CS 235: Artificial Intelligence

Week 1

Automation vs Artificial Intelligence
Solving Problem by Searching

Dr. Moumita Roy

CSE Dept., IIITG

Reference: http://ai.stanford.edu/~latombe/cs121/2011/schedule.htm

Automation vs AI

- Automation and AI are often used interchangeably, but they are not exactly same.
- Automation is designed something which runs itself with little or no human interaction by some specific patterns and rules to perform repetitive tasks.
- AI can be defined as the collection of different technologies that allow the machine to act as the human level of intelligence.
- Automation can or cannot be based on AI.
- Example:

  Automatic washing machine vs AI-powered automatic washing machine

  vacuum cleaner vs Robo clean

  Automatic car vs Self-driving car

Strong AI or Weak AI? Depending on the Design

Introduce a Problem: 8-Puzzle

8	2	
3	4	7
5	1	6

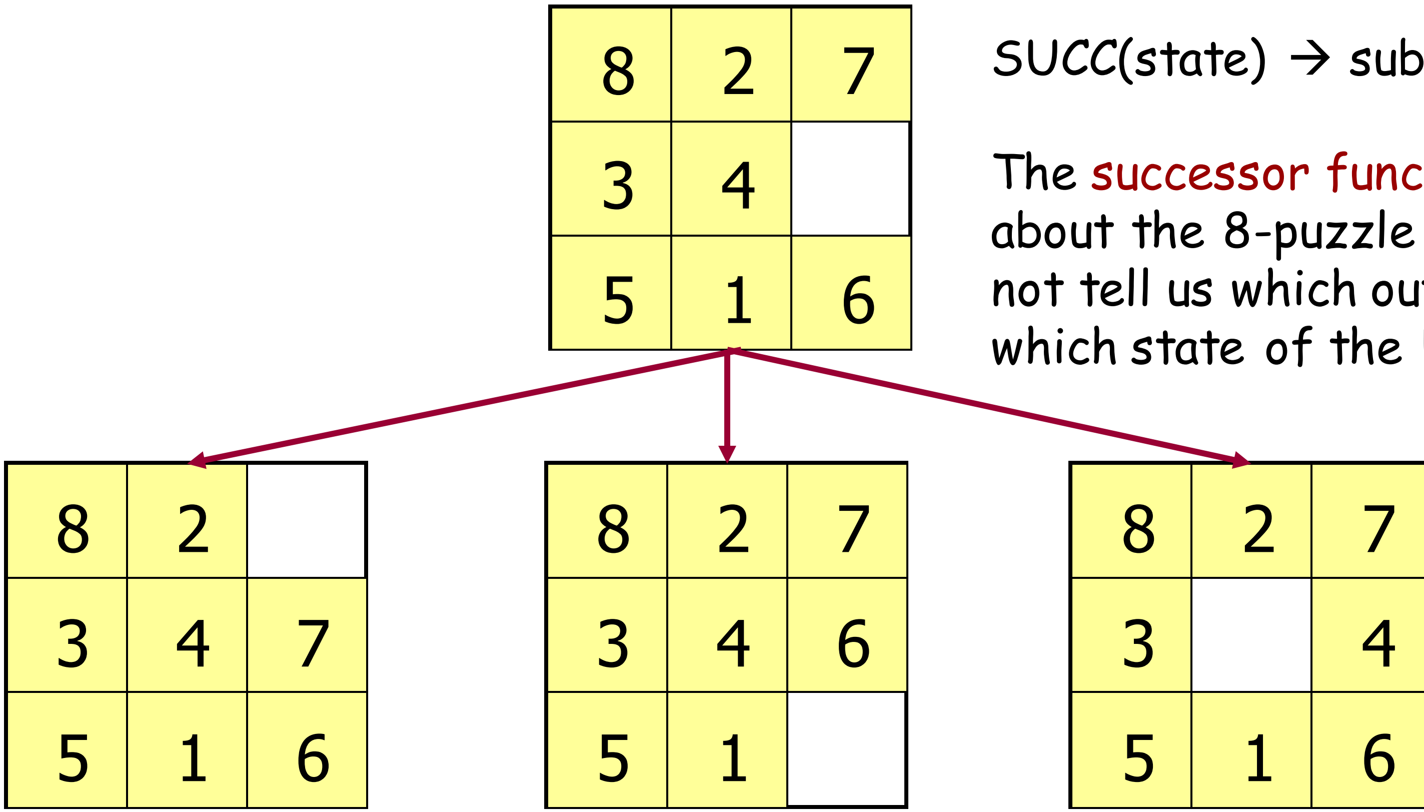
Initial state

1	2	3
4	5	6
7	8	

Goal state

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

8-Puzzle: Successor Function



SUCC(state) → subset of states

The successor function is knowledge about the 8-puzzle game, but it does not tell us which outcome to use, nor to which state of the board to apply it.

Search is about the exploration of alternatives

(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	

. . . .

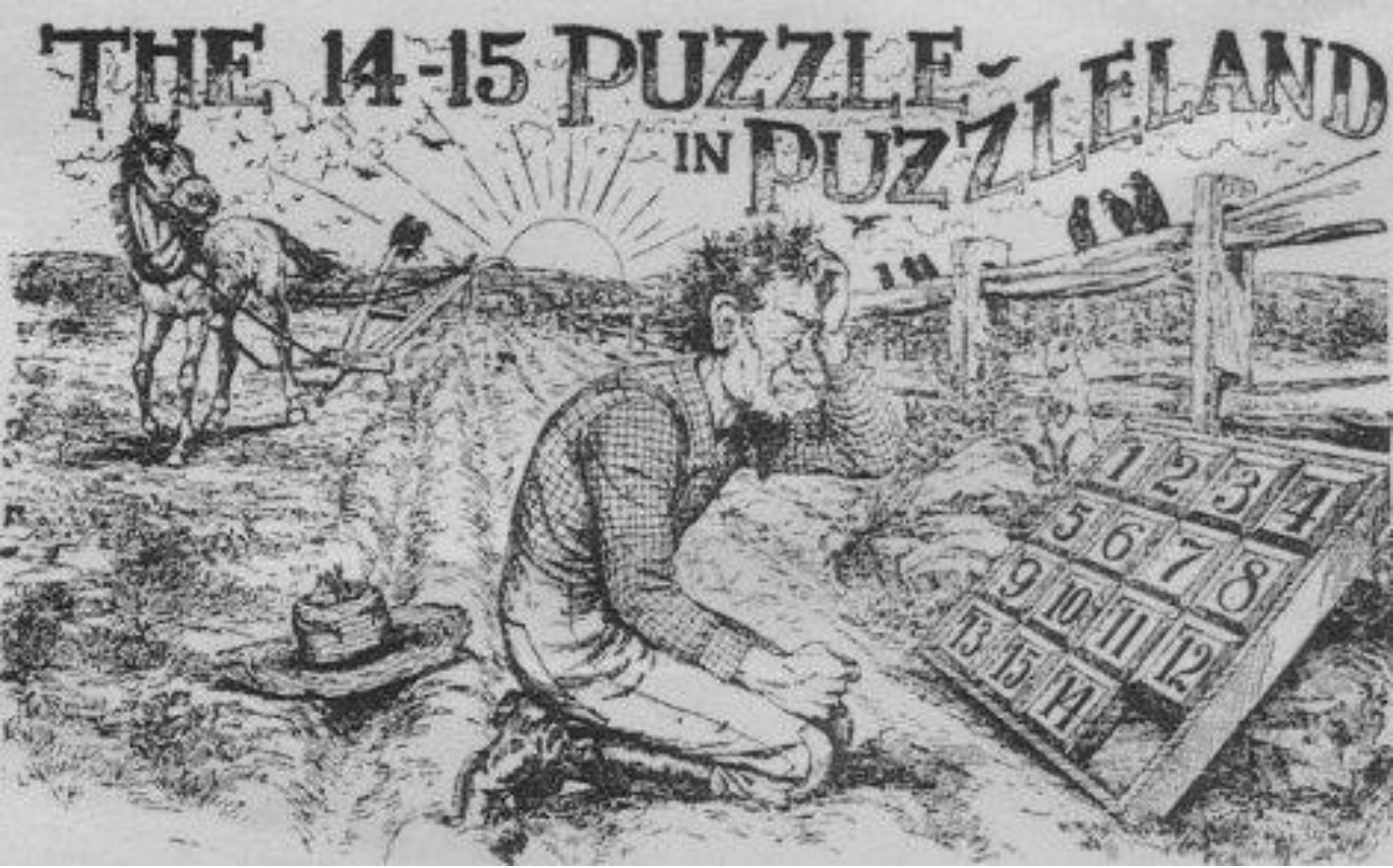
15-Puzzle

Sam Loyd offered $1,000 of his own money to the first person who would solve the following problem:

