INFORMED SEARCH

Lara J. Martin (she/they)
TA: Aydin Ayanzadeh (he)

9/19/2023

CMSC 671

By the end of class today, you will be able to:

1. Relate Uniform-Cost Search and Greedy (Best-First) Search to A*
2. Explain why the A* algorithm is complete, optimal, and efficient
3. Distinguish between admissible and inadmissible heuristics

Modified from slides by Dr. Cassandra Kent

ADDITION TO PAPER SUMMARY

(no points but required at the end of the summary) the names of anyone you talked to about the paper and/or how you used LLMs (if you did). Please see the Generative AI Policy for more information on how to cite LLM use. If you did not receive any help, please state that instead.

RECAP

UNINFORMED MODELS COMPARED

Breadth-First Search

- **Complete**
- **Optimal**
- $\boldsymbol{O(b^d)}$ **time complexity**
- $O(b^d)$ space complexity

Depth-First Search

- Not complete
- Not optimal
- $O(b^m)$ time complexity
- $\boldsymbol{O(bm)}$ **space complexity**

Iterative Deepening

- **Complete**
- **Optimal**
- $\boldsymbol{O(b^d)}$ **time complexity**
- $\boldsymbol{O(bd)}$ **space complexity**

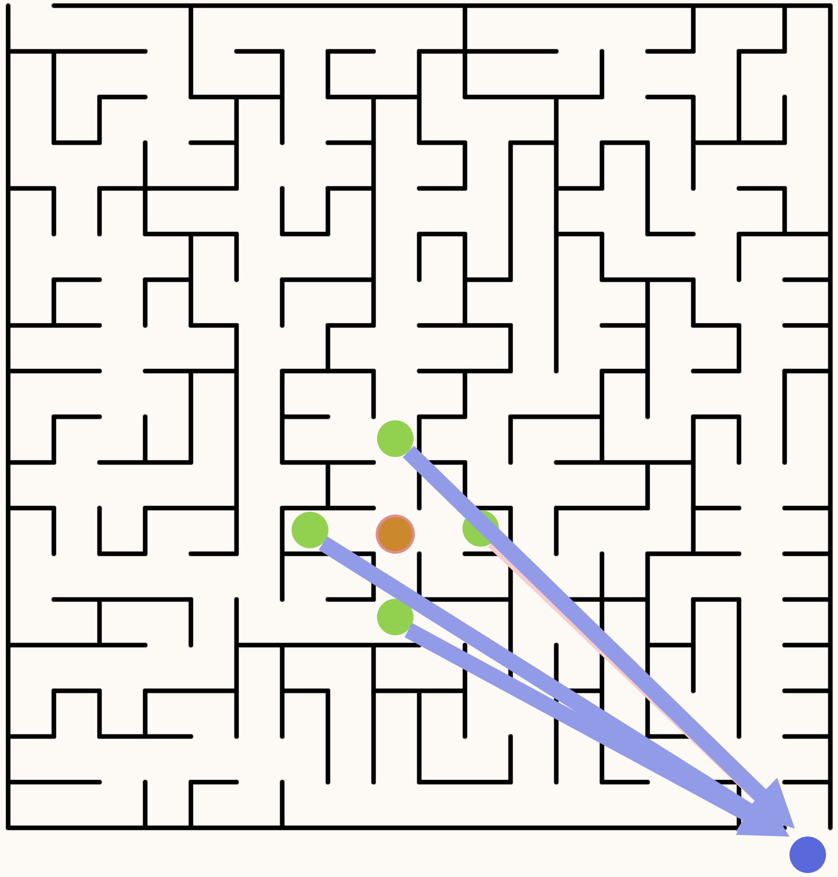
RECAP

What makes a search algorithm "informed"?

RECAP

INFORMED SEARCH INTRO

- down/right states are *closer* to the goal than the up/left states
- Define a function to represent approximate distance to the goal state: a **heuristic function $h(n)$**
  - Example: Euclidean (straight-line) distance
- Use $h(n)$ to select nodes for expansion



RECAP

Best-First Search uses what type of data structure for its frontier?

Priority queue

A* SEARCH

A* SEARCH

We need an optimal informed search algorithm

- Account for previous path length
  - Uniform-Cost Search $g(n)$
- Account for approximate remaining path length
  - Best-First Search $h(n)$
- Can we combine them?

A* SEARCH

```
function TREE-SEARCH(problem) returns a solution, or failure
  initialize the frontier using the initial state of problem
  loop do
      if the frontier is empty then return failure
      choose a leaf node and remove it from the frontier
      if the node contains a goal state then return the corresponding solution
      expand the chosen node, adding the resulting nodes to the frontier
```

Store frontier as a priority queue, prioritized by $f(n) = g(n) + h(n)$

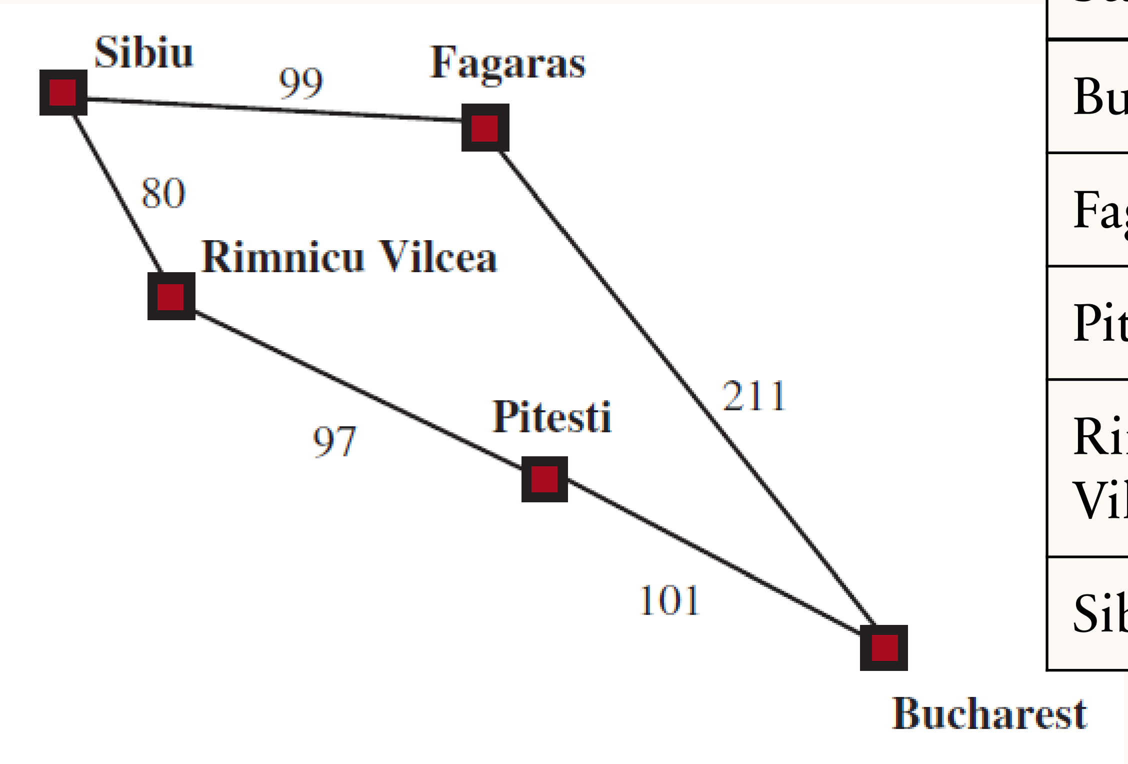
- $f(n)$ approximates the full solution cost passing through a node n
- $g(n)$ is the path cost and $h(n)$ is a heuristic approximating the cost-to-go

A* SEARCH EXAMPLE

Problem: find the shortest (in terms of distance) path from Sibiu to Bucharest

Frontier:
[Sibiu (0+253)]

