CS 161
Discussion 3
Informed Search

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.

However, uninformed strategies are incredibly inefficient in most cases …

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?

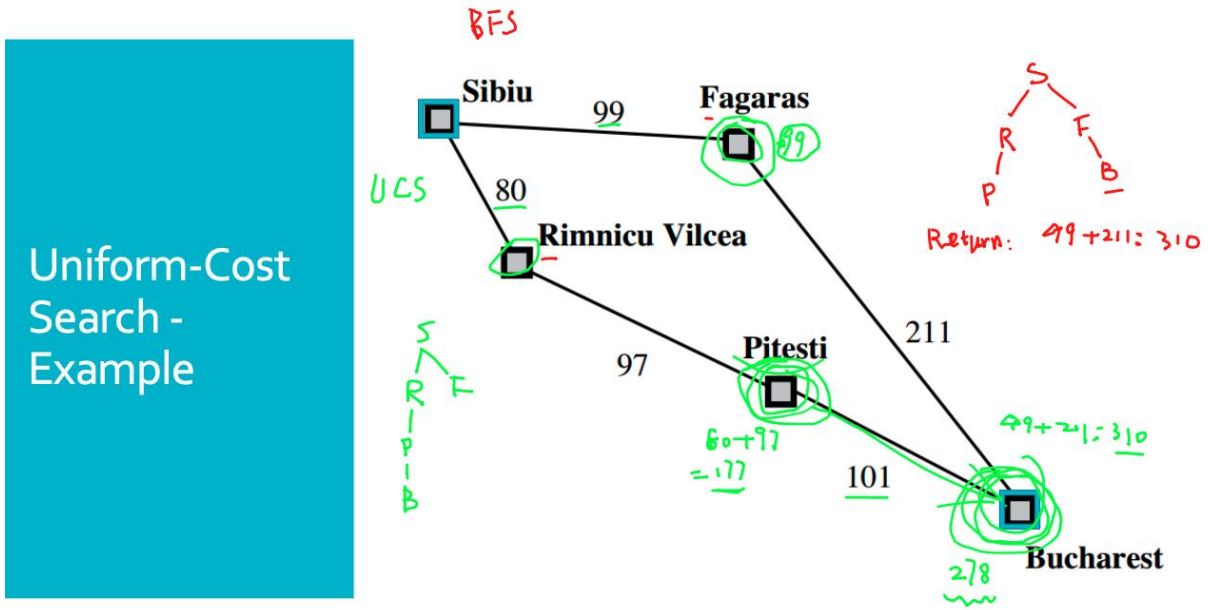
- Evaluation function $f(n)$ (A cost estimation for node n)
  - n is a node, not a state!
    - In uniform-cost search (an uninformed search strategy) we store state costs in the priority queue
  - Often involves a heuristic function $h(n)$
- Implementation: Order nodes in fringe by $f(n)$
- Two basic approaches:
  - Greedy best-first search
  - A* search

Is uniform-cost search informed search?

Is uniform-cost search informed search?

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

Uniform-Cost Search - Example



Greedy Best-First Search

- Complete?
- Optimal?
- Worst Time and Space complexity?

Demo: https://qiao.github.io/PathFinding.js/visual/

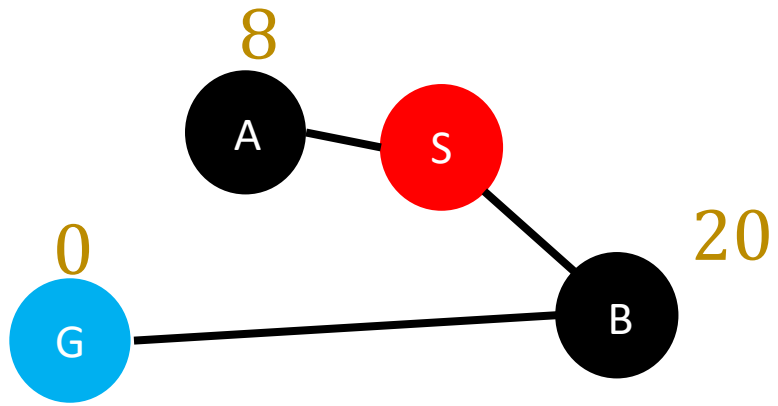
Greedy Best-First Search

- 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

- Complete?
  - No.
    - Can get stuck in loops
- Optimal?
  - No
- Worst Time and Space complexity $O(b^m)$
  - A good heuristic can give dramatic improvement

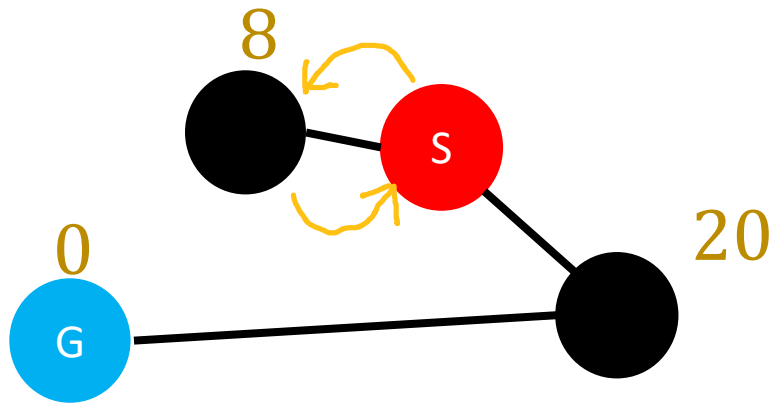
$$f(n) = h(n)$$



Greedy Best-First Search

- Complete?
  - No.
    - Can get stuck in loops
- Optimal?
  - No
- Worst Time and Space complexity $O(b^m)$
  - A good heuristic can give dramatic improvement

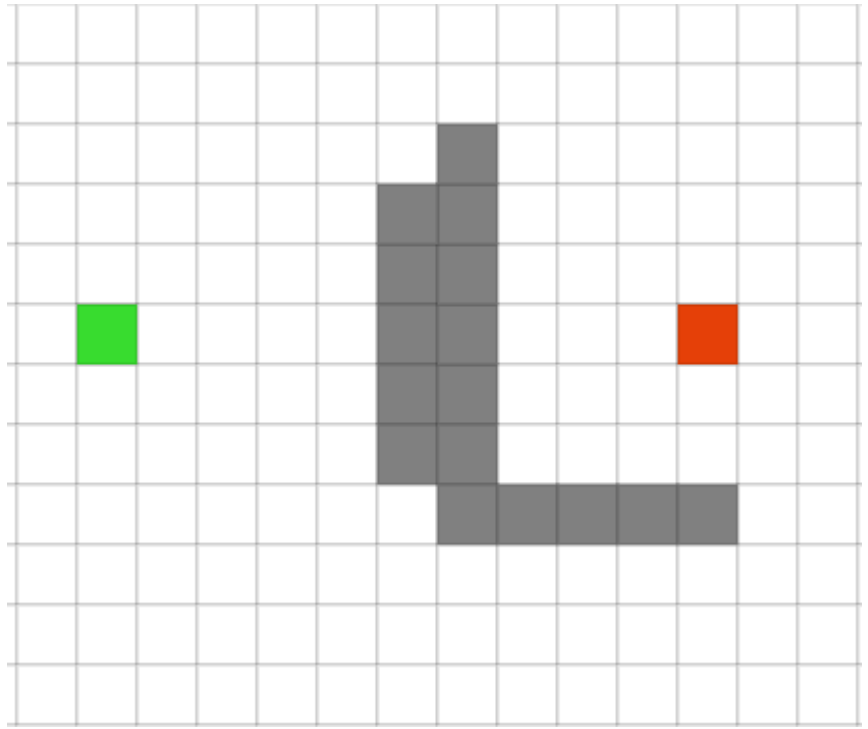
$$f(n) = h(n)$$



Greedy Best-First Search

Greedy Best first search

- Want: Path Green to Red
- Heuristic: Manhattan Distance to Red vs current node
- Which direction would we go? Is it optimal?

