# Discussion 3
# Informed Search

Jinghao Zhao

10/15/2020

# Uninformed Search Strategies

| Criterion | Breadth-First | Uniform-Cost | Depth-First | Depth-Limited | Iterative Deepening | Bidirectional (if applicable) |
|---|---|---|---|---|---|---|
| Complete? | Yes$^a$ | Yes$^{a,b}$ | No | No | Yes$^a$ | Yes$^{a,d}$ |
| Time | $O(b^d)$ | $O(b^{1+\lfloor C^*/\epsilon \rfloor})$ | $O(b^m)$ | $O(b^\ell)$ | $O(b^d)$ | $O(b^{d/2})$ |
| Space | $O(b^d)$ | $O(b^{1+\lfloor C^*/\epsilon \rfloor})$ | $O(bm)$ | $O(b\ell)$ | $O(bd)$ | $O(b^{d/2})$ |
| Optimal? | Yes$^c$ | Yes | No | No | Yes$^c$ | Yes$^{c,d}$ |

**Figure 3.21** Evaluation of tree-search strategies. $b$ is the branching factor; $d$ is the depth of the shallowest solution; $m$ is the maximum depth of the search tree; $l$ is the depth limit. Superscript caveats are as follows: $^a$ complete if $b$ is finite; $^b$ complete if step costs $\geq \epsilon$ for positive $\epsilon$; $^c$ optimal if step costs are all identical; $^d$ if both directions use breadth-first search.

- Please memorize completeness, optimality, time complexity, and space complexity for all of them.

# Informed Search

- Informed search
  - Leverage problem-specific knowledge

The general approach for informed search:

- Best-first search
  - Choose the "best" (the most <u>promising</u>) node to expand

# Informed Search

How to determine which node is the best?

- Evaluation function $f(n)$ (A cost estimation for node $n$)
  - $n$ is a node, not a state!
    - In uniform-cost search (an uninformed search strategy) we store <u>state</u> costs in the priority queue
  - Often involves a heuristic function $h(n)$
- Greedy best-first search
- A* search

# Is uniform-cost search informed search?

- No!
- It only looks backwards; has no ability to predict future costs.

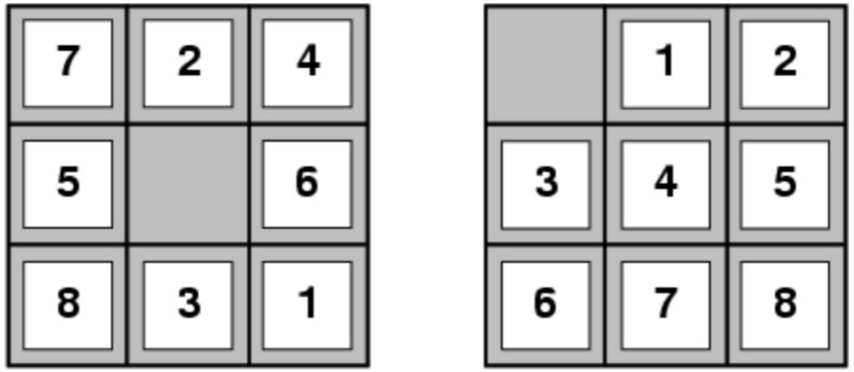
# How to figure out heuristics?

- Relaxed problem

A problem with fewer restrictions on the actions are relaxed problems.

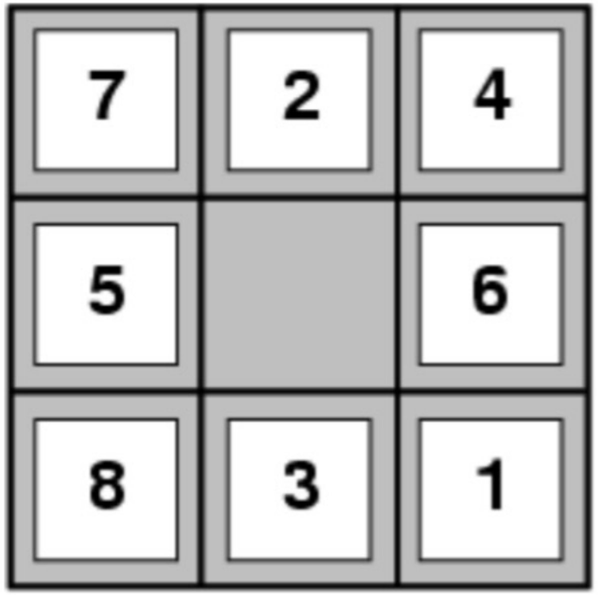
If the 8-puzzle is relaxed so the tile can move anywhere, which heuristic is better?

If the 8-puzzle is relaxed so that tile can move only to adjacent positions, which heuristic is better?

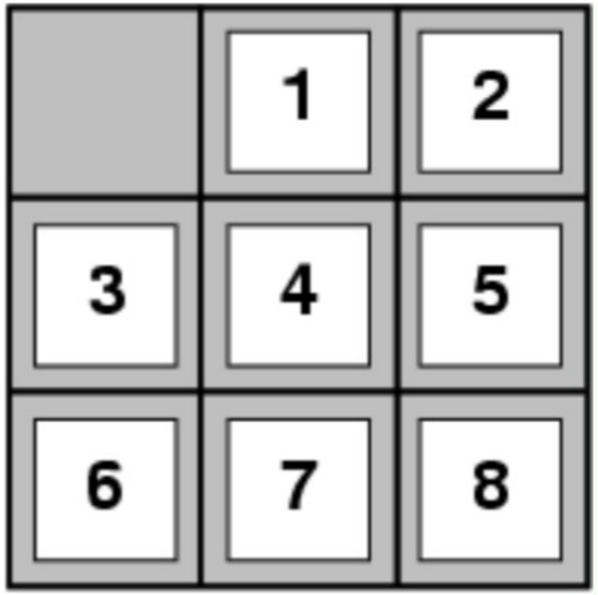



Start State Goal State

# Exercise 1




Start State




Goal State

h1(n): number of misplaced tiles h1(S) = ?

h2(n): number of Manhattan distance h2(S)=?