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

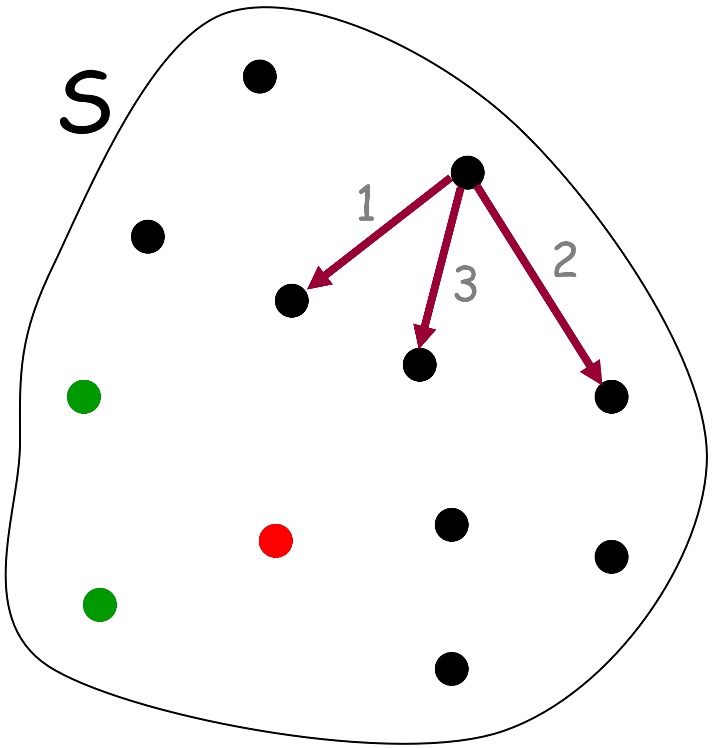
?

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



But no one ever won the prize !!

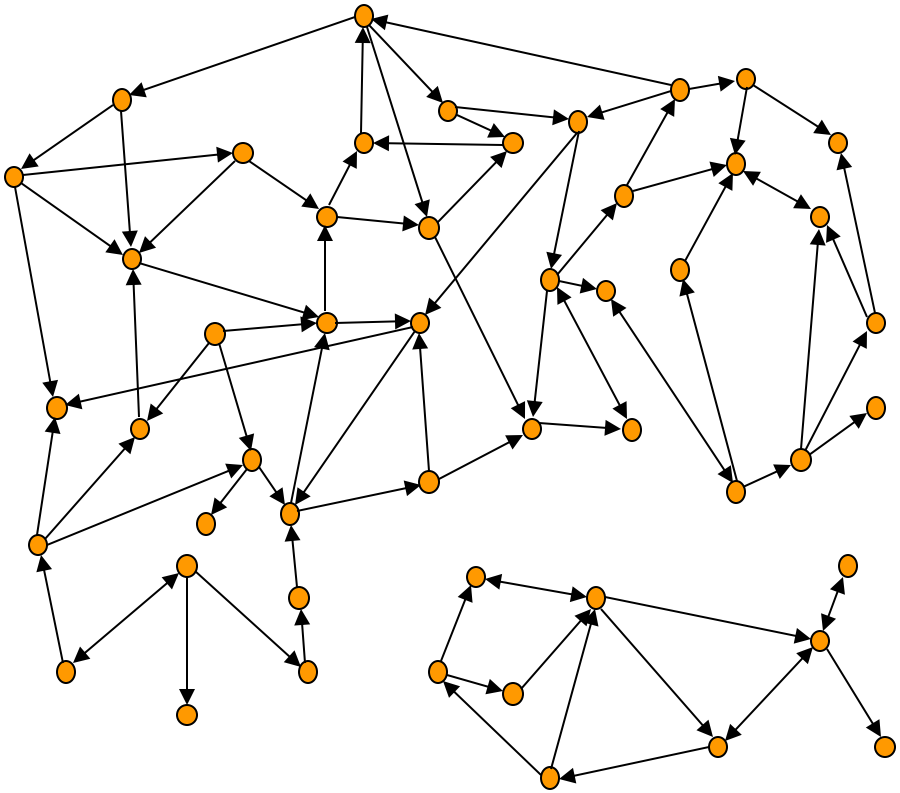
Stating a Problem as a Search Problem



- State space S
- Successor function:
  $x \in S \rightarrow \text{SUCCESSORS}(x) \in 2^S$
- Initial state s_0
- Goal test:
  $x \in S \rightarrow \text{GOAL?}(x) = T \text{ or } F$
- Arc cost

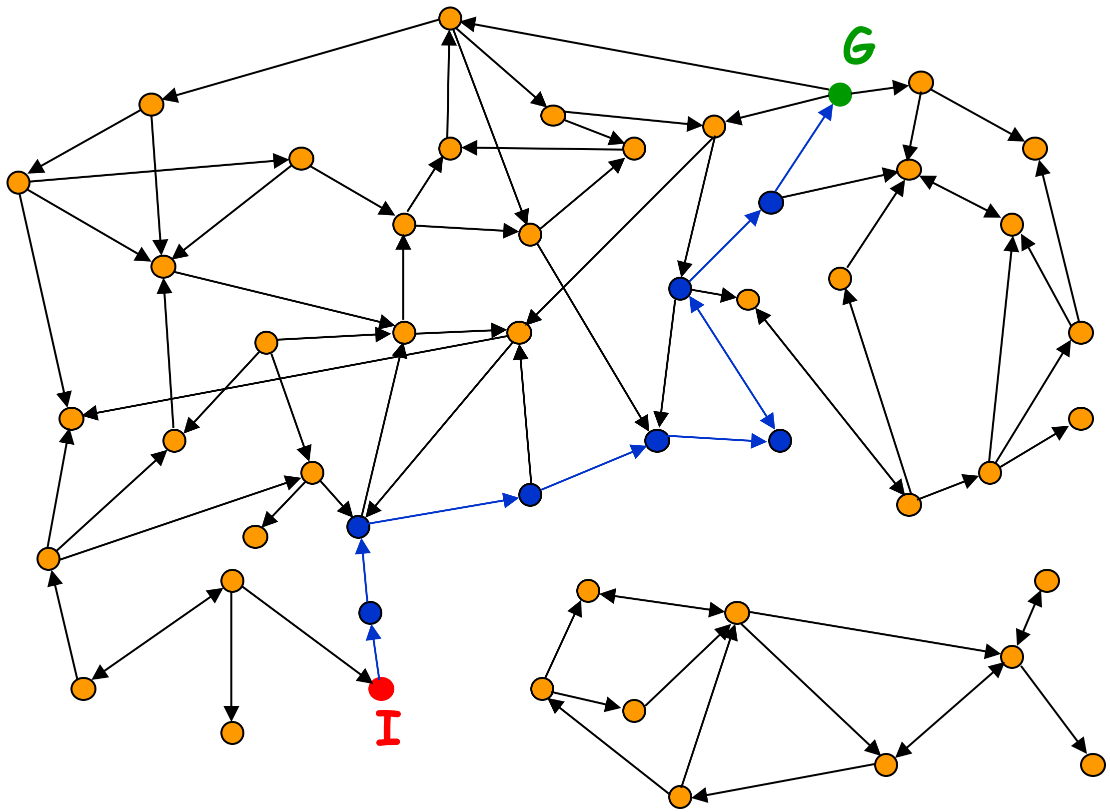
State Graph

- Each state is represented by a distinct node
- An arc (or edge) connects a node s to a node s' if s' $\in$ SUCCESSORS(s)
- The state graph may contain more than one connected component



