Discussion 4

Shirley Chen

10/18/2019

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 ϵ; [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 promising) 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 state costs in the priority queue
  - Often involves a heuristic function $h(n)$

- Implementation: Order nodes in fringe by $f(n)$

- 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?

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

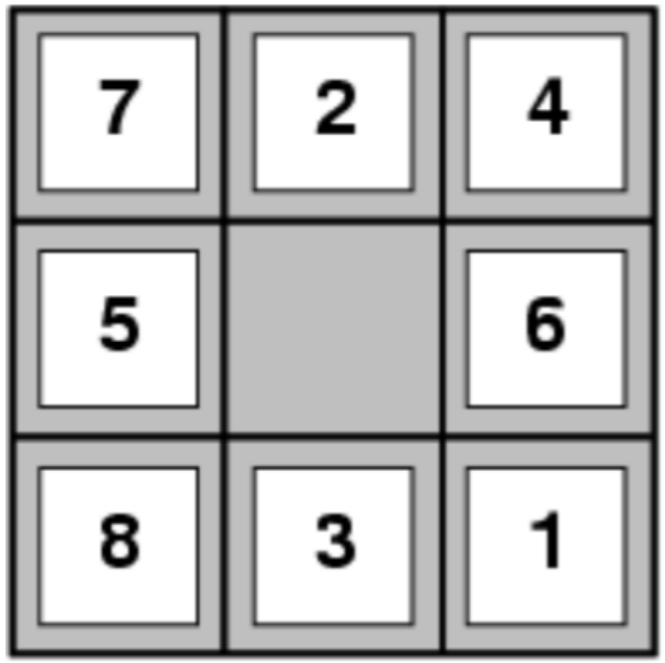
- h(n)=0

Is this admissible?

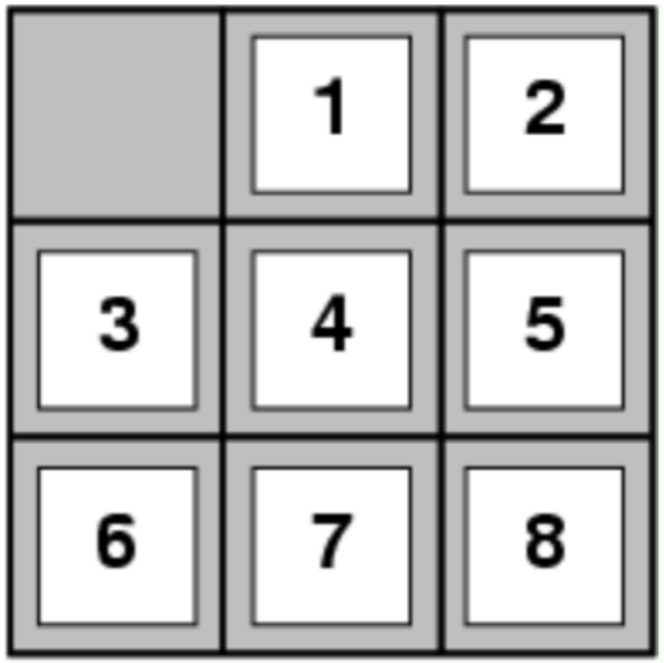
- h(n)=1

Is this admissible?

