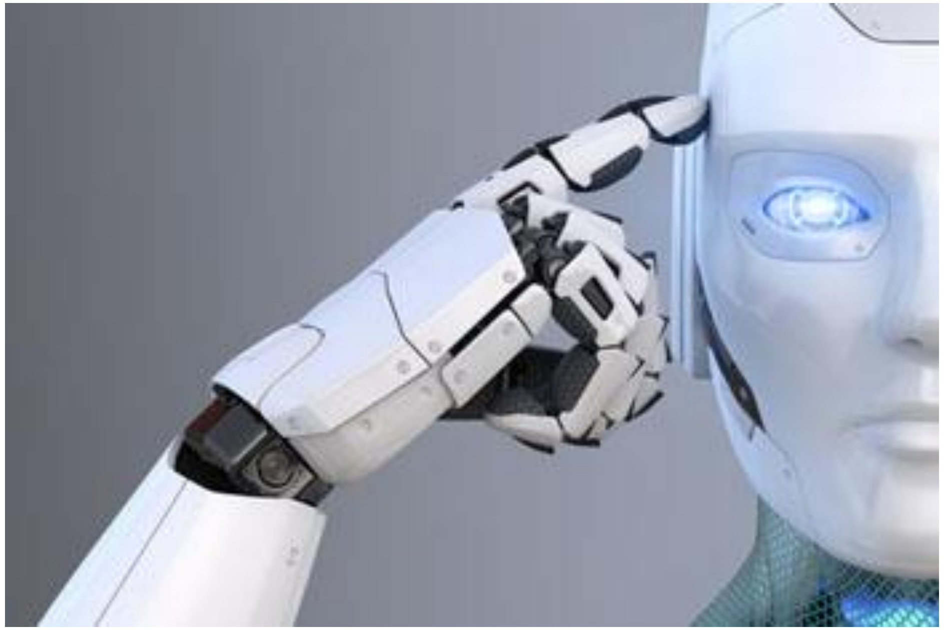
An entity that perceives its environment and acts upon it.





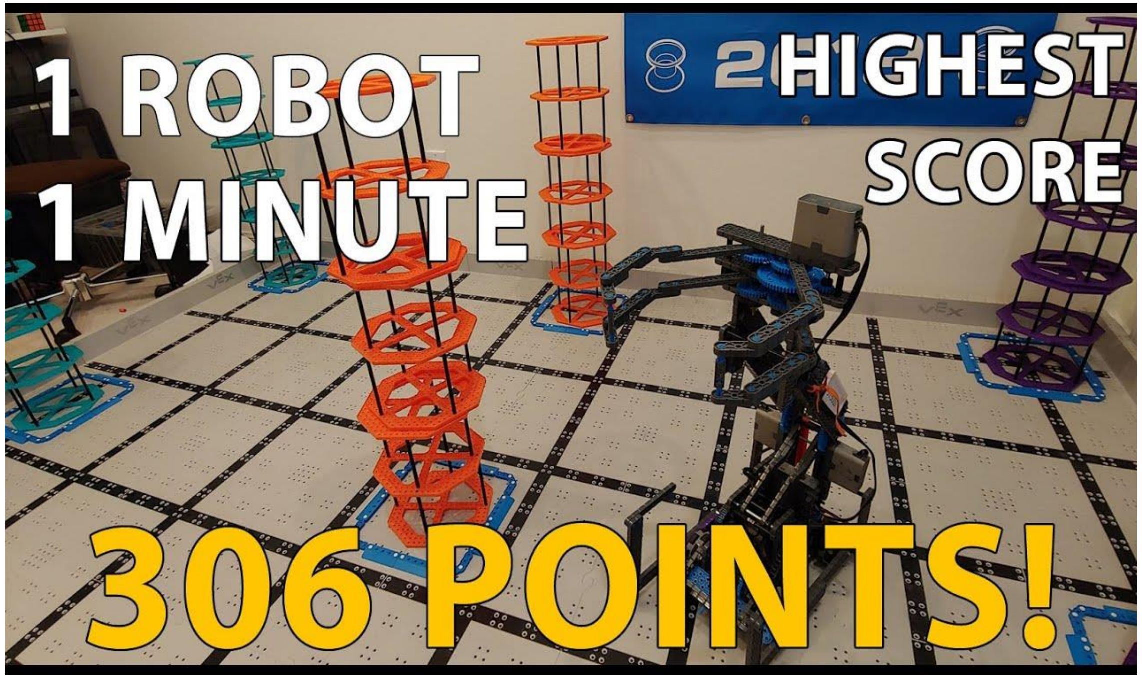
Actuators and Sensors

- **Actuators**: Devices that perform a physical action.
- **Sensors**: Devices that collect data from the environment.



Agent's Performance Measure

A criterion to evaluate the success of an agent's behavior.



Simple Reflex Agents

- Agents that directly map states to actions.
- Agents that select actions on the basis of current percepts, ignoring the rest of the percept history.
- Rule Matcher, Condition-Action Rule.
- Advantages: Speed, Simplicity
- Limitations: Lack of memory, No future planning

Simple Reflex Agents

- In Robotics
  - A vacuum cleaner robot
  - If dirt is detected → Suck
  - If obstacle ahead → Turn

Model-based Reflex Agents

- Agents that use a model of the world to make decisions.
- Agents that use memory or an internal state.

Goal-based Agents

- Agents that act to achieve specific goals.
- An intelligent program that can make decisions based on previous experiences, knowledge, user input, and the desired goal.

Utility-based Agents

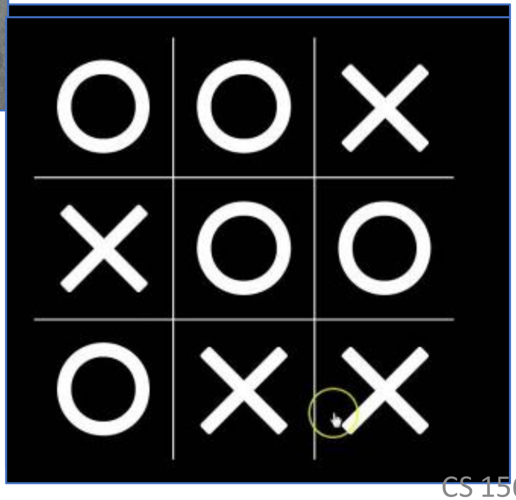
- Agents that act to maximize a predefined utility.

Fully vs. Partially Observable

- In some scenarios, an agent can see everything; in others, it can't.