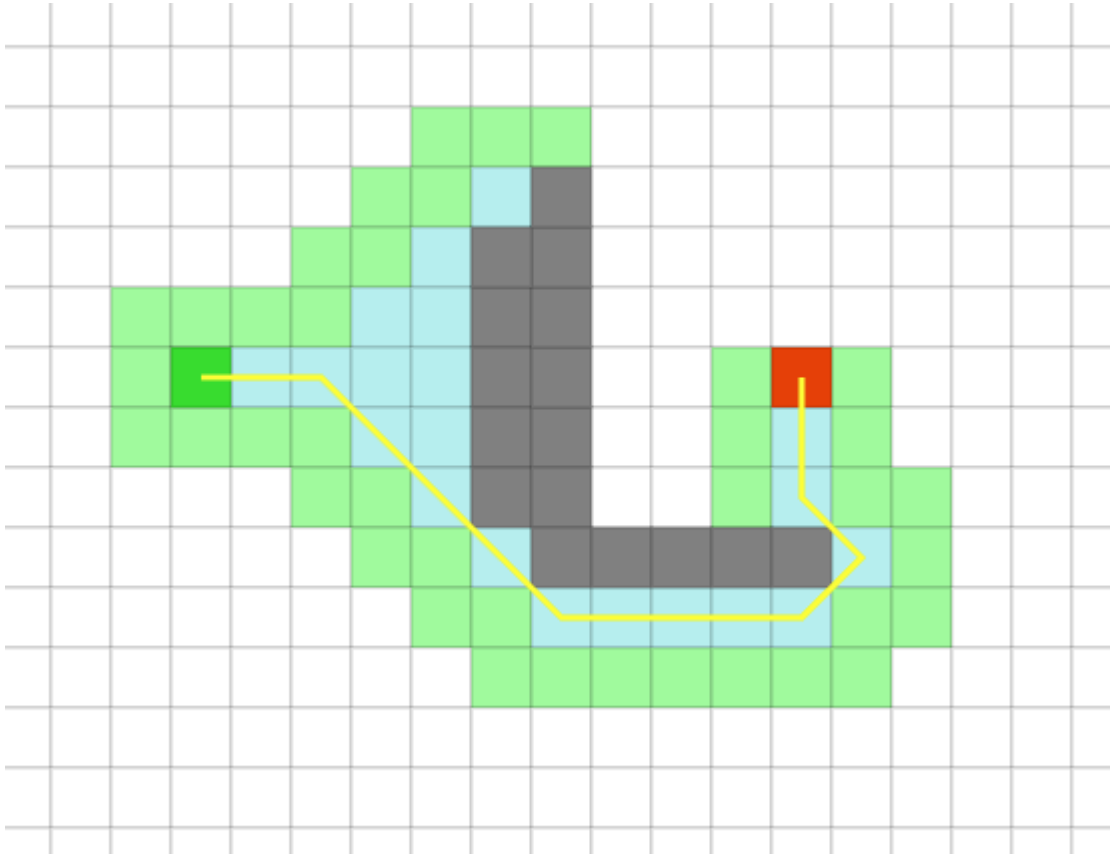


Greedy Best-First Search

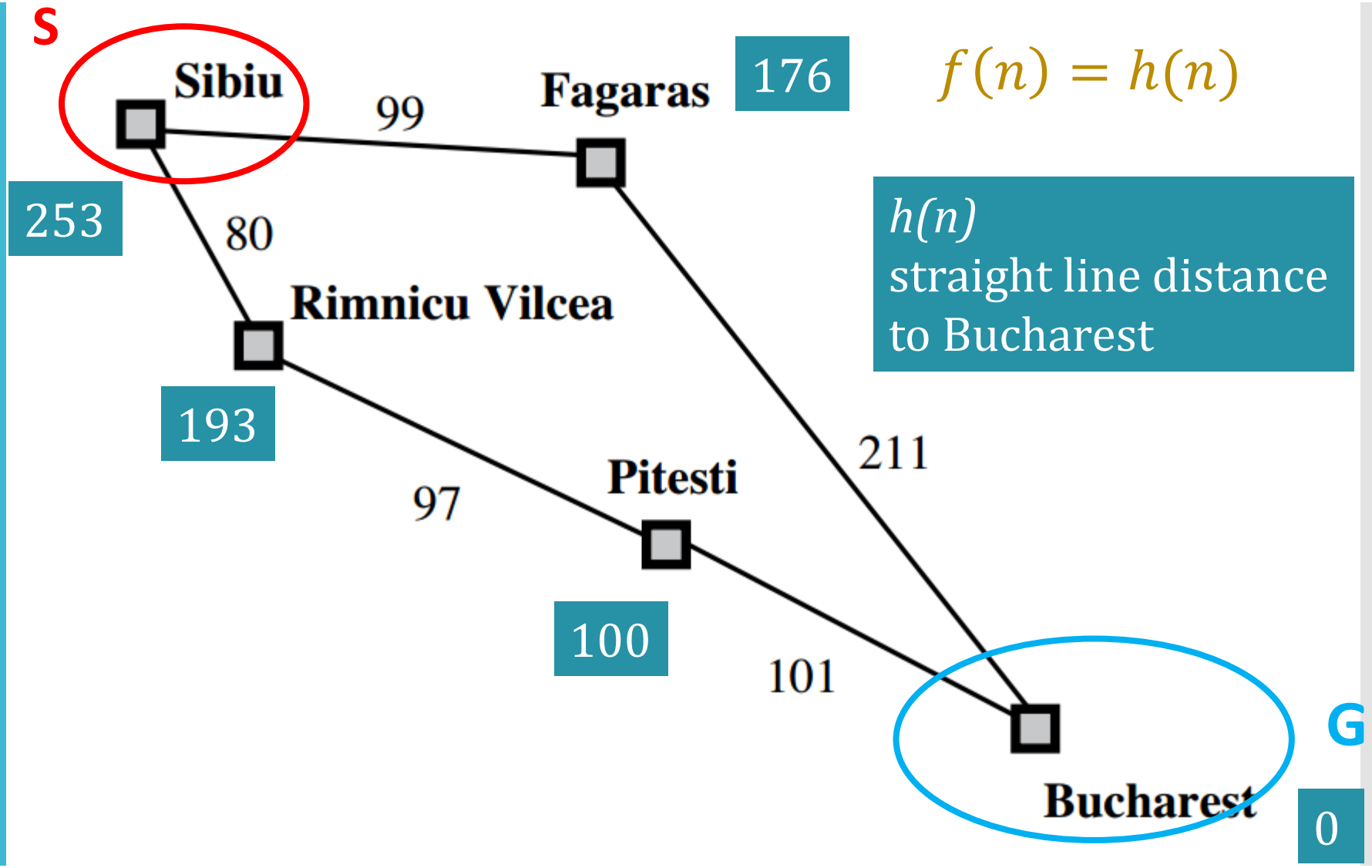
Greedy Best first search

- Want: Path Green to Red
- Heuristic: Manhattan Distance to Red vs current node
- Which direction would we go? Is it optimal?