Current node:
none



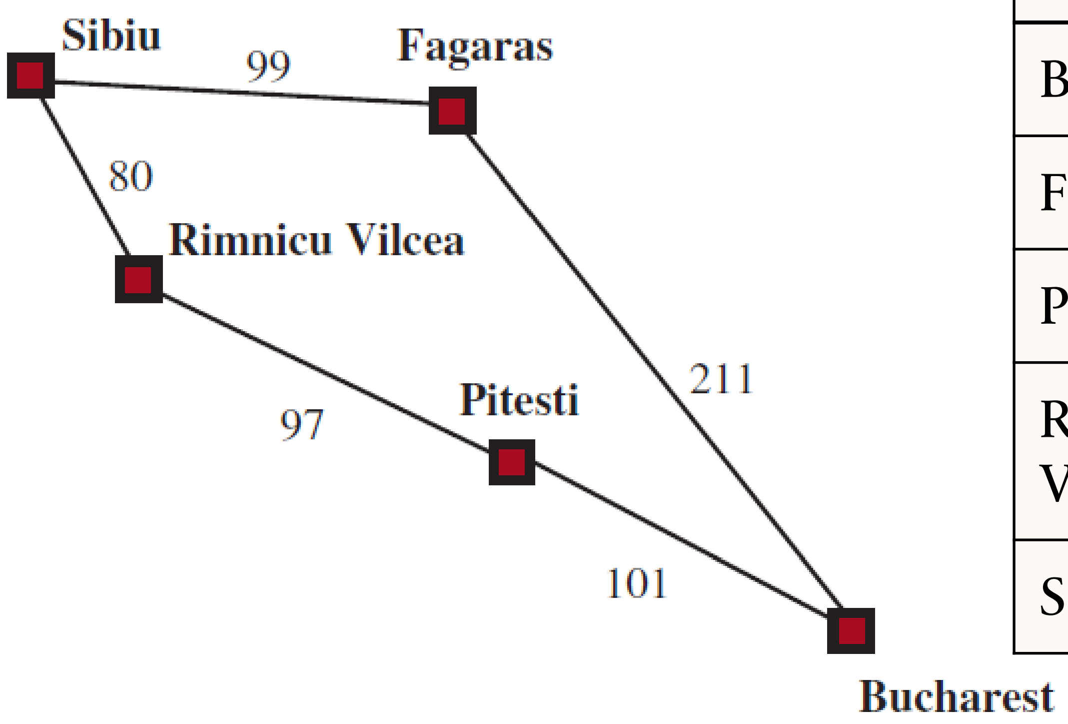
State	h(State)
Bucharest	0
Fagaras	176
Pitesti	100
Rimnicu Vilcea	193
Sibiu	253

A* SEARCH EXAMPLE

Problem: find the shortest (in terms of distance) path from Sibiu to Bucharest

Frontier:
[Sibiu (253)]

Current node:
none



State	h(State)
Bucharest	0
Fagaras	176
Pitesti	100
Rimnicu Vilcea	193
Sibiu	253

A* SEARCH EXAMPLE

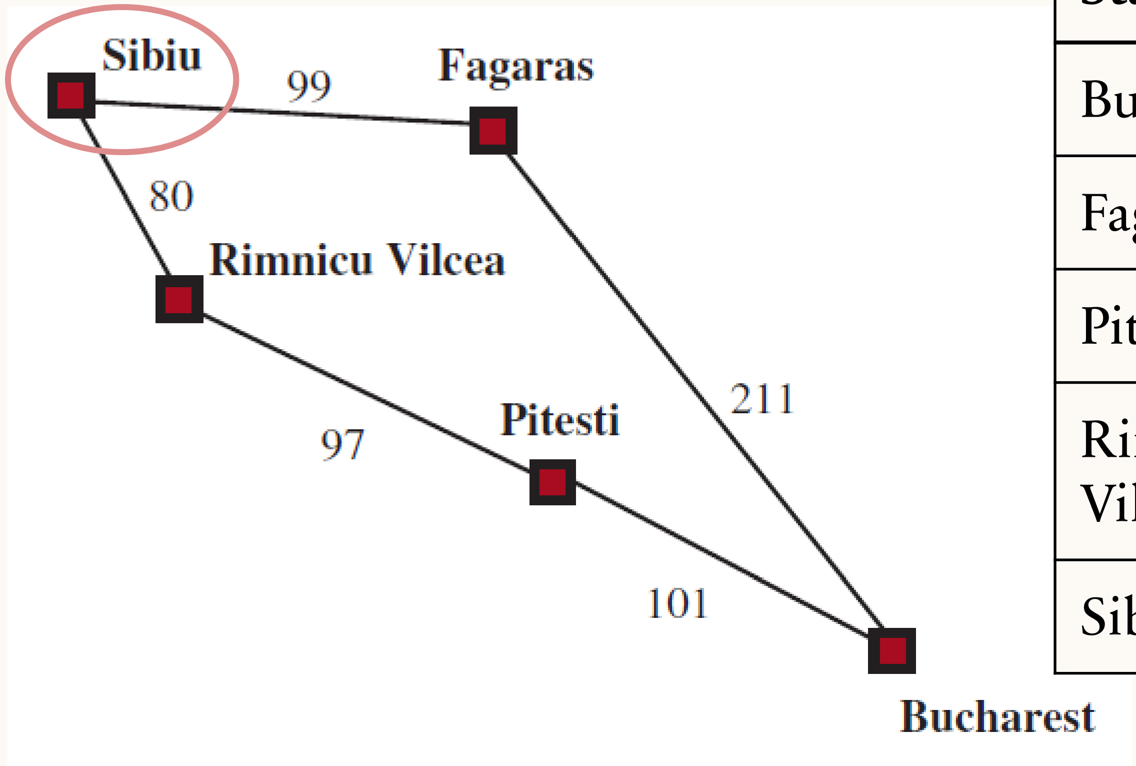
Problem: find the shortest (in terms of distance) path from Sibiu to Bucharest

Frontier:

[]

Current node:

Sibiu (253)



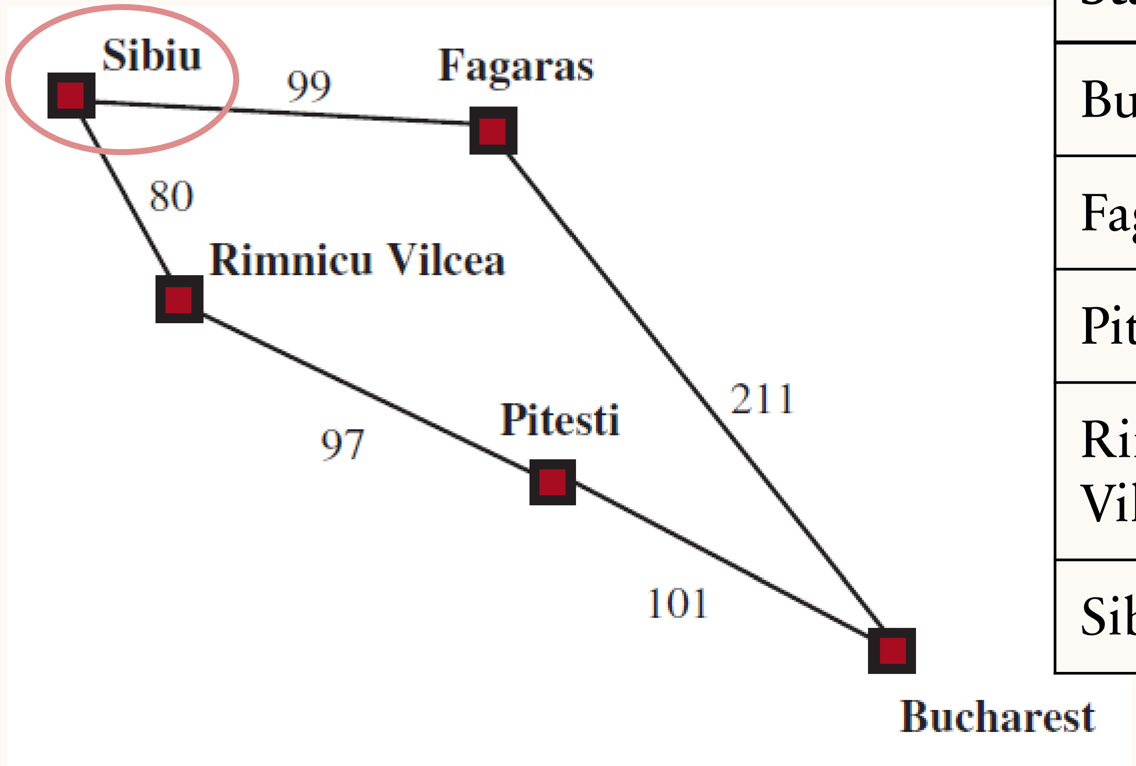
State	h(State)
Bucharest	0
Fagaras	176
Pitesti	100
Rimnicu Vilcea	193
Sibiu	253

A* SEARCH EXAMPLE

Problem: find the shortest (in terms of distance) path from Sibiu to Bucharest

Frontier:
[R.V. (80+193), F. (99+176)]

Current node:
Sibiu (253)