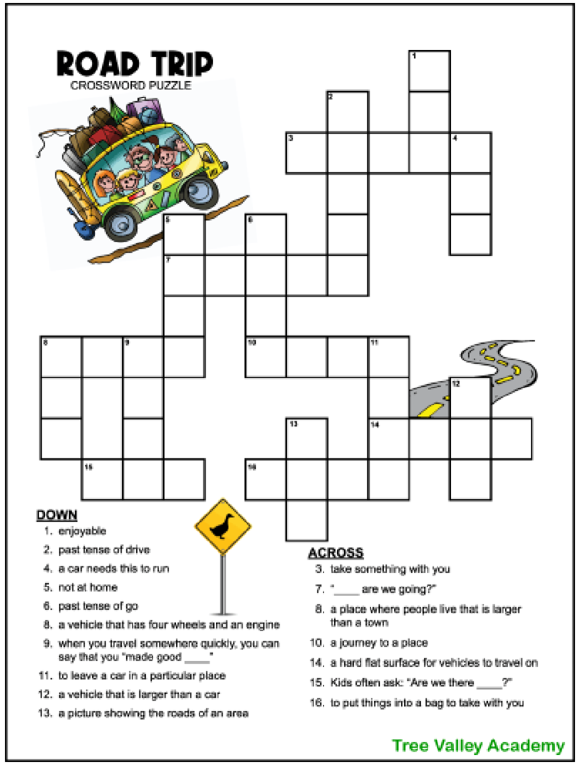
Deterministic vs. Stochastic

- Sometimes outcomes are predictable; other times, they involve randomness.



Static vs. Dynamic

- In some cases, the environment doesn't change while the agent deliberates, but in others, it does.



Search Problem?

Components of a Search Problem

- State space S:
  - Set of possible states
- Initial state S_0
- Goal states (one or more states)
- Action
  - Given a state s, ACTION(s) returns a finite set of actions that can be executed in s
- Transition model:
  - RESULT(s, a) returns the state that results from doing action a in state s
- Cost function:
  - c(s, a, s’) cost of action a in state s to reach state s’

Example Problems - Eight Puzzle

8	2	
3	4	7
5	1	6

Initial state

1	2	3
4	5	6
7	8	

Goal state

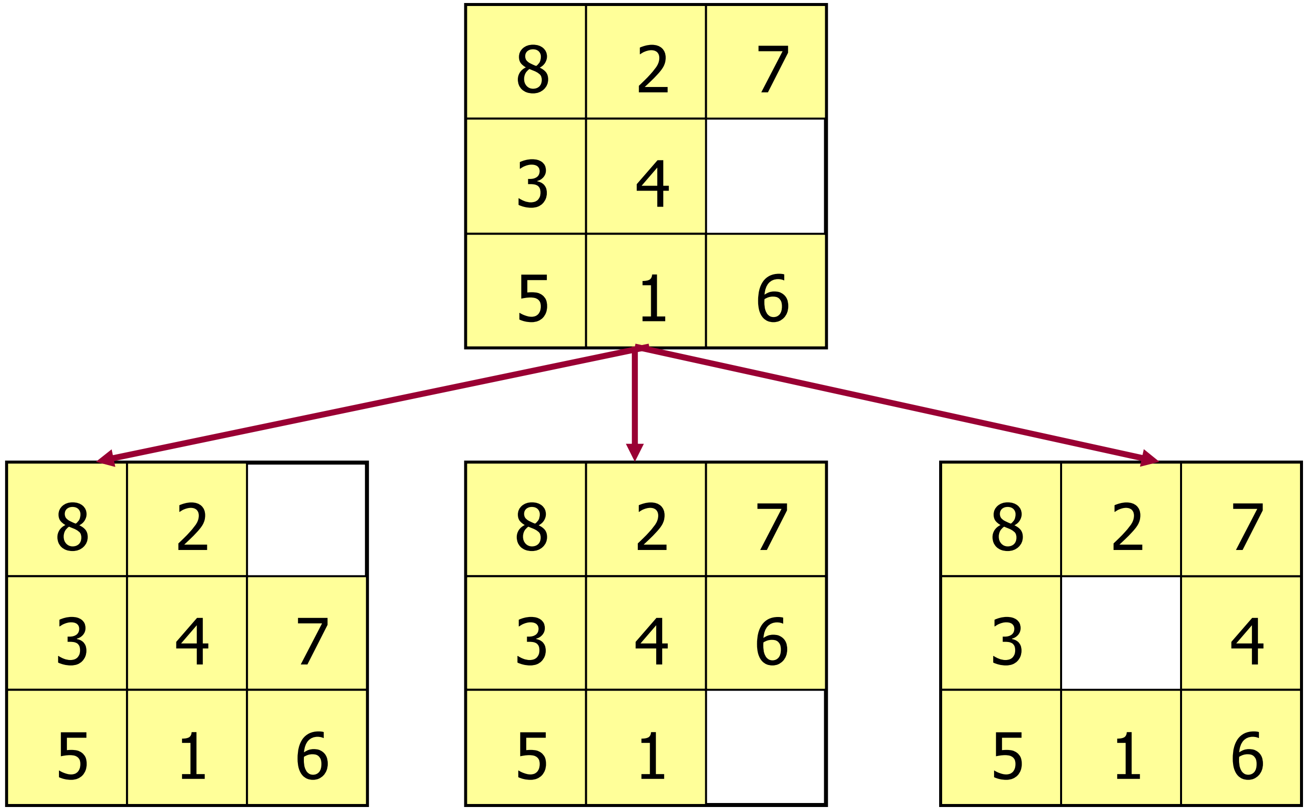
States: Any arrangement of 8 numbered tiles and an empty tile on a 3x3 board

Initial state: one specific tile configuration

Goal: tiles are numbered from one to eight around the square

Actions: move blank tile left, right, up, or down

Example Problems - Eight Puzzle



States: Any arrangement of 8 numbered tiles and an empty tile on a 3x3 board

Initial state: one specific tile configuration

Goal: tiles are numbered from one to eight around the square

Actions: move blank tile left, right, up, or down

Transition to new states

(n^2-1)-puzzle

8	2	
3	4	7
5	1	6

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

....

- How big is the state space of the (n^2-1)-puzzle?
  - 8-puzzle: 9! = 362,880 states
  - 15-puzzle: 16! ~ 2.09×10^{13} states
  - 24-puzzle: 25! ~ 10^{25} states

Example Problems – Rubik's Cube

States: list of colors for each cell on each face

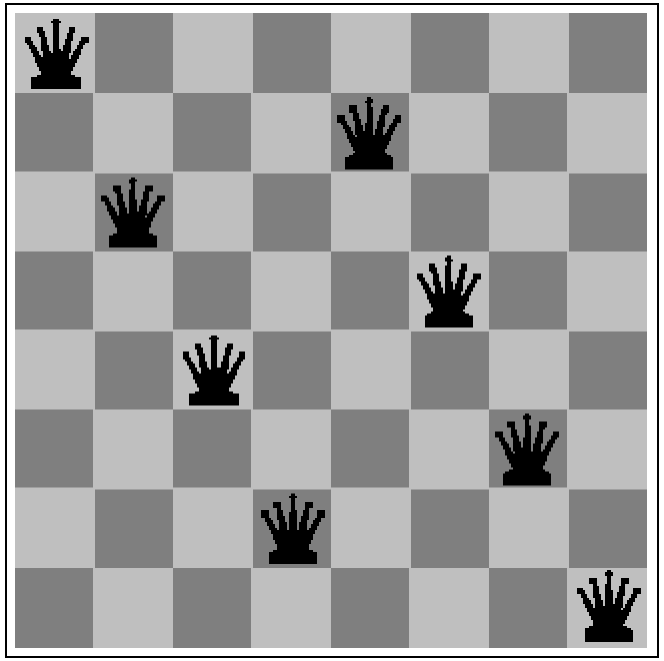
Initial state: one specific cube configuration