Down, Not Optimal



Example:
Straight-line distance in route planning



A* Search

- Avoid expanding paths that are already expensive
- Evaluation function: $f(n) = g(n) + h(n)$
  - *n: node*
  - *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
- Expands no nodes with $f(n)$ >= f^*

Properties of Heuristics

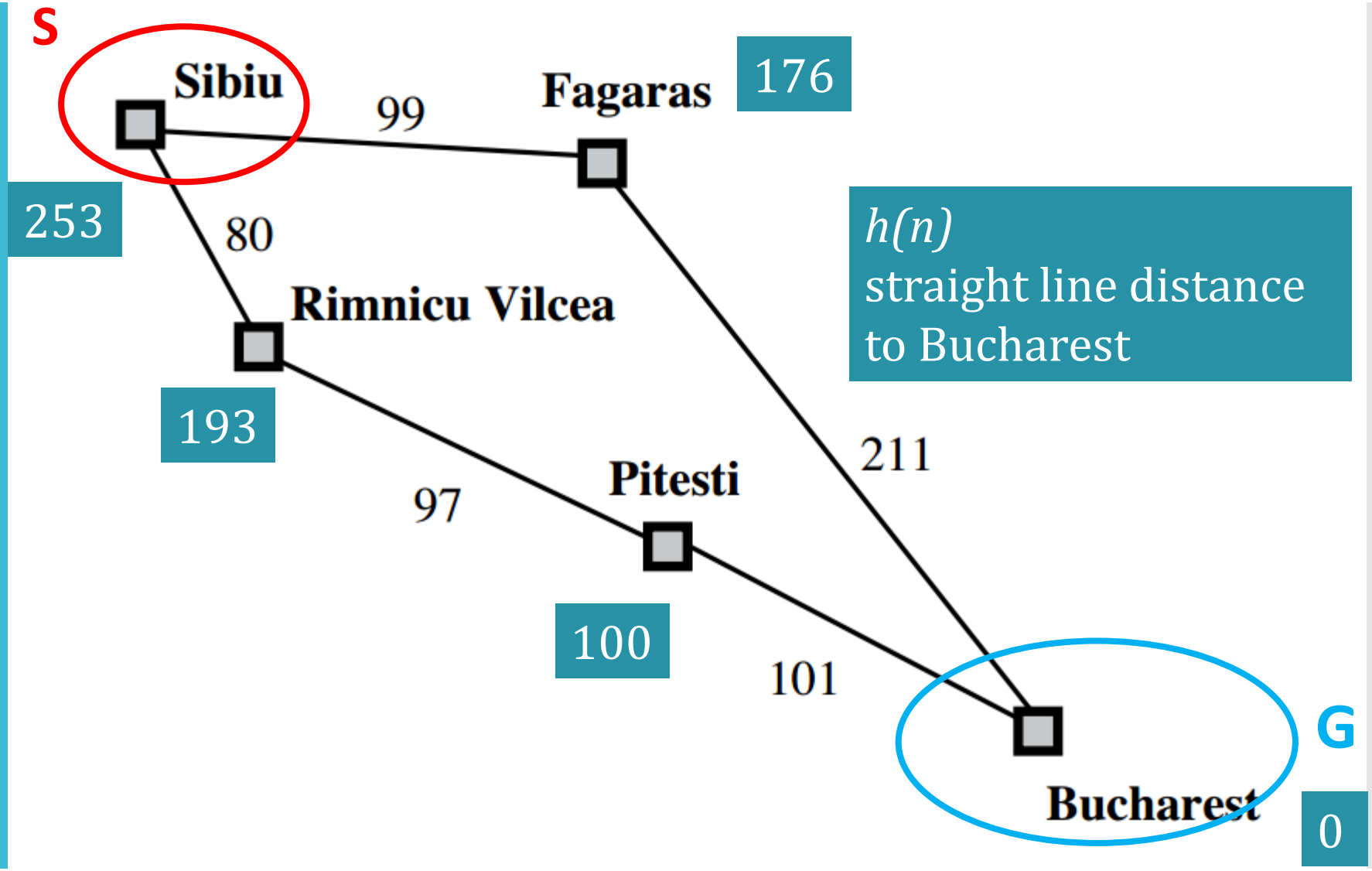
- *Admissibility*
  - $h(n) \leq h^*(n)$
    - $h^*(n)$: true cost from node n to goal state
  - **Theorem**: If *h(n)* is admissible, A* using TREE-SEARCH is optimal
  - **Example**: straight-line distance

- *Consistency (Monotonicity)*
  - $h(n) \leq cost(n, n') + h(n')$ (n' is successor of n)
    - $f(n)$ is non-decreasing along any path: *f(n') = max(f(n), g(n') + h(n'))*
    - 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.