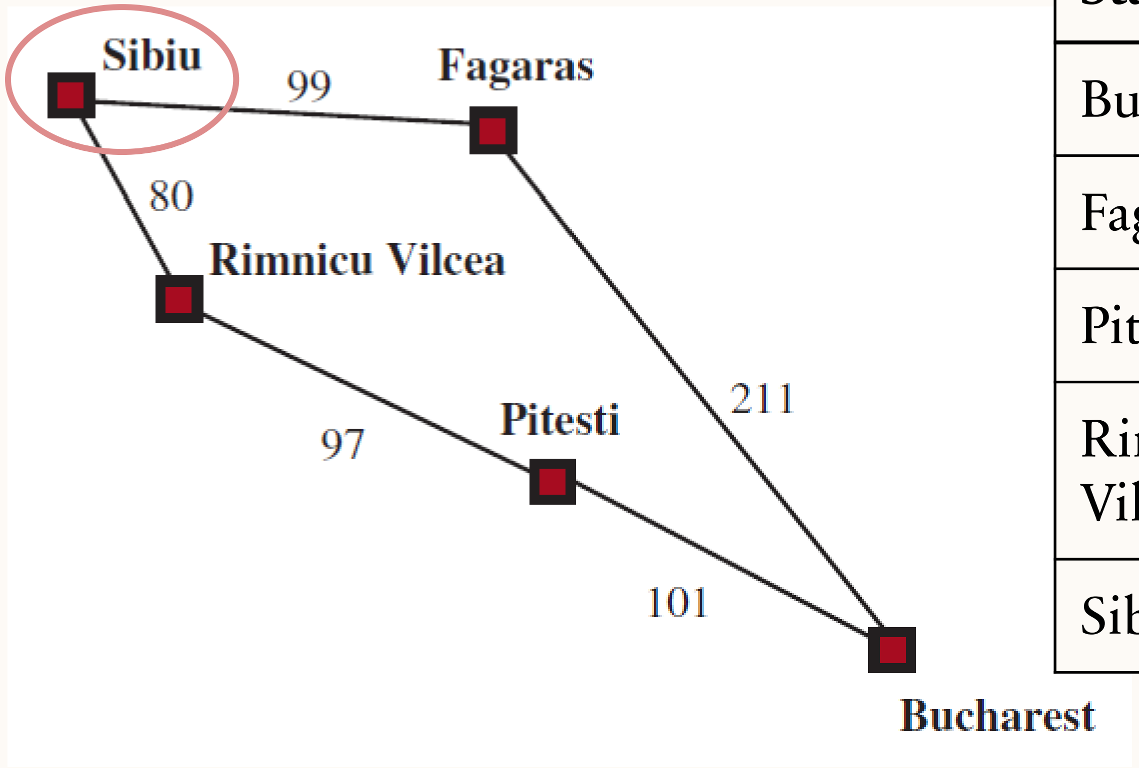
State	h(State)
Bucharest	0
Fagaras	176
Pitesti	100
Rimnicu Vilcea	193
Sibiu	253

A* SEARCH EXAMPLE

Problem: find the shortest (in terms of distance) path from Sibiu to Bucharest

Frontier:
[R.V. (273), F. (275)]

Current node:
Sibiu (253)



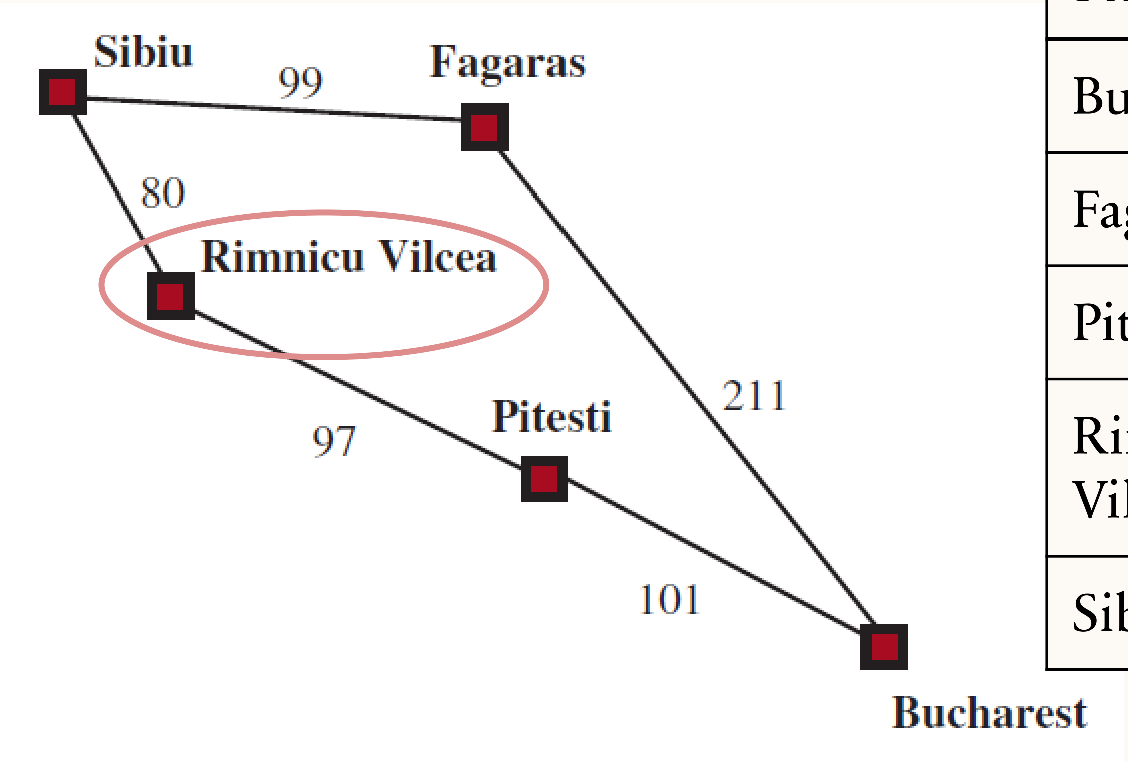
State	h(State)
Bucharest	0
Fagaras	176
Pitesti	100
Rimnicu Vilcea	193
Sibiu	253

A* SEARCH EXAMPLE

Problem: find the shortest (in terms of distance) path from Sibiu to Bucharest

Frontier:
[F. (275)]

Current node:
R.V. (273)



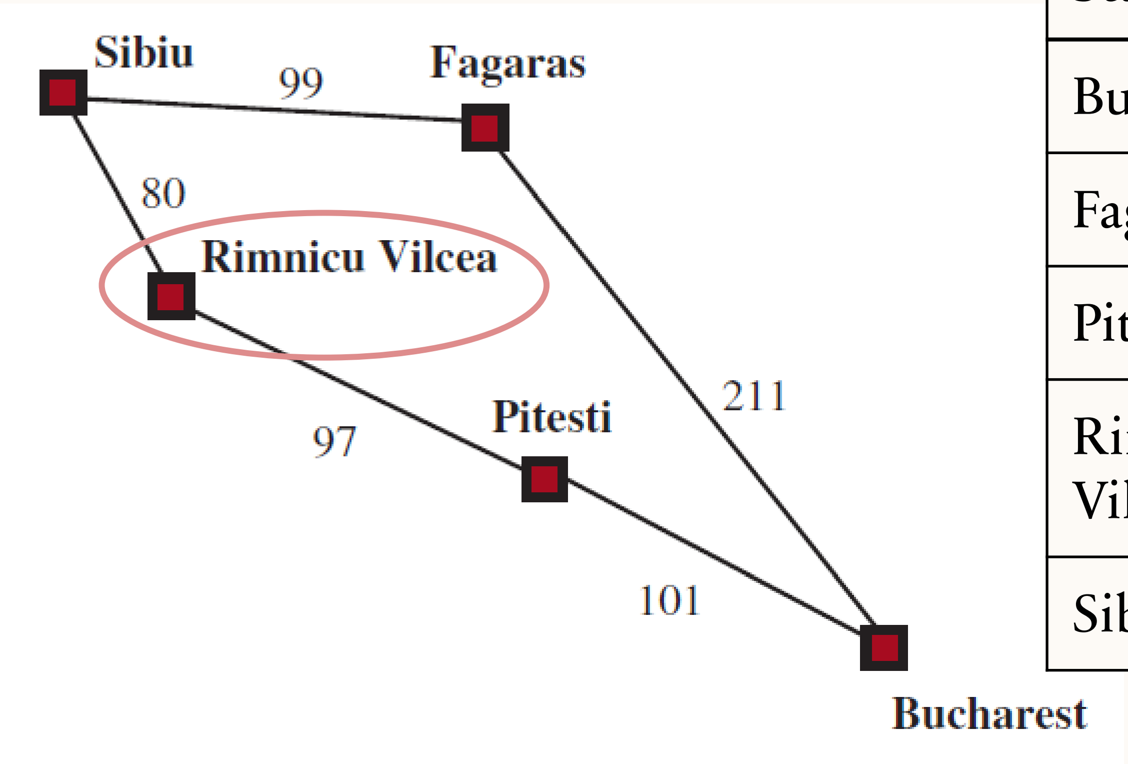
State	h(State)
Bucharest	0
Fagaras	176
Pitesti	100
Rimnicu Vilcea	193
Sibiu	253

A* SEARCH EXAMPLE

Problem: find the shortest (in terms of distance) path from Sibiu to Bucharest

Frontier:
[F. (275), P. (80+97+100)]

Current node:
R.V. (273)