Actions: rotate row x or column y on face z direction a

Goal: configuration has only one color on each face

Path cost: 1 per move

Example Problems – Eight Queens (#1)



States: locations of 8 queens on chess board

Initial state: 0 queens on the board

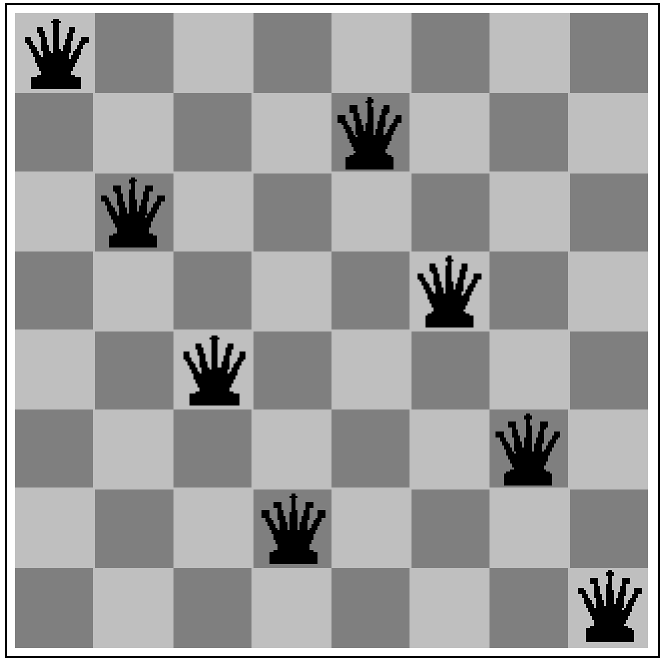
Actions: each of the successors is obtained by adding one queen in an empty square

Goal: 8 queens are on the board, with no queens attacking each other

Path cost: 0 per move

→ ~ 64x63x…x57 ~ 3×10^{14} states

Example Problems – Eight Queens (#2)



States: locations of 8 queens on chess board

Initial state: 0 queens on the board

Actions: each successor is obtained by adding one queen in any square that is not attacked by any queen already in the board, in the leftmost empty column

Goal: 8 queens are on the board, with no queens attacking each other

Path cost: 0 per move

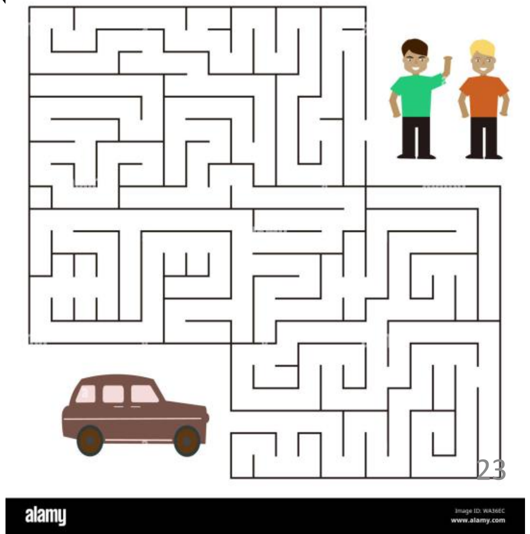
→ 2,057 states

Sample Search Problems

- Graph coloring
- Protein folding
- Game playing
- Airline travel
- Proving algebraic equalities
- Robot motion planning

Problem Solving in AI

- **Definition:** Problem-solving in AI refers to the methodology an AI system uses to find a solution from a given initial state using specified actions, eventually reaching a goal state.
- **Purpose**: AI systems are essentially problem solvers - from recognizing patterns in vast data sets to playing chess, every AI task can be viewed as a problem-solving task



Well-Defined Problems

Components of a well-defined problem:

- **Initial State**: Where we begin.
- **Actions**: What are the possibilities from a given state?
- **Transition Model**: Result of an action.
- **Goal Test**: Determine if a state is a goal state.
- **Path Cost**: Numeric cost to move from one state to another.

Well-Defined Problems

Components of a well-defined problem:

- **Initial State**: Where we begin.
- **Actions**: What are the possibilities from a given state?
- **Transition Model**: Result of an action.
- **Goal Test**: Determine if a state is a goal state.
- **Path Cost**: Numeric cost to move from one state to another.

- Objective: shortest? fastest? most scenic?
- Actions: go straight, turn left, turn righ