Exercise 1

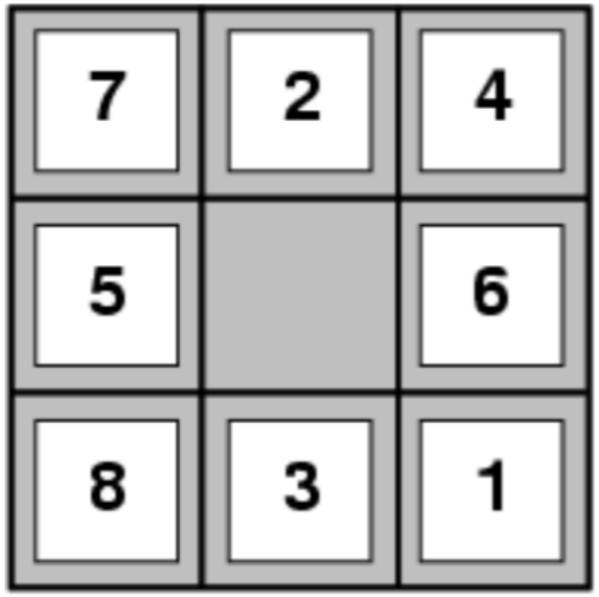


Start State

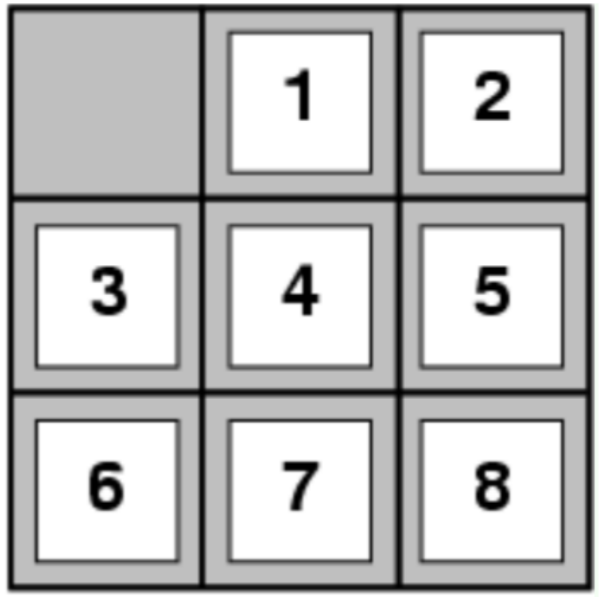


Goal State

Exercise 1



Start State

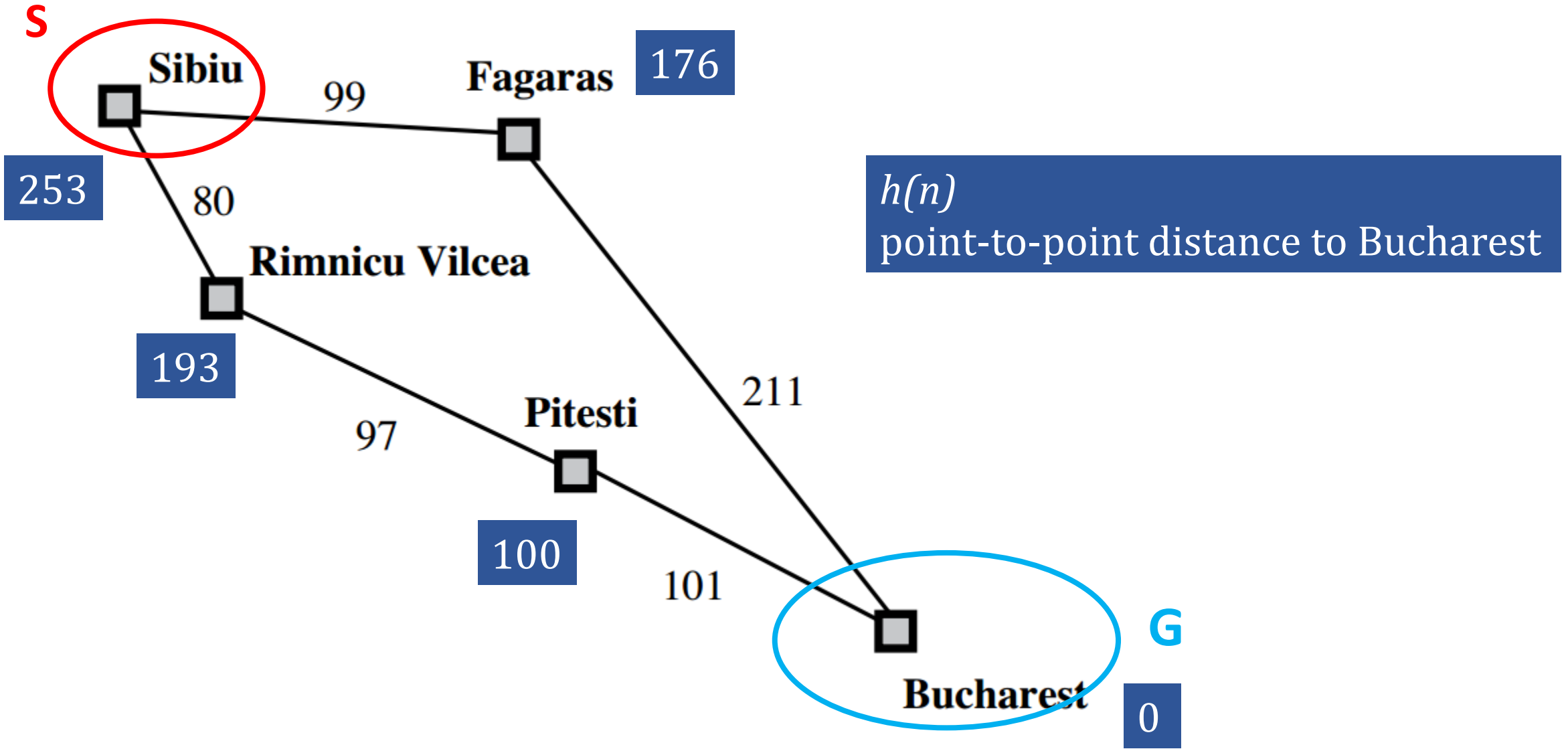


Goal State

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

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

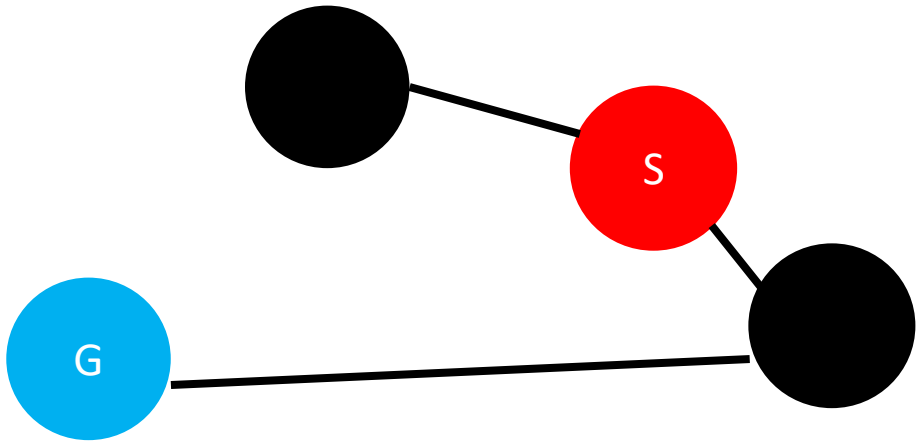
Example – point to point distance in route planning