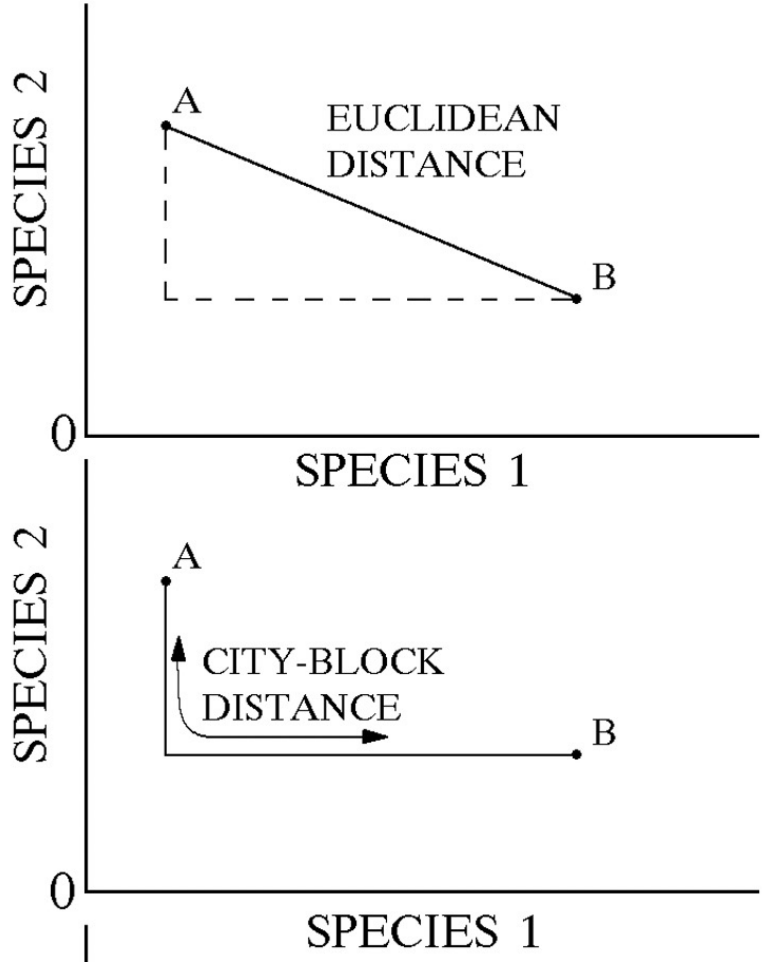
# FYI-Euclidean & Manhattan Distance

- Euclidean distance

$$ED_{i,h} = \sqrt{\sum_{j=1}^{p} (a_{i,j} - a_{h,j})^2}$$

- City-block distance (= Manhattan distance)

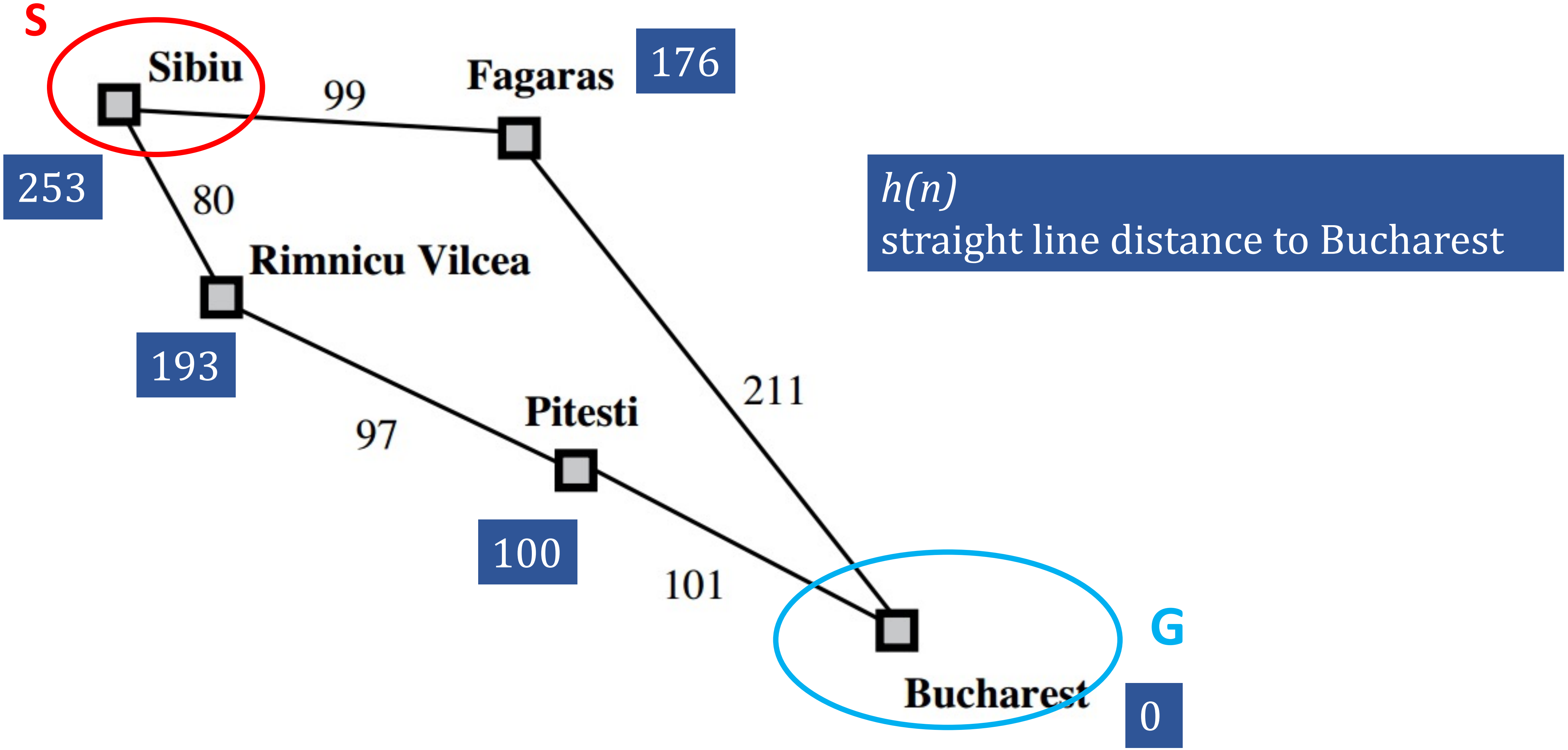
$$CB_{i,h} = \sum_{j=1}^{p} |a_{i,j} - a_{h,j}|$$



# A good heuristic function

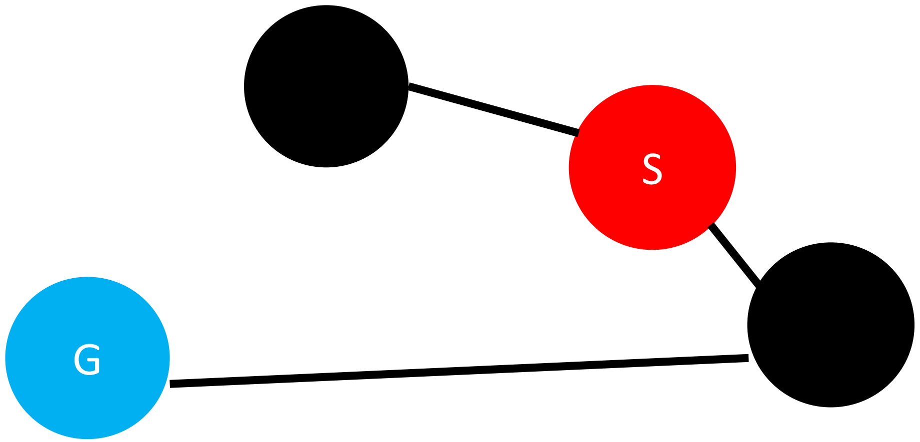
- Admissibility
  - Heuristic cost should **never overestimate** the actual cost of a node
    - I.e., it must be “optimistic”
      - they think the cost of solving the problem is less than it actually is
    - So that we never overlook a node that is actually good