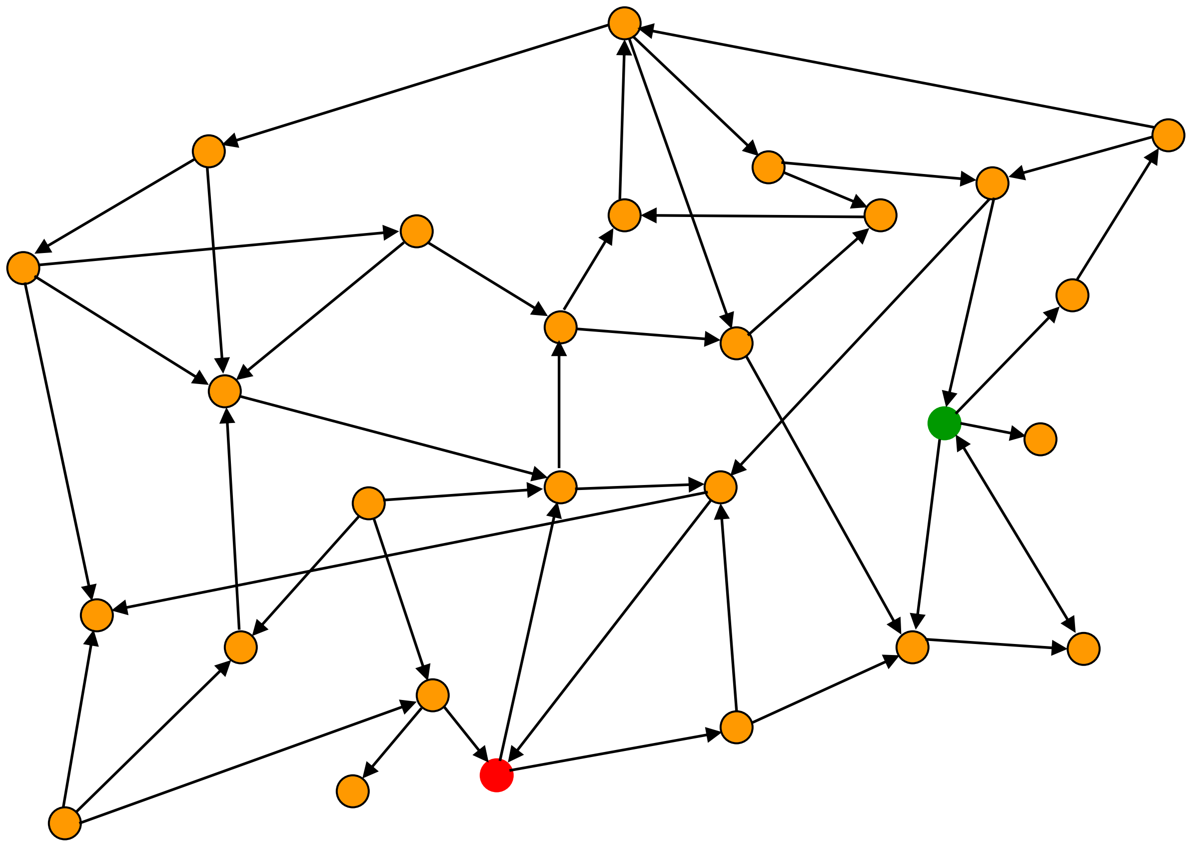
Problem Space

- **Definition**: The environment in which the search takes place.
- **Visualization**: Show a tree or graph to illustrate how states are connected.

Challenges in Problem Solving

- **Large Problem Spaces**: The sheer number of possibilities in some problems.
- **Unknown Environments**: When the AI does not have full knowledge of all possible states.
- **Dynamic Environments**: When the environment can change while the AI is deciding.

Searching the State Space



- It is often not feasible (or too expensive) to build a complete representation of the state graph