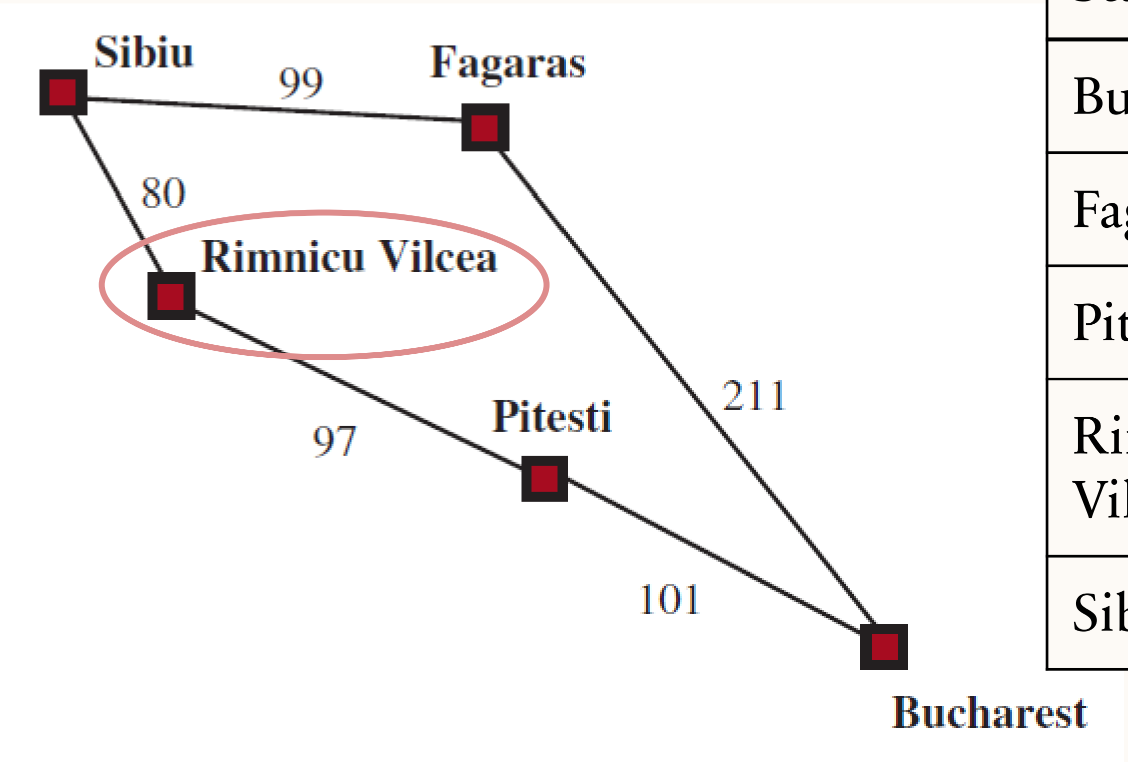
State	h(State)
Bucharest	0
Fagaras	176
Pitesti	100
Rimnicu Vilcea	193
Sibiu	253

A* SEARCH EXAMPLE

Problem: find the shortest (in terms of distance) path from Sibiu to Bucharest

Frontier:
[F. (275), P. (277)]

Current node:
R.V. (273)



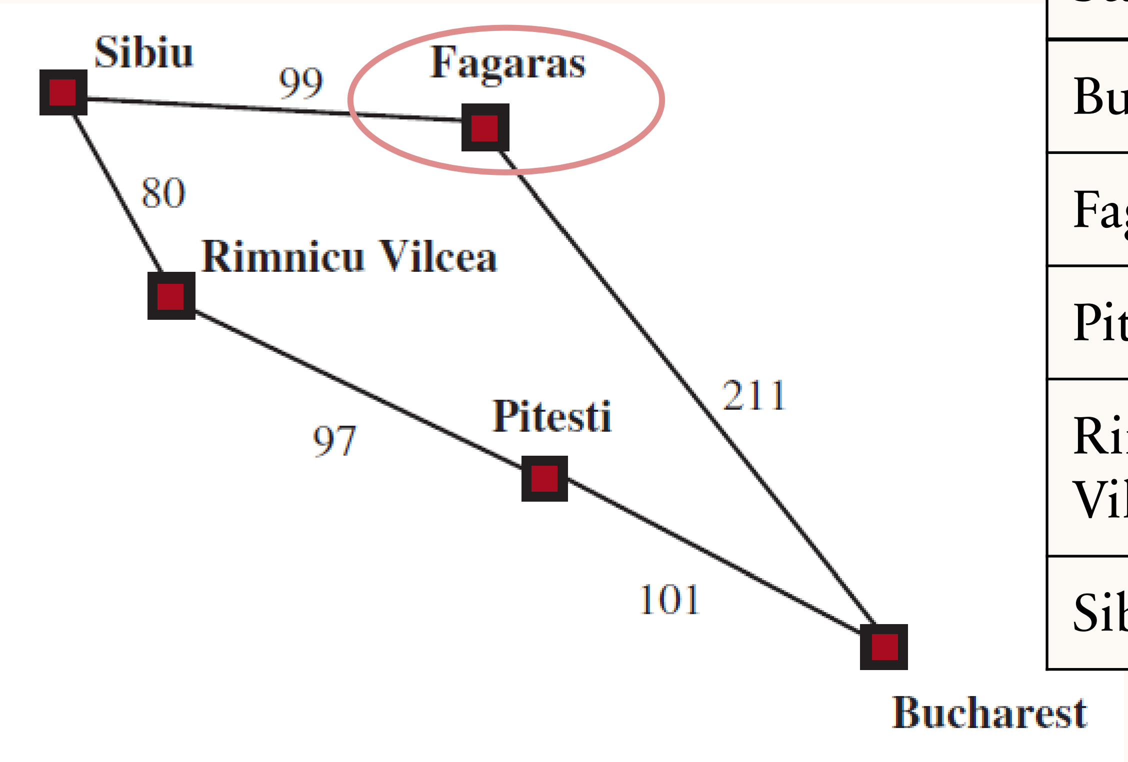
State	h(State)
Bucharest	0
Fagaras	176
Pitesti	100
Rimnicu Vilcea	193
Sibiu	253

A* SEARCH EXAMPLE

Problem: find the shortest (in terms of distance) path from Sibiu to Bucharest

Frontier:
[P. (277)]

Current node:
F. (275)



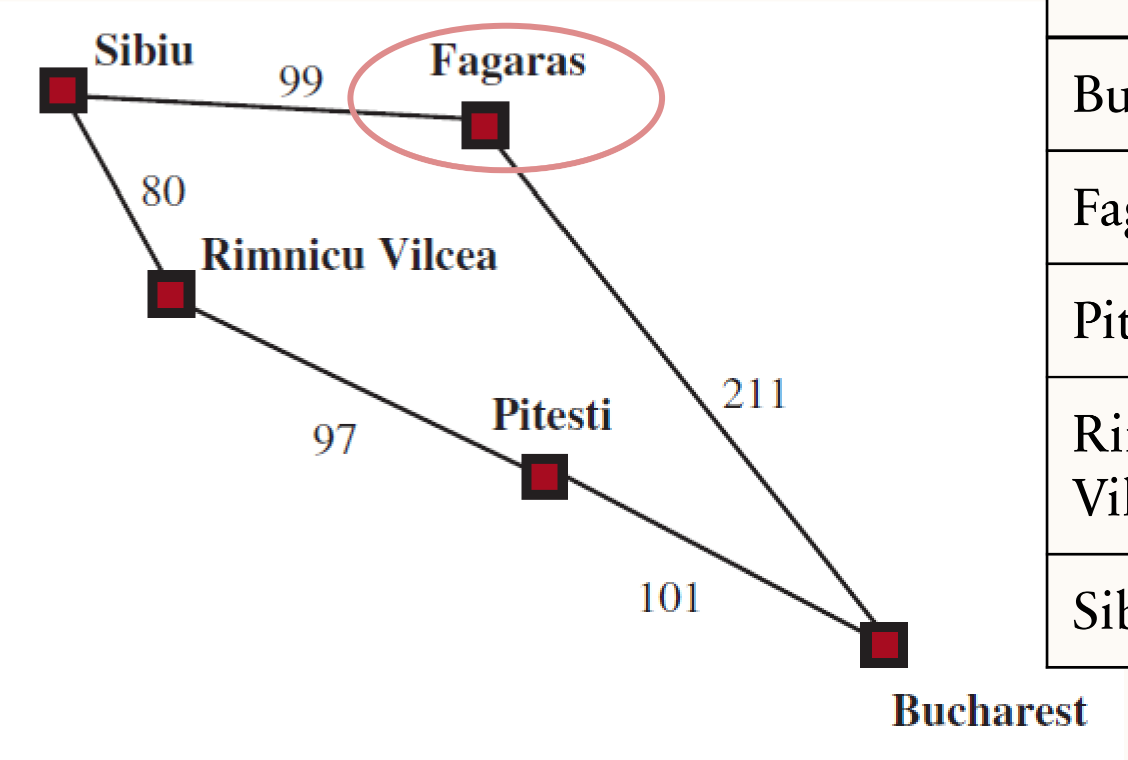
State	h(State)
Bucharest	0
Fagaras	176
Pitesti	100
Rimnicu Vilcea	193
Sibiu	253

A* SEARCH EXAMPLE

Problem: find the shortest (in terms of distance) path from Sibiu to Bucharest

Frontier:
[P. (277), B.(99+211+0)]

Current node:
F. (275)