# Example – straight line distance in route planning



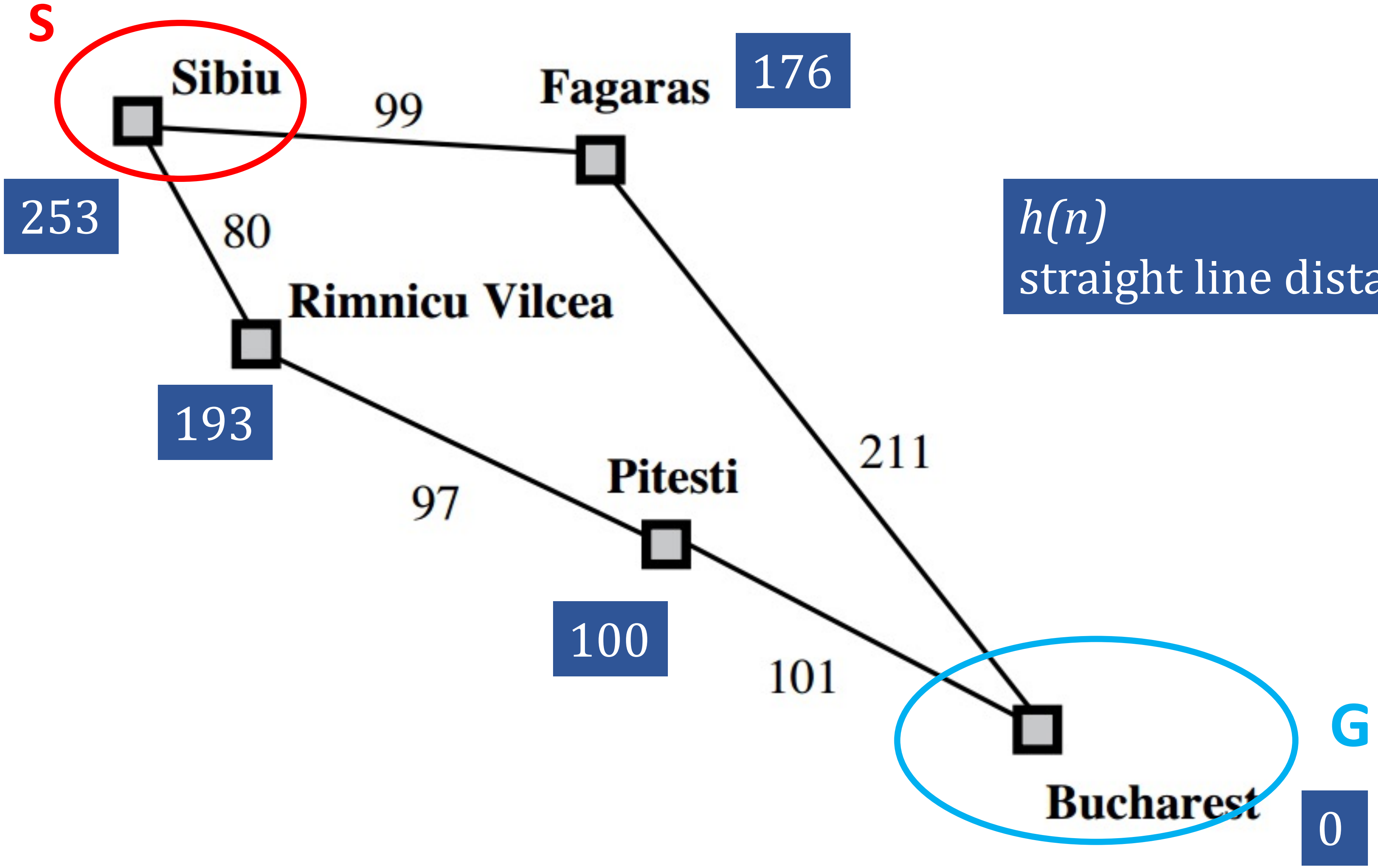
# Greedy best-first search $f(n) = h(n)$

- Complete?
  - No.
    - Can get stuck in loops




- Optimal?
  - No
- Worst Time and Space complexity $O(b^m)$
  - A good heuristic can give dramatic improvement

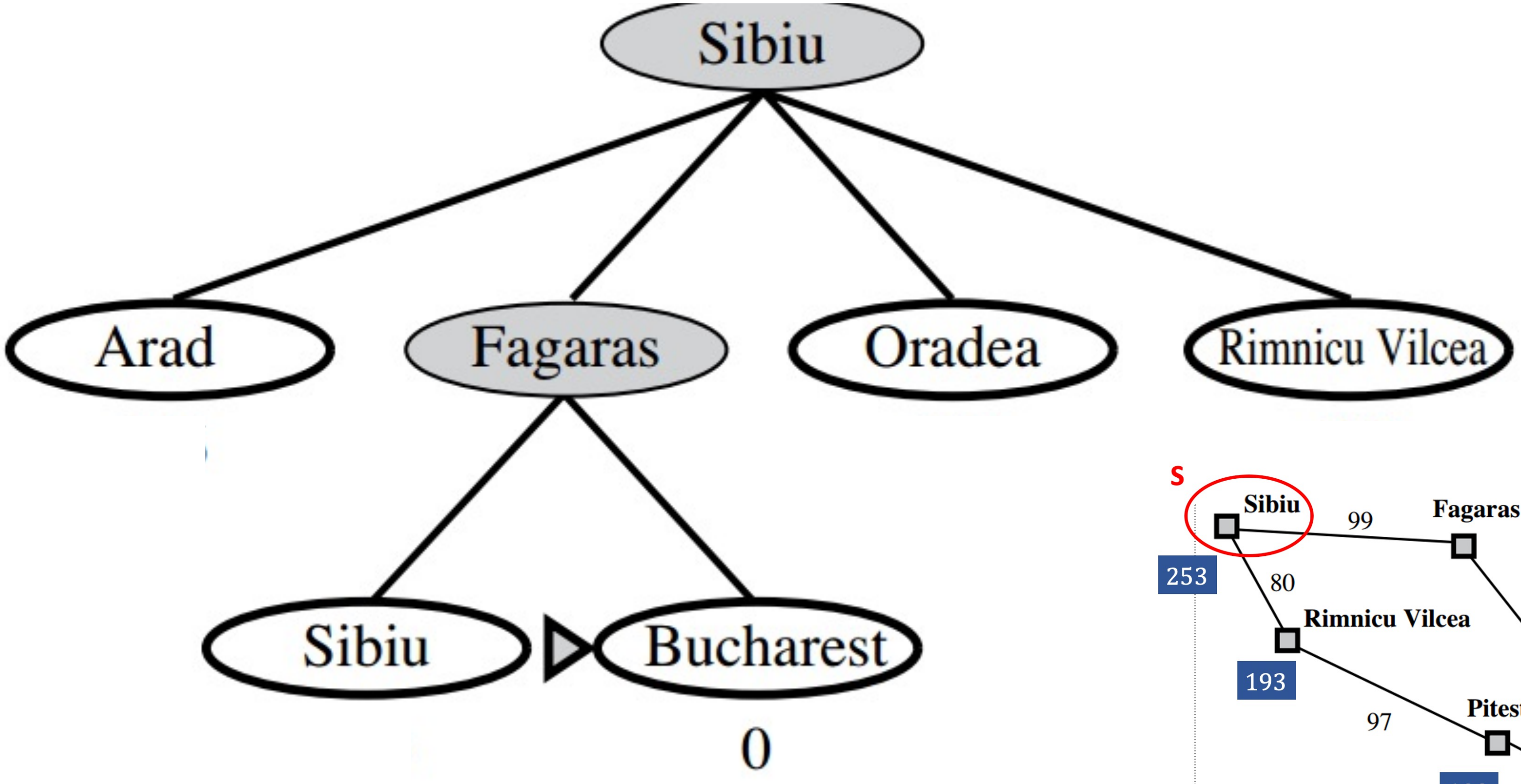
# Greedy best-first search $f(n) = h(n)$