8-, 15-, 24-Puzzles

8-puzzle → 362,880 states

0.036 sec

15-puzzle → 2.09×10^{13} states

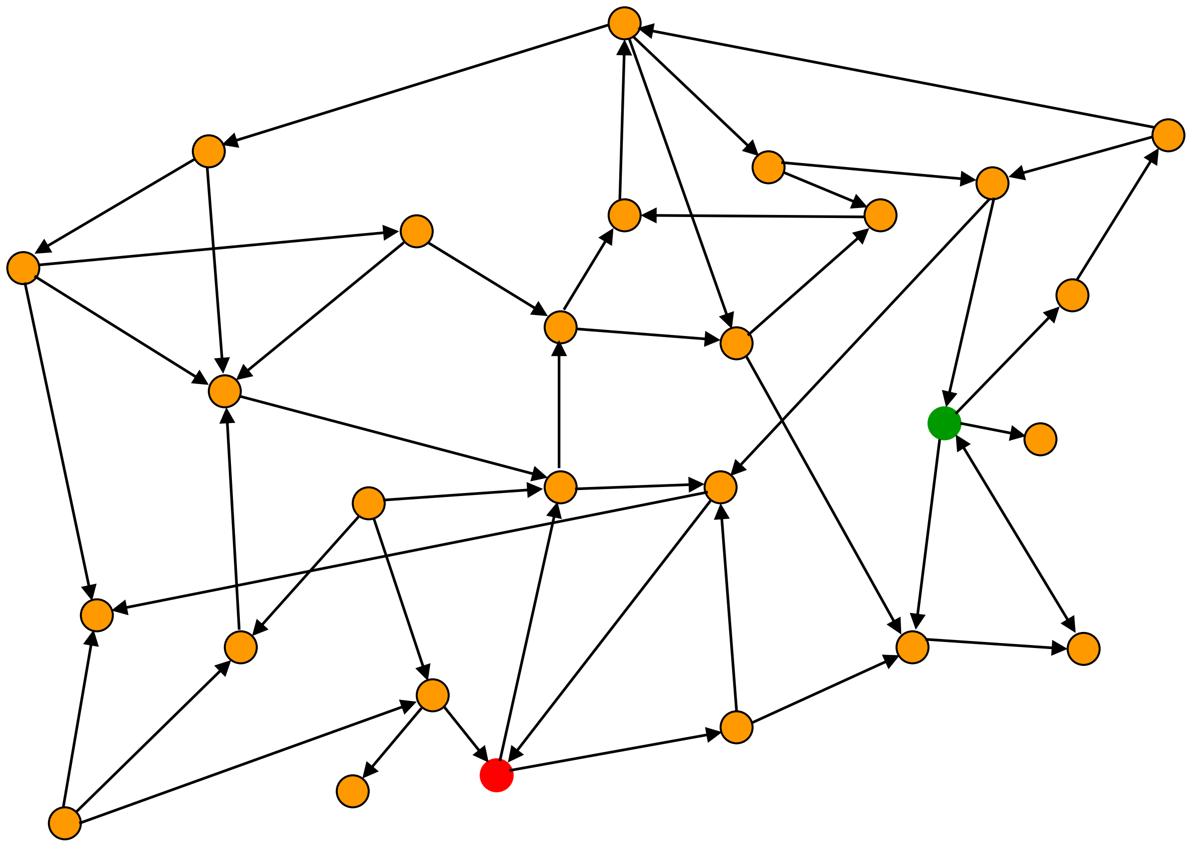
~ 55 hours

24-puzzle → 10^{25} states

> 10^{9} years

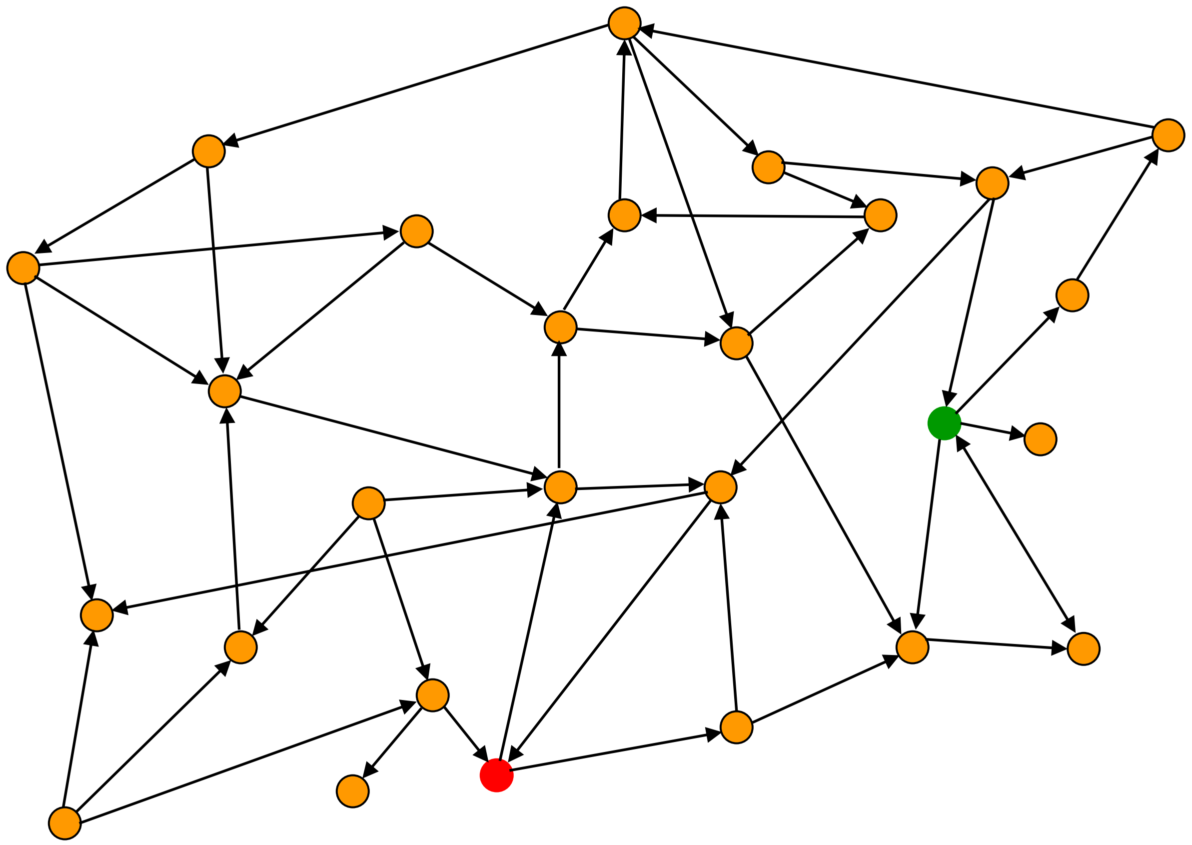
100 millions states/sec

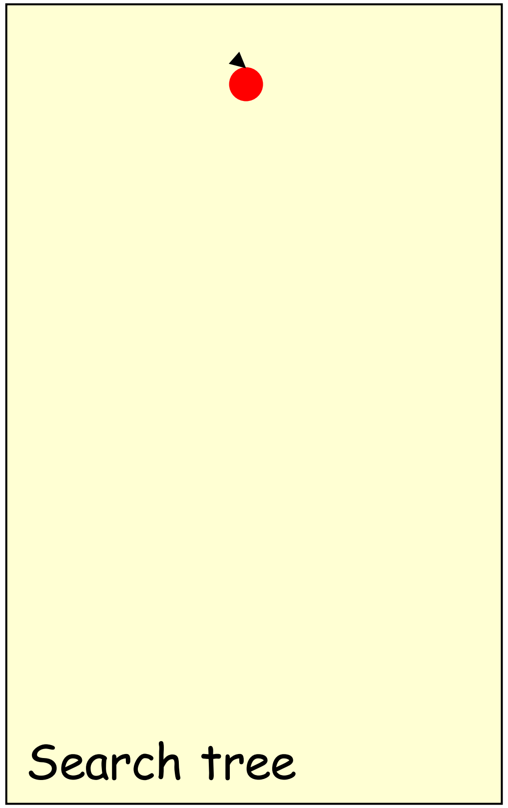
Searching the State Space



- Often it is not feasible (or too expensive) to build a complete representation of the state graph
- A problem solver must construct a solution by exploring a small portion of the graph