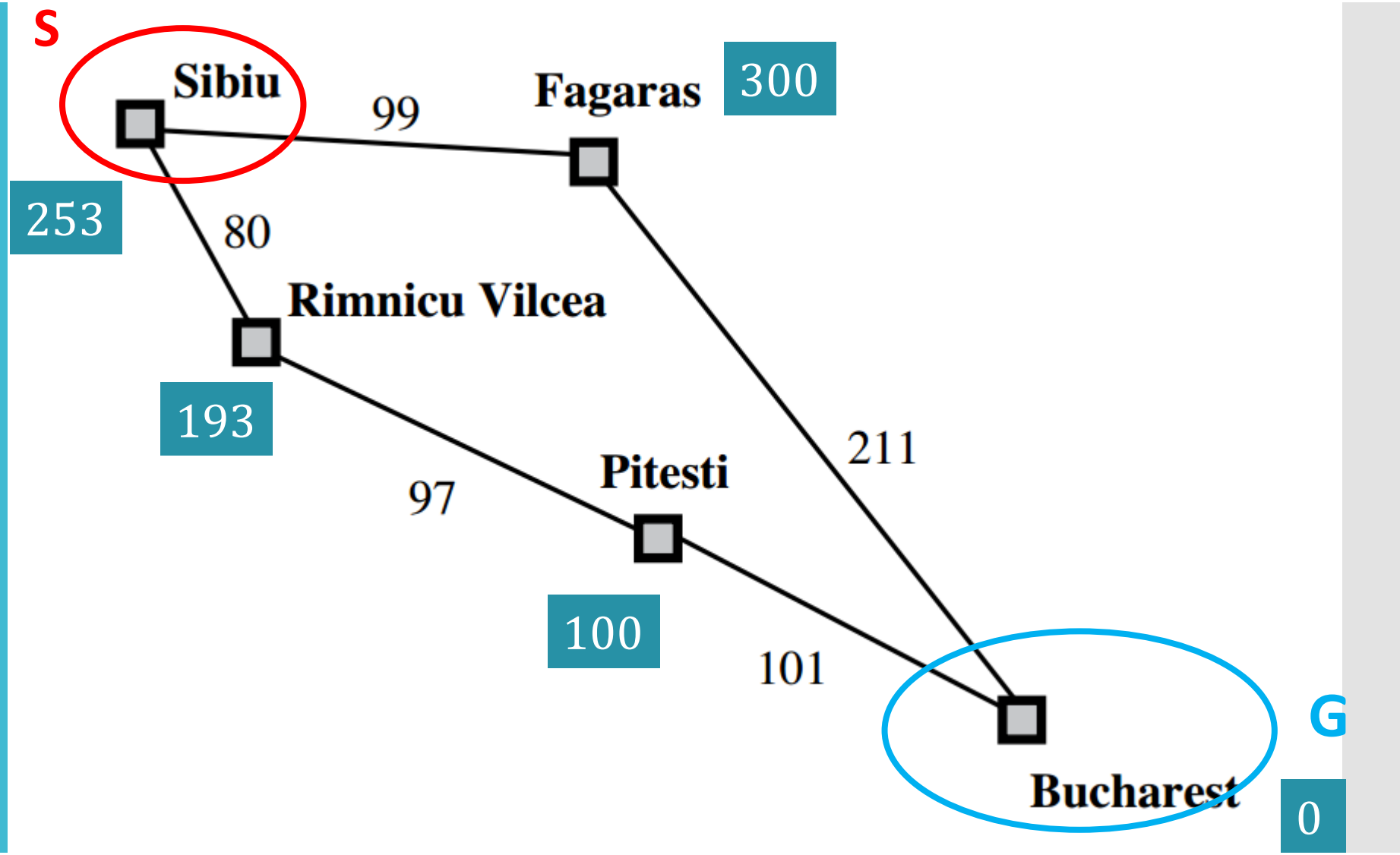
Properties of Heuristics

- *Dominating:*
  - If h2 (n) ≥ h1 (n) for every state n then h2 dominates h1
  - **Question:** if h2 dominates h1, which heuristic is better? Is it always the case? In what cases?

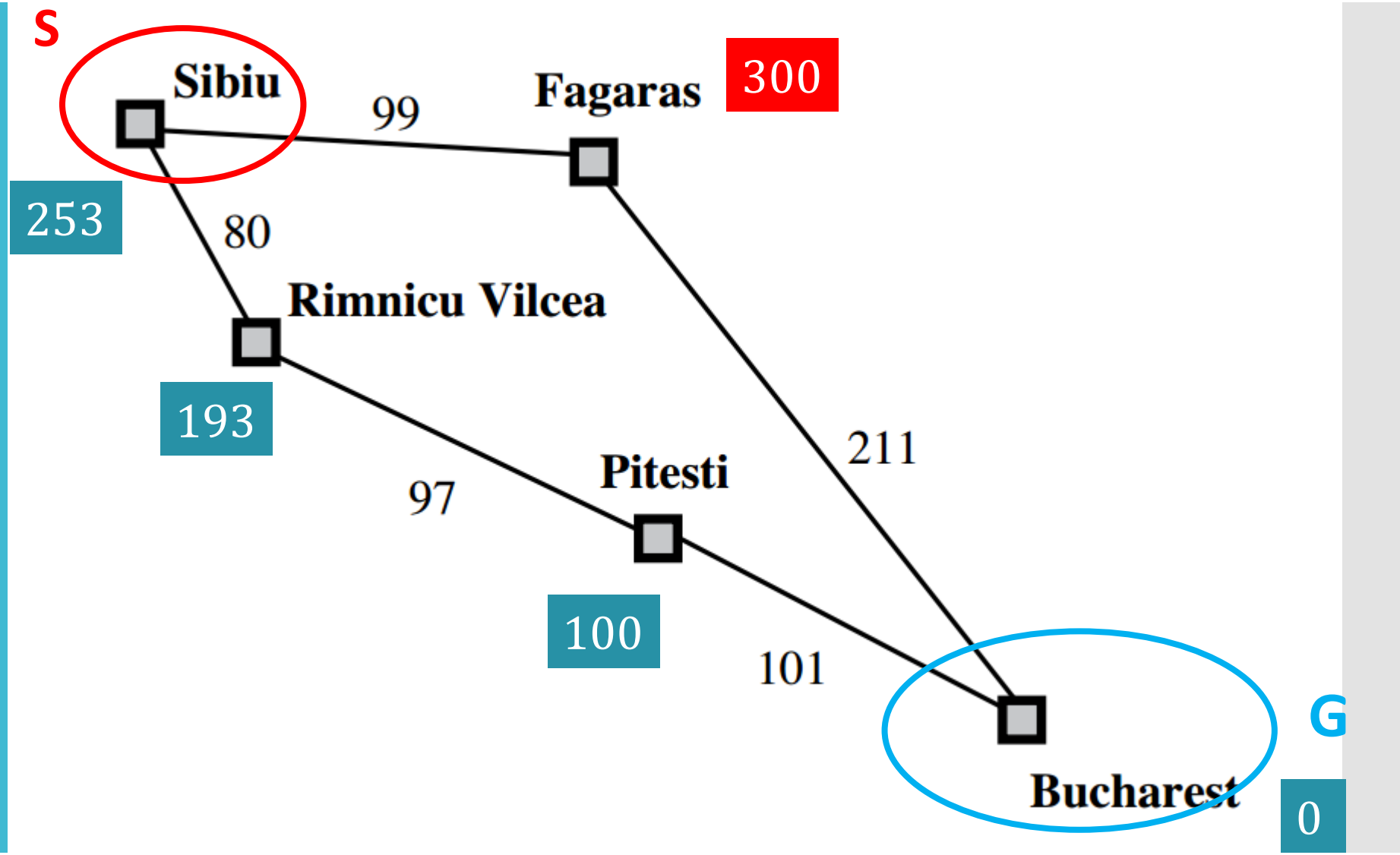
Is this h(n) admissible?



Is this h(n) admissible?