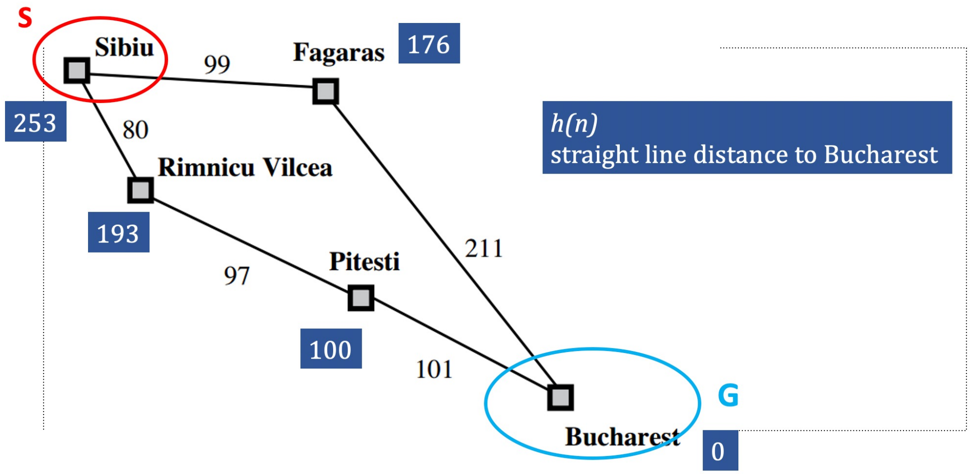



$h(n)$
straight line distance to Bucharest

# Example - Straight-line distance in route planning

**Greedy best-first search** $f(n) = h(n)$



# A* search

- Avoid expanding paths that are already expensive

- $f(n) = g(n) + h(n)$

*g(n)* = cost so far to reach *n*
*h(n)* = estimated cost from *n* to goal
*f(n)* = estimated total cost of path through *n* to goal