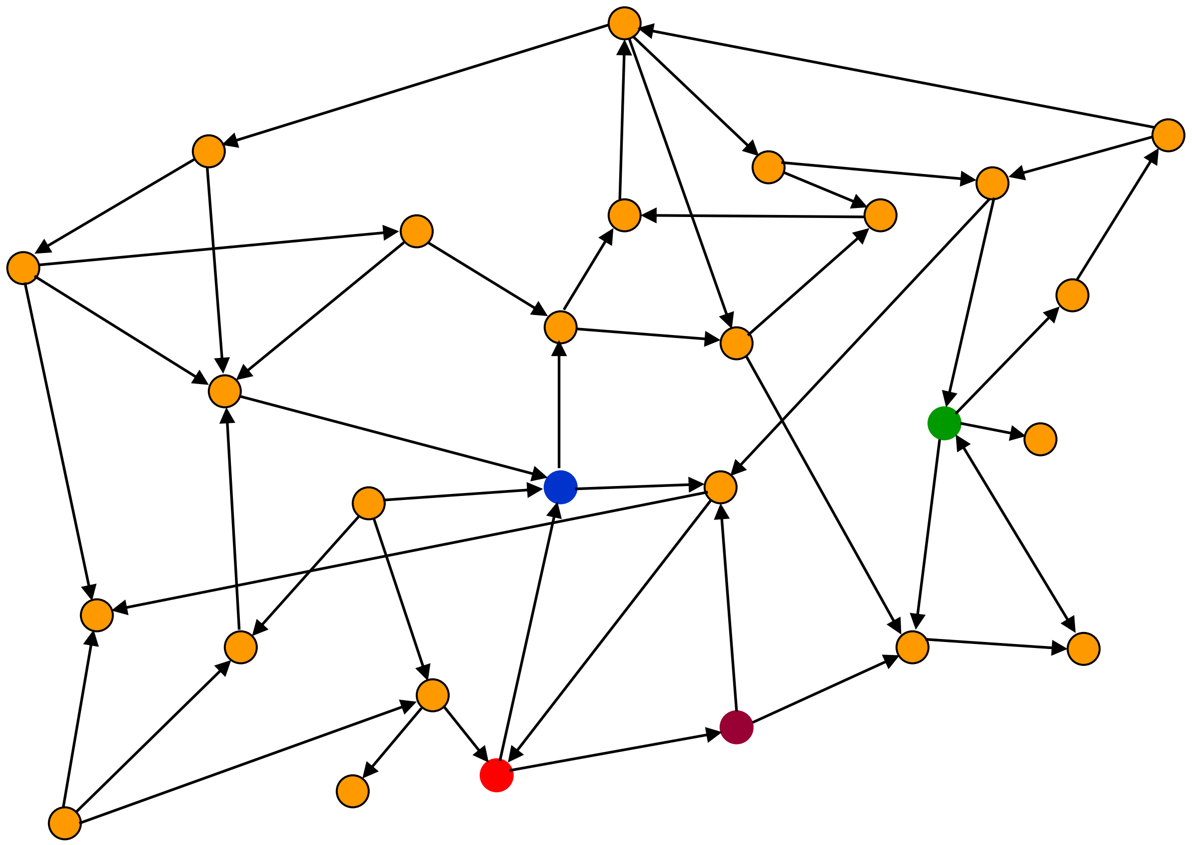
Searching the State Space

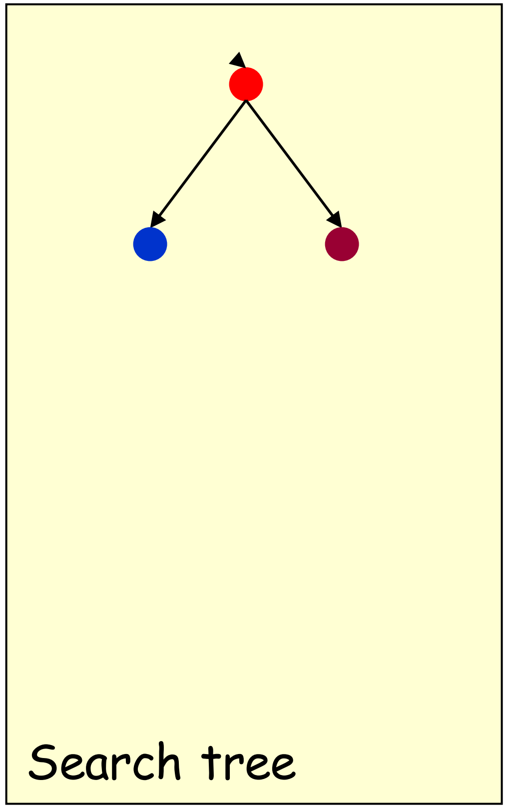




Visualize Search Space as a Tree

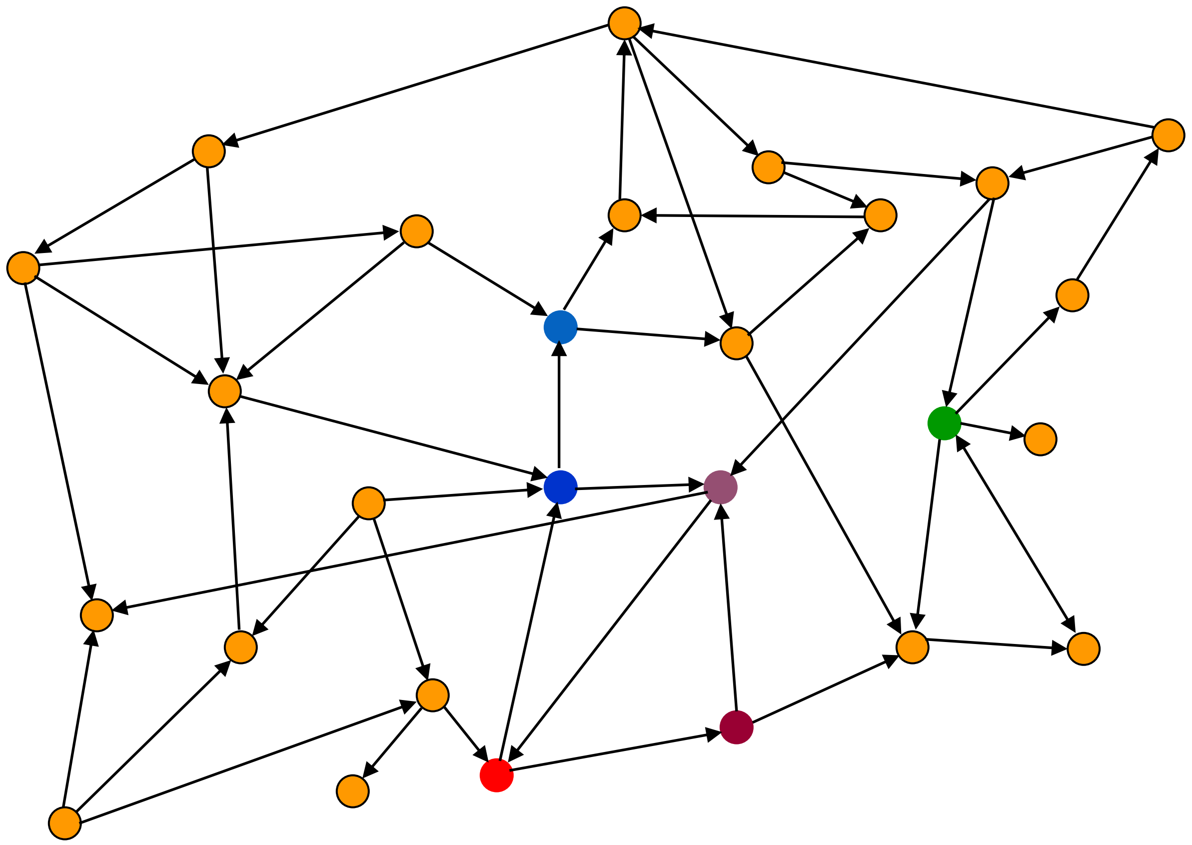
Searching the State Space

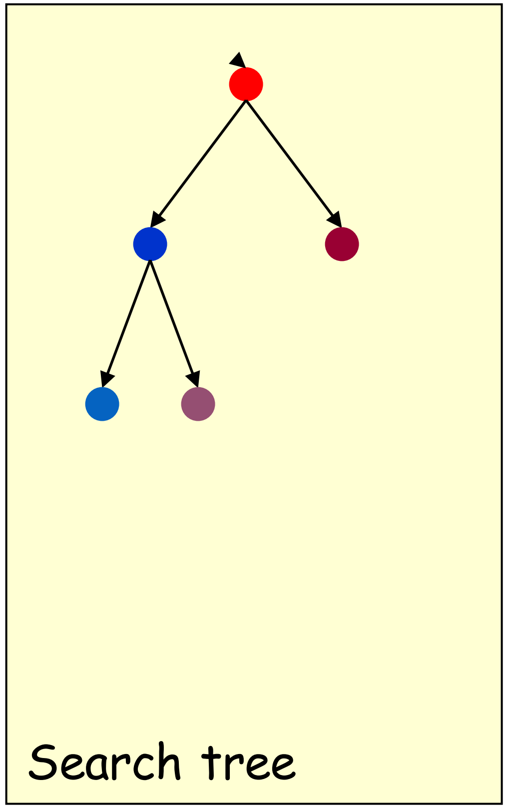