h(n)
point-to-point distance to Bucharest

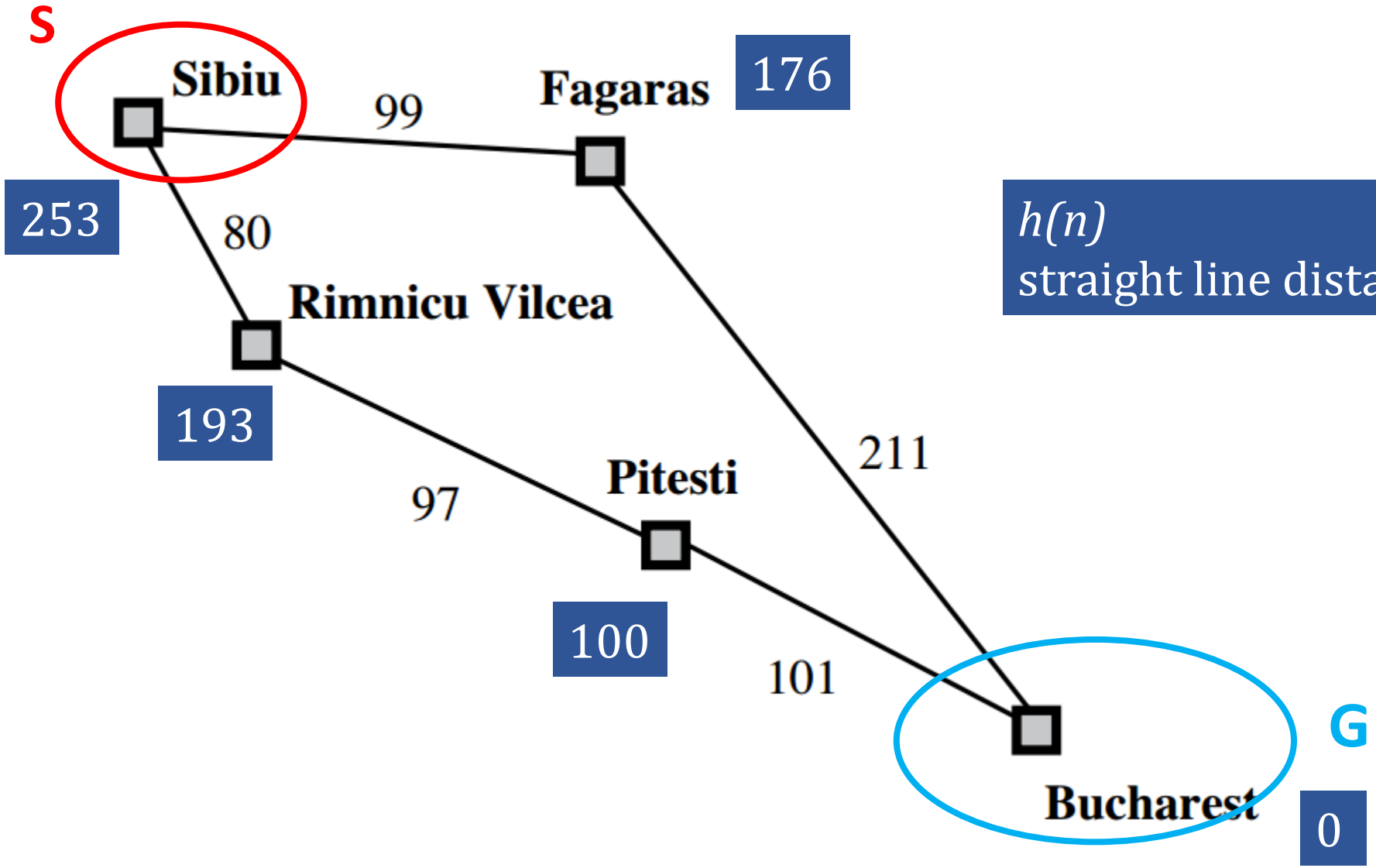
Greedy best-first search $f(n) = h(n)$

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



- Worst Time and Space complexity $O(b^m)$
  - A good heuristic can give dramatic improvement

