Lecture notes for *Principles of Artificial Intelligence* (COMS 4720/5720)
Yan-Bin Jia, Iowa State University

Problem Solving

Outline

I. Problem formulation

II. Example problems

* Figures are from the textbook site (or drawn by the instructor) unless the source is specifically cited.

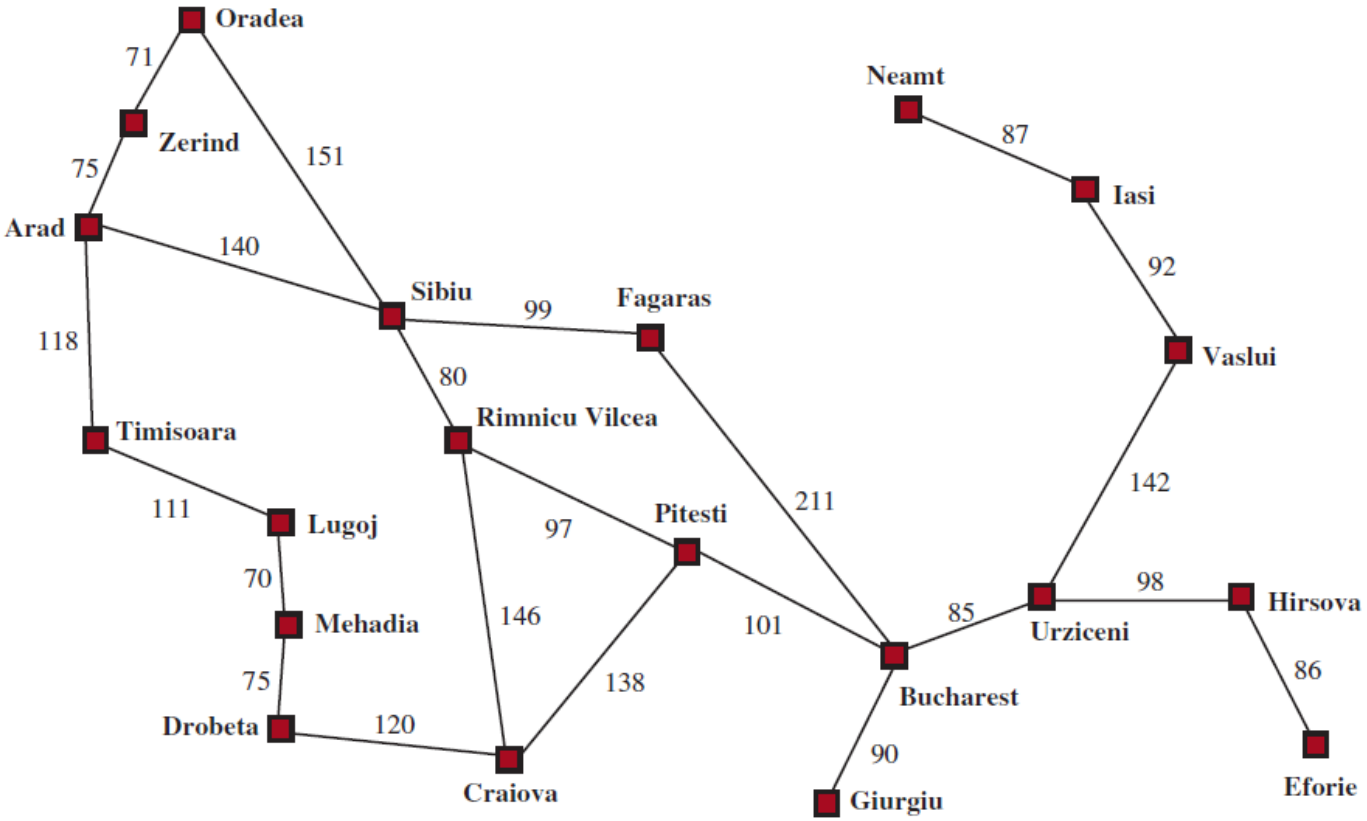
I. Problem-Solving Agent

♣ A reflex agent cannot operate when the mapping from states to to actions becomes too large to store.

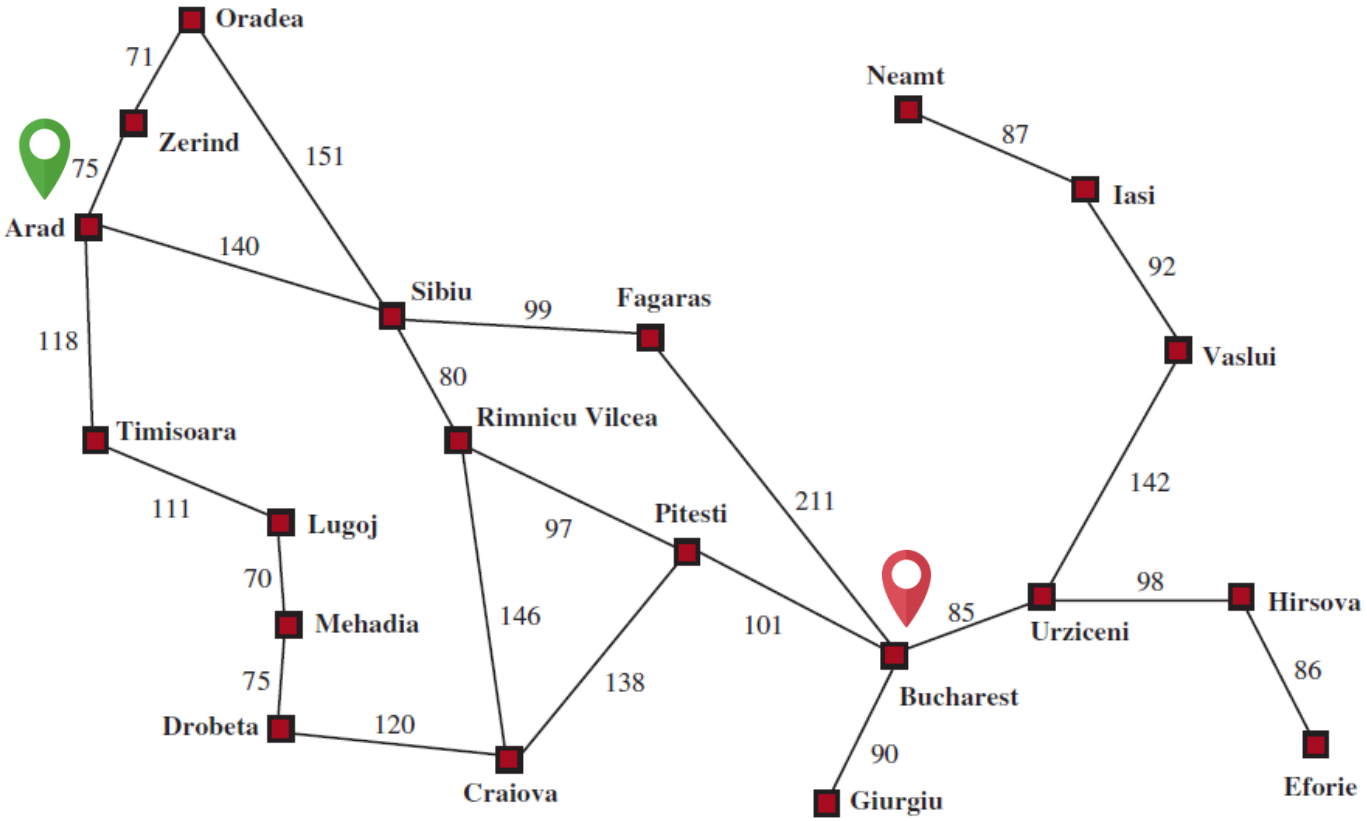
```
if current percepts then action
```

- A problem-solving agent is one kind of *goal-based* agent:
  - ♦ It considers states with no internal structure, i.e., in *atomic* representations.
- Problems are solved by general-purpose *search* algorithms.