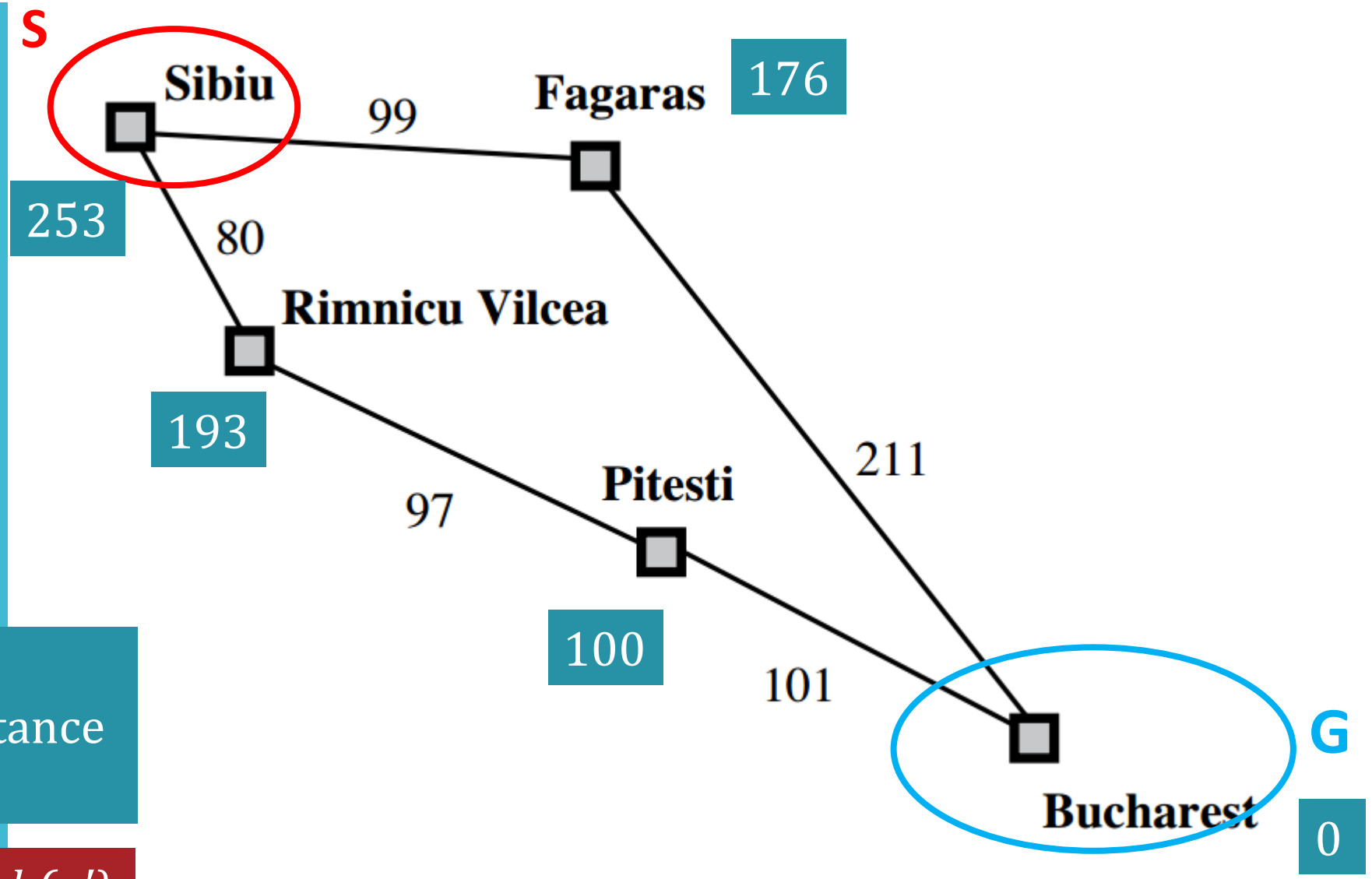
Is this h(n) admissible?



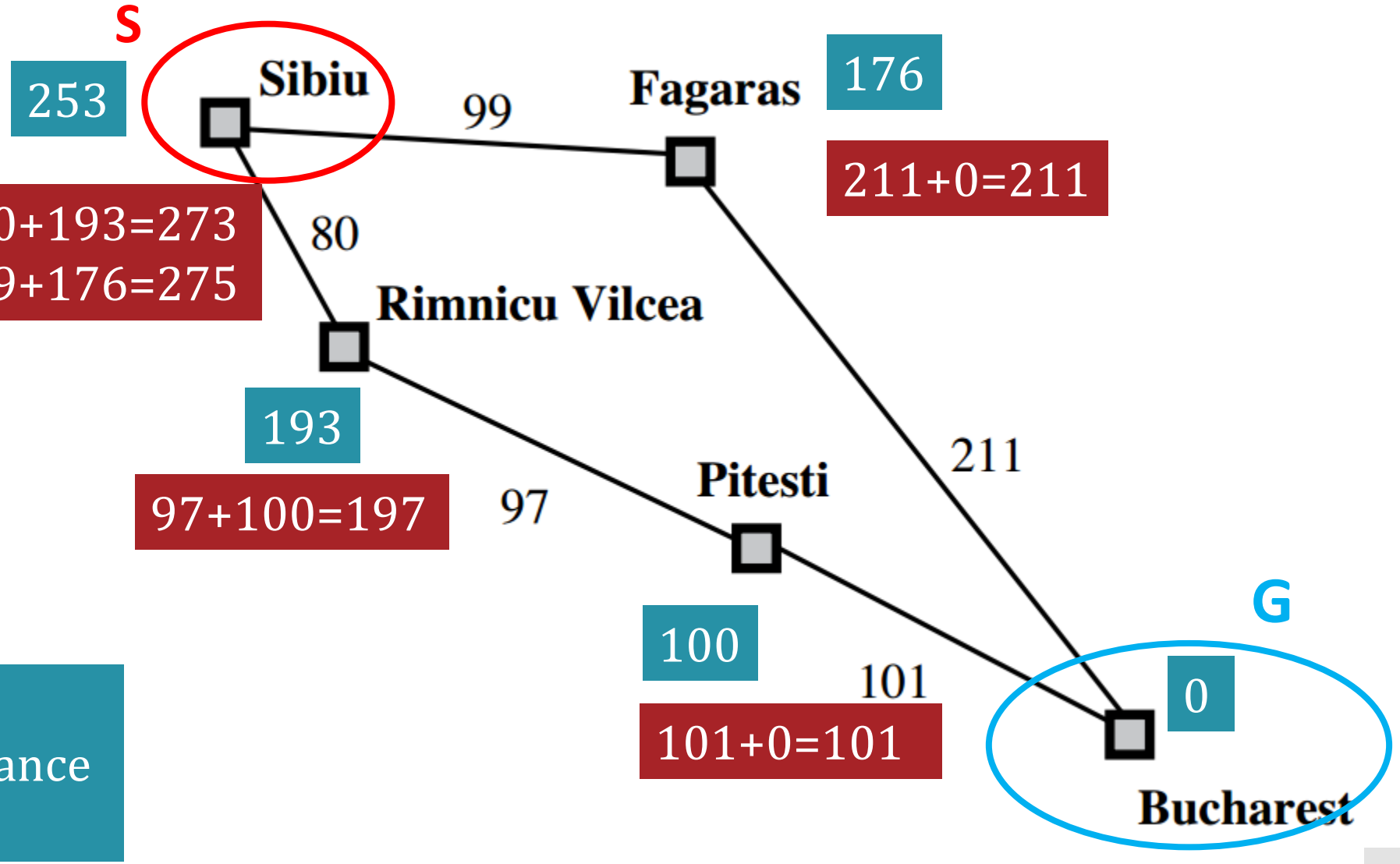
Is this h(n) monotonic?