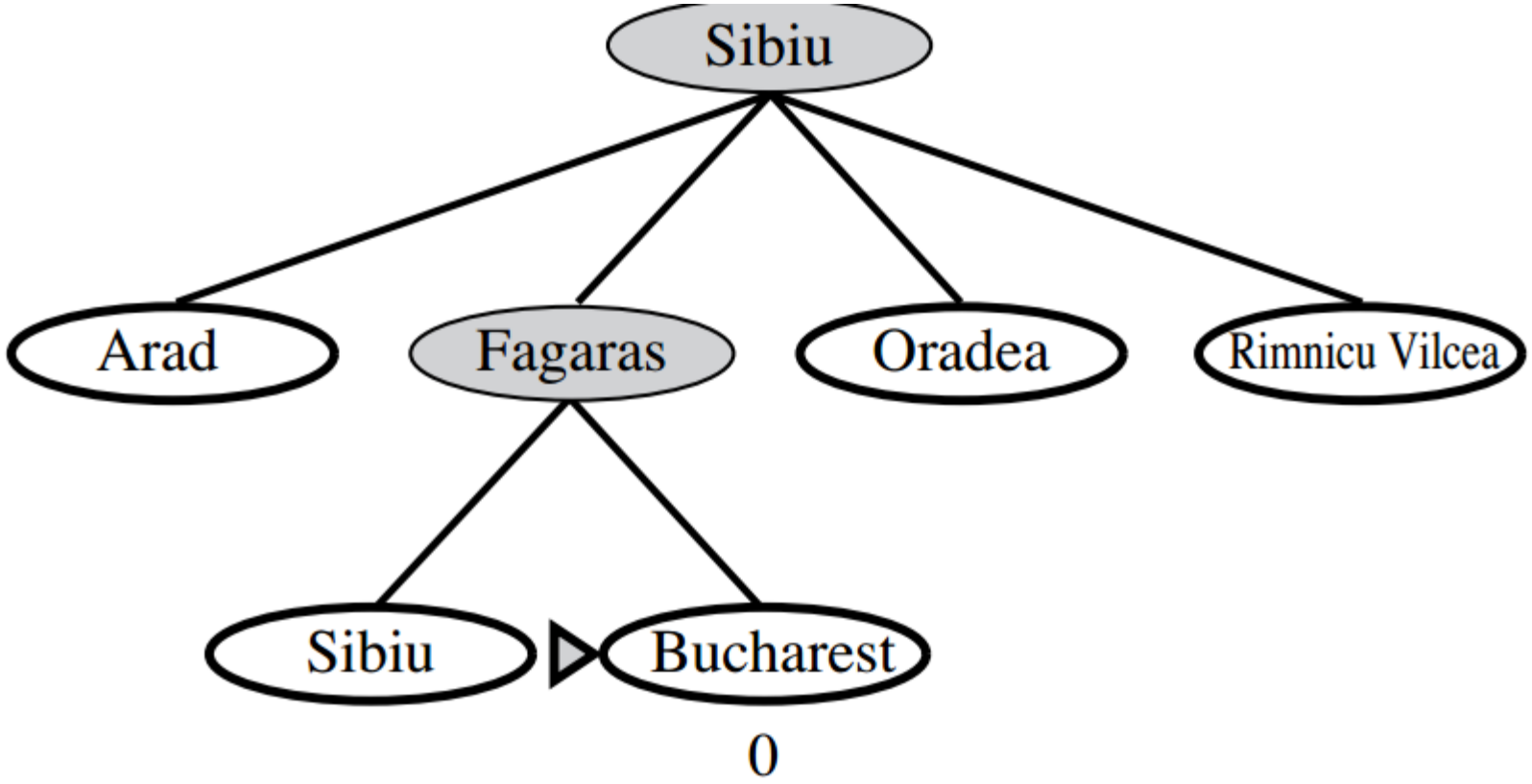
Greedy best-first search $f(n) = h(n)$



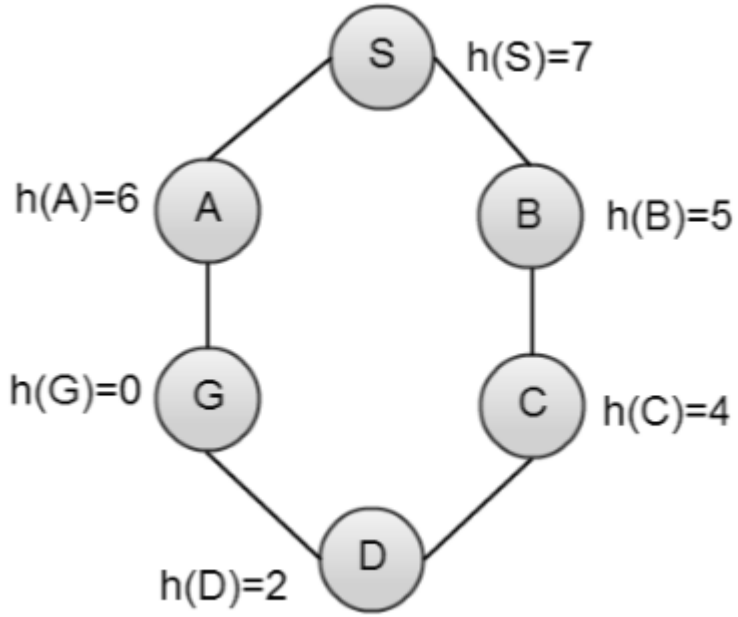
$h(n)$
straight line distance to Bucharest