Sightseeing Trip in Romania

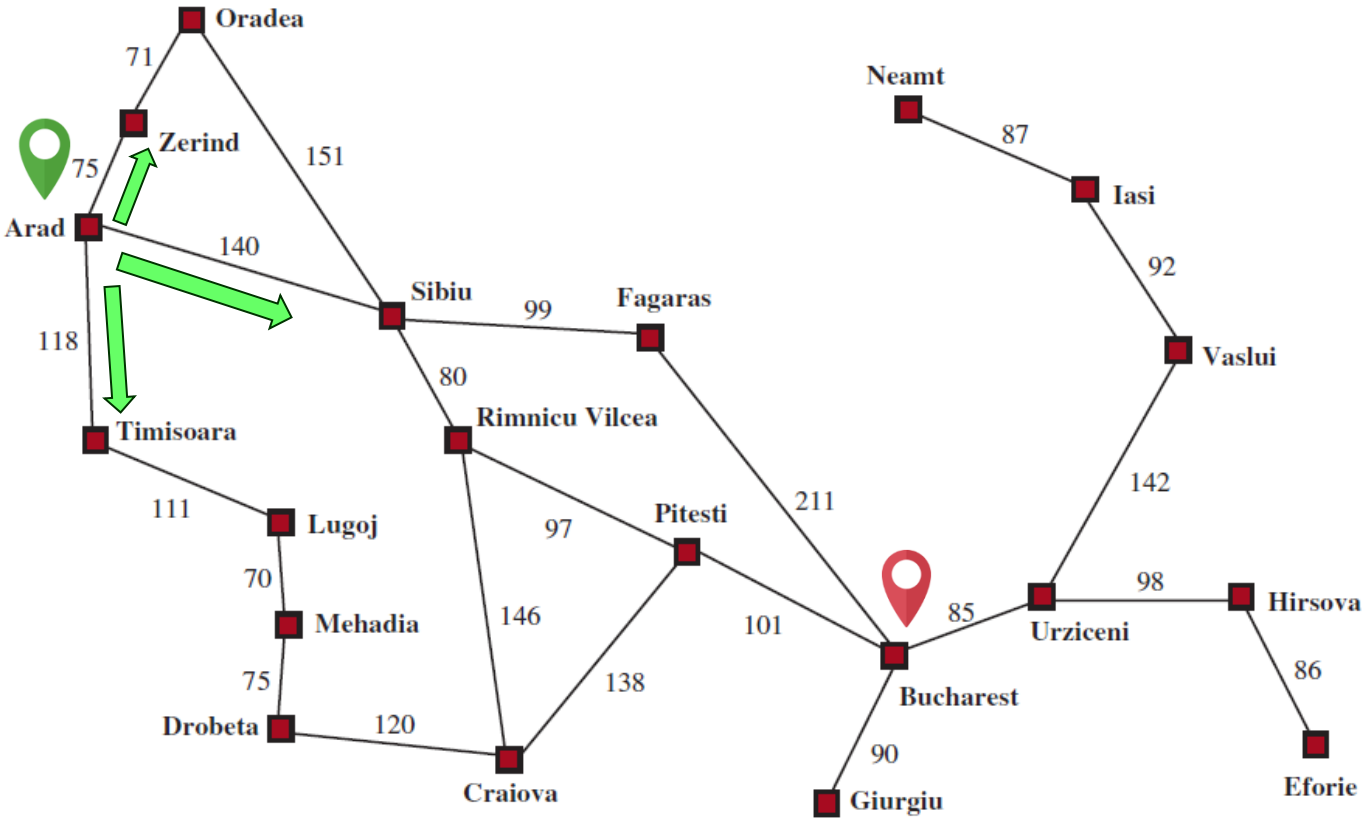


Sightseeing Trip in Romania



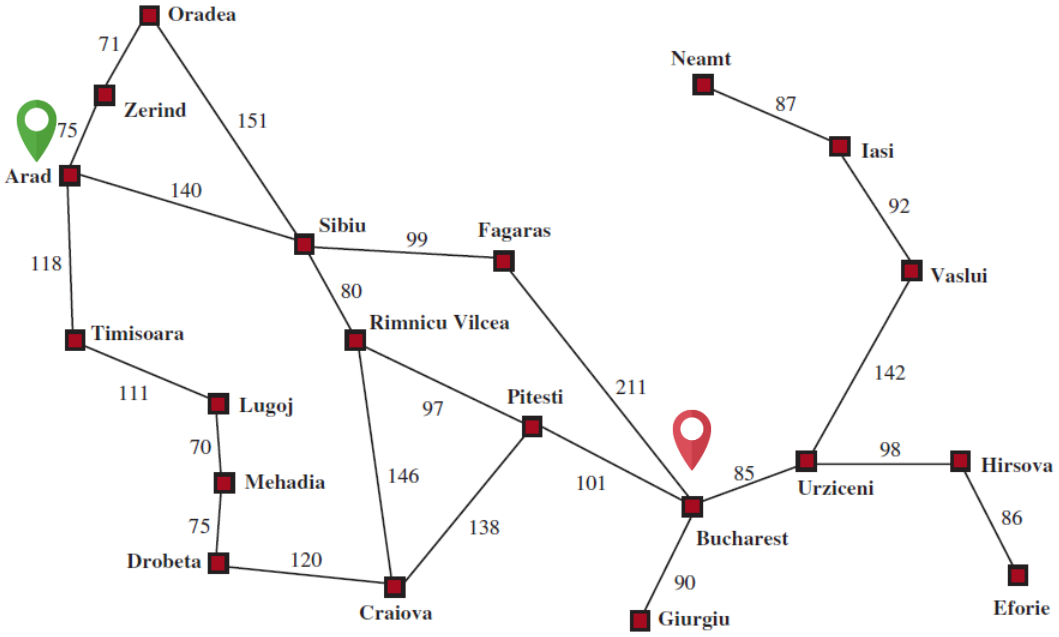
How to get to Bucharest from Arad?

Sightseeing Trip in Romania



How to get to Bucharest from Arad?

Four-Phase Problem Solving



- Goal formulation
- Problem formulation

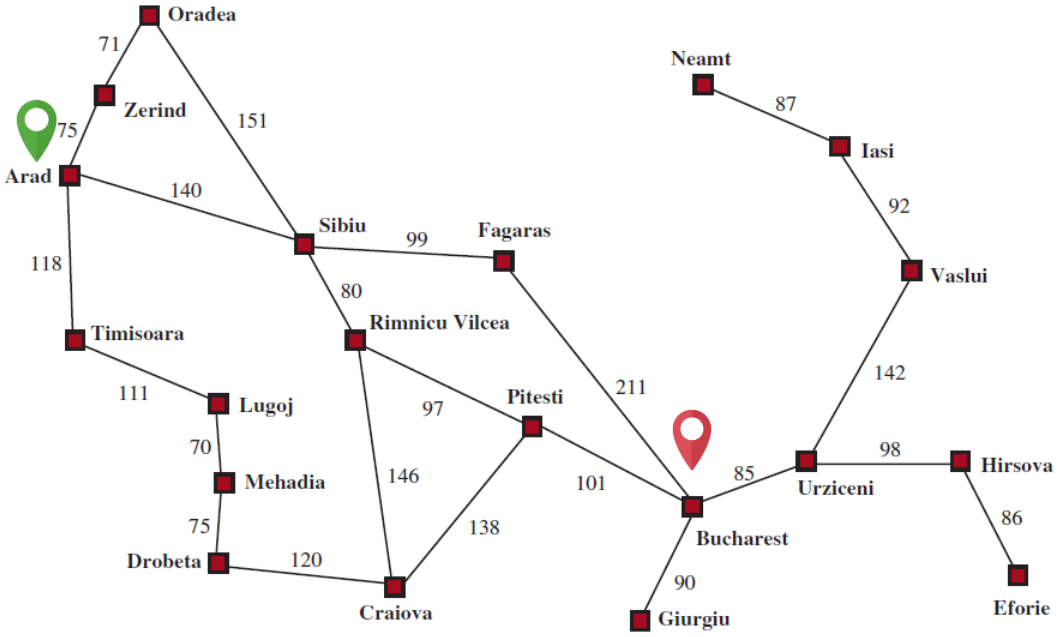
  states: cities

  action: travel from one city to an adjacent city
- Search

  Find a solution.

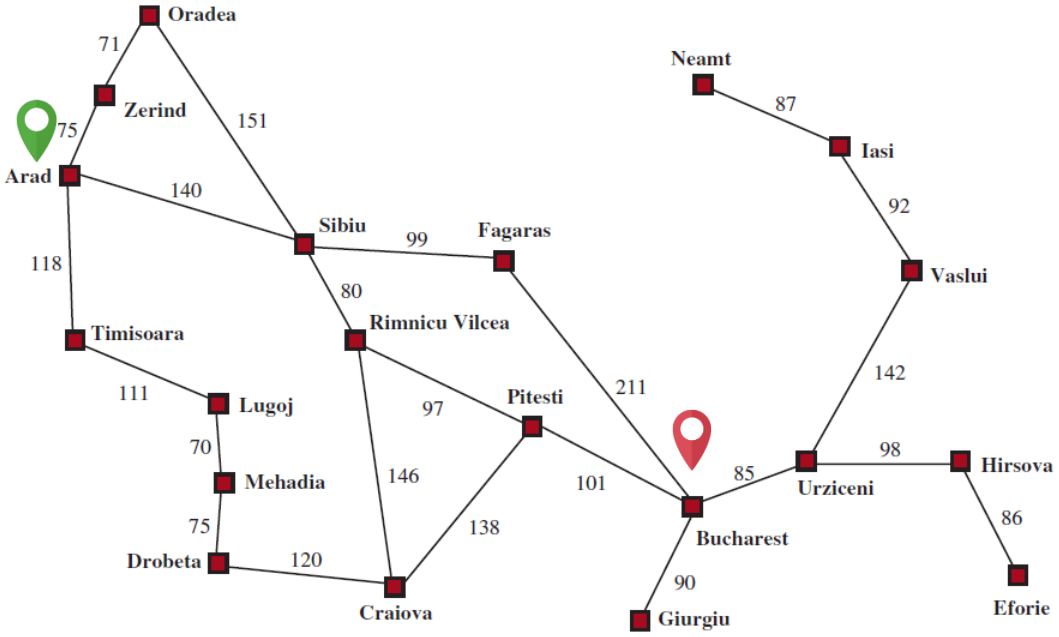
  A sequence of actions to reach the goal
- Execution