Visualize Search Space as a Tree

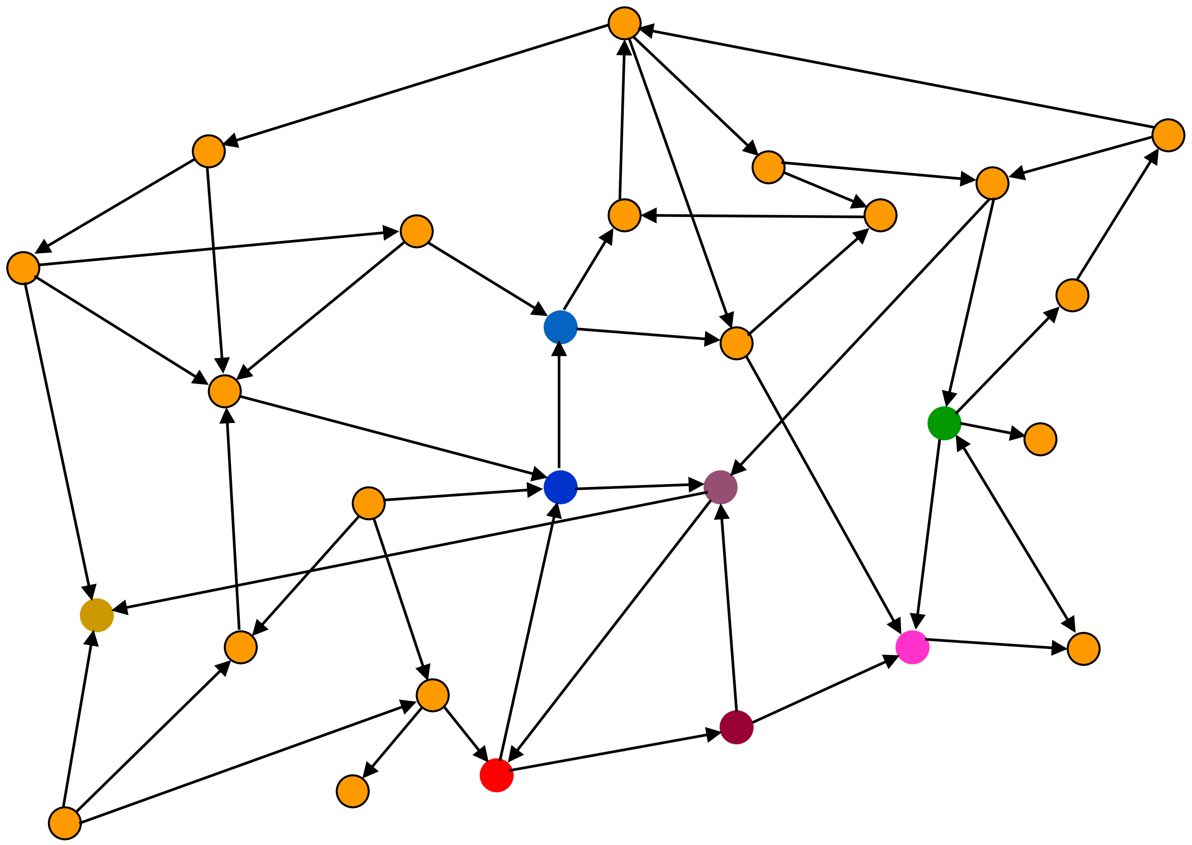
Searching the State Space

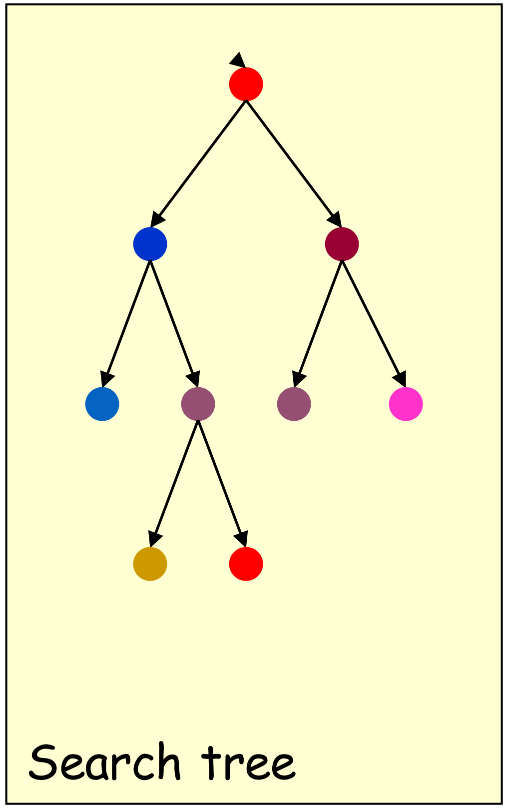




Visualize Search Space as a Tree

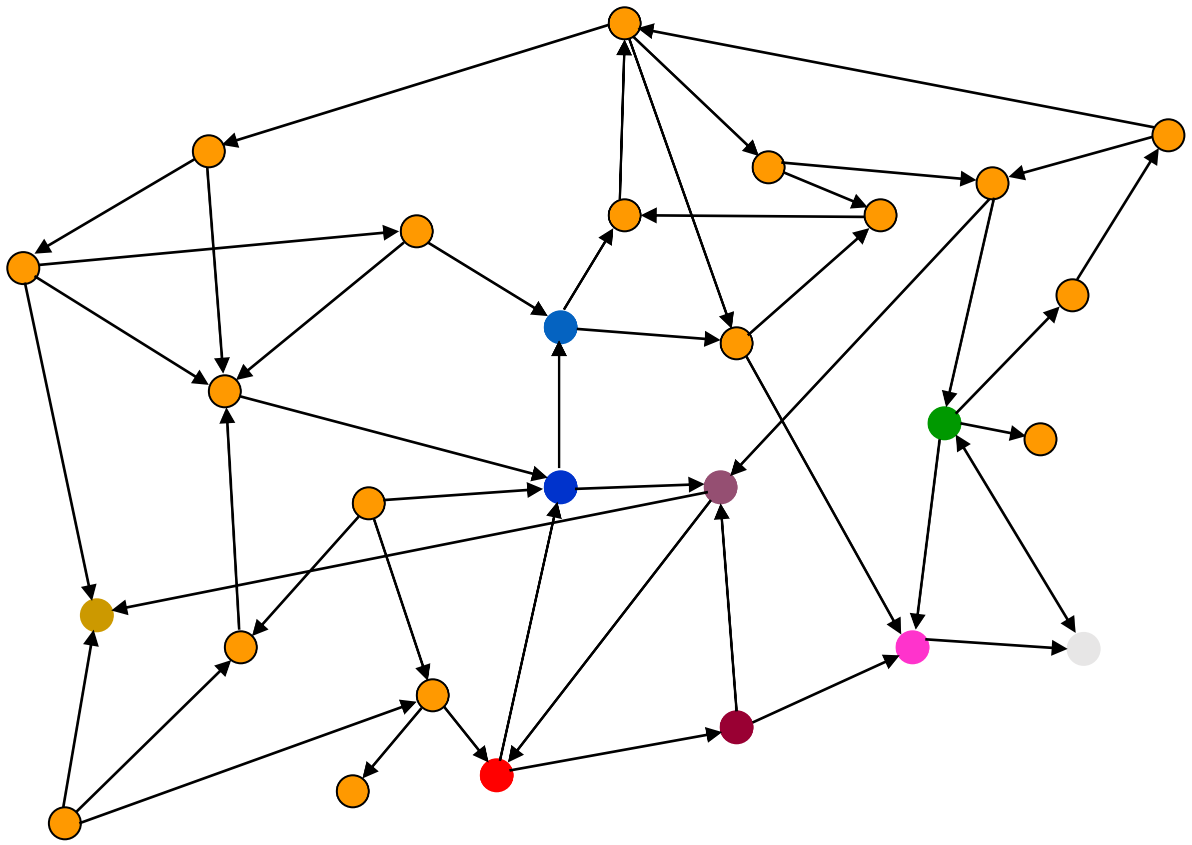
Searching the State Space

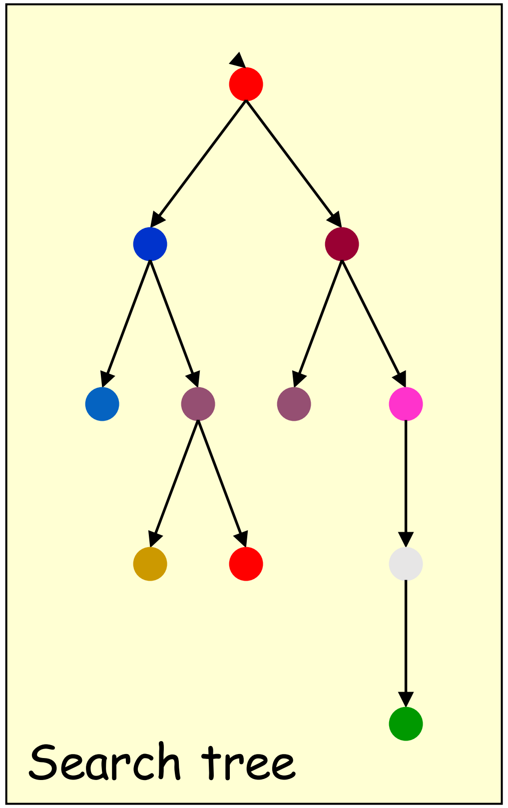