A$*$ expands no nodes with f(n) > C$*$

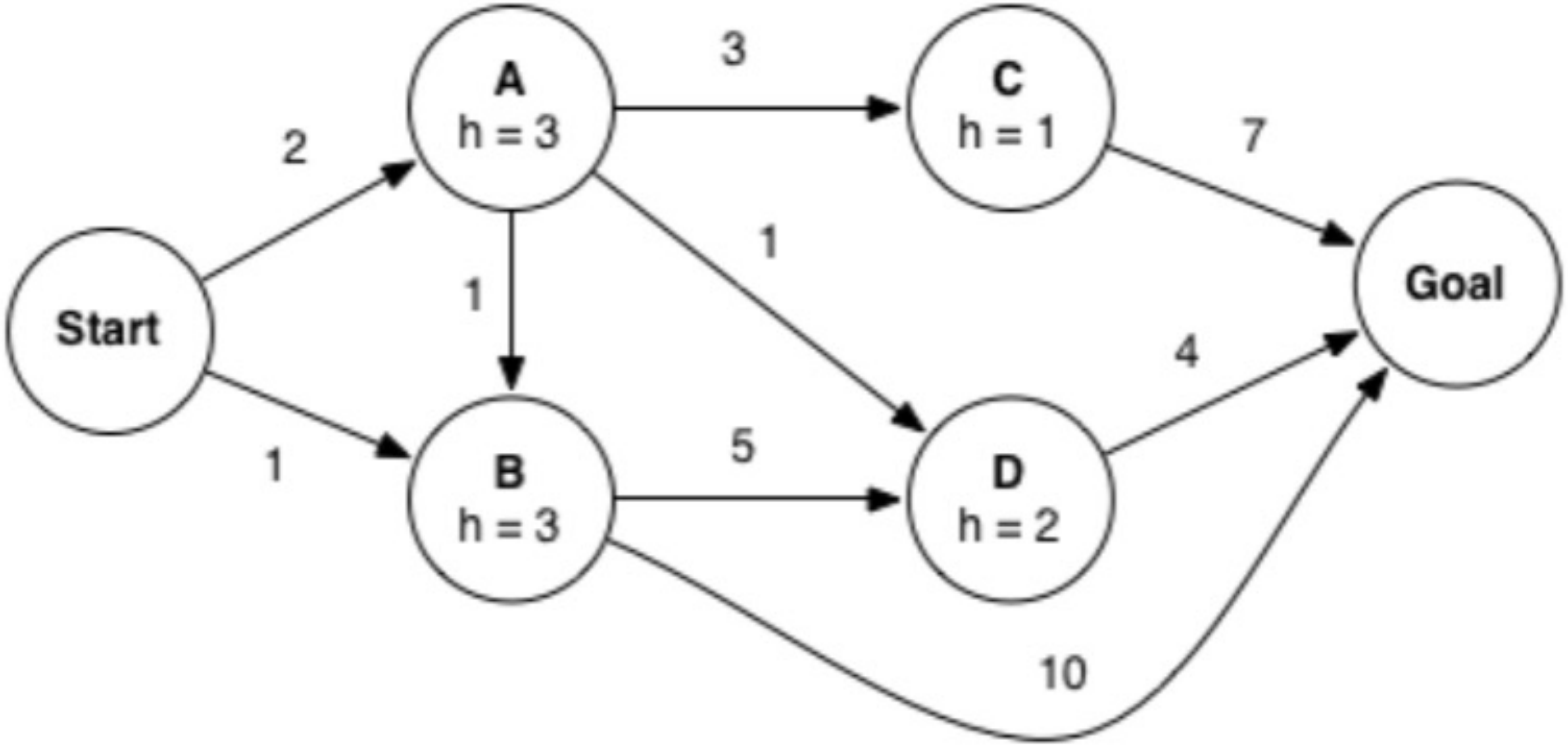
# Properties of Heuristics

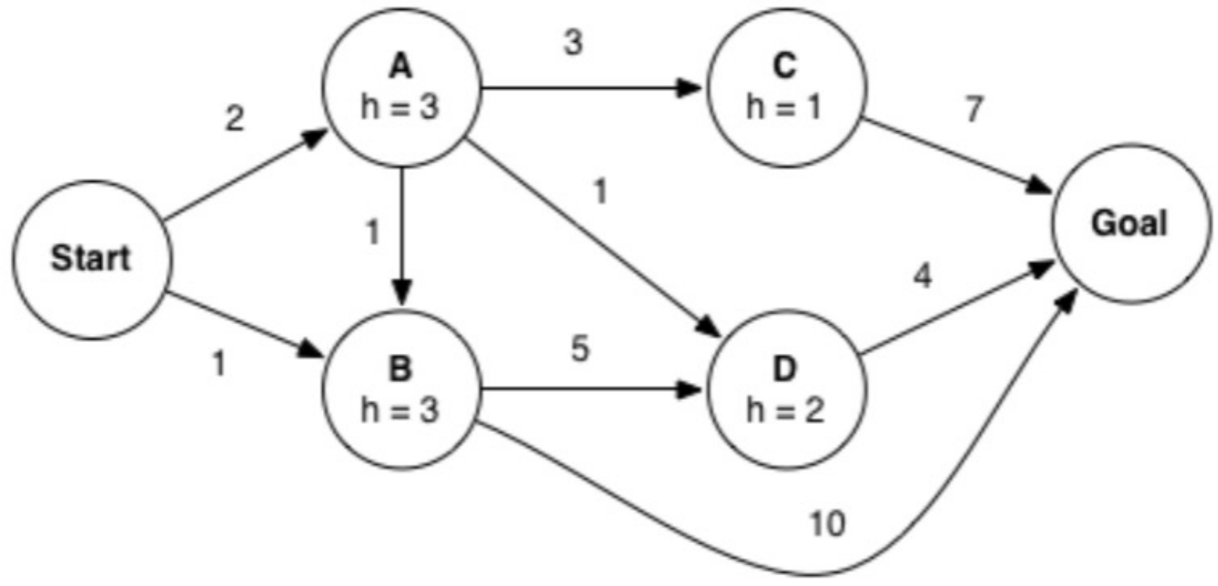
- Admissibility
  - $h(n) \leq h^*(n)$
    - $h^*(n)$: true cost from node n to goal state
  - Theorem: If *h(n)* is admissible, A* using TREE-SEARCH is optimal

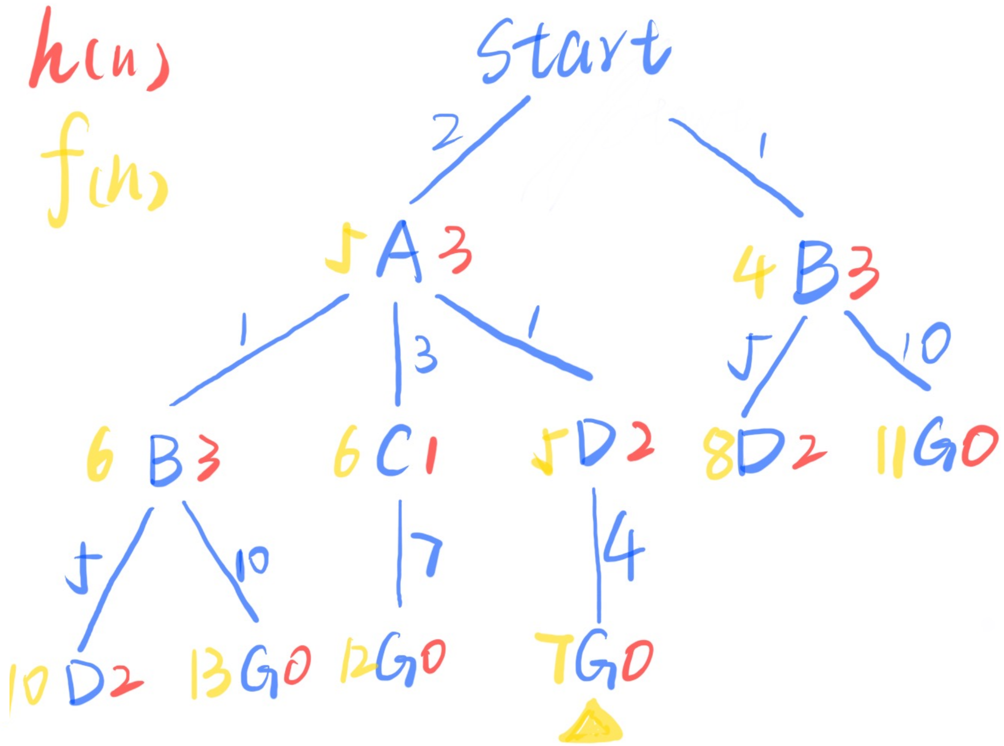
- Consistency (Monotonicity)
  - $h(n) \leq cost(n, n') + h(n')$ ($n'$ is successor of $n$)
    - $f(n)$ is non-decreasing along any path
    - Each node is reached, consistency guarantees that the path length to that point is equal to the minimal path-length from the start node.
    - The first goal node **selected for expansion** must be an optimal solution because f is the true cost for goal nodes
      (which have h = 0) and all later goal nodes will be at least as expensive.