Search Problem



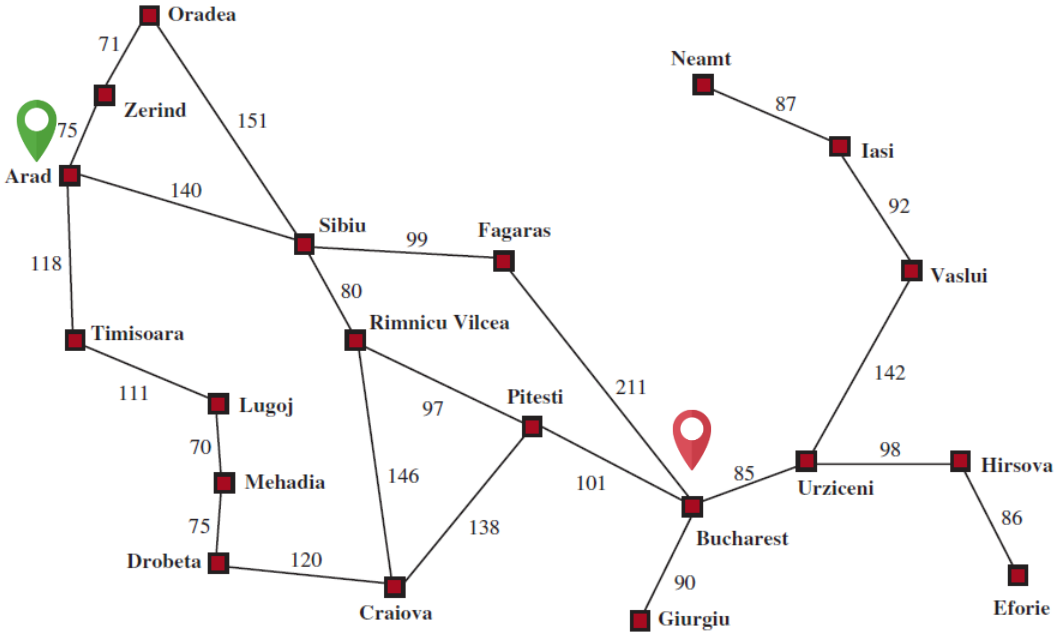
- State space (as a graph)

Search Problem



- State space (as a graph)
- Initial state (e.g., Arad)

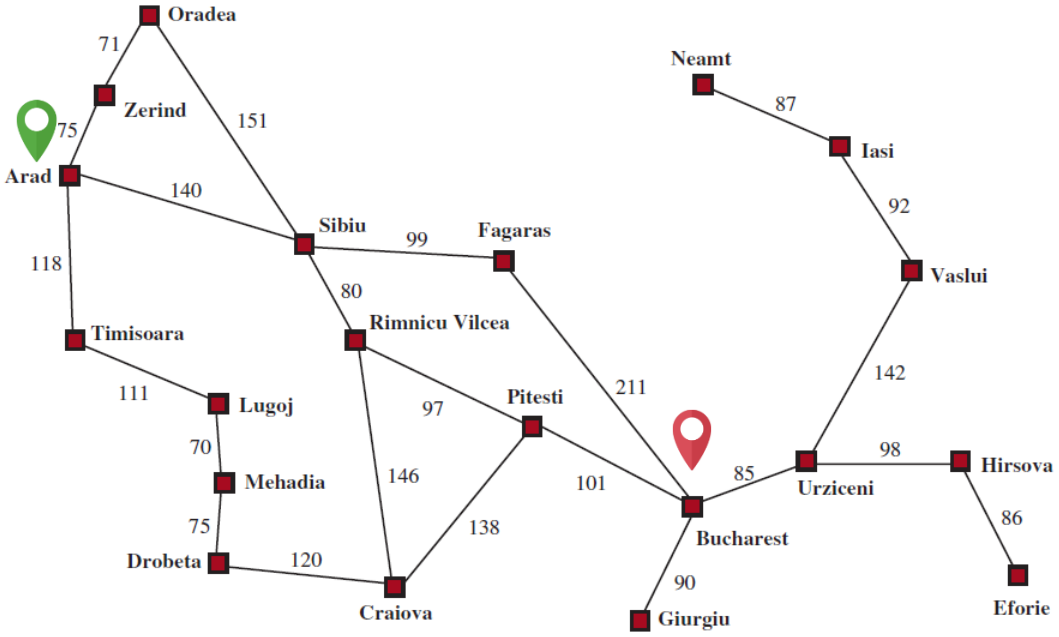
Search Problem



- State space (as a graph)
- Initial state (e.g., Arad)
- Goal state(s) (e.g., Bucharest)

```
IS-GOAL(Fagaras)
```

Search Problem

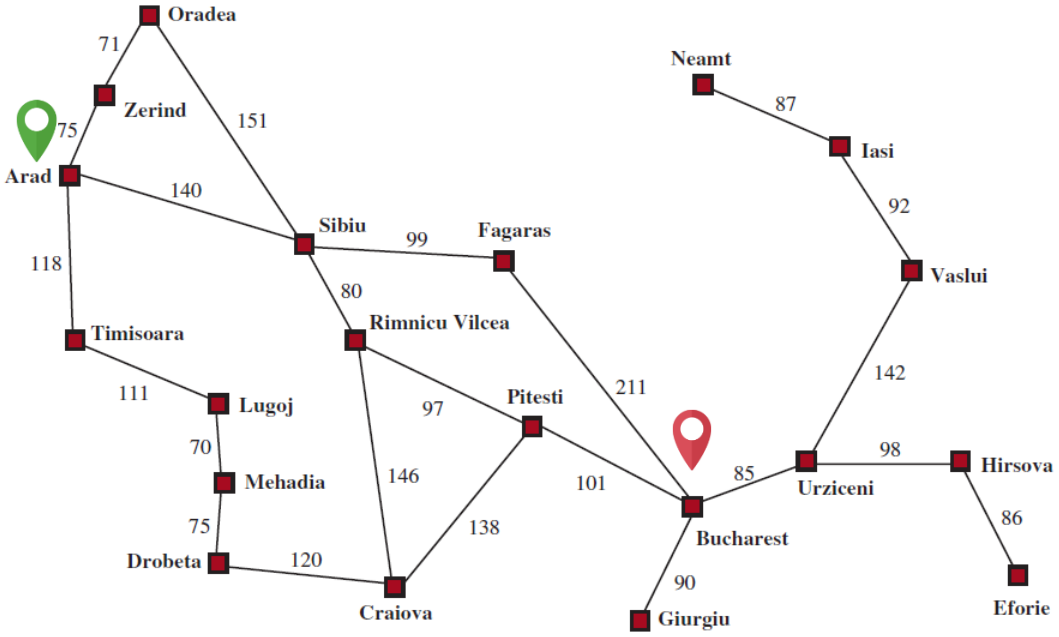


- State space (as a graph)
- Initial state (e.g., Arad)
- Goal state(s) (e.g., Bucharest)

```
IS-GOAL(Fagaras)
```

- Actions

Search Problem



- State space (as a graph)
- Initial state (e.g., Arad)
- Goal state(s) (e.g., Bucharest)

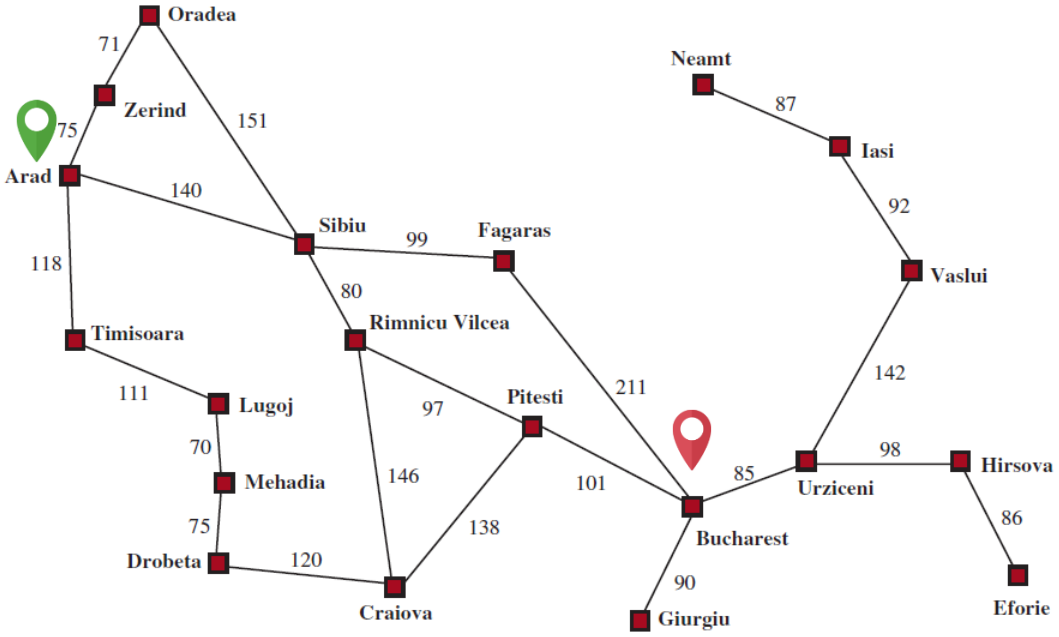
```
IS-GOAL(Fagaras)
```

- Actions

ACTIONS(s): a finite set of actions executable at state s.