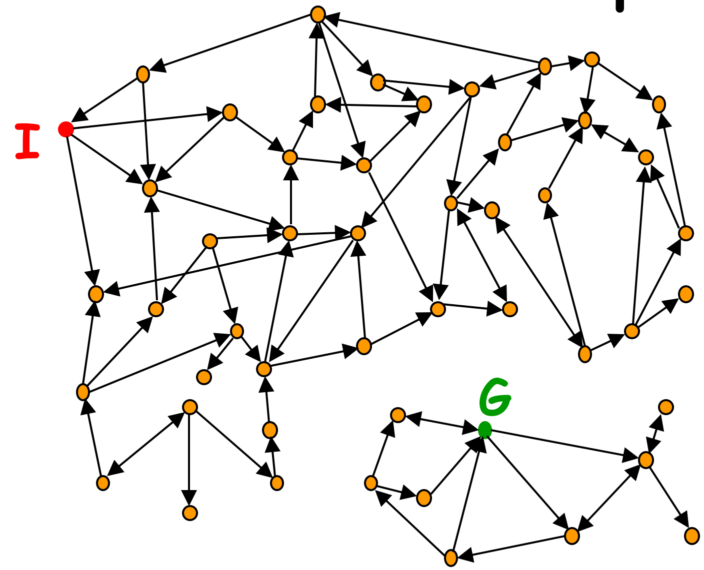
Solution to the Search Problem

- A solution is a path connecting the initial node to a goal node (any one)



Solution to the Search Problem

- A solution is a path connecting the initial node to a goal node (any one)
- The cost of a path is the sum of the arc costs along this path
- An optimal solution is a solution path of minimum cost
- There might be no solution !



How big is the state space of the (n^2-1)-puzzle?

- 8-puzzle → ?? states

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

But <u>only half</u> of these states are reachable from any given state
(but you may not know that in advance)

15-Puzzle

Sam Loyd offered $1,000 of his own money to the first person who would solve the following problem:

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

?

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

N = 4

N = 5

So, the second state is not reachable from the first, and Sam Loyd took no risk with his money ...

What is the Actual State Space?

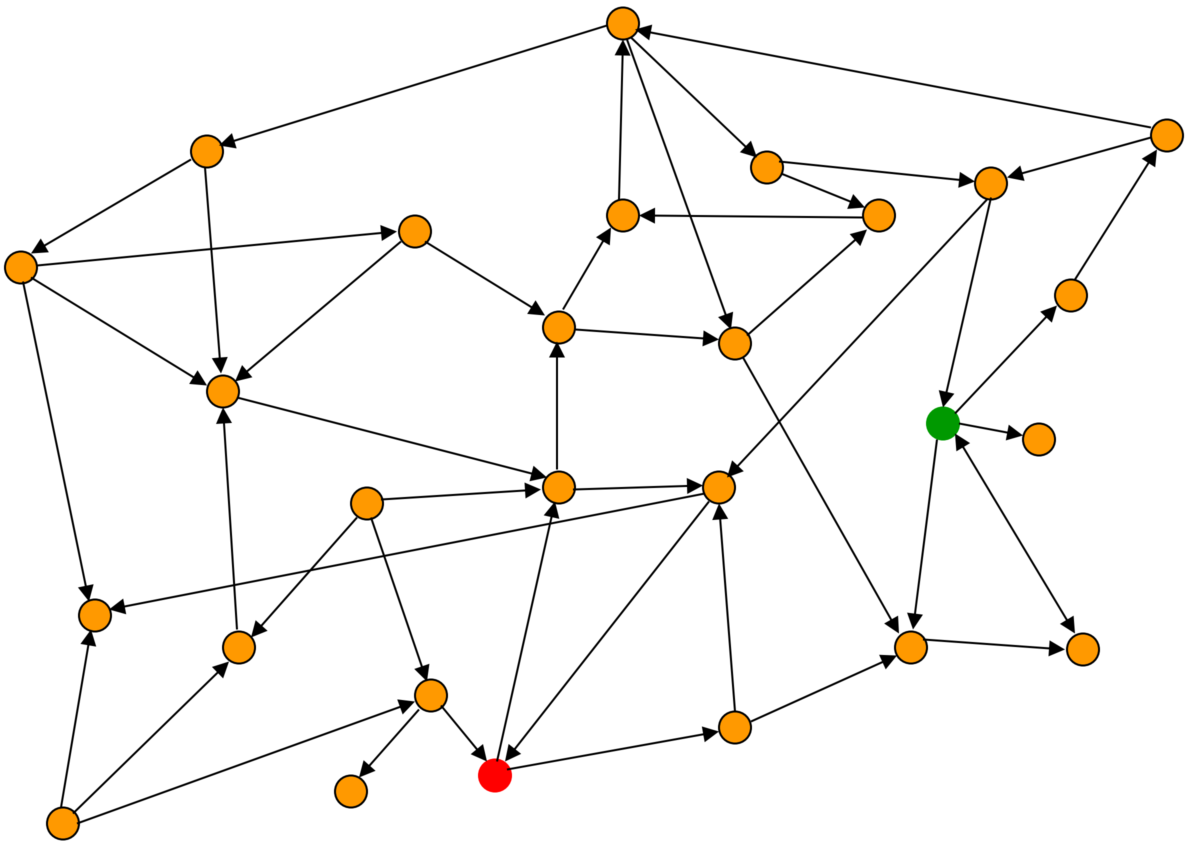
a) The set of all states?
   [e.g., a set of 16! states for the 15-puzzle]

b) The set of all states reachable from a given initial state?
   [e.g., a set of 16!/2 states for the 15-puzzle]

In general, the answer is a)
[because one does not know in advance which states are reachable]

But a fast test determining whether a state is reachable from another is very useful, as search techniques are often **inefficient** when a problem has no solution