Visualize Search Space as a Tree

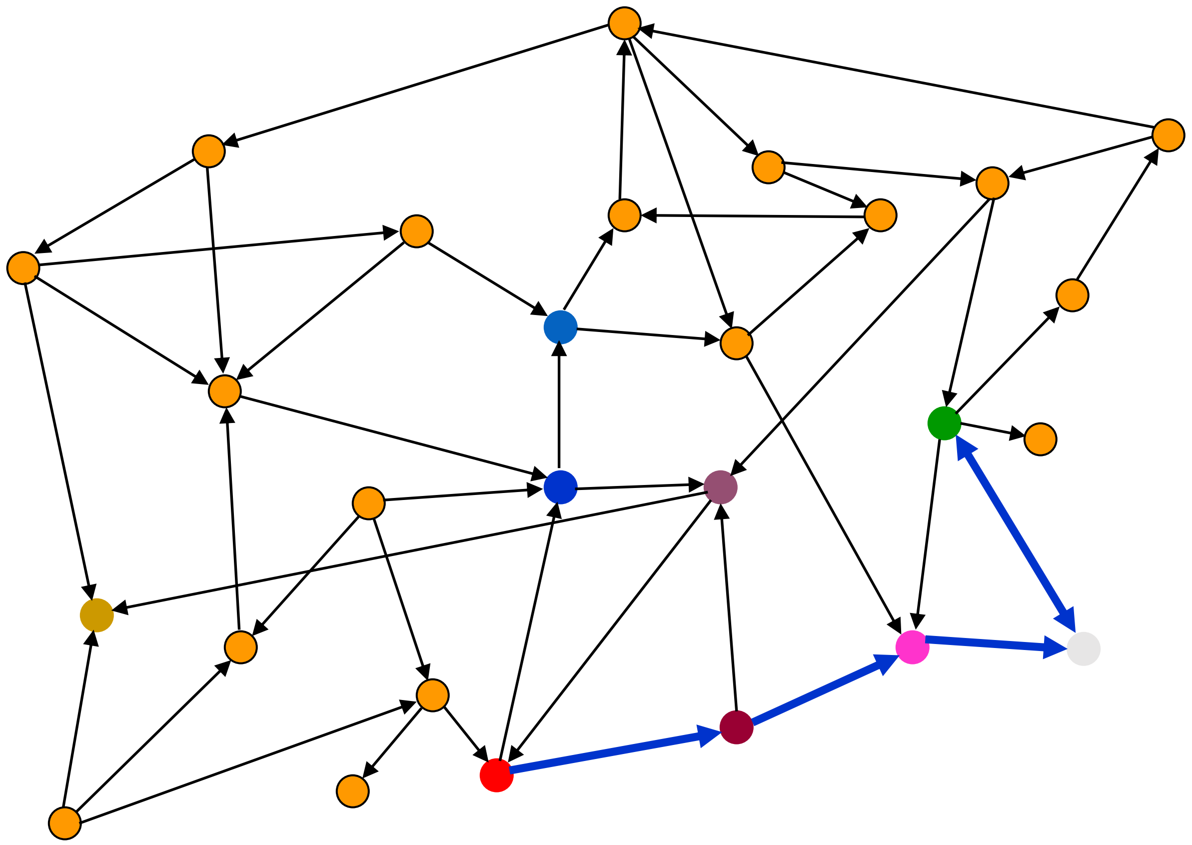
Searching the State Space

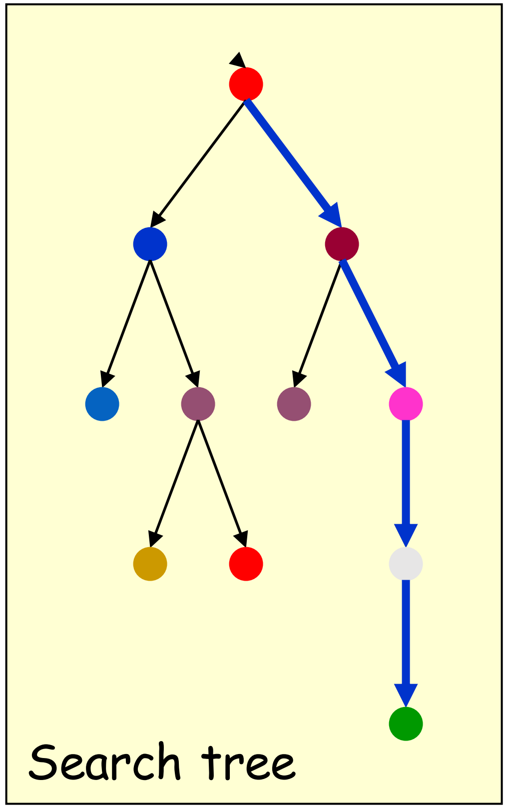




Visualize Search Space as a Tree

Searching the State Space





Visualize Search Space as a Tree