ACTIONS($Arad$) = $\{ToSibiu, ToTimisoara, ToZerind\}$

Search Problem



- State space (as a graph)
- Initial state (e.g., Arad)
- Goal state(s) (e.g., Bucharest)

```
IS-GOAL(Fagaras)
```

- Actions

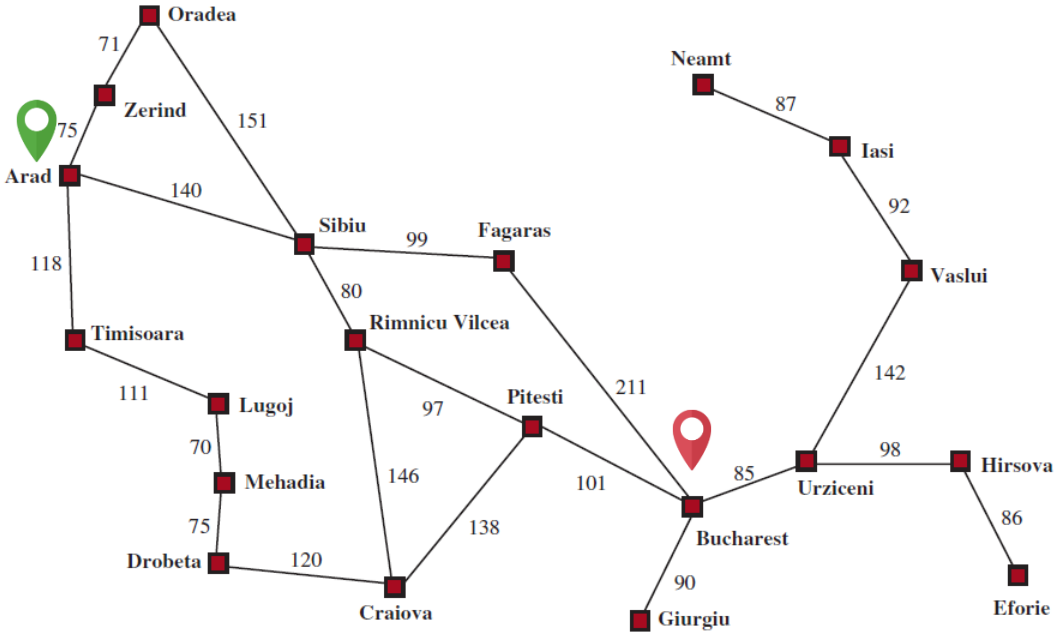
ACTIONS(s): a finite set of actions executable at state s.

ACTIONS($Arad$) = $\{ToSibiu, ToTimisoara, ToZerind\}$

- Transition model

RESULT($Arad, ToZerind$) = $Zerind$

Search Problem



- State space (as a graph)
- Initial state (e.g., Arad)
- Goal state(s) (e.g., Bucharest)

```
IS-GOAL(Fagaras)
```

- Actions

ACTIONS(s): a finite set of actions executable at state s.

ACTIONS($Arad$) = $\{ToSibiu, ToTimisoara, ToZerind\}$

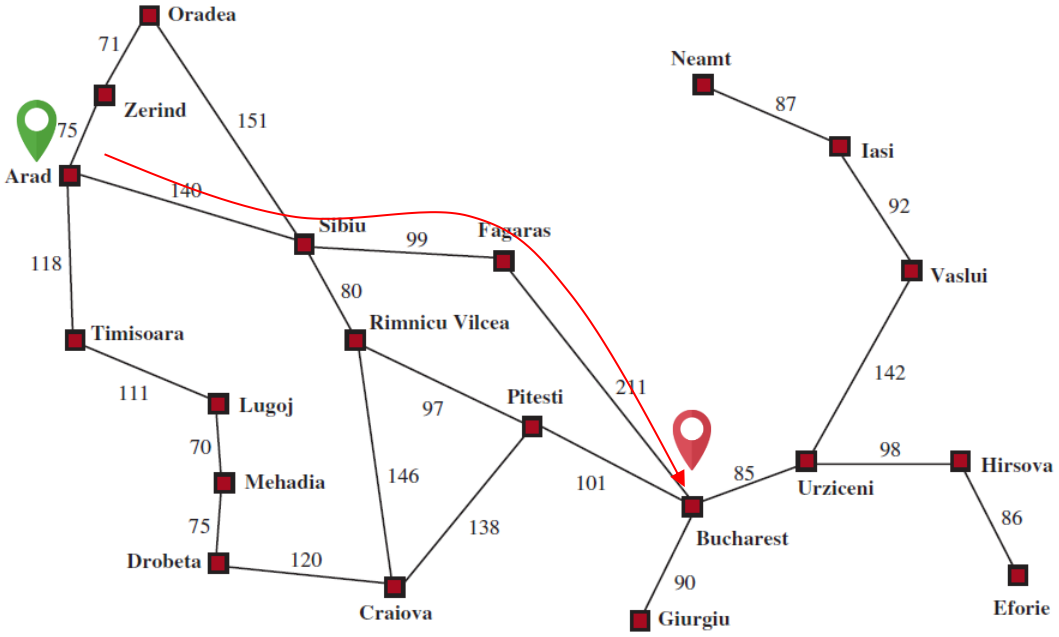
- Transition model

RESULT($Arad, ToZerind$) = $Zerind$

- Action cost function

$c(s, a, s')$: cost of applying action a in state s to reach state s'.

Search Problem



- State space (as a graph)
- Initial state (e.g., Arad)
- Goal state(s) (e.g., Bucharest)

```
IS-GOAL(Fagaras)
```

- Actions

ACTIONS(s): a finite set of actions executable at state s.

ACTIONS($Arad$) = $\{ToSibiu, ToTimisoara, ToZerind\}$

- Transition model

RESULT($Arad, ToZerind$) = $Zerind$

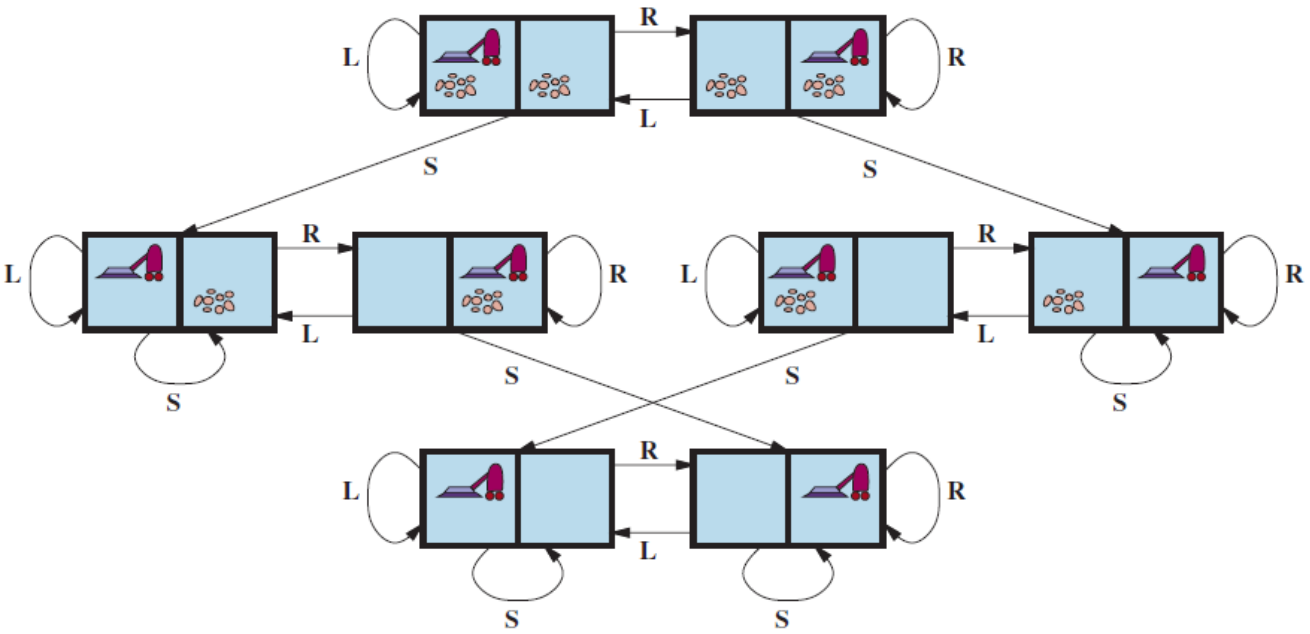
- Action cost function

$c(s, a, s')$: cost of applying action a in state s to reach state s'.

- Solution: initial state ⇝ goal state (e.g., Arad – Sibiu – Fagaras – Bucharest)

II. Example 1: Vacuum World

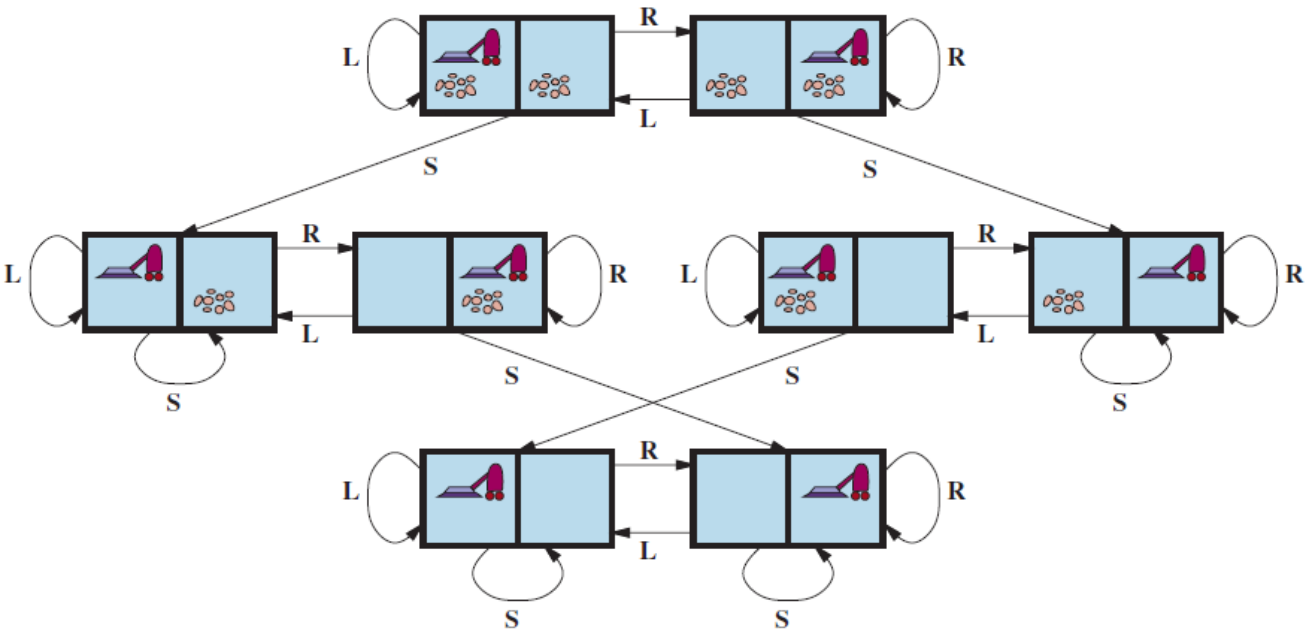
State-space graph:



- Actions: *Suck*, *Left*, *Right*
- Goal: every cell is clean.