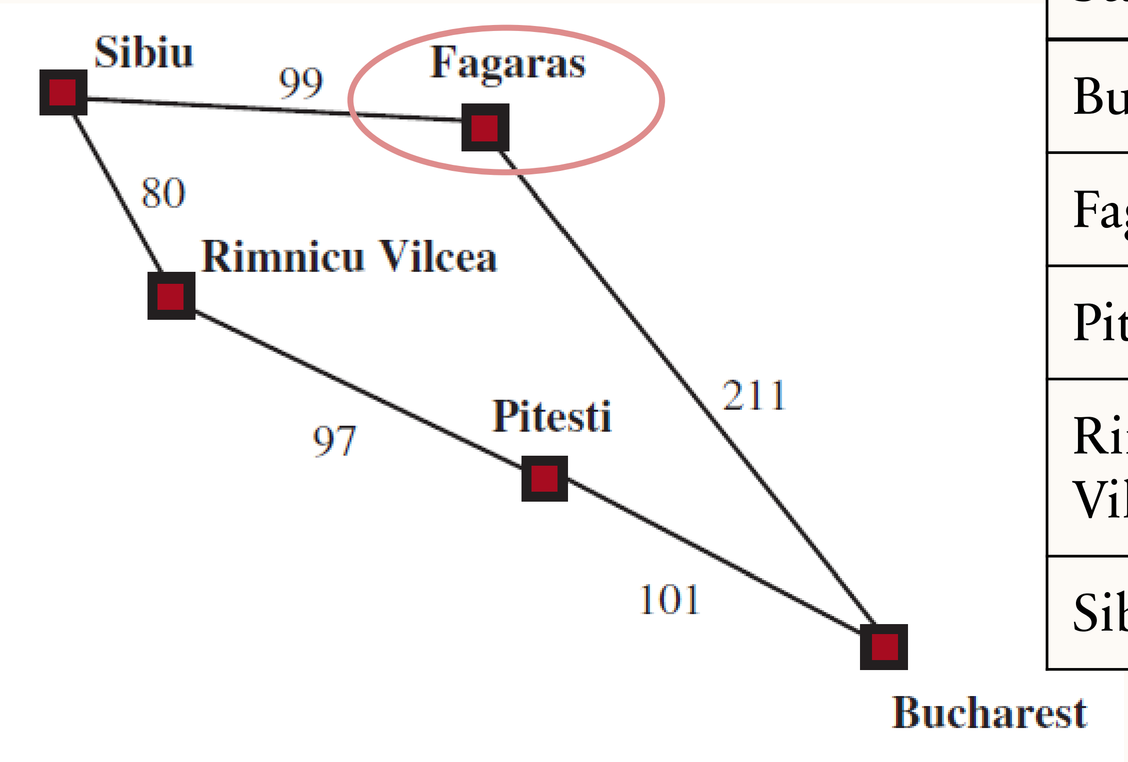
State	h(State)
Bucharest	0
Fagaras	176
Pitesti	100
Rimnicu Vilcea	193
Sibiu	253

A* SEARCH EXAMPLE

Problem: find the shortest (in terms of distance) path from Sibiu to Bucharest

Frontier:
[P. (277), B.(310)]

Current node:
F. (275)



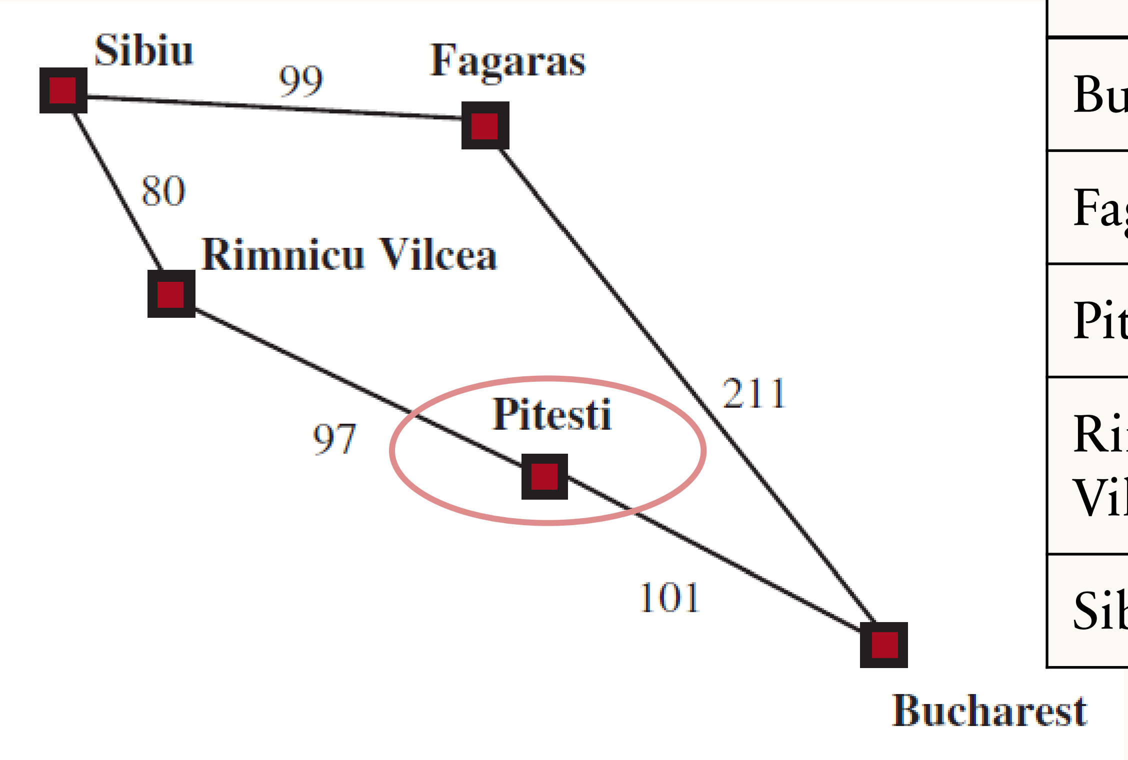
State	h(State)
Bucharest	0
Fagaras	176
Pitesti	100
Rimnicu Vilcea	193
Sibiu	253

A* SEARCH EXAMPLE

Problem: find the shortest (in terms of distance) path from Sibiu to Bucharest

Frontier:
[B.(310)]

Current node:
P. (277)



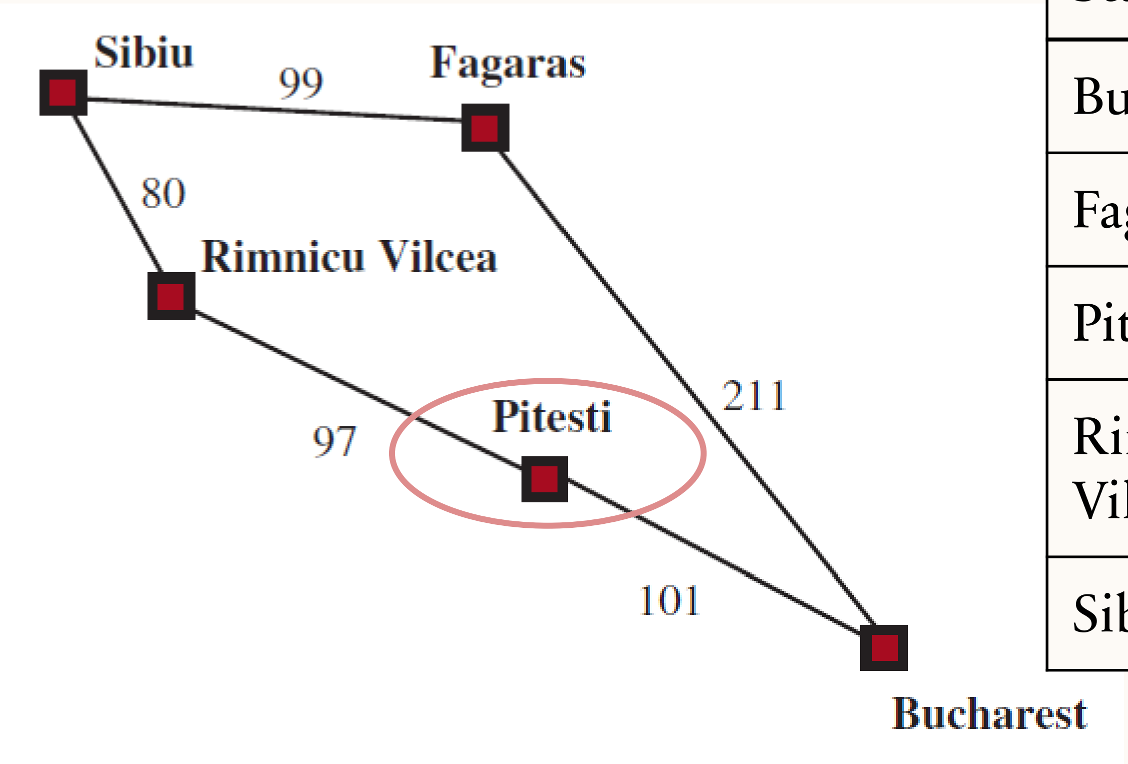
State	h(State)
Bucharest	0
Fagaras	176
Pitesti	100
Rimnicu Vilcea	193
Sibiu	253