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

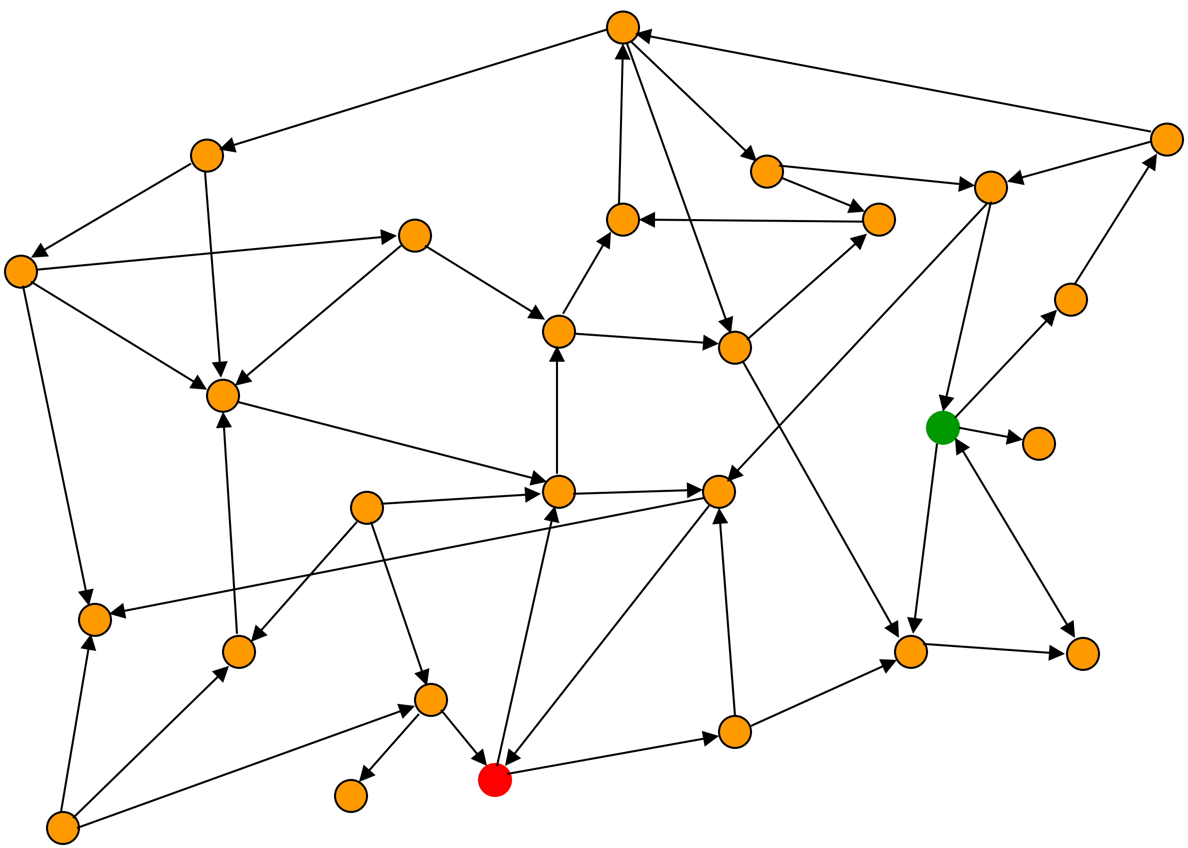
~ 55 hours

24-puzzle → 10^{25} states

> 10^{9} years

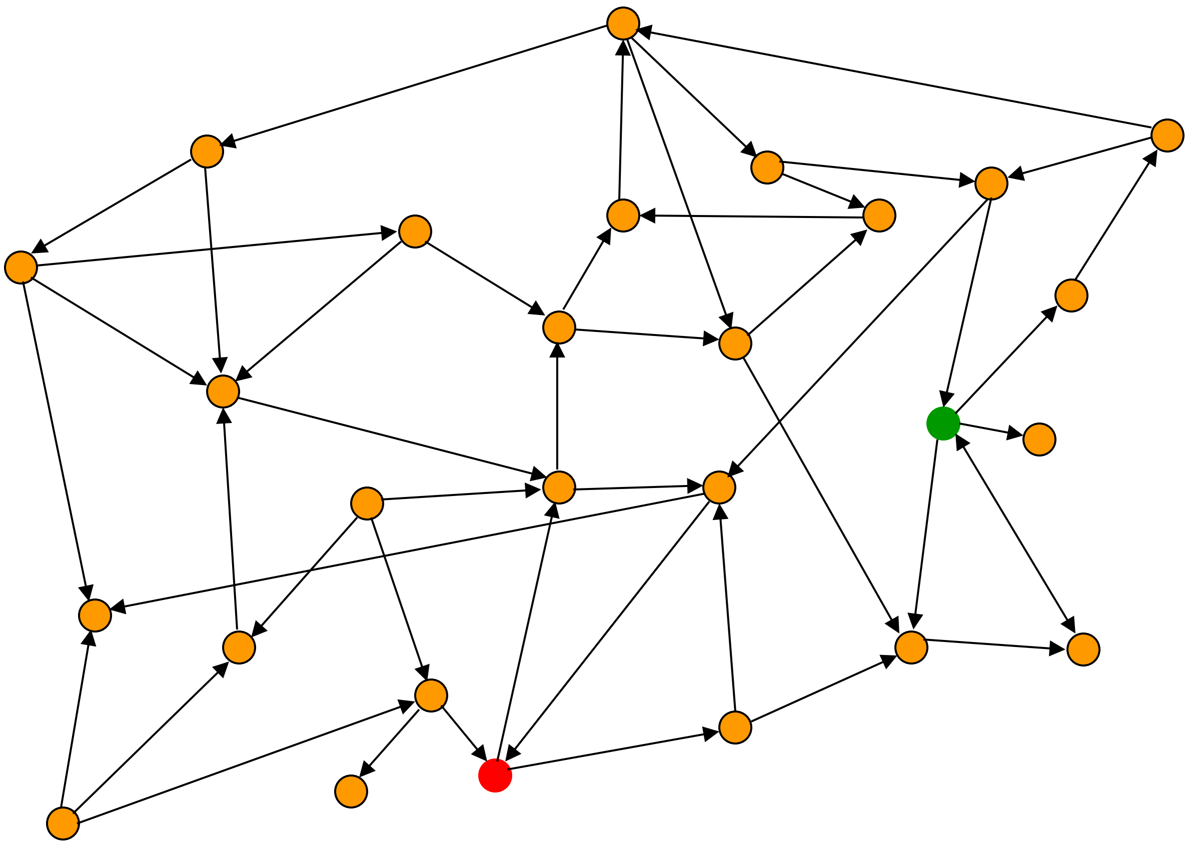
100 millions states/sec

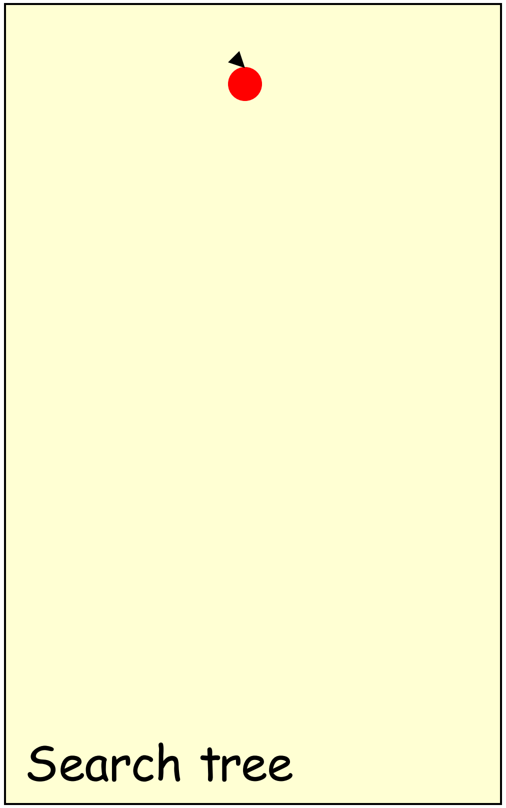
Searching the State Space



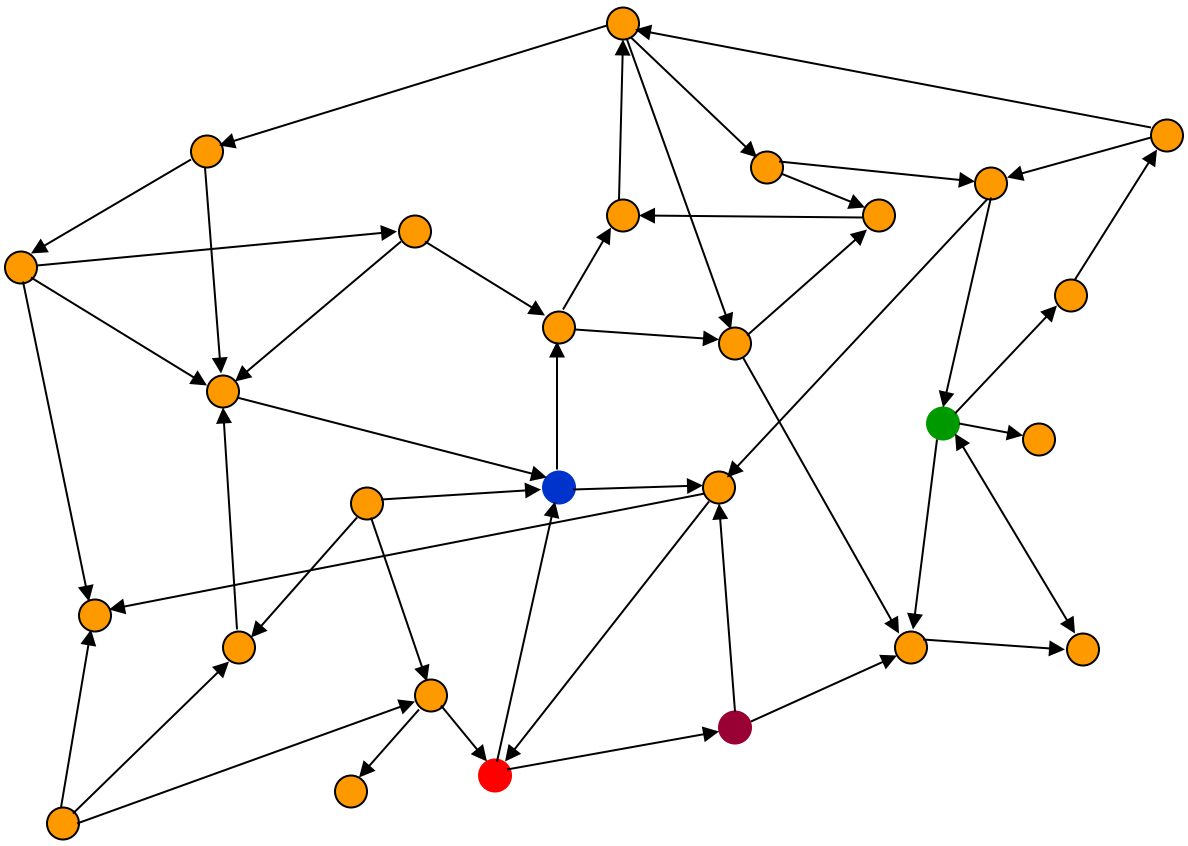
- Often it is not feasible (or too expensive) to build a complete representation of the state graph
- A problem solver must construct a solution by exploring a small portion of the graph