II. Example 1: Vacuum World

State-space graph:



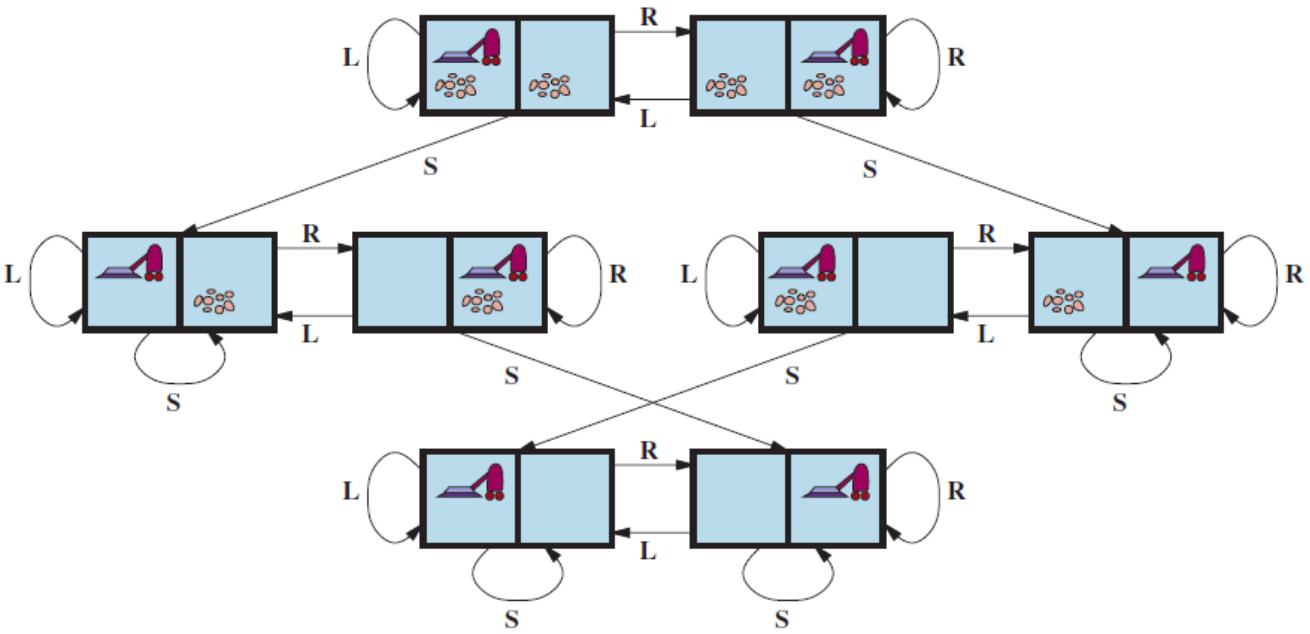
Agent in either cell

2

- Actions: *Suck*, *Left*, *Right*
- Goal: every cell is clean.

II. Example 1: Vacuum World

State-space graph:



Agent in either cell

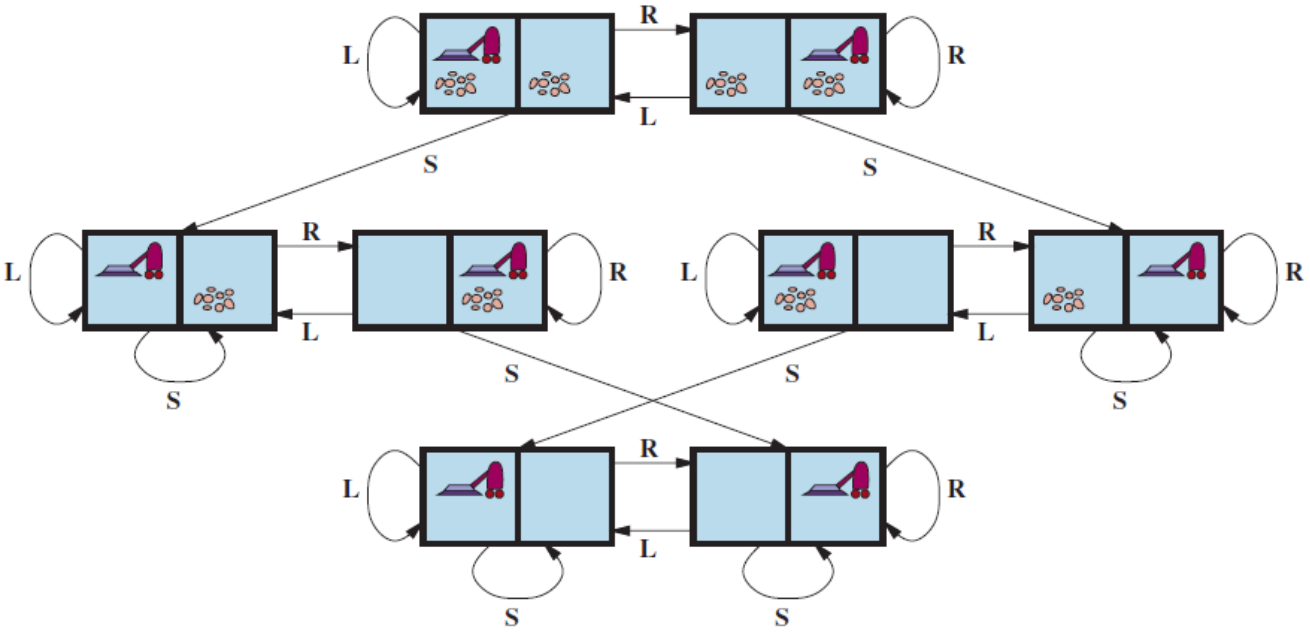
$2 \cdot 2$

Left cell has dirt or not

- Actions: *Suck*, *Left*, *Right*
- Goal: every cell is clean.

II. Example 1: Vacuum World

State-space graph:



- Actions: *Suck*, *Left*, *Right*
- Goal: every cell is clean.

Agent in either cell

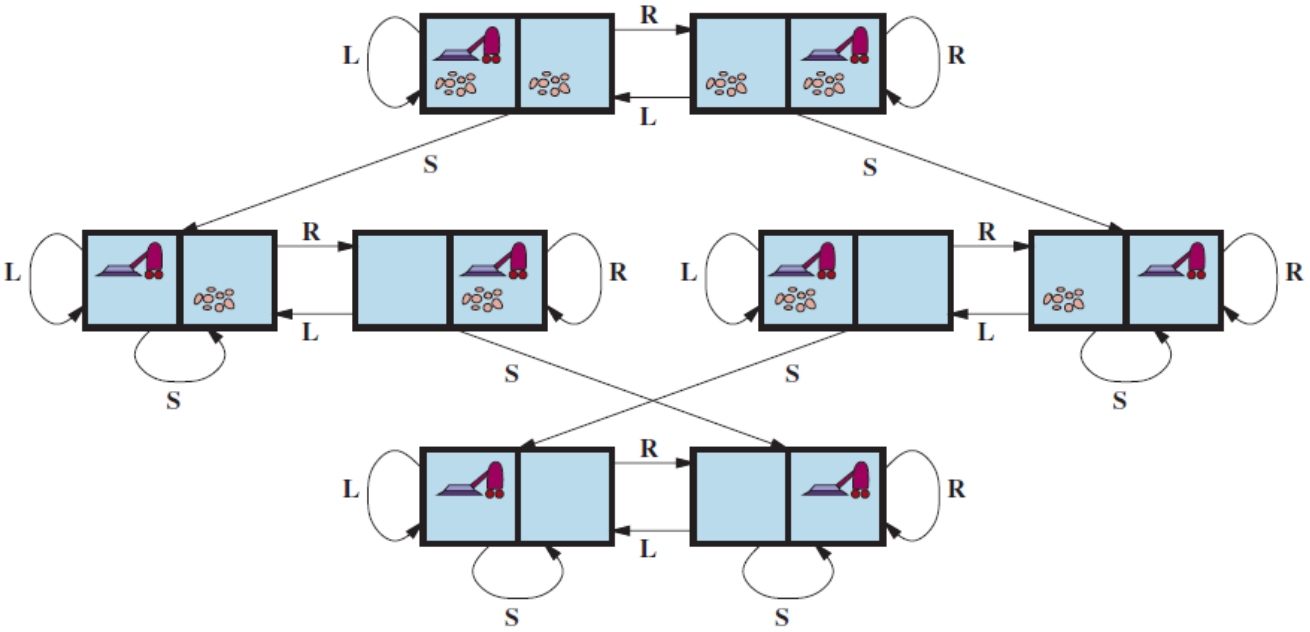
$2 \cdot 2 \cdot 2 = 8$ states

Left cell has dirt or not

Right cell has dirt or not

II. Example 1: Vacuum World

State-space graph:



Agent in either cell

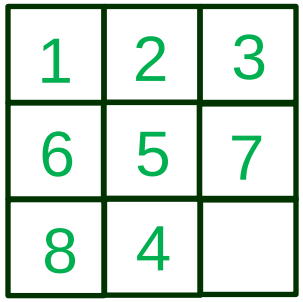
$$2 \cdot 2 \cdot 2 = 8 \text{ states}$$

Left cell has dirt or not

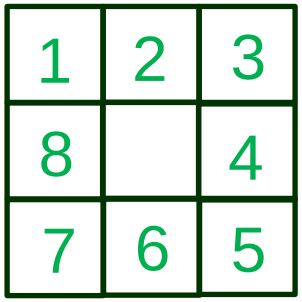
Right cell has dirt or not

- Actions: *Suck*, *Left*, *Right*
- Goal: every cell is clean.
- Cost: 1 for each action

Example 2: 8-Puzzle



Initial state



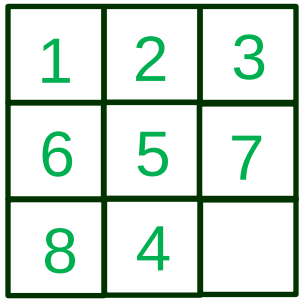
Goal state

- Actions: ways of sliding a tile (adjacent to the blank space).

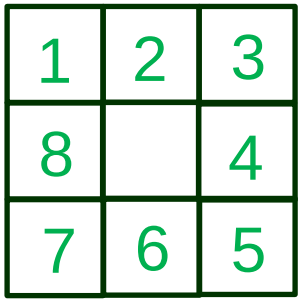
Left, Right, Up, Down

Applied to the
right neighbor
of the blank space

Example 2: 8-Puzzle



Initial state



Goal state

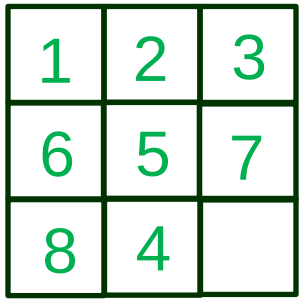
- Actions: ways of sliding a tile (adjacent to the blank space).

Left, Right, Up, Down

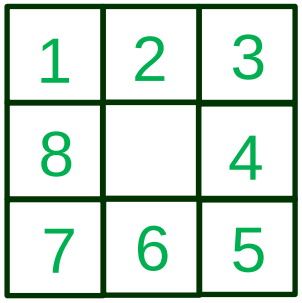
Applied to the right neighbor of the blank space

Only two possible actions at the initial state: *Right* and *Down.*

Example 2: 8-Puzzle



Initial state



Goal state

- Actions: ways of sliding a tile (adjacent to the blank space).

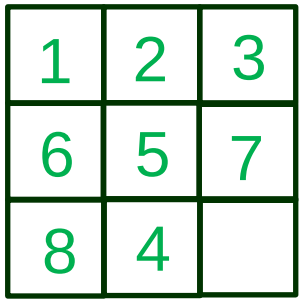
Left, *Right*, *Up*, *Down*

Applied to the right neighbor of the blank space

Only two possible actions at the initial state: *Right* and *Down.*

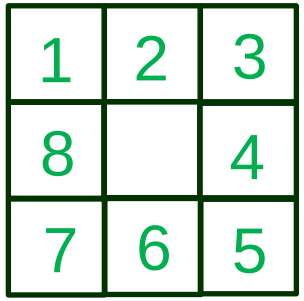
Cost 1 for each action.

Solution to an 8-Puzzle

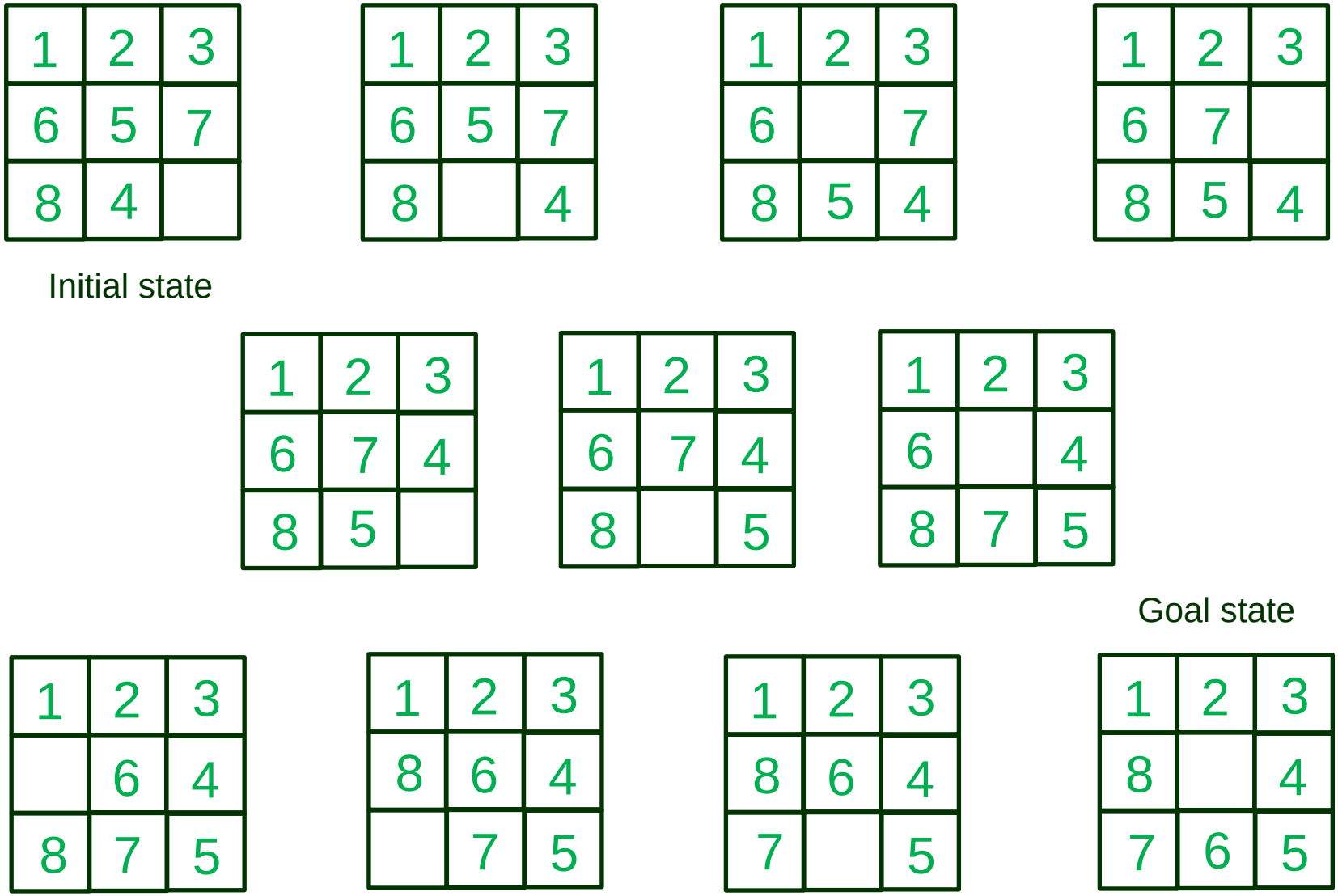


Initial state

Goal state

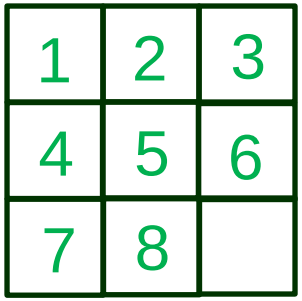


Solution to an 8-Puzzle



Inversions in an 8-Puzzle

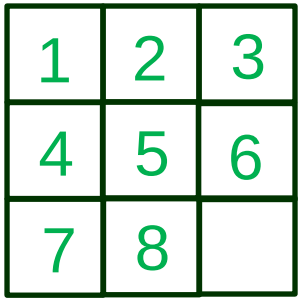
State in which all the tiles appear in the correct order.



Every tile with a smaller number must be either above or, if in the same row, to the left of any tile with a larger number.

Inversions in an 8-Puzzle

State in which all the tiles appear in the correct order.

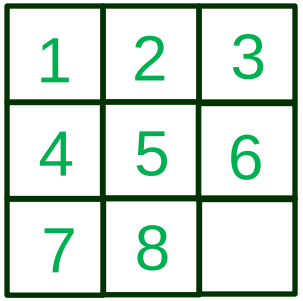


Every tile with a smaller number must be either above or, if in the same row, to the left of any tile with a larger number.

Every violation of this order is called an *inversion*.

Inversions in an 8-Puzzle

State in which all the tiles appear in the correct order.



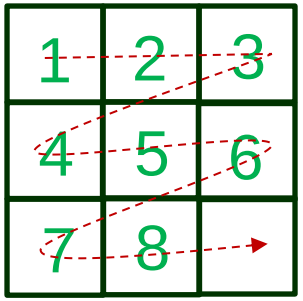
Every tile with a smaller number must be either above or, if in the same row, to the left of any tile with a larger number.