A* SEARCH EXAMPLE

Problem: find the shortest (in terms of distance) path from Sibiu to Bucharest

Frontier:
[B.(80+97+101+0), B.(310)]

Current node:
P. (277)



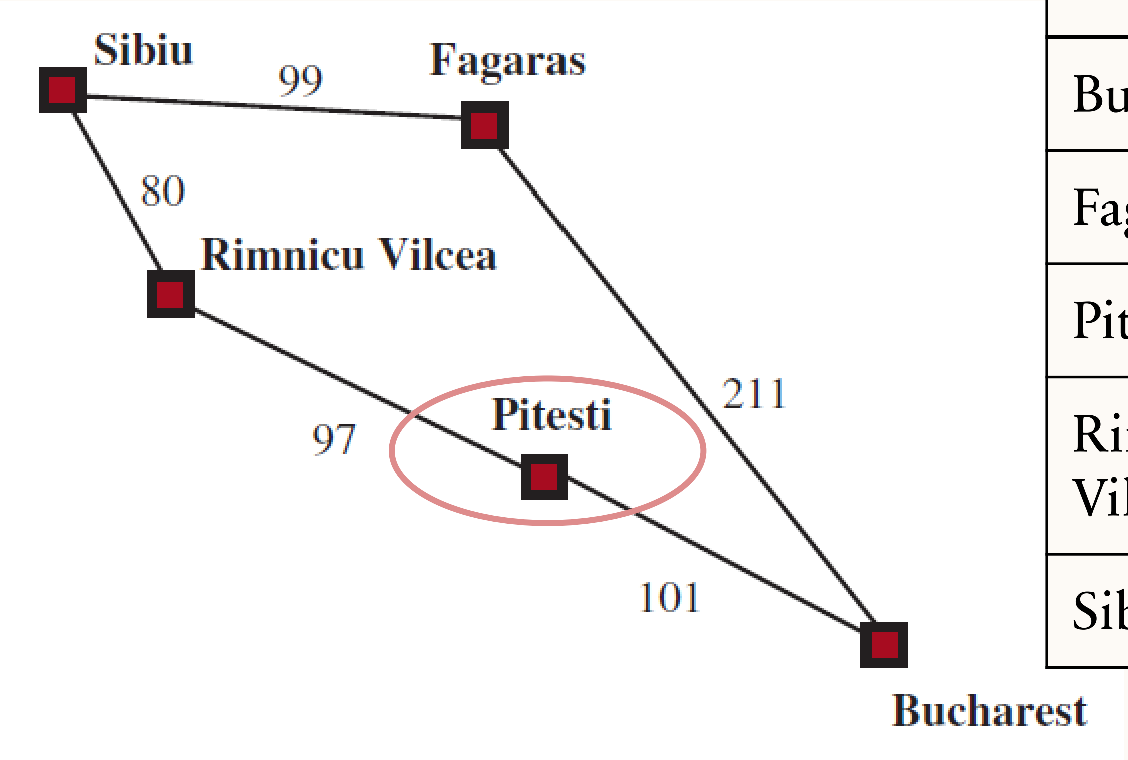
State	h(State)
Bucharest	0
Fagaras	176
Pitesti	100
Rimnicu Vilcea	193
Sibiu	253

A* SEARCH EXAMPLE

Problem: find the shortest (in terms of distance) path from Sibiu to Bucharest

Frontier:
[B.(278), B.(310)]

Current node:
P. (277)



State	h(State)
Bucharest	0
Fagaras	176
Pitesti	100
Rimnicu Vilcea	193
Sibiu	253

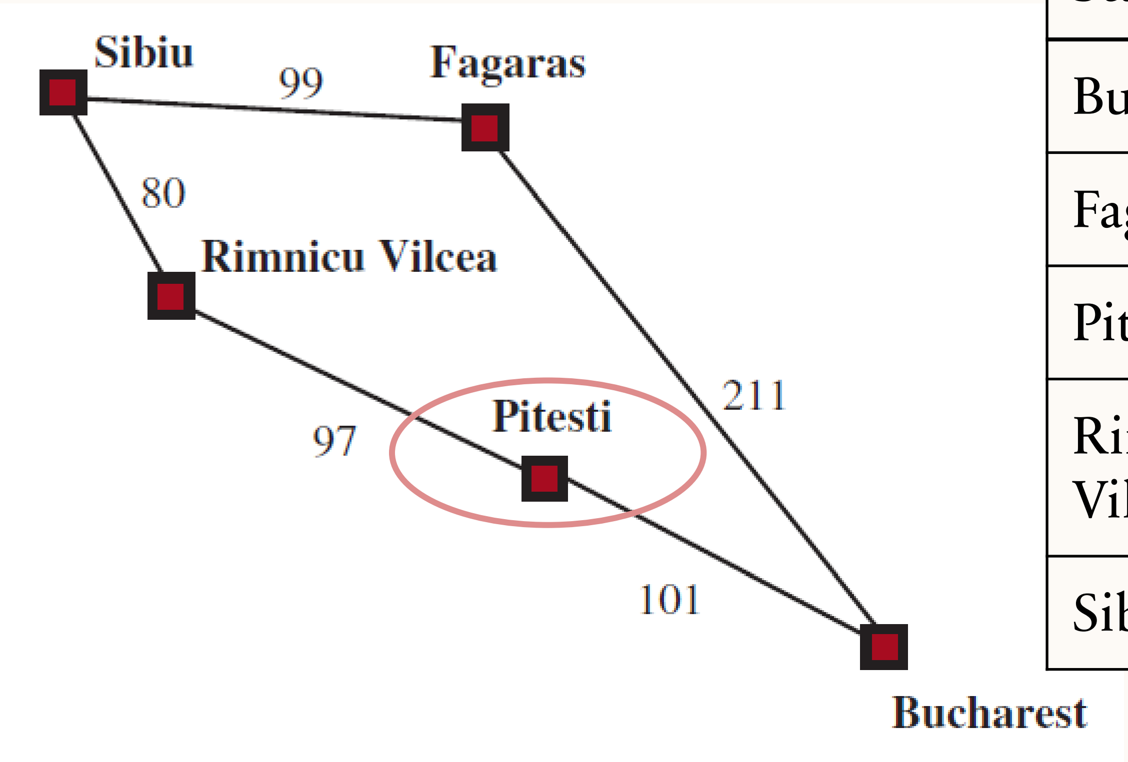
A* SEARCH EXAMPLE

Problem: find the shortest (in terms of distance) path from Sibiu to Bucharest

Frontier:
[B.(310)]

Current node:
B.(278),

Return:
[Sibiu, Rimnicu Vilcea, Pitesti, Bucharest]



State	h(State)
Bucharest	0
Fagaras	176
Pitesti	100
Rimnicu Vilcea	193
Sibiu	253

A* SEARCH PROPERTIES

A* Search

- **Complete**
- **Optimal**[1]
- **Exponential time complexity (decreased by good heuristics)**
- Exponential space complexity (decreased by good heuristics)
- **Optimally efficient**[1]**:** A* guaranteed to find optimal solution faster than any other informed search algorithm for a given heuristic $h(n)$

[1]For admissible heuristics (consistent heuristics for graph search)

ADMISSIBLE HEURISTICS

ADMISSIBLE HEURISTICS

A heuristic is **admissible** if it never over-estimates the cost to the goal:

optimal solution cost — optimal path cost — perfect estimate

$$f^*(n) = g^*(n) + h^*(n)$$

Admissible heuristic: $h(n) \leq h^*(n)$

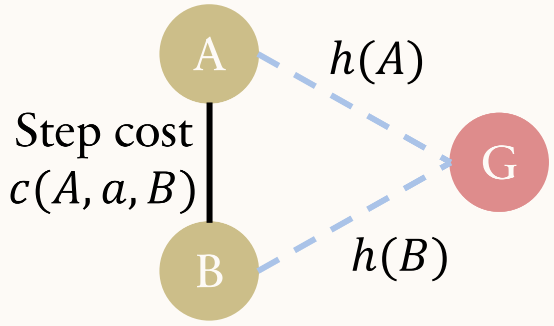
$$f(n) = g(n) + h(n) \leq C^*$$

actual cost

A heuristic is **consistent** if it meets the **triangle inequality**:

$$h(n) \leq c(n, a, n') + h(n')$$

(one side of a triangle cannot be longer than the sum of the two other sides)



A* SEARCH PROPERTIES

If $h(n)$ is admissible: $f(n) = g(n) + h(n) \leq C^*$

- A* is optimal because it expands all nodes where $f(n) < C^*$, and some nodes where $f(n) = C^*$
- A* is efficient because it **prunes** (ignore parts of search space) where $f(n) > C^*$

Some heuristics are better than others:

- What happens if $h(n) = 0$?
  - Equivalent to UCS (still optimal, but uninformed)
- What happens if $h(n) = h^*(n)$, i.e. it estimates the *exact* cost to the goal?
  - Every step will follow the optimal path