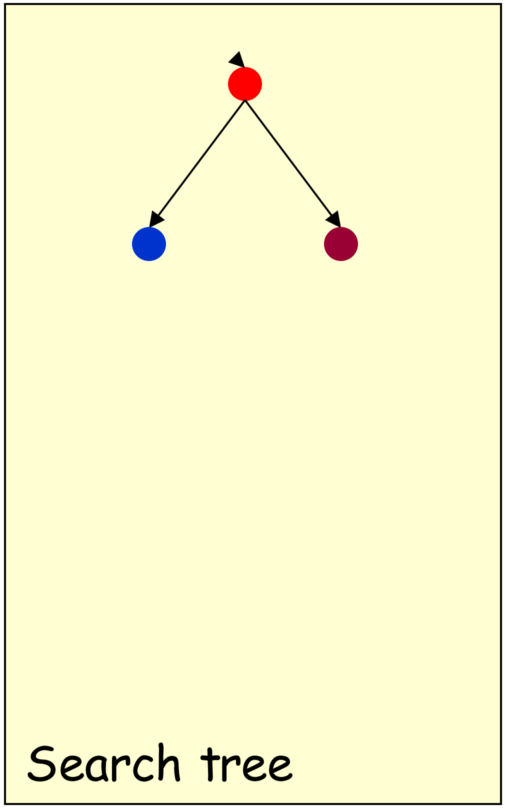
Searching the State Space



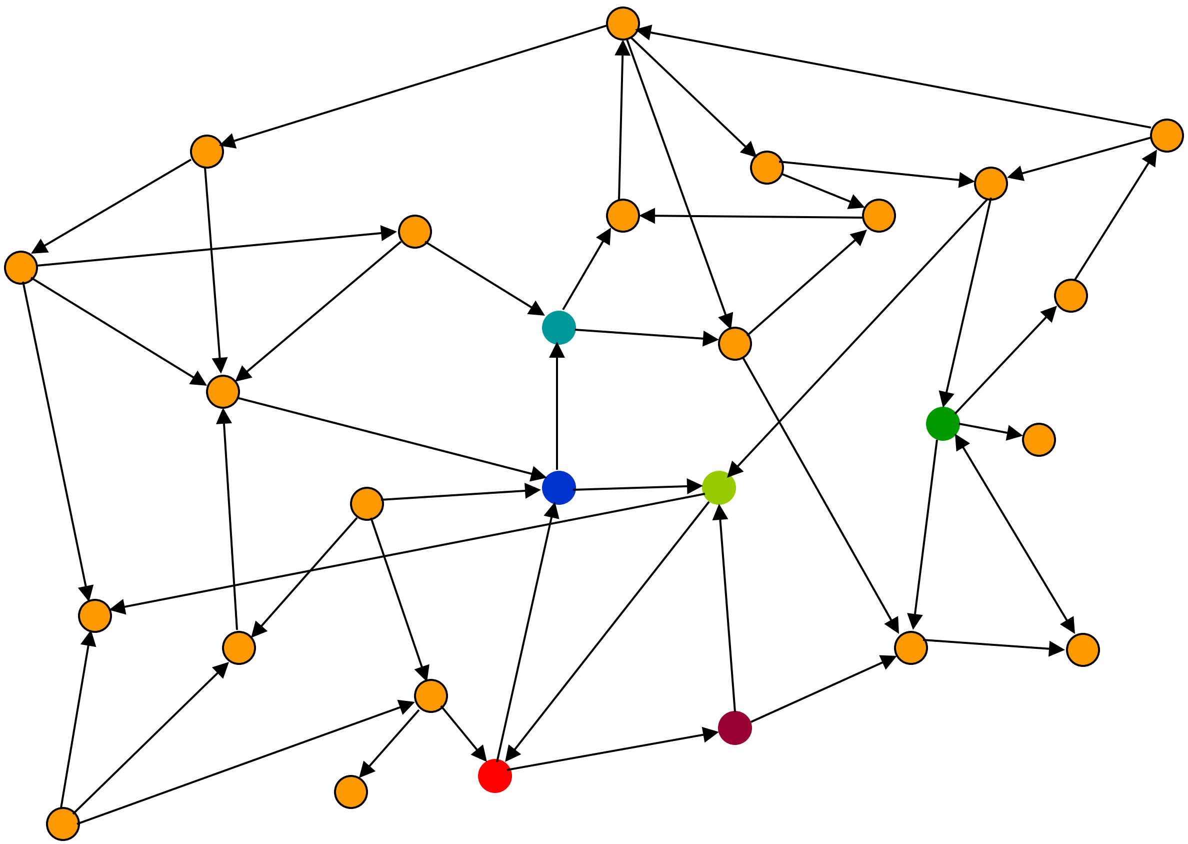


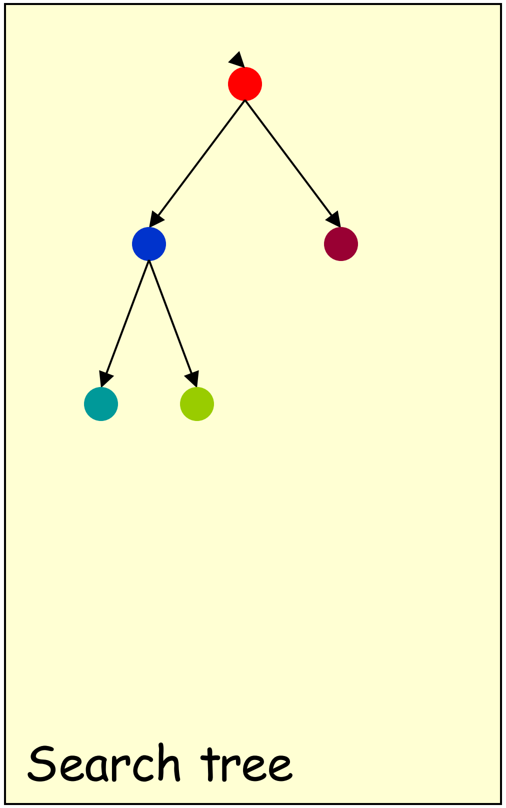
Searching the State Space



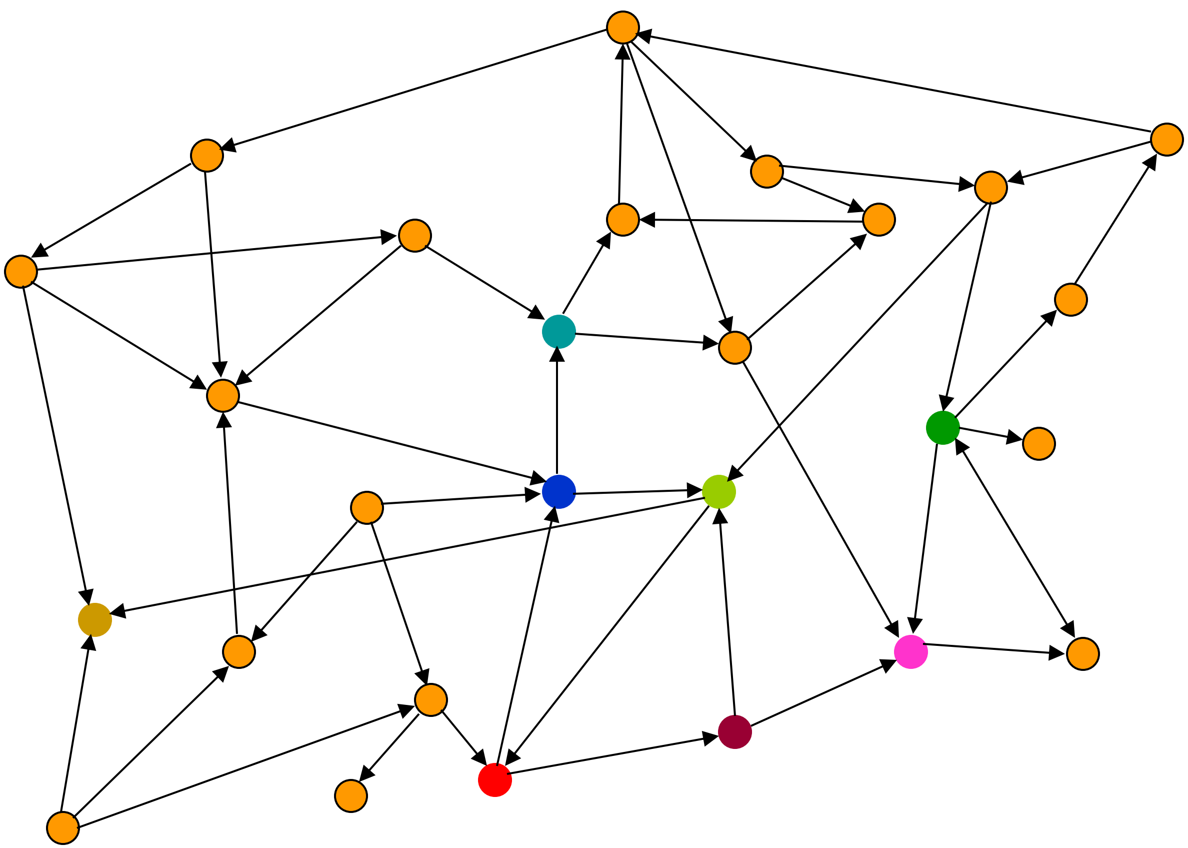


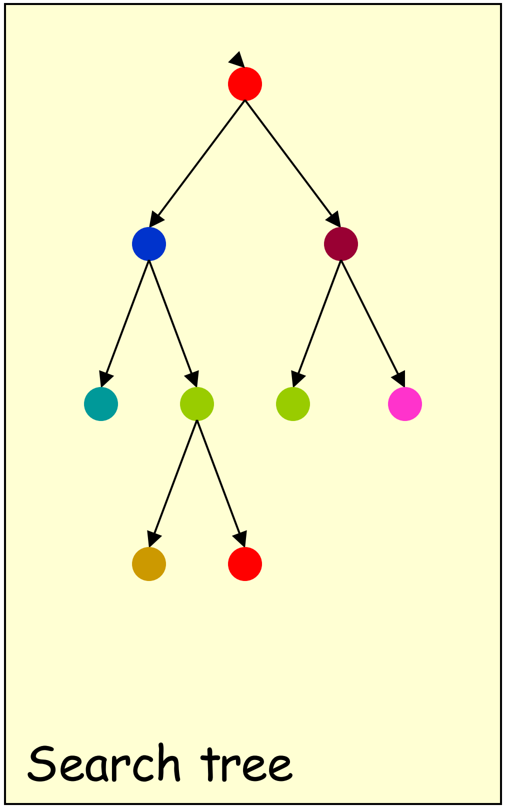