h(n)
straight line distance to Bucharest

$h(n) \leq cost(n,n') + h(n')$



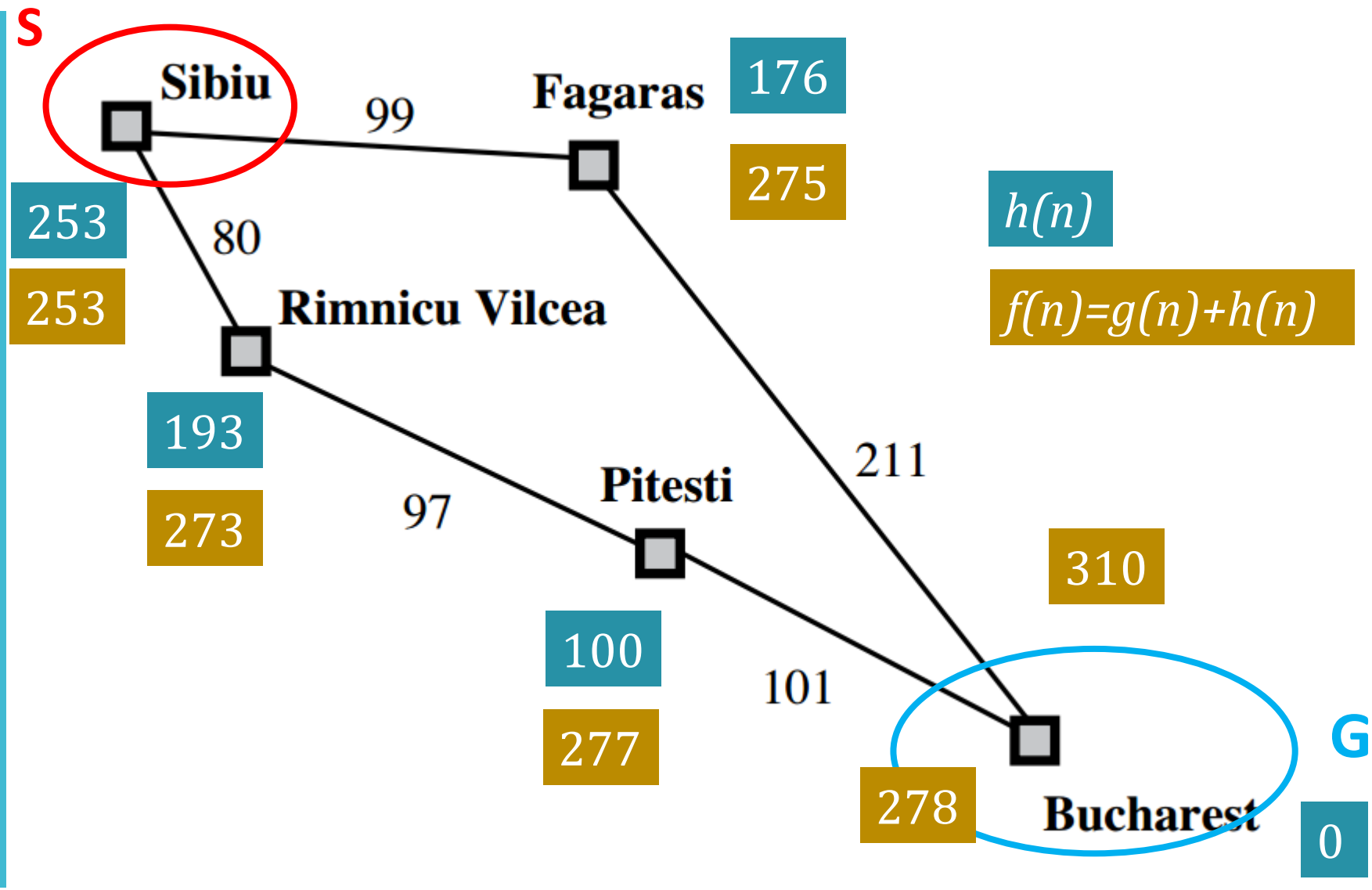
Is this h(n) monotonic?



h(n)
straight line distance to Bucharest

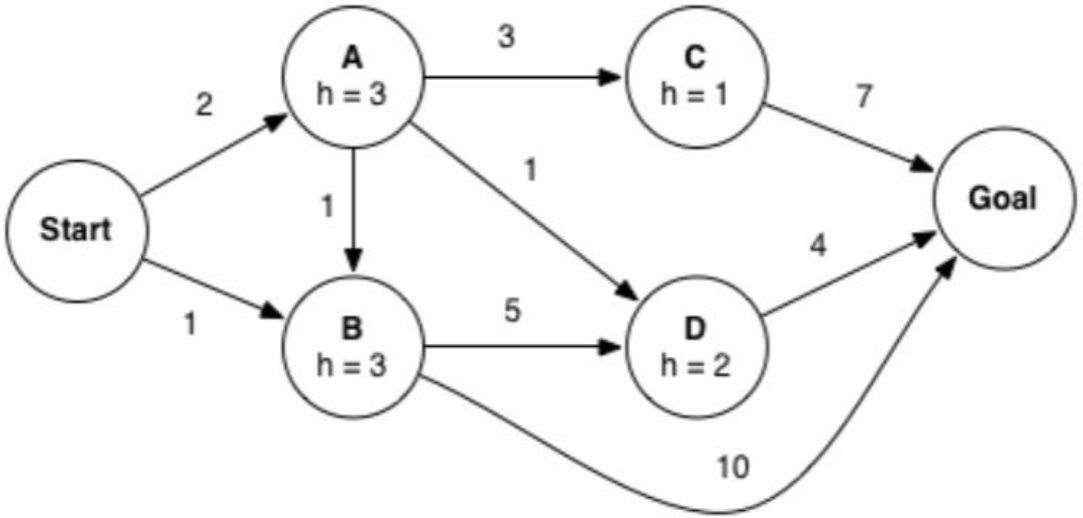
$h(n) \leq cost(n,n') + h(n')$

Is A* optimal in this problem?