EXAMPLES OF (SPATIAL) HEURISTICS

- **Manhattan** (city block) **distance:** sum of horizontal and vertical distance, minimum distance on a 4-connected grid

$$h(n) = |g_x - n_x| + |g_y - n_y|$$

- **Euclidean** (straight-line) **distance:** shortest distance from start to goal

$$h(n) = \sqrt{(g_x - n_x)^2 + (g_y - n_y)^2}$$

- Octile **distance:** minimum distance on a grid with diagonal moves (8-connected)

A* SEARCH IN PRACTICE

Experiment URL (try it for yourself!):
https://qiao.github.io/PathFinding.js/visual/

Experiment summary:

- A* succeeds where Best-First Search fails and UCS is inefficient
- Only holds for admissible heuristics
- Inadmissible heuristics may produce sub-optimal paths
- In the limit as $h(n)$ increases, A* behaves like greedy search
  - $h(n)$ dominates $g(n)$, so $f(n) \approx h(n)$

IS THIS HEURISTIC ADMISSIBLE? ROUTE FINDING

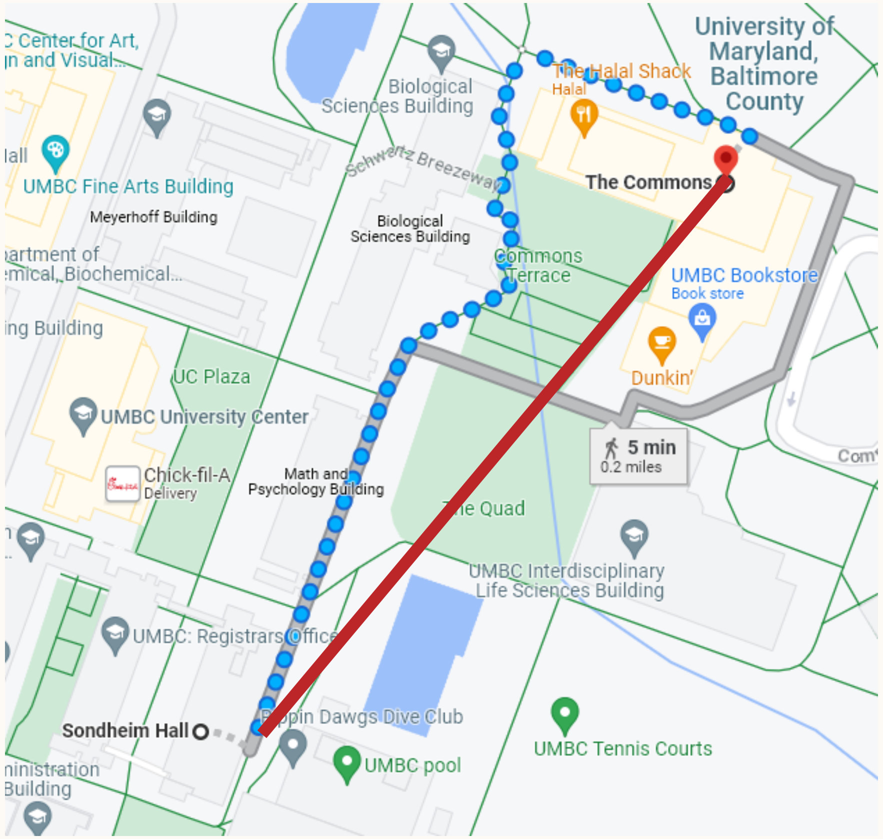
State: agent location (which path intersection are you at)

Actions: follow path *x* to the next intersection

Goal: find the shortest path to the end location

Heuristic: **Euclidean distance**

Answer: Yes, because the shortest distance between two points is a line



IS THIS HEURISTIC ADMISSIBLE? ROUTE FINDING

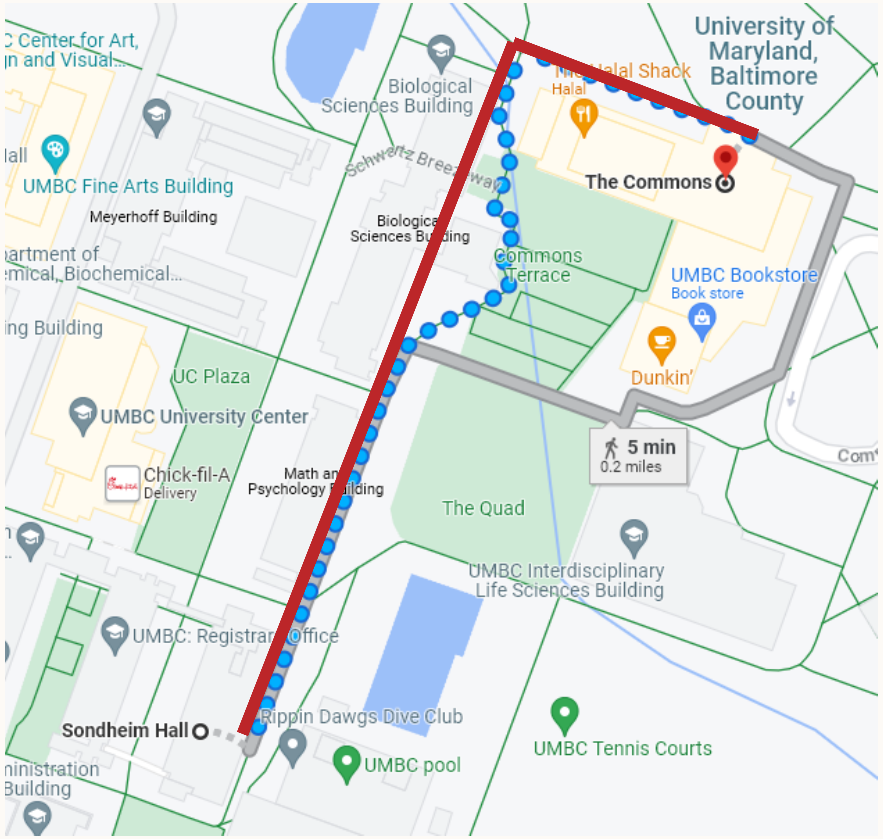
State: agent location (which path intersection are you at)

Actions: follow path x to the next intersection

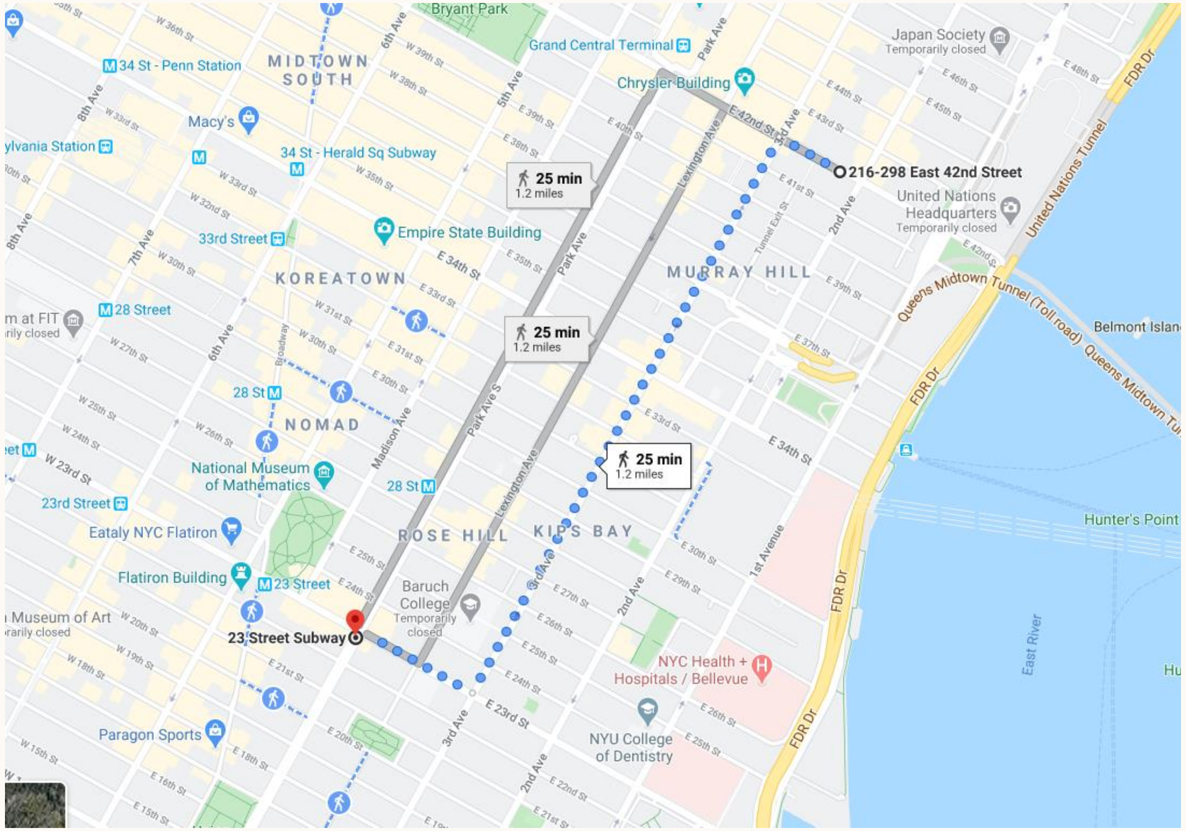
Goal: find the shortest path to the end location