Searching the State Space





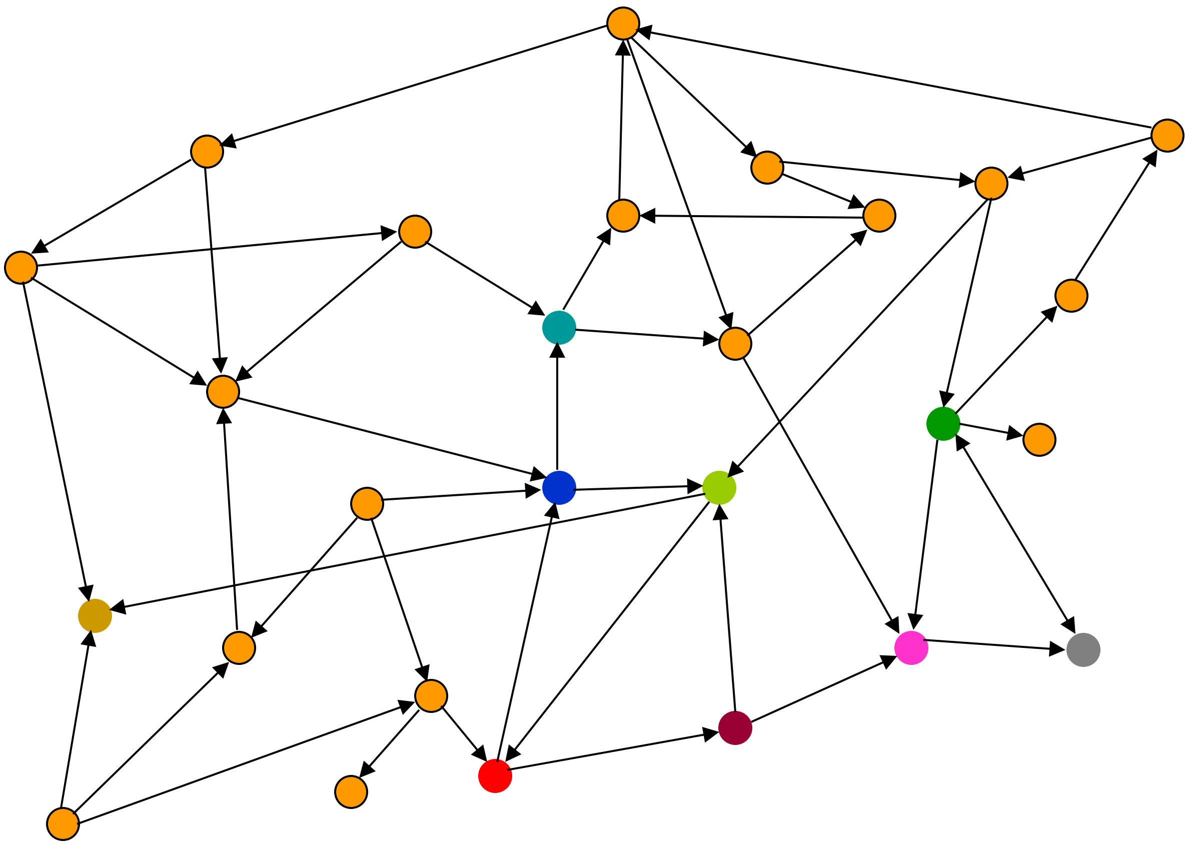
Searching the State Space

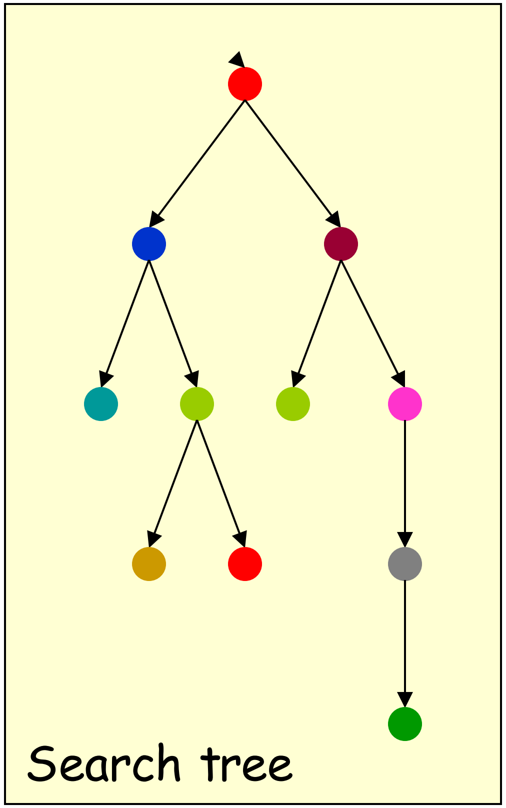




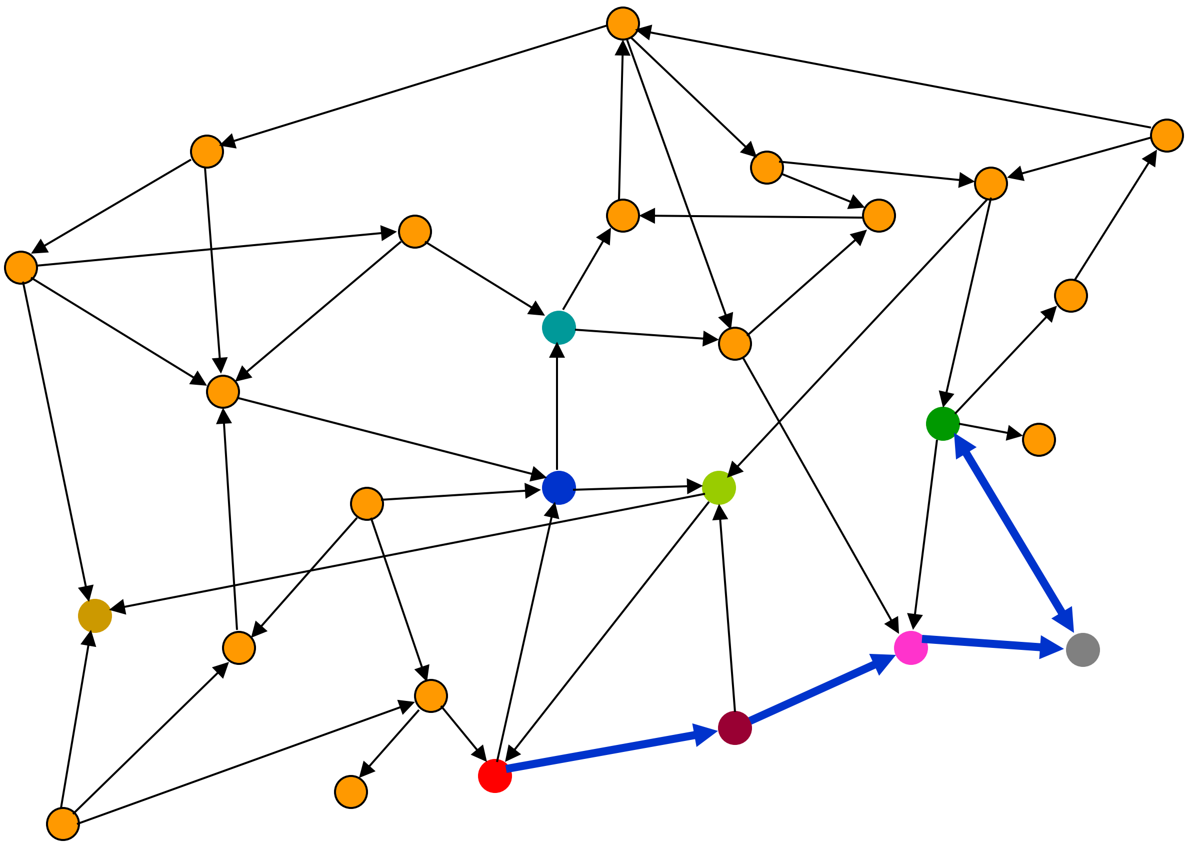
Search tree

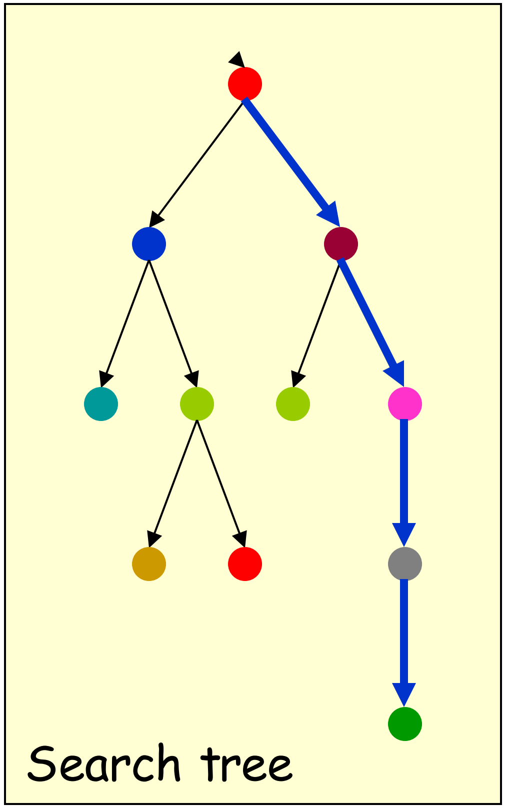
Searching the State Space