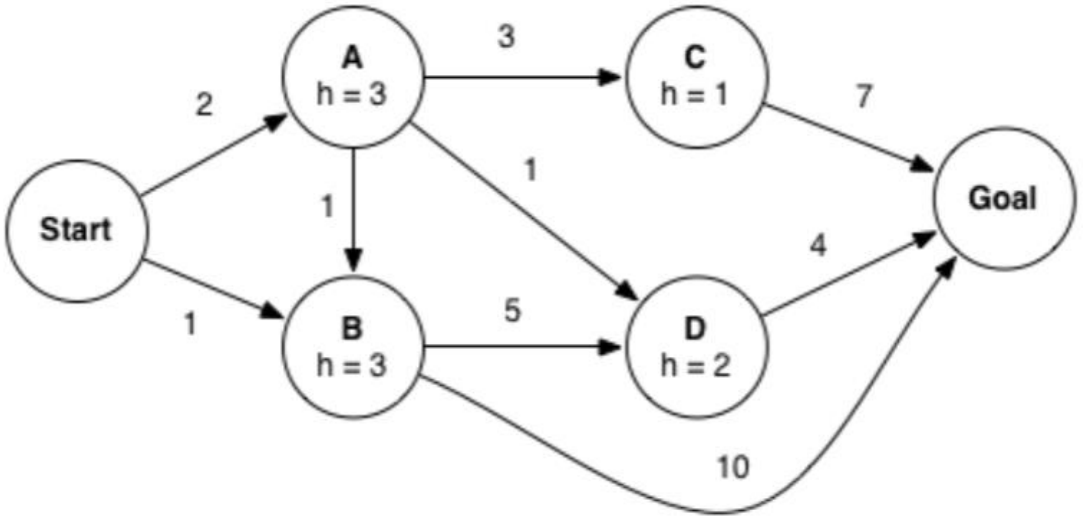
Exercise – A*

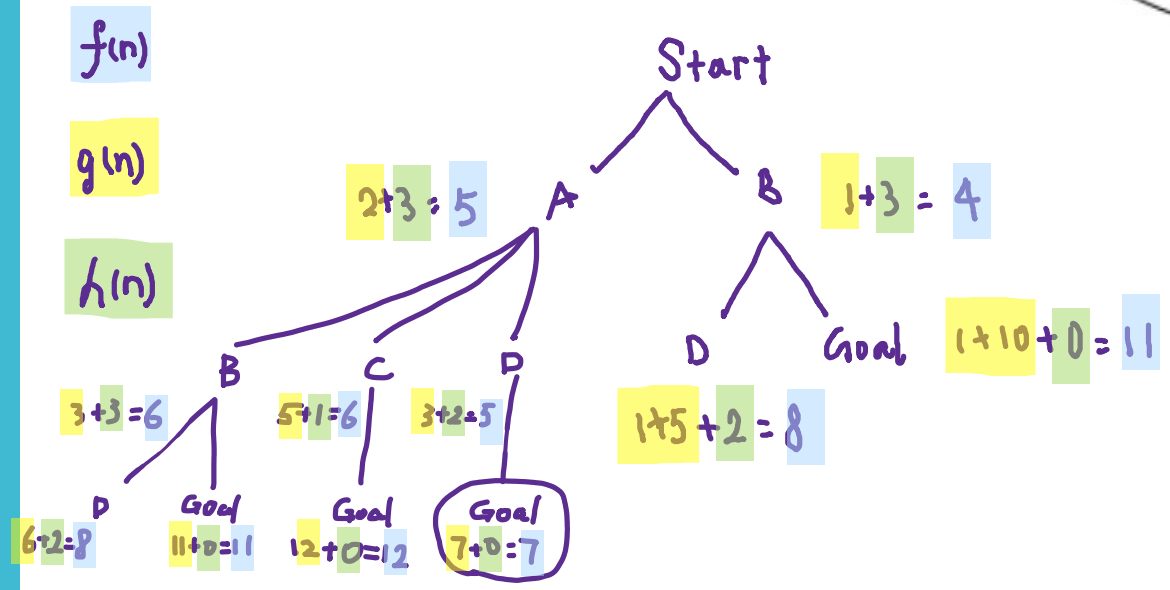
Quiz: try to draw the search tree



Exercise – A*

Quiz: try to draw the search tree





Return: Start → A → D → Goal

Important claims

- *consistency* implies *admissibility* whereas the opposite is not necessarily true

- 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?**

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

- *consistency* implies *admissibility* whereas the opposite is not necessarily true

- 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

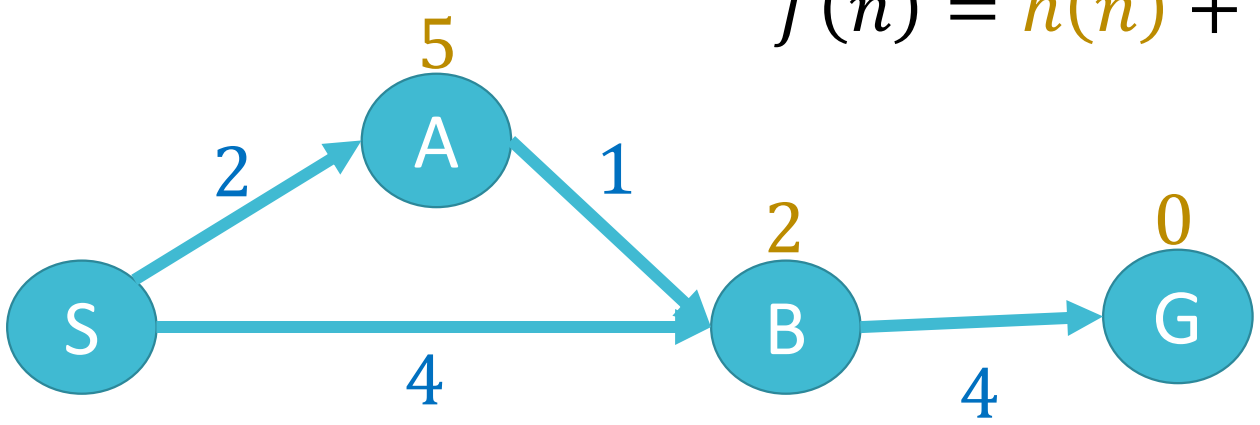
- Why would GRAPH-SEARCH + inconsistent h(n) may stop A* from being optimal?

Inconsistency

What happen if inconsistent but admissible heuristic?

TREE-SEARCH is still optimal.
GRAPH-SEARCH may not necessarily find the best solution.

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

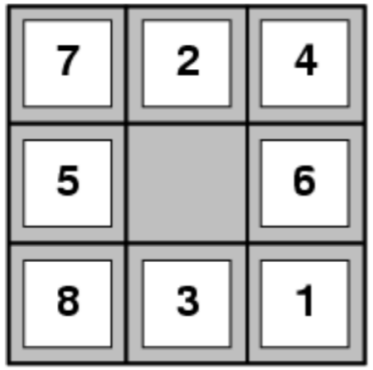


How to figure out heuristics?

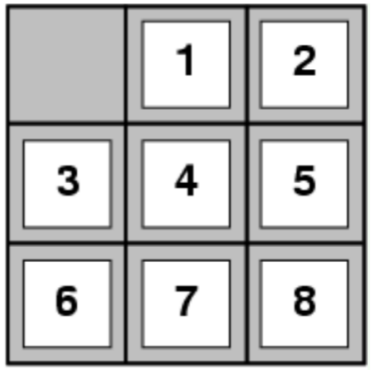
- Relaxed problem

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?



Start State



Goal State

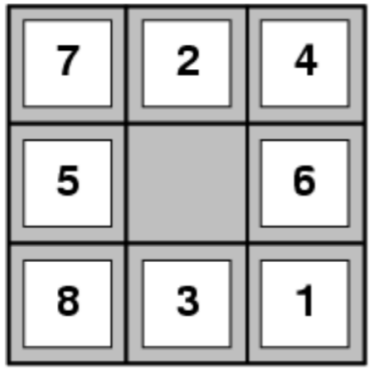
How to figure out heuristics?

- Relaxed problem

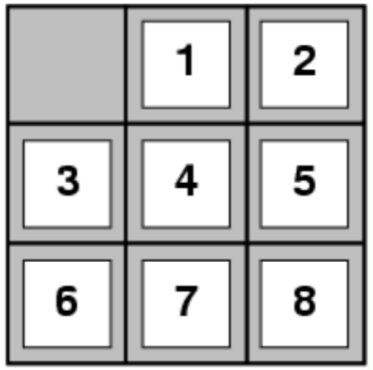
A problem with fewer restrictions on the actions are relaxed problems.