Example - Straight-line distance in route planning

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



Greedy best-first search



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* togoal

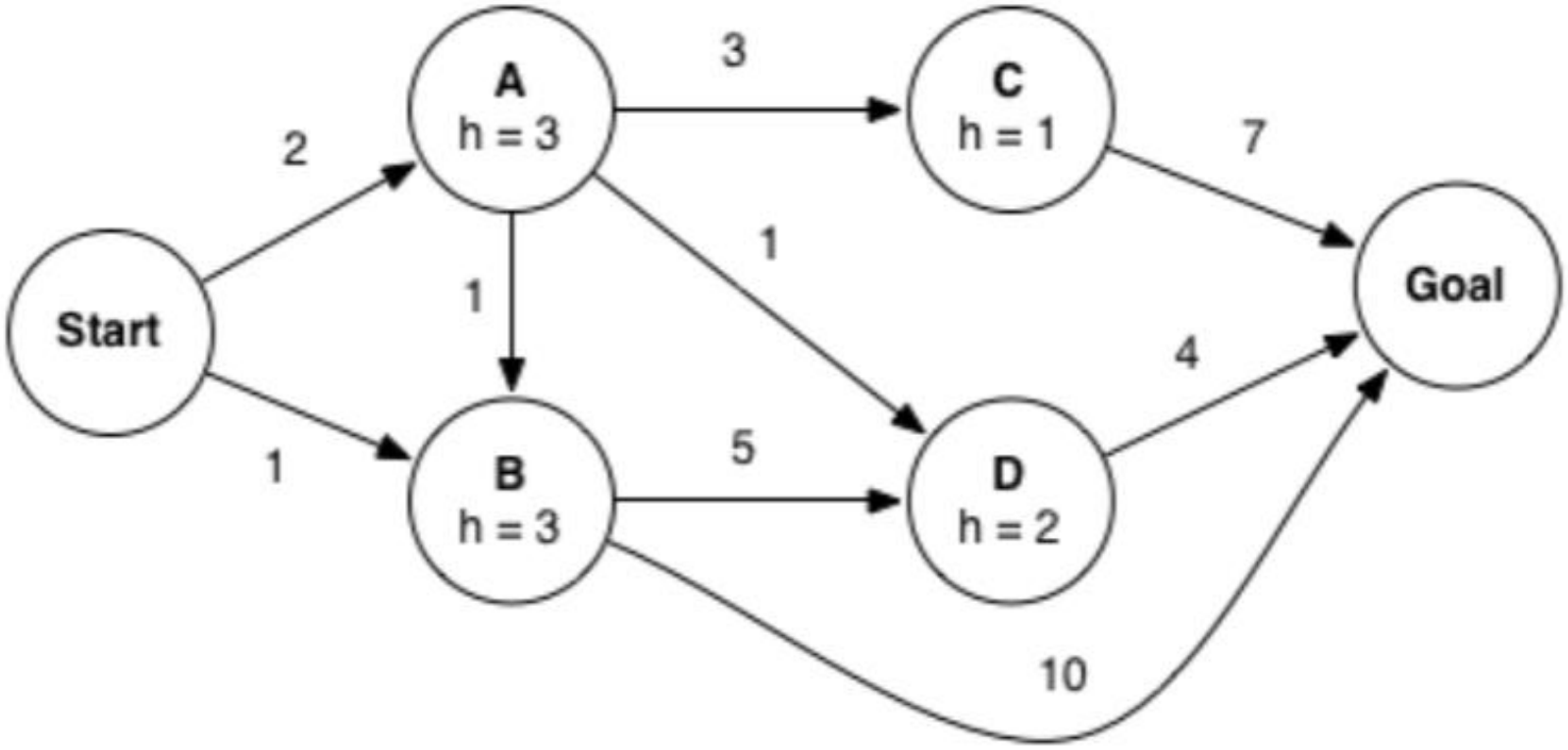
A∗ expands no nodes with f(n) > C∗

Properties of Heuristics

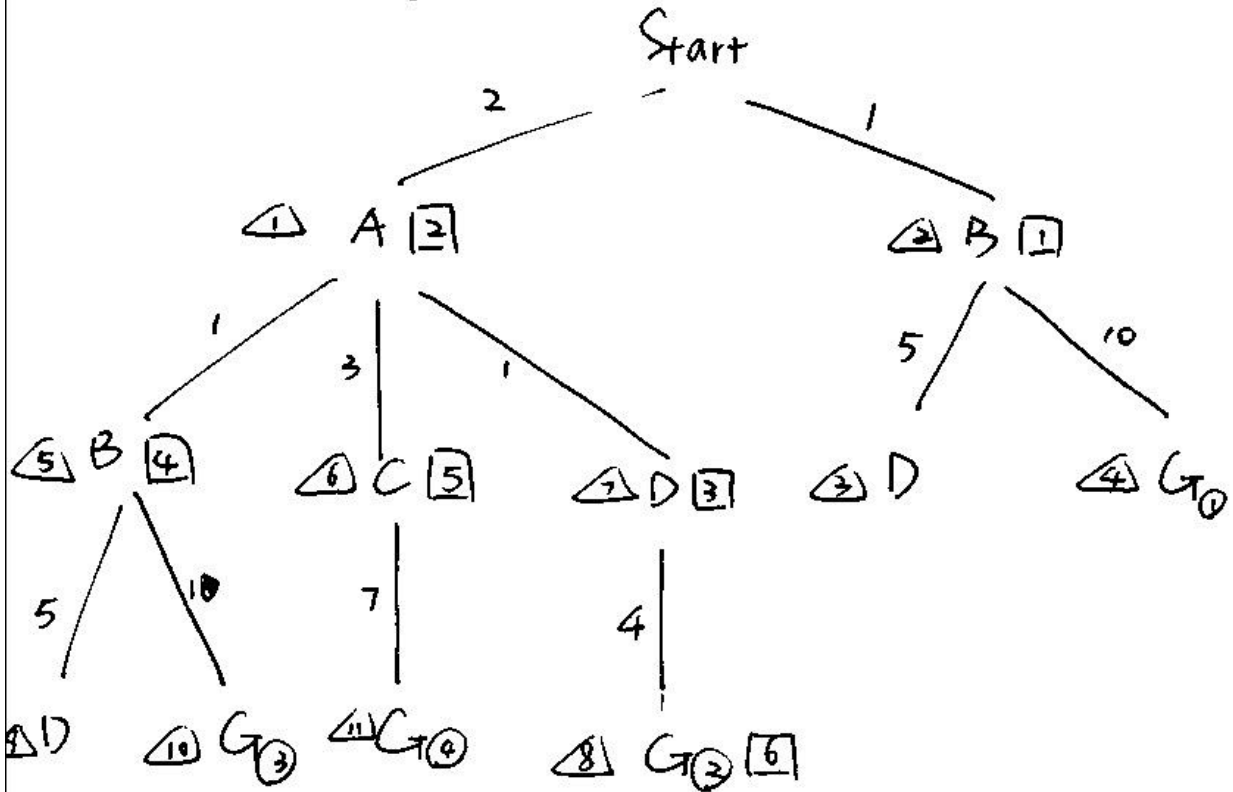
- Admissibility
  - $h(n) \leq h^*(n)$
    - $h^*(n)$: true cost from node n to goal state
  - Theorem: If *h(n)* is admissible, A* using TREE-SEARCH is optimal

- 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.
  - What does this mean?

Exercise – A*



A*. Tree-Search.



Fringe.