- Consistency implies admissibility

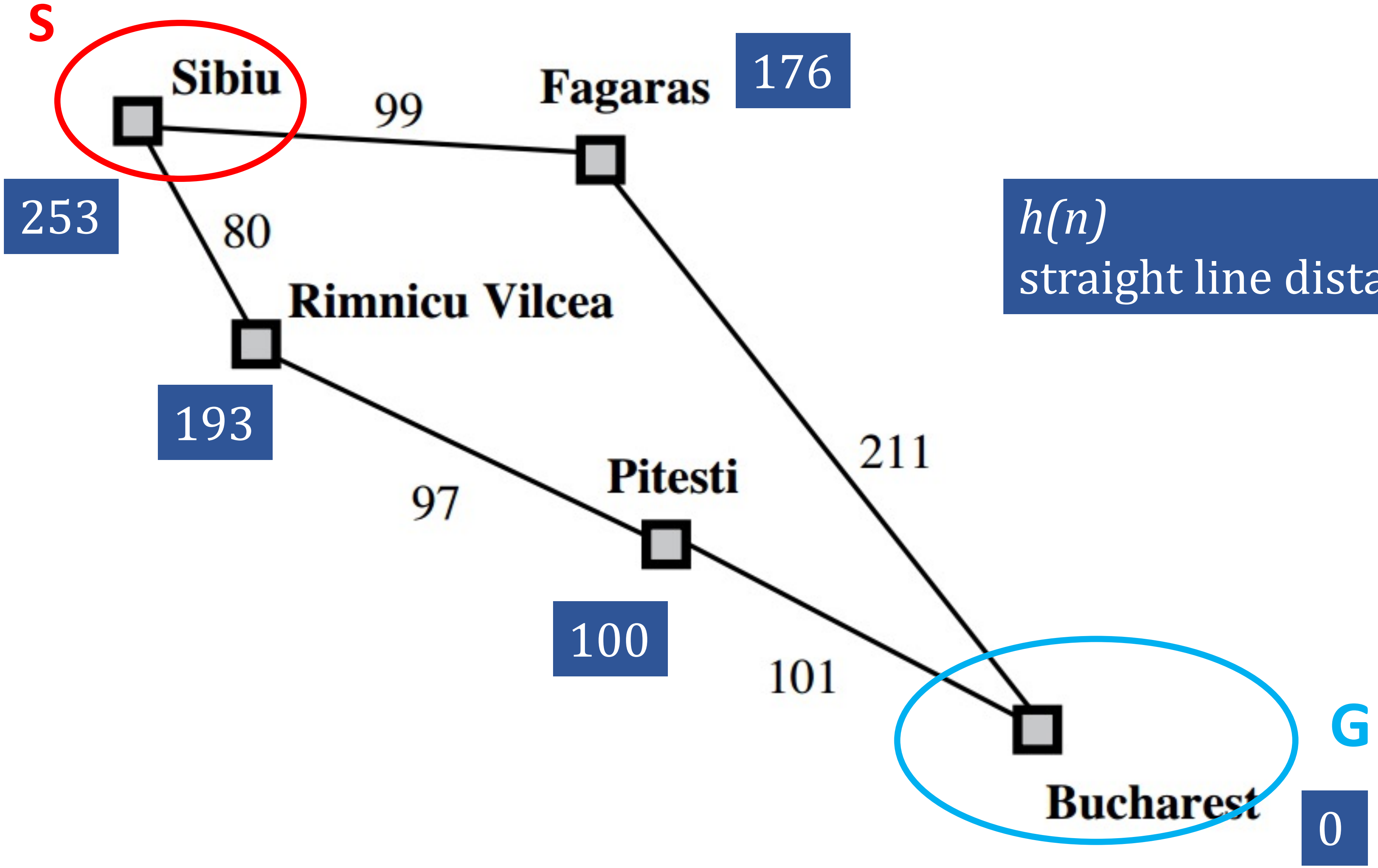
# Exercise – A*




Start
A
h = 3
B
h = 3
C
h = 1
D
h = 2
Goal
2
1
1
1
3
5
7
4
10

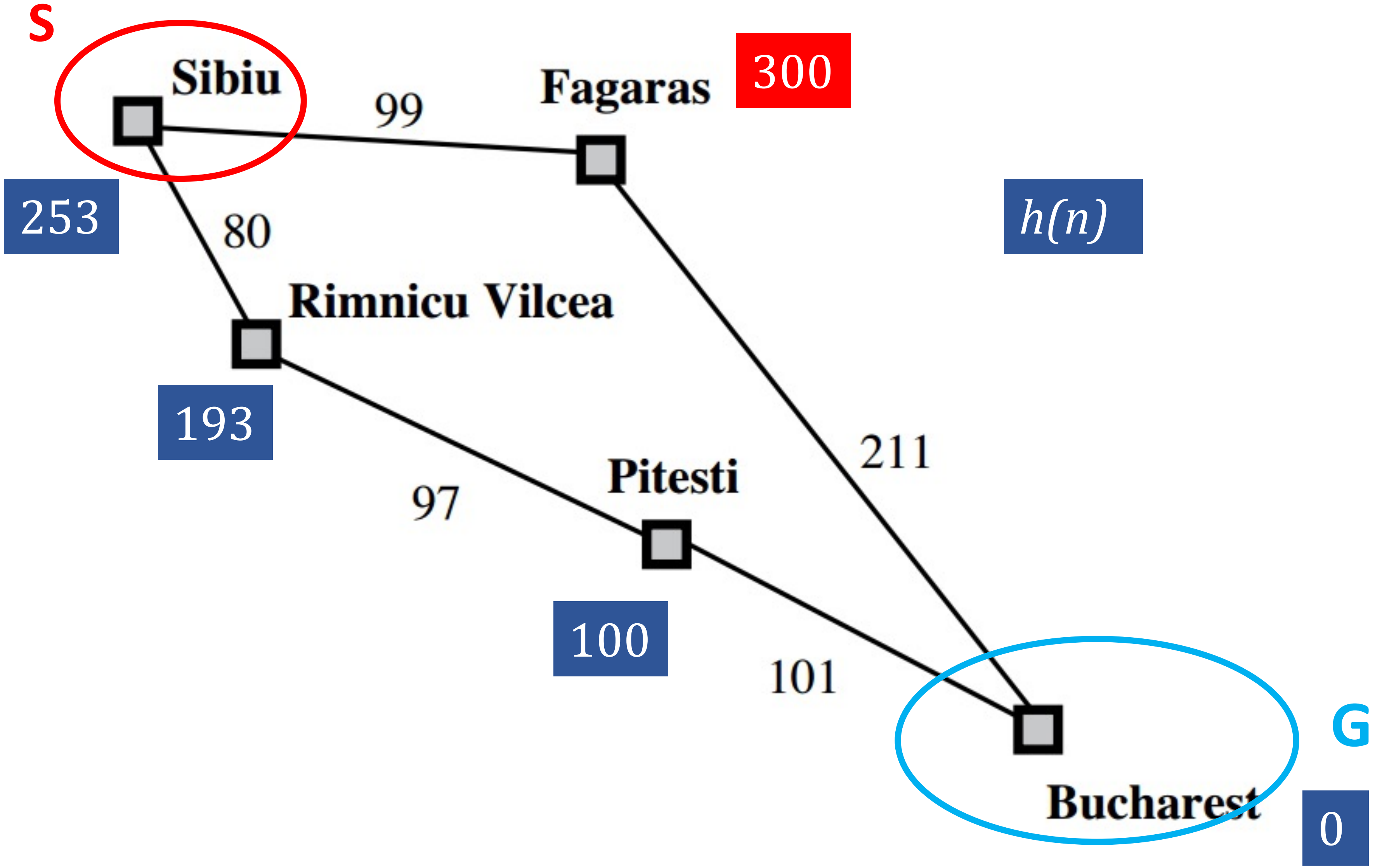

h(n)
f(n)
Start
2
1
5 A 3
4 B 3
1
3
1
5
10
6 B 3
6 C 1
5 D 2
8 D 2
11 G 0
5
10
7
4
10 D 2
13 G 0
12 G 0
7 G 0