Searching the State Space



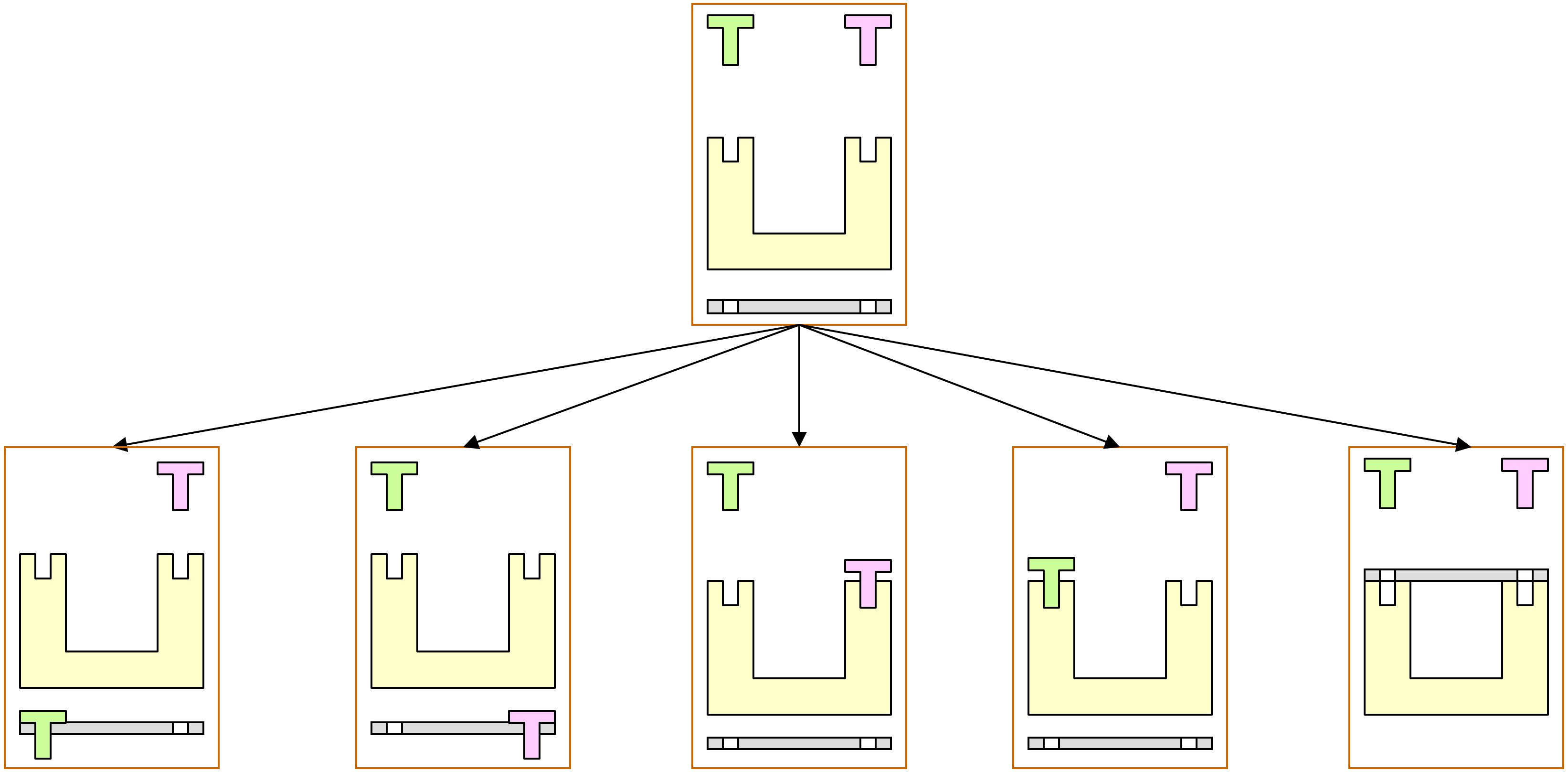


Simple Problem-Solving-Agent Algorithm

1. I ← sense/read initial state
2. GOAL? ← select/read goal test
3. Succ ← select/read successor function
4. solution ← **search**(I, GOAL?, Succ)
5. perform(solution)