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?

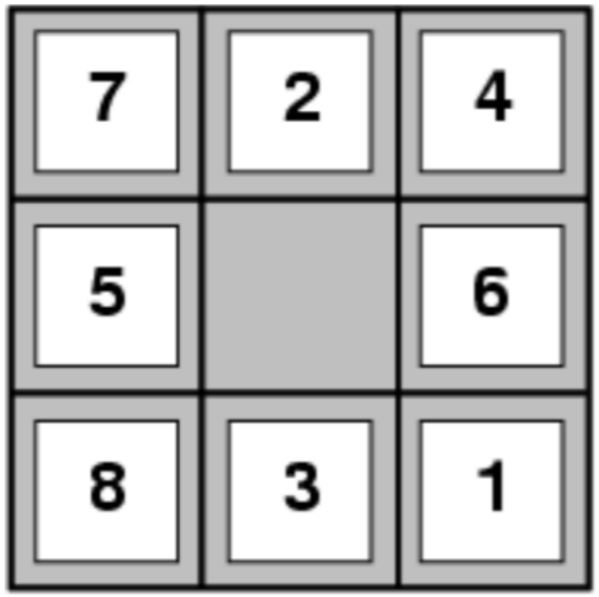


Start State

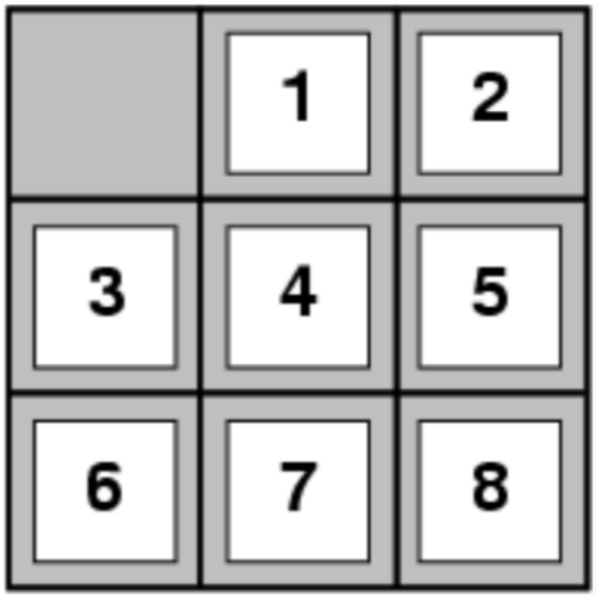


Goal State

Exercise



Start State



Goal State

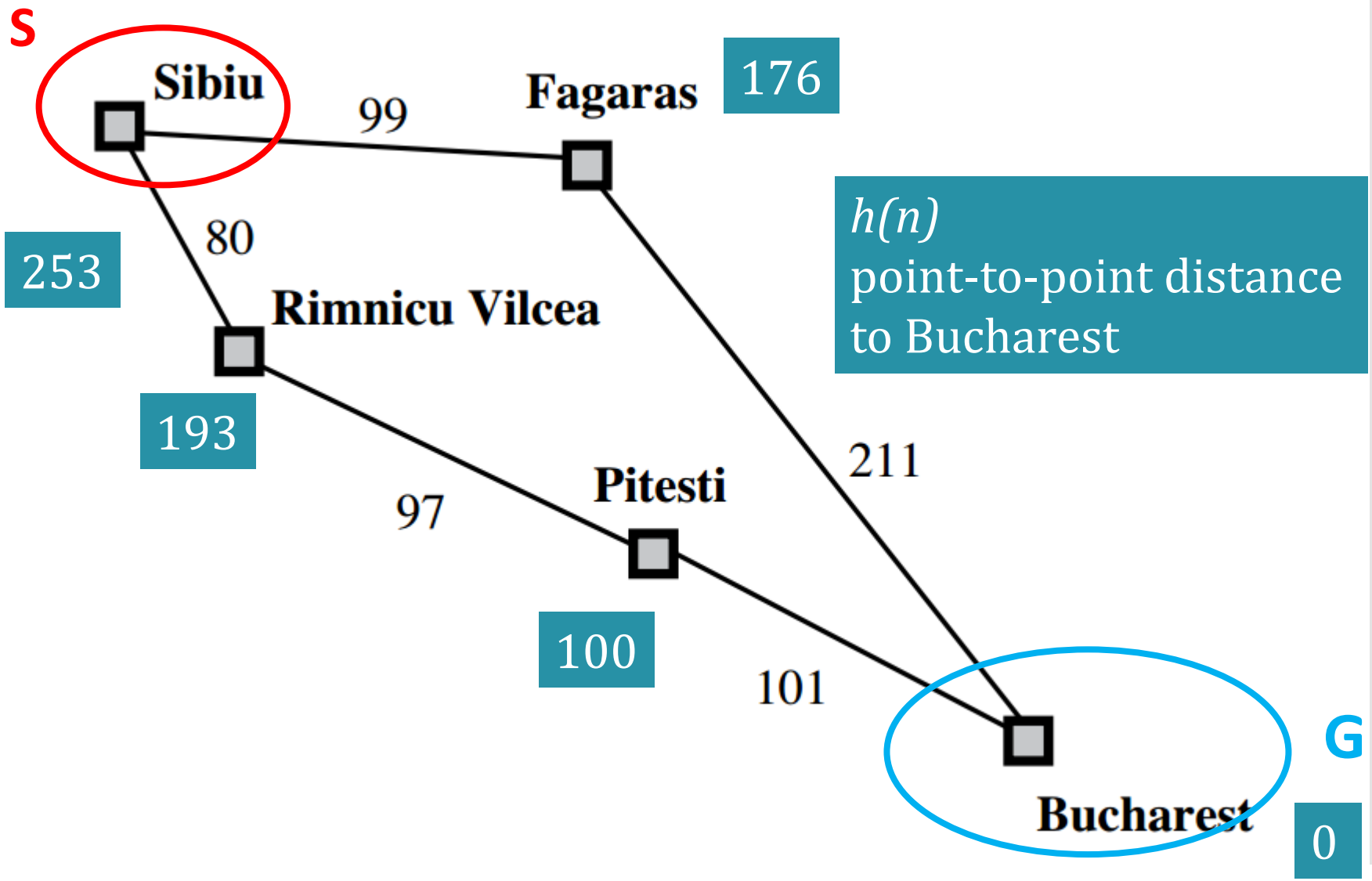
h1(n): number of misplaced tiles

h1(S) = ?

h2(n): sum of Manhattan distances to destination

h2(S)=?

Example – point to point distance in route planning



Generate heuristic function

- Given a collection of *admissible* heuristics, h_1, …, h_m, if there is not "clearly best" heuristics (i.e., neither heuristic dominates another one), what heuristics should we use? (Hint: can combine the heuristics somehow)

Generate heuristic function

- Given a collection of *admissible* heuristics, h_1 , … , h_m, if there is not "clearly best" heuristics, what heuristics should we use?

- A composite heuristics: $h(n) = max(h_1(n) , … , h_m(n))$
  - admissible
  - Dominates all of the individual heuristics