~~① B: 1+3=4~~ ③ ~~D: 3+2=5~~

② A: 2+3=5 ④ ~~B: 3+3=6~~

⑤ ~~C: 5+1=6~~

⑥ ~~G②: 7~~

D: 6+2=8

D: 8+2=10

G①: 11 G④: 12

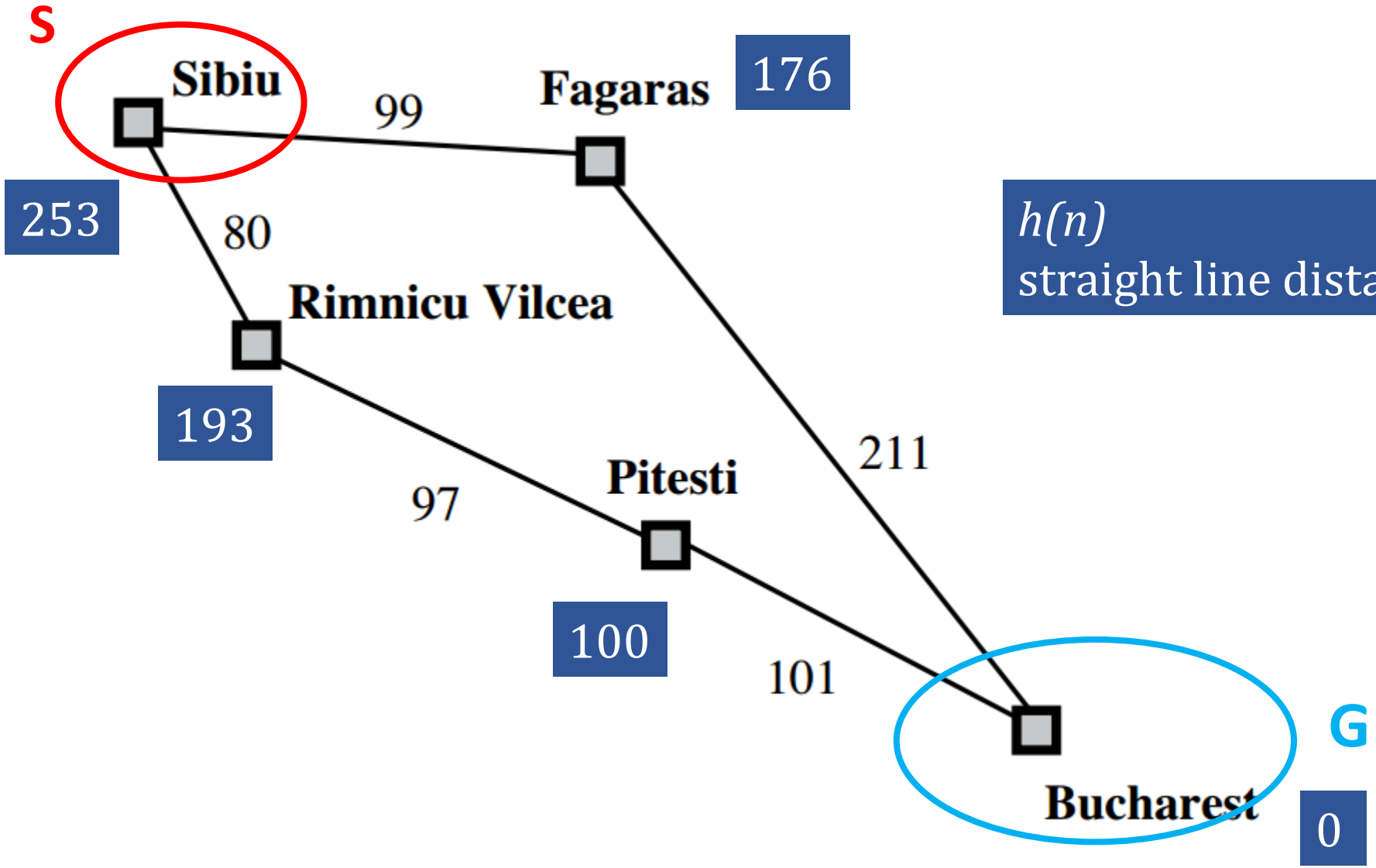
G③: 13

Expand G②. Algorithm terminates.

(No need to clear fringe)

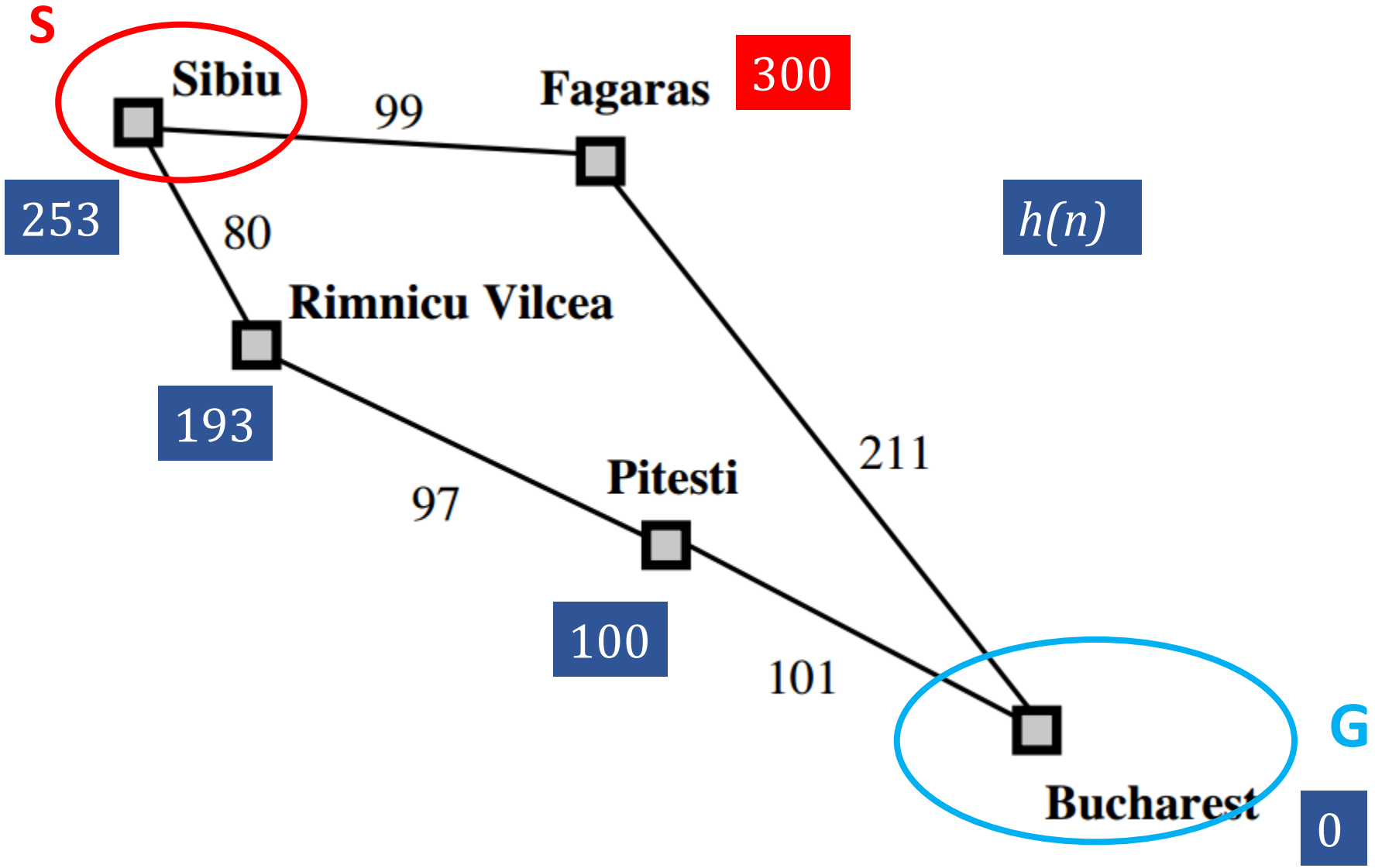
(Different from discussion talk!)

Is this h(n) admissible?