# Is this h(n) admissible?

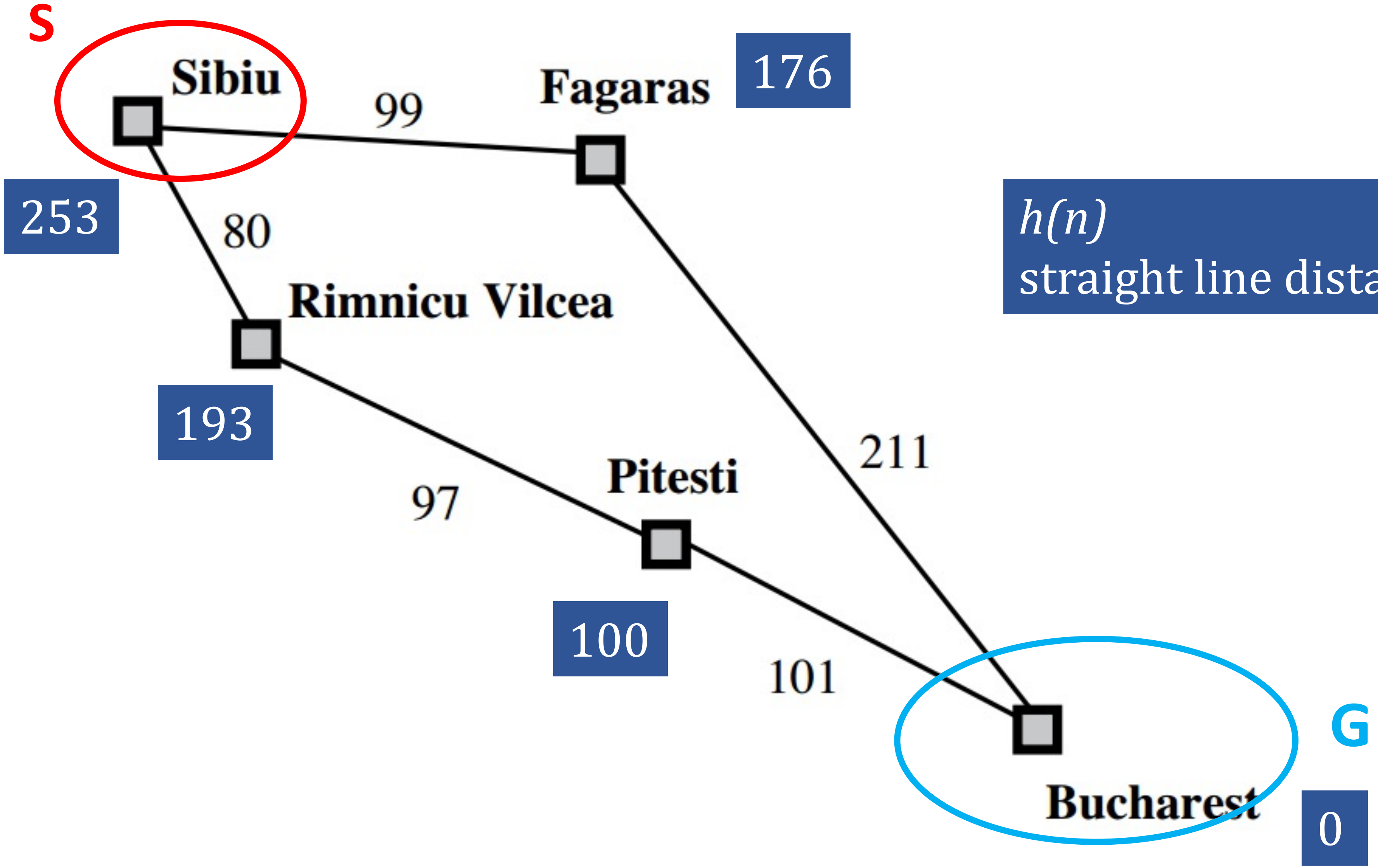



*h(n)*
straight line distance to Bucharest

# Is this h(n) admissible?

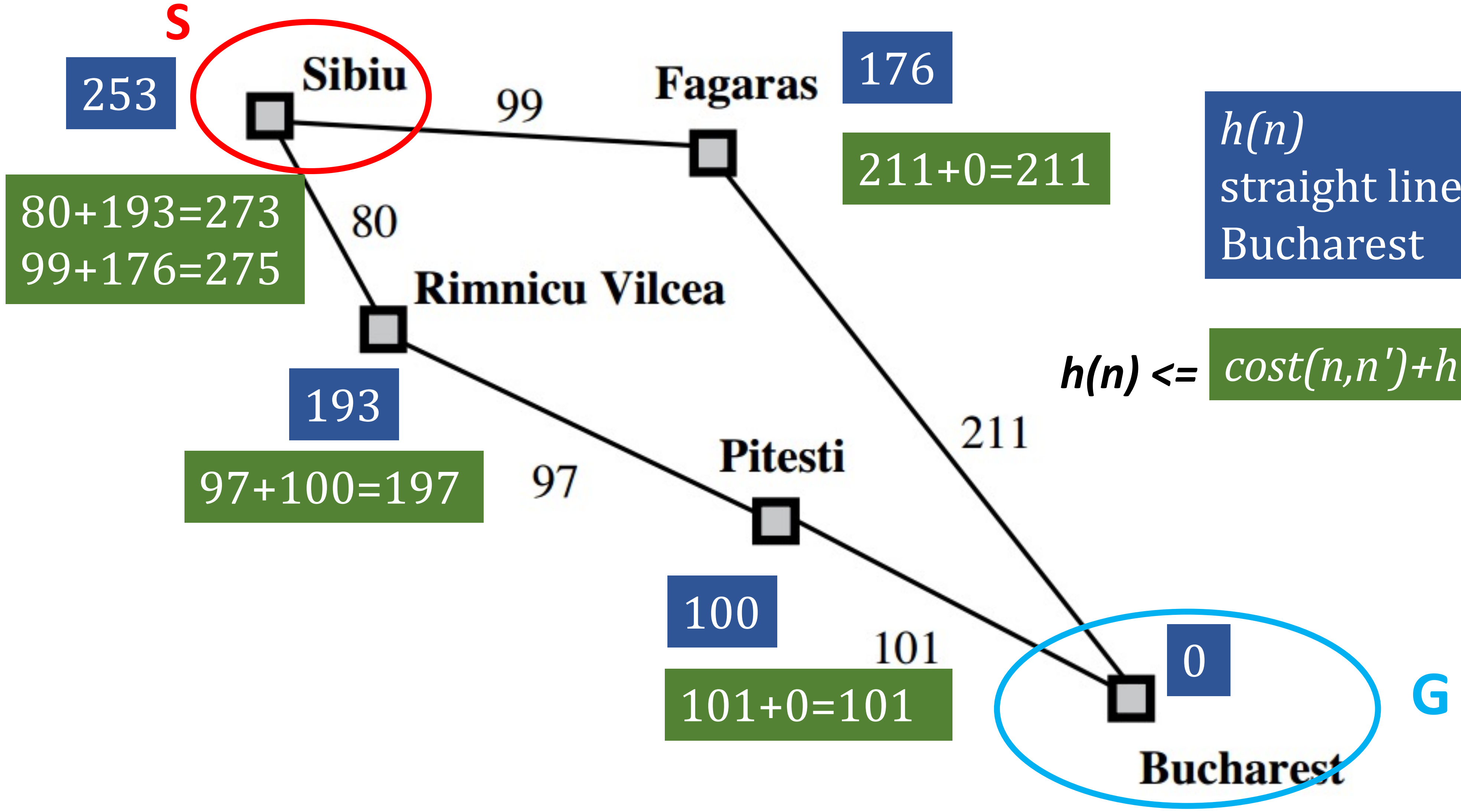


# Is this h(n) monotonic?




*h(n)*
straight line distance to Bucharest

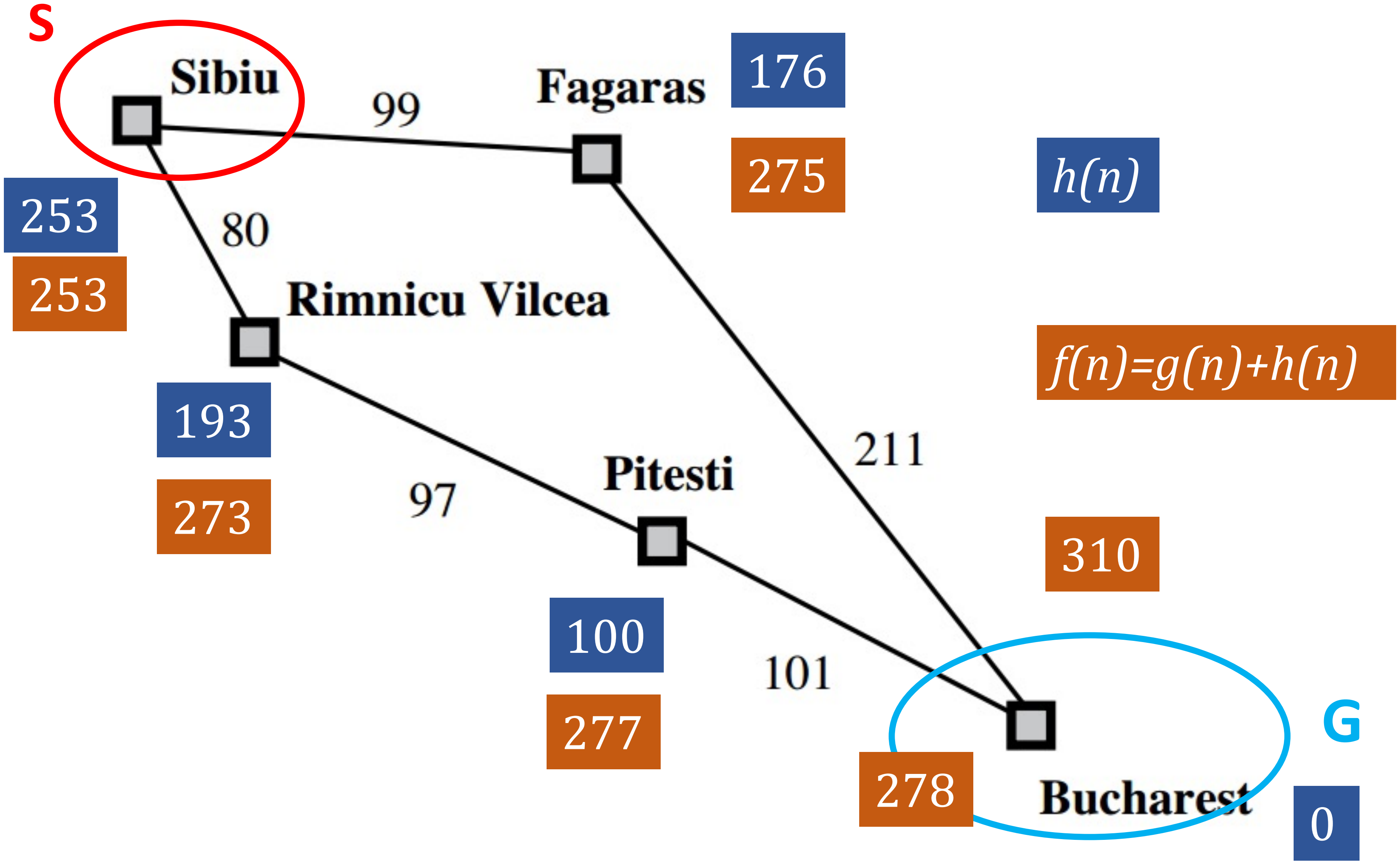
# Is this h(n) monotonic?




*h(n)*
straight line distance to Bucharest

**$h(n) <=$** $cost(n,n')+h(n')$

# Is A* optimal in this problem?



# Tree-Search and Graph-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

---

**function** GRAPH-SEARCH(*problem*) **returns** a solution, or failure
  initialize the frontier using the initial state of *problem*
  ***initialize the explored set to be empty***
  **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
    ***add the node to the explored set***
    expand the chosen node, adding the resulting nodes to the frontier
      ***only if not in the frontier or explored set***

# Important claims

- If *h(n)* is admissible, A* using TREE-SEARCH is optimal
- If *h(n)* is consistent, A* using GRAPH-SEARCH is optimal
  - Example: https://stackoverflow.com/questions/51684682/why-does-a-with-admissible-non-consistent-heuristic-find-non-optimal-solution