Successor Function

- It implicitly represents all the actions that are feasible in each state



Successor Function

- It implicitly represents all the actions that are feasible in each state
- Only the results of the actions (the successor states) and their costs are returned by the function
- The successor function is a “black box”: its content is unknown

Path Cost

- An arc cost is a positive number measuring the “cost” of performing the action corresponding to the arc, e.g.:
  - 1 in the 8-puzzle example
- We will assume that for any given problem the cost c of an arc always verifies: $c \geq \varepsilon > 0$, where ε is a constant

Goal State

- It may be explicitly described:

1	2	3
4	5	6
7	8	

- or partially described:

1	a	a
a	5	a
a	8	a

("a" stands for "any" other than 1, 5, and 8)

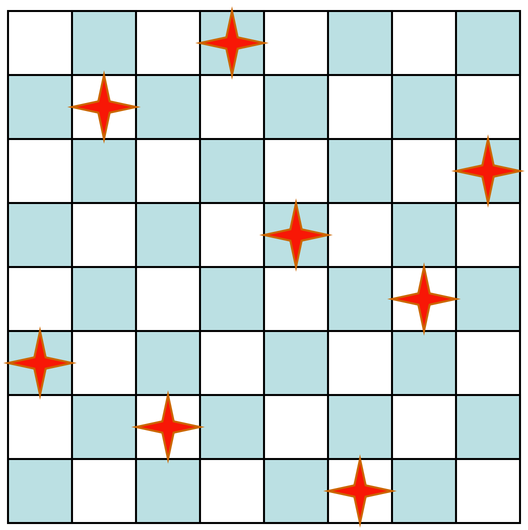
- or defined by a condition,
  e.g., the sum of every row, of every column, and of every diagonal equals 30

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

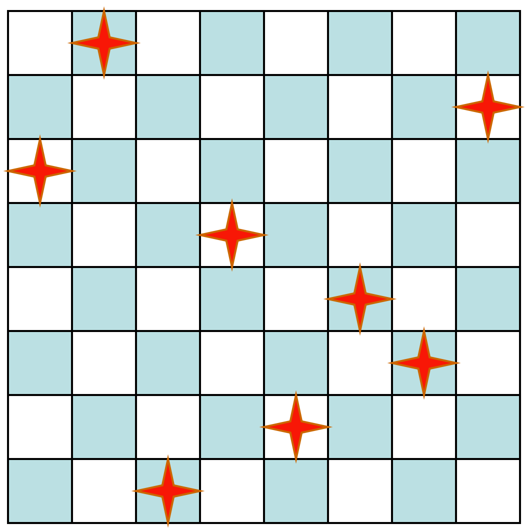
Other examples

8-Queens Problem

Place 8 queens in a chessboard so that no two queens are in the same row, column, or diagonal.

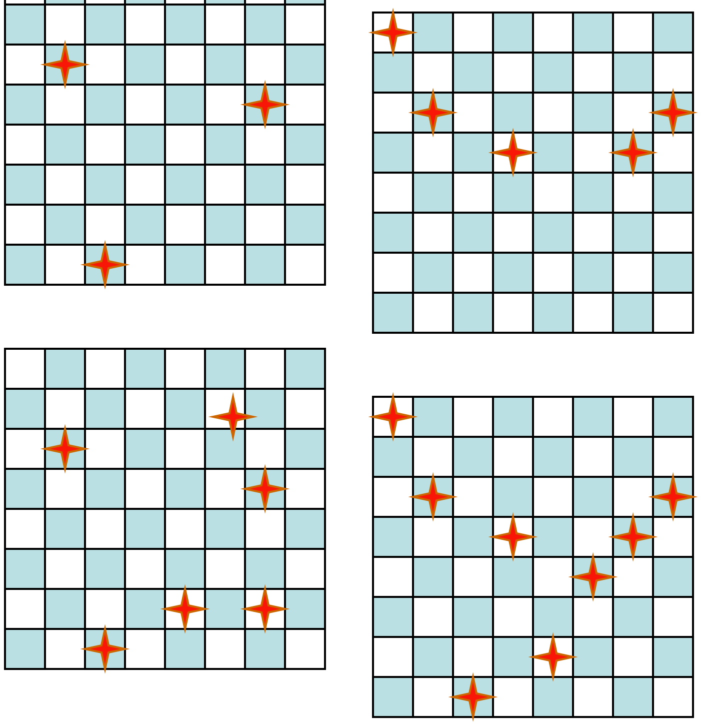


A solution



Not a solution

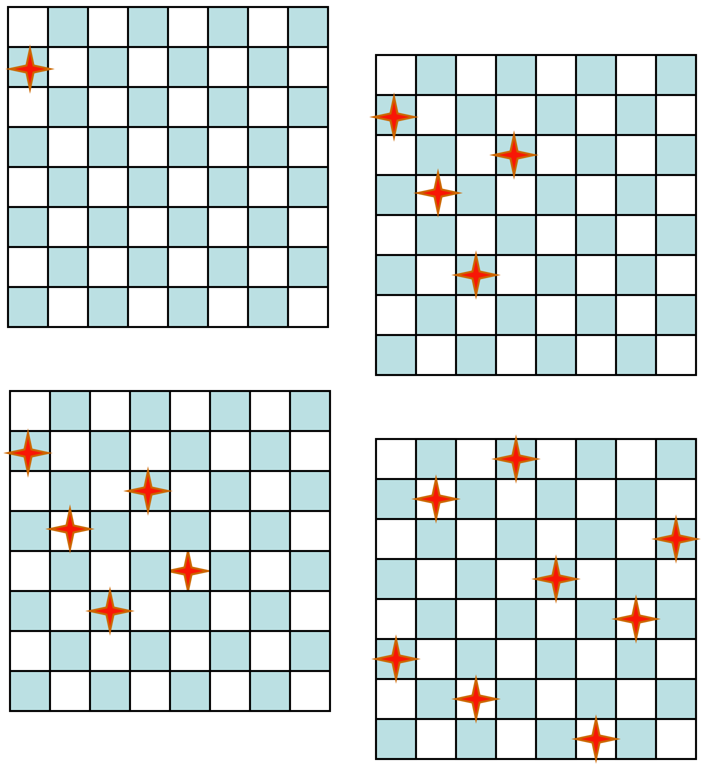
Formulation #1



- **States:** all arrangements of 0, 1, 2, ..., 8 queens on the board
- **Initial state:** 0 queens on the board
- **Successor function:** each of the successors is obtained by adding one queen in an empty square
- **Arc cost:** irrelevant
- **Goal test:** 8 queens are on the board, with no queens attacking each other

→ ~ 64x63x…x57 ~ 3×10^{14} states

Formulation #2



→ 2,057 states

- **States:** all arrangements of k = 0, 1, 2, ..., 8 queens in the k leftmost columns with no two queens attacking each other
- **Initial state:** 0 queens on the board
- **Successor function:** 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
- **Arc cost:** irrelevant
- **Goal test:** 8 queens are on the board

n-Queens Problem

- A solution is a goal node, not a path to this node (typical of design problem)
- Number of states in state space:
  - 8-queens → 2,057
  - 100-queens → 10^{52}
- But techniques exist to solve n-queens problems efficiently for large values of n

  They exploit the fact that there are many solutions well distributed in the state space

Assumptions in Basic Search

- The world is static
- The world is discretized
- The world is observable
- The actions are deterministic

But many of these assumptions can be removed, and search still remains an important problem-solving tool