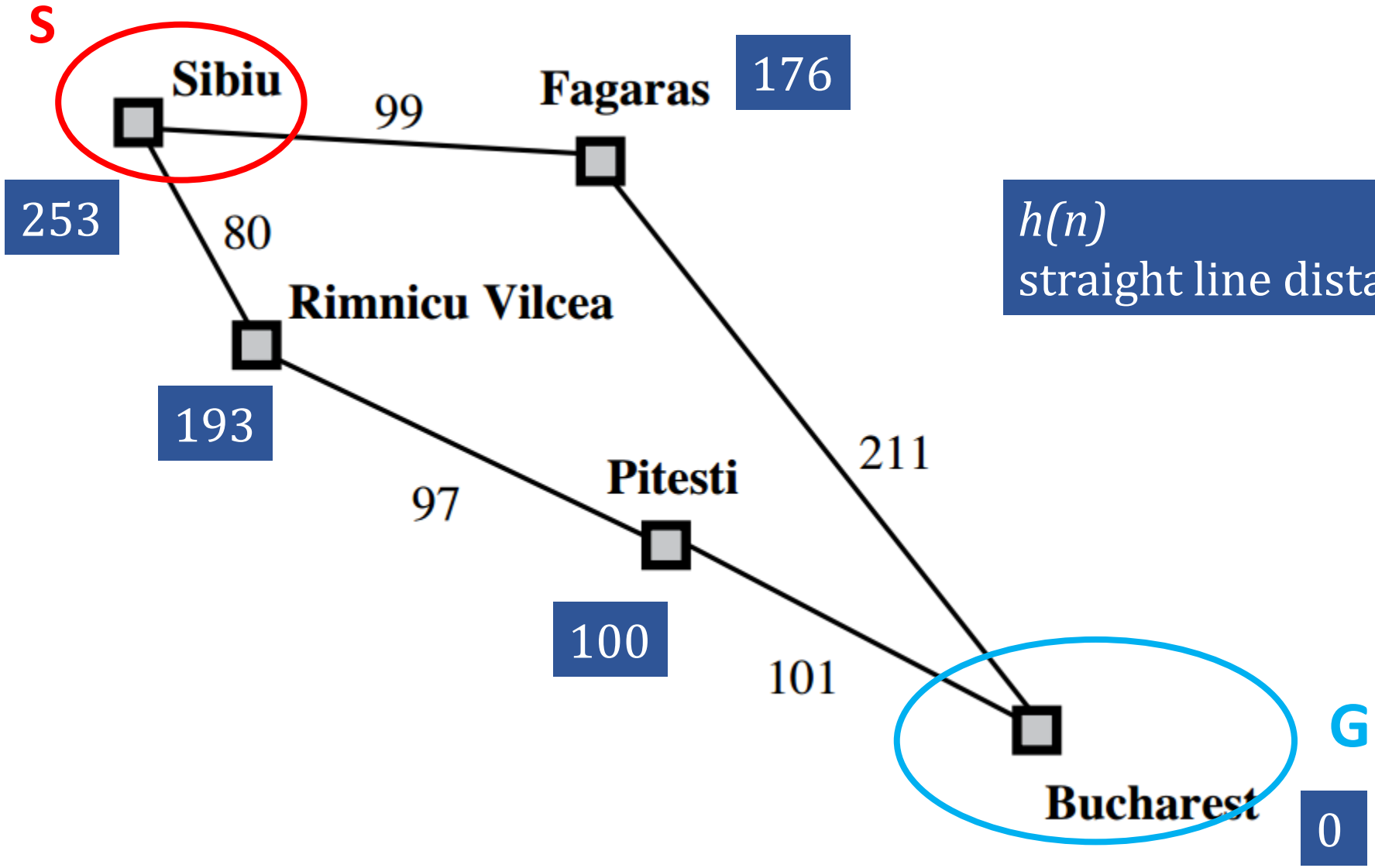


h(n)
straight line distance to Bucharest

Is this h(n) admissible?