- **What's the difference between GRAPH-SEARCH and TREE-SEARCH?**
  - They both terminate when EXPANDING a goal node
  - GRAPH-SEARCH has an `explored set` so it won't expand the same node again

- Why would GRAPH-SEARCH + inconsistent h(n) may stop A* from being optimal?

# Example-Consistent & Inconsistent

```
       (8)
        A
       / \
+1    /   \ +3
     /     \
   B ----- C ----- D
  (7)  +1  (6)  +6  (0)
```

Consistent

```
       (8)
        A
       / \
+1    /   \ +3
     /     \
   B ----- C ----- D
  (7)  +1  (0)  +6  (0)
```

Inconsistent

# Optimality of A*

- A*+Admissibility => Optimality (Tree-Search)
- A*+Consistency => Optimality (Graph-Search)

Assumption:

arc costs are strictly positive

branching factor is finite

# Optimality of A* - Proof

- **Assume:** *h* admissible; *f* non-decreasing along any path.
  **Proof:**
  Assume C* is cost of optimal solution, G2 is suboptimal goal (so h(G2)=0)
  f(G2)=g(G2)+h(G2)=g(G2) > C*

Assume node *n* is some node on the optimal path
f(n)=g(n)+h(n)<= C*
So f(n)<= C* < f(G2)

so *n* will always be expanded before G2.