Heuristic: **Manhattan distance**

Answer: No (e.g. it will overestimate diagonal paths)



IS THIS HEURISTIC ADMISSIBLE? ROUTE FINDING



Heuristic: **Manhattan distance** (maybe yes if you're in Manhattan)

IS THIS HEURISTIC ADMISSIBLE? 8-PUZZLE

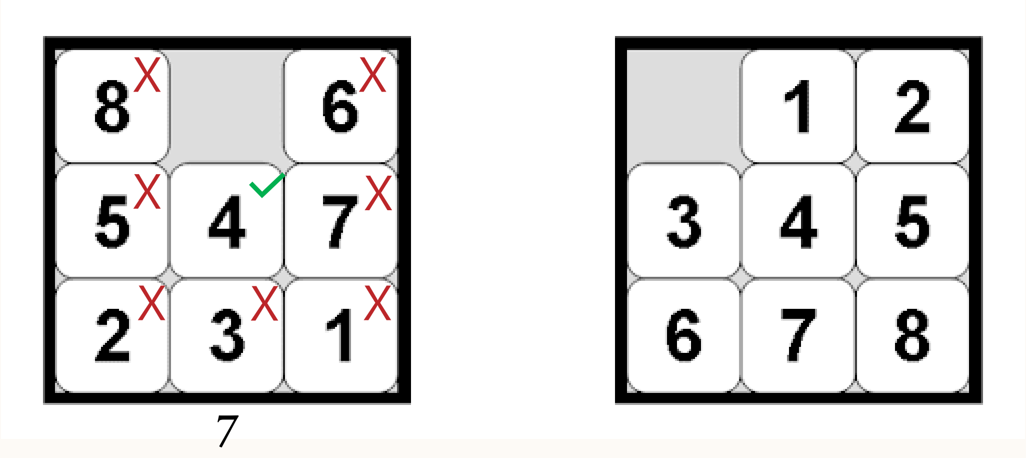
State: configuration of tiles

Actions: slide any single tile to the open position

Goal: reach solved state in the fewest moves

Heuristic: **# of tiles out of solved position**

Answer: Yes (we can only move one tile at a time, and any out of place tile will need to be moved at least once)



IS THIS HEURISTIC ADMISSIBLE? 8-PUZZLE

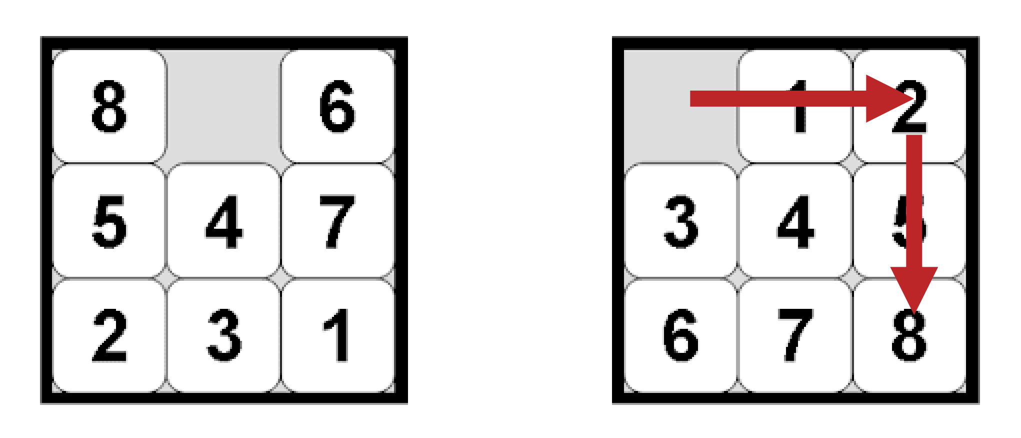
State: configuration of tiles

Actions: slide any single tile to the open position

Goal: reach solved state in the fewest moves

Heuristic: **Manhattan distance for each tile to solved position**

Answer: Yes (each tile can only move one space at a time, on a 4-connected grid)



IS THIS HEURISTIC ADMISSIBLE? RUBIK'S CUBE

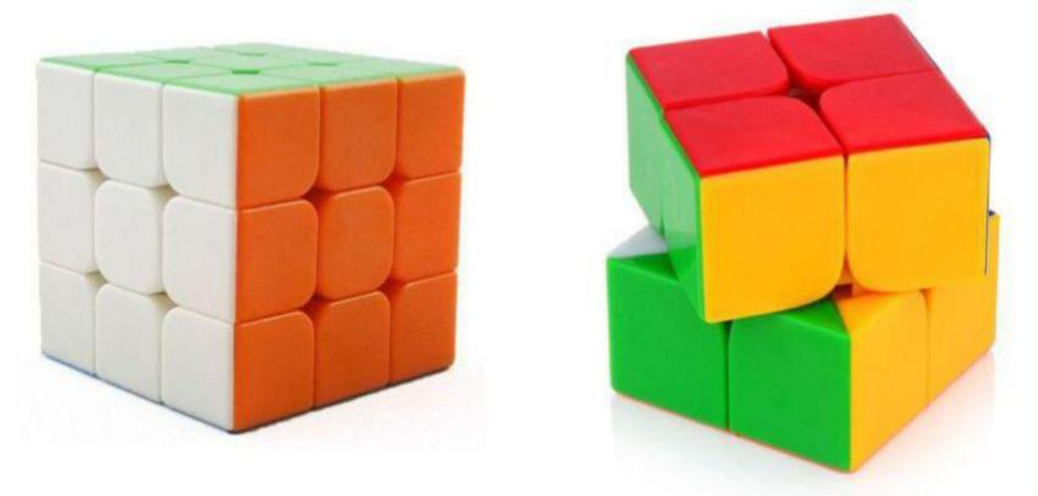
State: configuration of blocks on the cube

Actions: rotate each face clockwise/counterclockwise 90 degrees

Goal: reach solved state in the fewest moves

Heuristic: **# of blocks out of solved position**

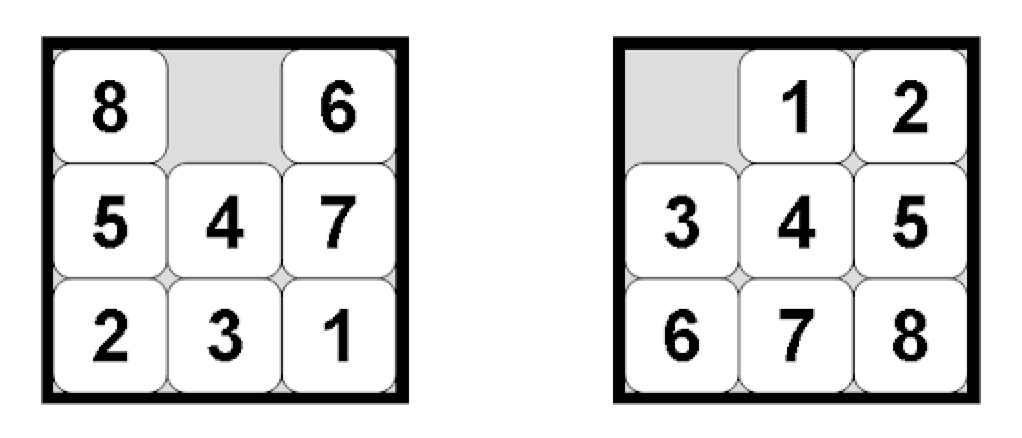
Answer: No (1 single move can move 8 blocks at a time, so this can over estimate)



WHICH HEURISTIC IS BETTER? 8-PUZZLE

Heuristic 1: **# of tiles out of solved position**
Heuristic 2: **Manhattan distance for each tile to solved position**

Answer: Heuristic 2, because it *dominates* Heuristic 1 (see later slides)



EVALUATING HEURISTICS

A* performance depends on how good your heuristic is.

How do we know how good a heuristic is?

- **Informedness**: a more informed heuristic will explore a smaller portion of the state space

$$informedness(h) = \frac{\#\ states\ in\ search\ space}{avg\ \#\ of\ states\ explored\ with\ h}$$

- **Dominance**: a heuristic h_1 **dominates** h_2 if

$$h_2(s_i) \leq h_1(s_i) \leq h^*(s_i) \text{ for all states } s_i$$

OPTIMAL SEARCH ALGORITHM CHEATSHEET

Search Algorithm	Frontier Implementation	Notes
Breadth-First Search (BFS)	Queue	
Depth-First Search (DFS)	Stack	Optimal with iterative deepening
Depth-Limited Search	Stack	Optimal with known solution depth
Uniform-Cost Search (UCS)	Priority Queue	Path cost $g(n)$
A* Search	Priority Queue	Estimated solution cost $f(n) = g(n) + h(n)$

FOR NEXT CLASS

- Finish reading Chapter 3.5-3.7
- If you have Module 1/Search for your paper presentation
  - **Summaries are due tomorrow!**
  - Presentation on Thursday Sept 21
- You should be ready to do all of HW 1 now