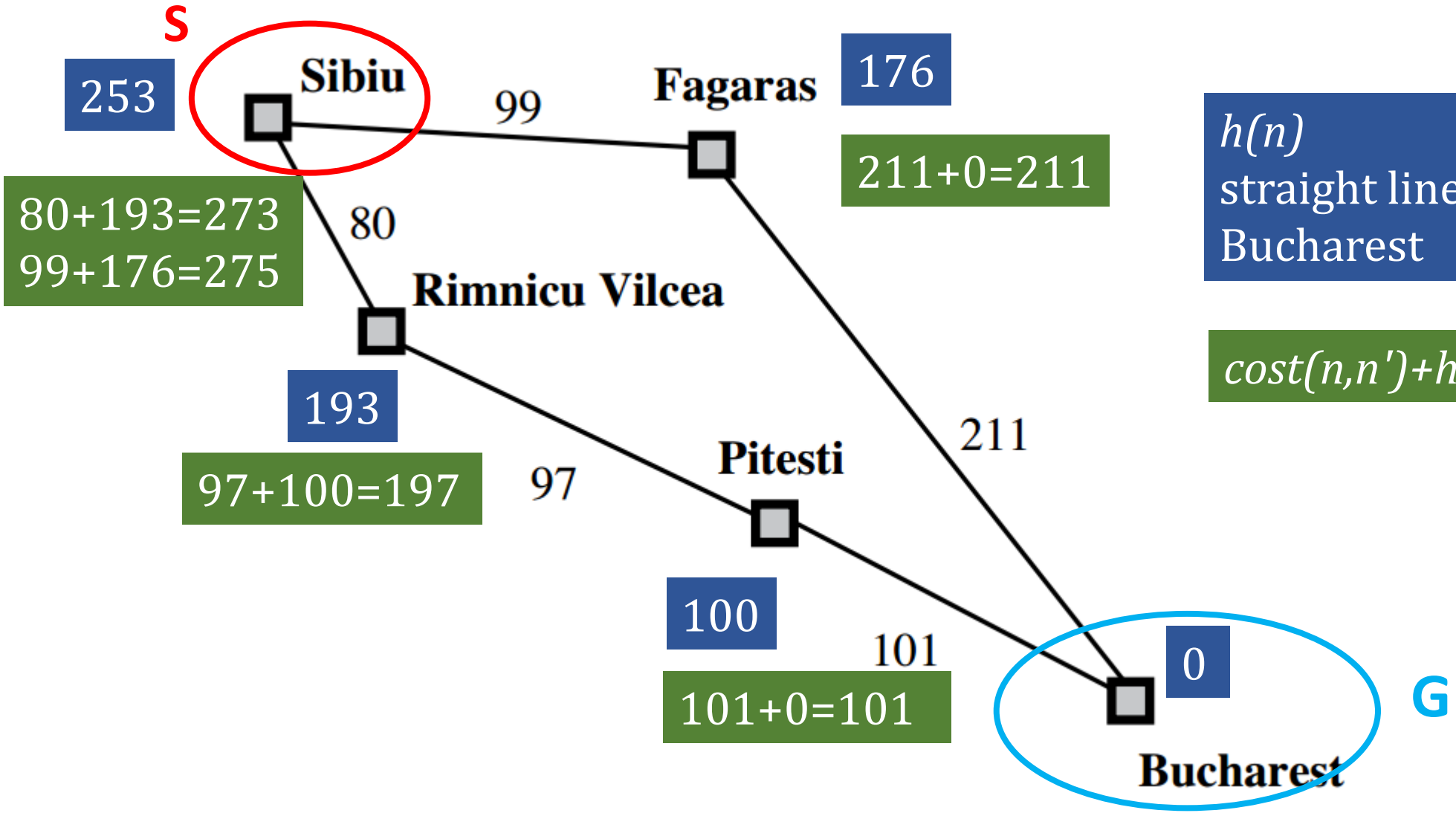


Is this h(n) monotonic?



h(n)
straight line distance to Bucharest

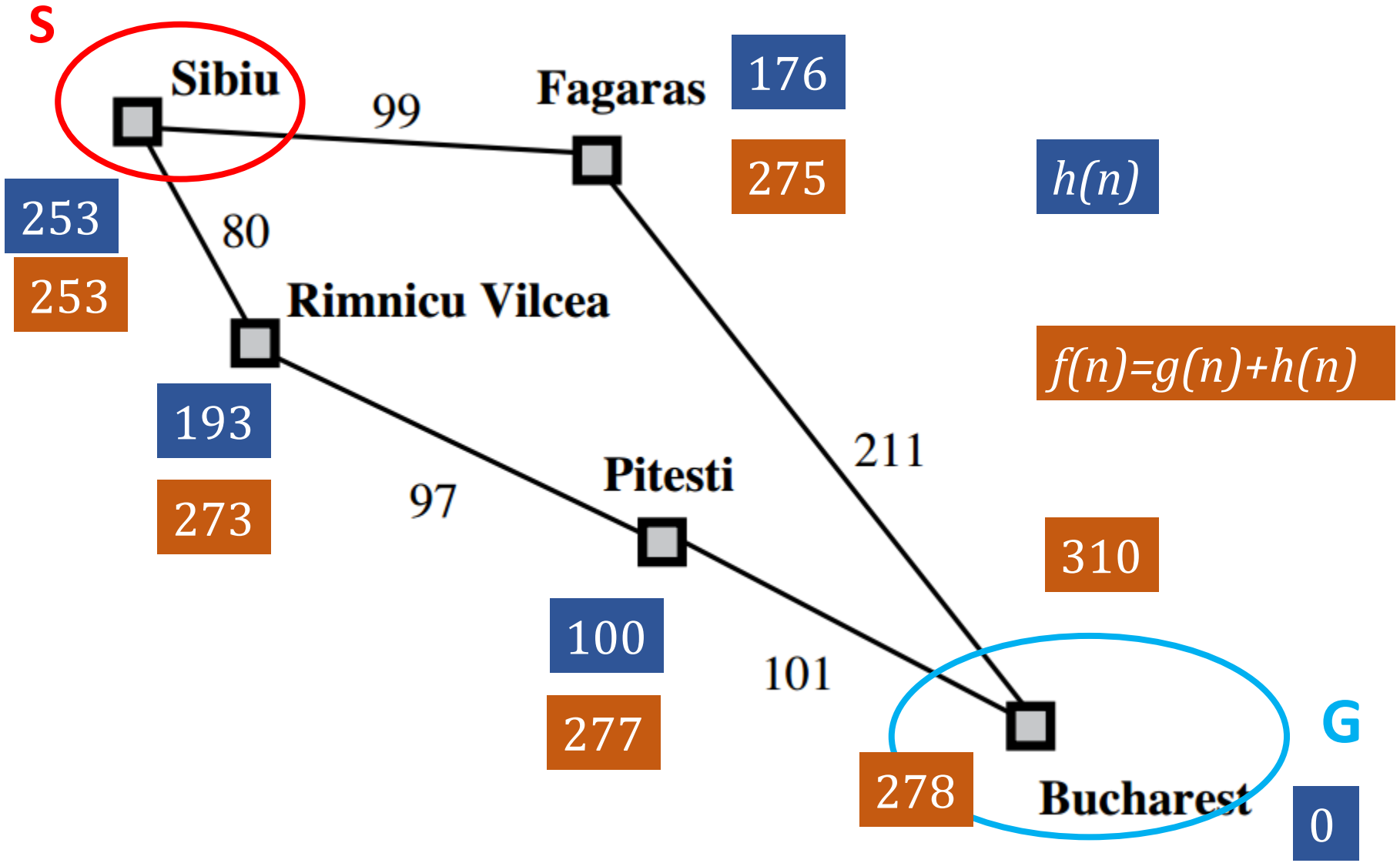
Is this h(n) monotonic?



h(n)
straight line distance to Bucharest

cost(n,n')+h(n')

Is A* optimal in this problem?



Important claims

- If *h(n)* is admissible, A* using TREE-SEARCH is optimal
- If *h(n)* is consistent, A* using GRAPH-SEARCH is optimal

- **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?

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

Some Claims

- Consistency implies admissibility
  - Why?

Optimality of A*

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

Assumption:

arc costs are strictly positive

branching factor is finite

Some Claims

Theorem 1: If *h(n)* is admissible, A* using TREE-SEARCH is optimal

Theorem 2: If *h(n)* is consistent, A* using GRAPH-SEARCH is optimal

- What's the difference between GRAPH-SEARCH and TREE-SEARCH?
- Why would GRAPH-SEARCH + inconsistent h(n) may stop A* from being optimal?