Every violation of this order is called an *inversion*.

- An inversion is caused by a larger number appearing either above a smaller number or to the left of a smaller number in the same row.

Inversions in an 8-Puzzle

State in which all the tiles appear in the correct order.



Every tile with a smaller number must be either above or, if in the same row, to the left of any tile with a larger number.

Every violation of this order is called an *inversion*.

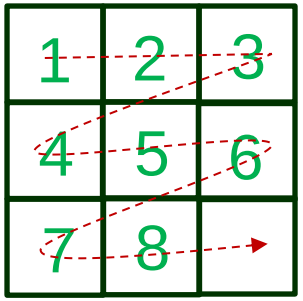
- An inversion is caused by a larger number appearing either above a smaller number or to the left of a smaller number in the same row.

Starting from the top, we scan the tiles row by row, and within each row, from left to right. Ignore the blank tile.

Inversions in an 8-Puzzle

State in which all the tiles appear in the correct order.

0 inversion



Every tile with a smaller number must be either above or, if in the same row, to the left of any tile with a larger number.

Every violation of this order is called an *inversion*.

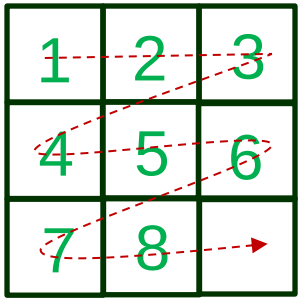
- An inversion is caused by a larger number appearing either above a smaller number or to the left of a smaller number in the same row.

Starting from the top, we scan the tiles row by row, and within each row, from left to right. Ignore the blank tile.

Inversions in an 8-Puzzle

State in which all the tiles appear in the correct order.

0 inversion



Every tile with a smaller number must be either above or, if in the same row, to the left of any tile with a larger number.

Every violation of this order is called an *inversion*.

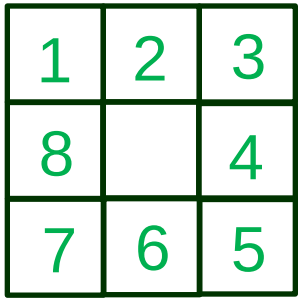
- An inversion is caused by a larger number appearing either above a smaller number or to the left of a smaller number in the same row.

Starting from the top, we scan the tiles row by row, and within each row, from left to right. Ignore the blank tile.

- In the resulting sequence, every instance of a number appearing before a smaller one is counted as one inversion.

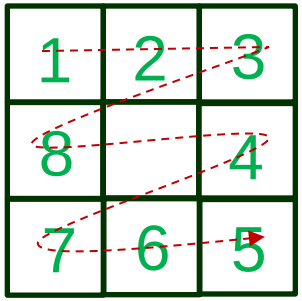
Counting Inversions

How many inversions are in the goal state?



Counting Inversions

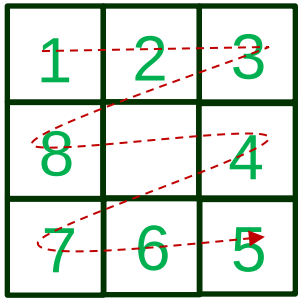
How many inversions are in the goal state?



Sequence from the scan: 1, 2, 3, 8, 4, 7, 6, 5.

Counting Inversions

How many inversions are in the goal state?

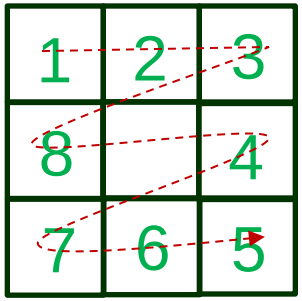


Sequence from the scan: 1, 2, 3, 8, 4, 7, 6, 5.

6 comes before 5
7 comes before 5
7 comes before 6
8 comes before 4
8 comes before 5
8 comes before 6
8 comes before 7

Counting Inversions

How many inversions are in the goal state?



Sequence from the scan: 1, 2, 3, 8, 4, 7, 6, 5.

6 comes before 5
7 comes before 5
7 comes before 6
8 comes before 4
8 comes before 5
8 comes before 6
8 comes before 7

7 inversions!

Solvability of an 8-Puzzle

Theorem An 8-puzzle with an initial state and a goal state is solvable if and only if the two states differ by an ***even*** number of inversions

Solvability of an 8-Puzzle

Theorem An 8-puzzle with an initial state and a goal state is solvable if and only if the two states differ by an ***even*** number of inversions

If we fix the goal state at

1	2	3
8		4
7	6	5

(7 inversions)

the puzzle is solvable if and only if the initial state has an *odd* number of inversions.

Solvability of an 8-Puzzle

Theorem An 8-puzzle with an initial state and a goal state is solvable if and only if the two states differ by an ***even*** number of inversions

If we fix the goal state at

1	2	3
8		4
7	6	5

(7 inversions)

the puzzle is solvable if and only if the initial state has an *odd* number of inversions.

Initial state 1:

4	1	2
3	5	
8	6	7

Solvability of an 8-Puzzle

Theorem An 8-puzzle with an initial state and a goal state is solvable if and only if the two states differ by an ***even*** number of inversions

If we fix the goal state at

1	2	3
8		4
7	6	5

(7 inversions)

the puzzle is solvable if and only if the initial state has an *odd* number of inversions.

Initial state 1:

4	1	2
3	5	
8	6	7

5 inversions:

4 comes before 1, 2, 3

8 comes before 6, 7

Solvability of an 8-Puzzle

Theorem An 8-puzzle with an initial state and a goal state is solvable if and only if the two states differ by an ***even*** number of inversions

If we fix the goal state at

1	2	3
8		4
7	6	5

(7 inversions)

the puzzle is solvable if and only if the initial state has an *odd* number of inversions.

Initial state 1:

4	1	2
3	5	
8	6	7

5 inversions: ⇨ Solvable

4 comes before 1, 2, 3

8 comes before 6, 7

Solvability of an 8-Puzzle

Theorem An 8-puzzle with an initial state and a goal state is solvable if and only if the two states differ by an ***even*** number of inversions

If we fix the goal state at

1	2	3
8		4
7	6	5

(7 inversions) the puzzle is solvable if and only if the initial state has an *odd* number of inversions.

Initial state 1:

4	1	2
3	5	
8	6	7

5 inversions: ⇨ Solvable

4 comes before 1, 2, 3

8 comes before 6, 7

Initial state 2:

4	1	2
5	3	
8	6	7

Solvability of an 8-Puzzle

Theorem An 8-puzzle with an initial state and a goal state is solvable if and only if the two states differ by an ***even*** number of inversions

If we fix the goal state at

1	2	3
8		4
7	6	5

(7 inversions) the puzzle is solvable if and only if the initial state has an *odd* number of inversions.

Initial state 1:

4	1	2
3	5	
8	6	7

5 inversions: ⇨ Solvable

4 comes before 1, 2, 3

8 comes before 6, 7

Initial state 2:

4	1	2
5	3	
8	6	7

6 inversions:

4 comes before 1, 2, 3

5 comes before 3

8 comes before 6, 7

Solvability of an 8-Puzzle

Theorem An 8-puzzle with an initial state and a goal state is solvable if and only if the two states differ by an ***even*** number of inversions

If we fix the goal state at

1	2	3
8		4
7	6	5

(7 inversions)

the puzzle is solvable if and only if the initial state has an *odd* number of inversions.

Initial state 1:

4	1	2
3	5	
8	6	7

5 inversions: ⇨ Solvable

4 comes before 1, 2, 3

8 comes before 6, 7

Initial state 2:

4	1	2
5	3	
8	6	7

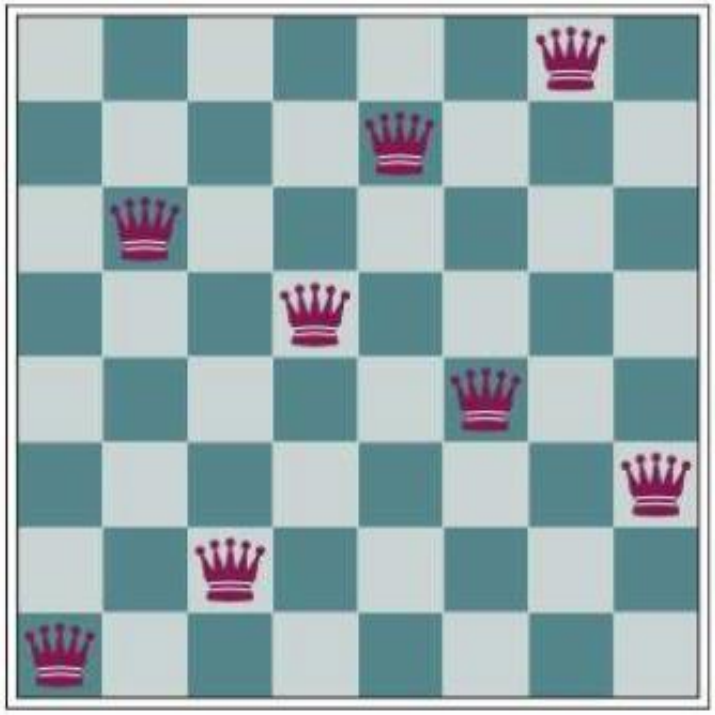
6 inversions: ⇨ Not solvable

4 comes before 1, 2, 3

5 comes before 3

8 comes before 6, 7

The 8-Queens Problem



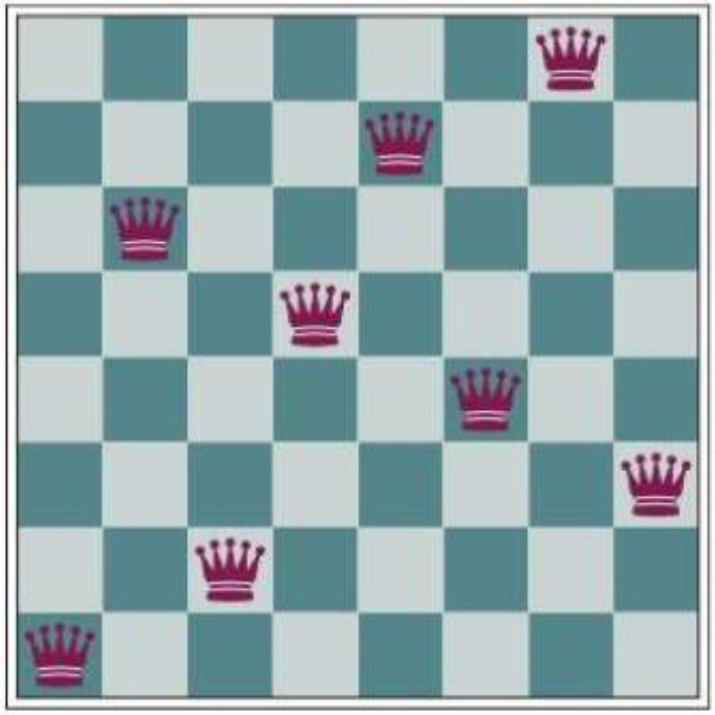
- Goal: A placement of 8 queens on the chess board in which no queen attacks another.

  ↓

  Same row, column, or diagonal

- Initial state: no queen on the board.

The 8-Queens Problem



Close to a solution

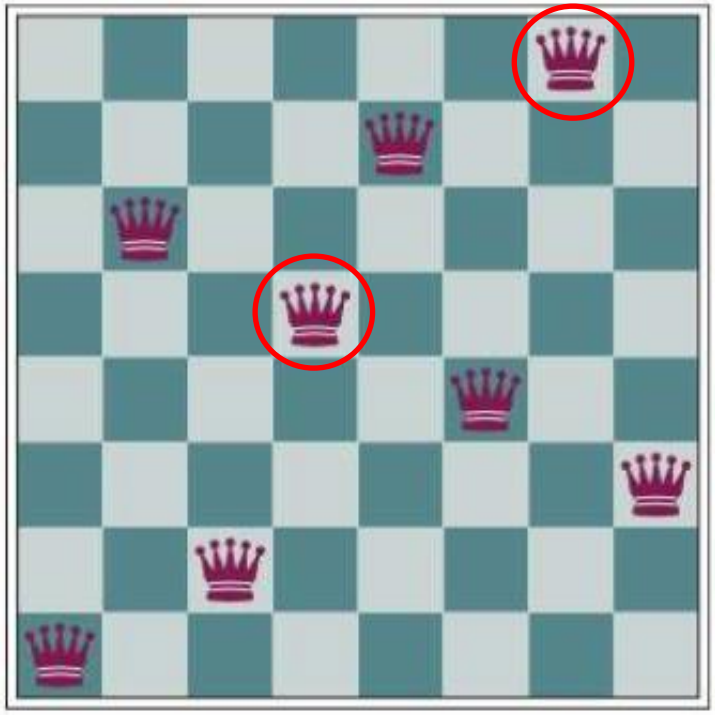
- Goal: A placement of 8 queens on the chess board in which no queen attacks another.

  ↓

  Same row, column, or diagonal

- Initial state: no queen on the board.

The 8-Queens Problem



Close to a solution

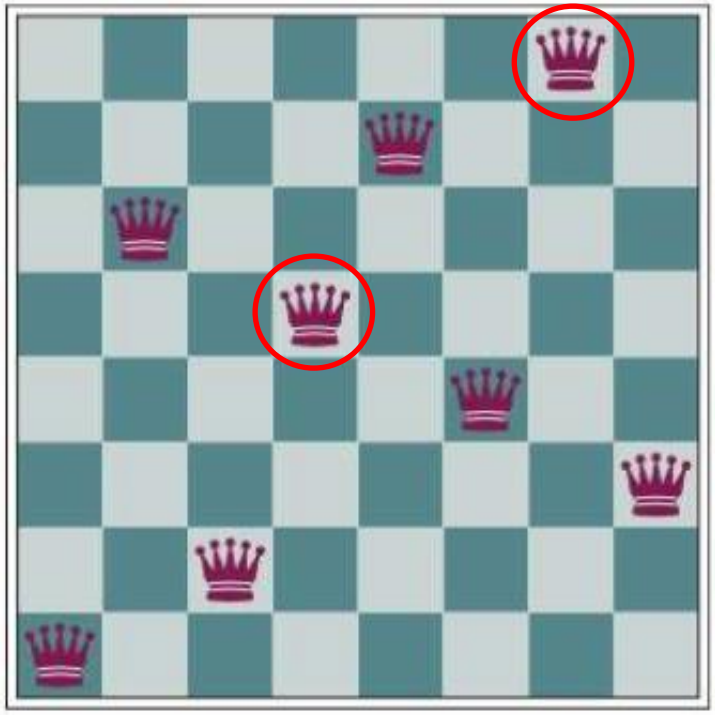
- Goal: A placement of 8 queens on the chess board in which no queen attacks another.

  ↓

  Same row, column, or diagonal

- Initial state: no queen on the board.

The 8-Queens Problem



Close to a solution

- Goal: A placement of 8 queens on the chess board in which no queen attacks another.

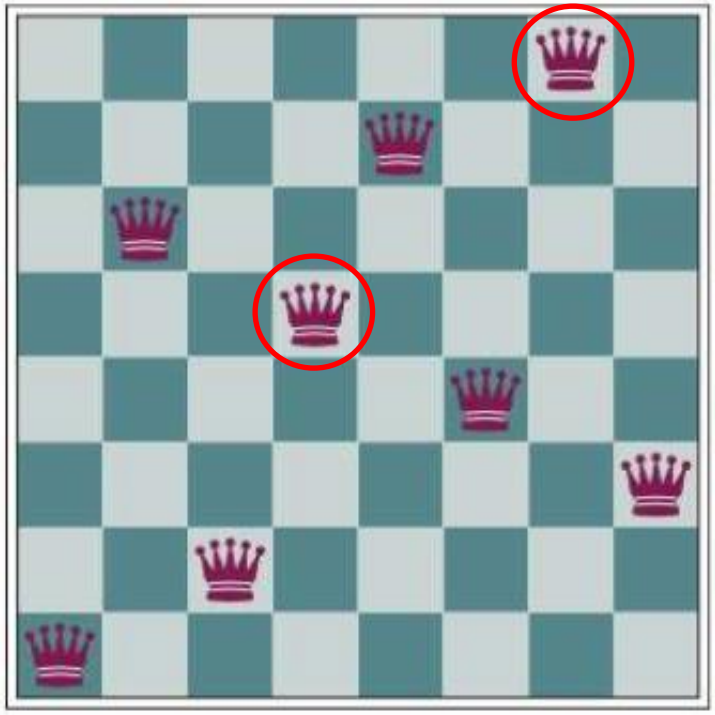
  ↓

  Same row, column, or diagonal

- Initial state: no queen on the board.

Constraint satisfaction!

The 8-Queens Problem



Close to a solution

- Goal: A placement of 8 queens on the chess board in which no queen attacks another.

  ↓

  Same row, column, or diagonal

- Initial state: no queen on the board.

Constraint satisfaction!

The n-queens problem

Place n queens on an $n \times n$ chess board so that no queen attacks another.

Knuth's Conjecture (1964)

Donald Knuth (Stanford)

"father of the analysis of algorithms"
ACM Turing Award (1974)
National Medal of Science (1979)

Any integer > 4 can be reached from 4 via a sequence of square root, floor, and factorial operations.

$$\left\lfloor \sqrt{\sqrt{\sqrt{\sqrt{(4!)!}}}} \right\rfloor = 5$$

* Photo from https://amturing.acm.org/byyear.cfm.

Knuth's Conjecture (1964)

Donald Knuth (Stanford)

"father of the analysis of algorithms"
ACM Turing Award (1974)
National Medal of Science (1979)

Any integer > 4 can be reached from 4 via a sequence of square root, floor, and factorial operations.

$$\left\lfloor \sqrt{\sqrt{\sqrt{\sqrt{(4!)!}}}} \right\rfloor = 5$$

- States: positive real numbers.
- Initial state: 4.
- Goal state: the desired integer > 4.
- Actions: square root, floor, or factorial operation.
- Action cost: 1.

* Photo from https://amturing.acm.org/byyear.cfm.

Real-World Problems

- Route finding (e.g., from Ames, IA to Mountain View, CA)
- Traveling salesman problem

Round trip visiting 13,509 US cities
with population size > 500 exactly once*

* Figure from http://www.crpc.rice.edu/CRPC/newsletters/sum98/news_tsp.html.

Real-World Problems

- Route finding (e.g., from Ames, IA to Mountain View, CA)
- Traveling salesman problem

Round trip visiting 13,509 US cities
with population size > 500 exactly once*

- VLSI layout

Positioning of millions of components and connections on a chip.

- Robot navigation
- Autonomous assembly sequencing (e.g., protein design)

* Figure from http://www.crpc.rice.edu/CRPC/newsletters/sum98/news_tsp.html.