CPSC 322: Introduction to Artificial Intelligence

Search: Best-First Search and A* Search

Textbook reference: [3.5.3, 3.5.4, 3.6]

Instructor: Varada Kolhatkar
University of British Columbia

Credit: These slides are adapted from the slides of the previous offerings of the course. Thanks to all instructors for creating and improving the teaching material and making it available!

Announcements

- Midterm time and location

  Time: Friday, Oct 25th, from 6pm to 7pm
  Location: Woodward 2
  (Instructional Resources Centre-IRC) (WOOD) - 2

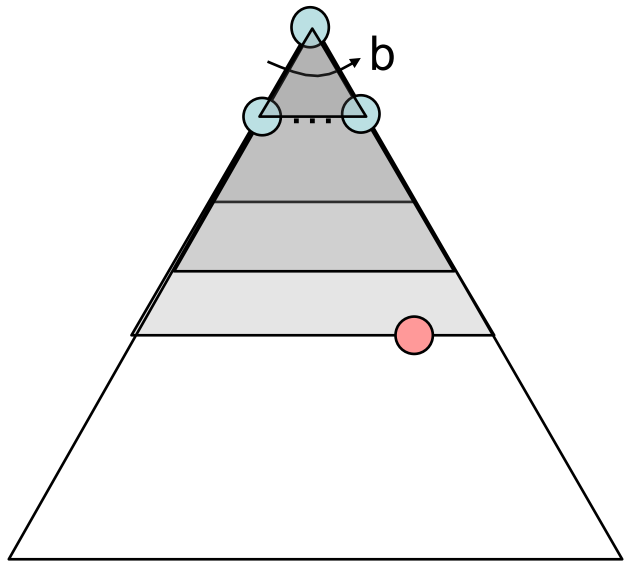
- Assignment 1 is due on Sept. 30th at 11:59pm

Lecture outline

- **Recap from last lecture (~10 mins)** 👈
- Admissible heuristic (~20 mins)
- Best-first search (~10 mins)
- A* search (~30 mins)
- Summary and wrap-up (~5 mins)

IDS: Linear space complexity with completeness and optimality

- Define a depth bound b
- Run a DFS from scratch with depth limit d = 1. If no solution…
- Run a DFS from scratch with depth limit d = 2. If no solution…
- Run a DFS from scratch with depth limit d = 3. …



Credit: Berkeley AI course material

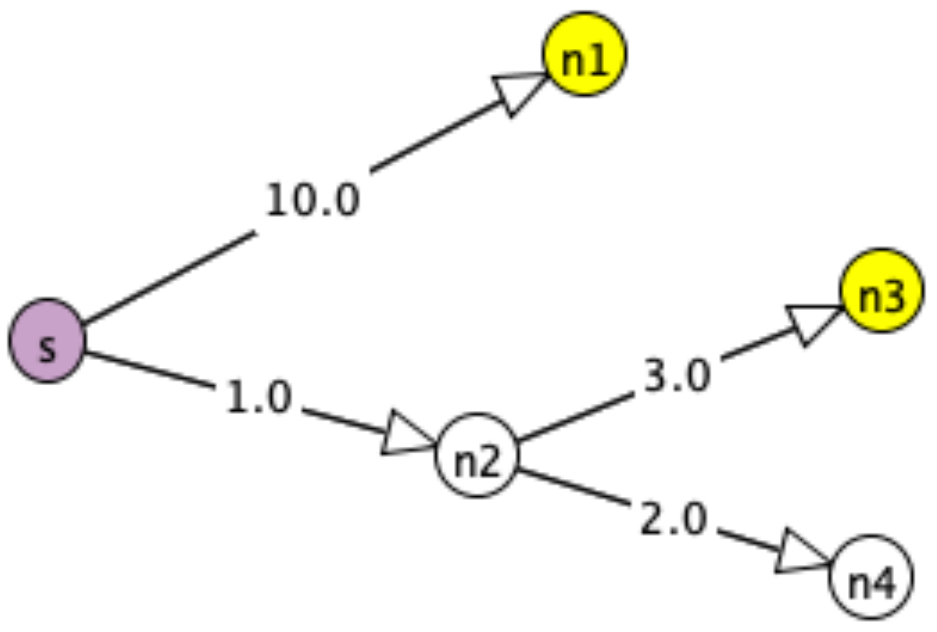
LCFS

- Sometimes there are costs associated with arcs and cost of a path is the sum of the costs of its arcs

- Optimal solution is not the one that minimizes the number of arcs but the one that minimizes **cost**.

- What path would BFS choose?

- What path would LCFS choose?



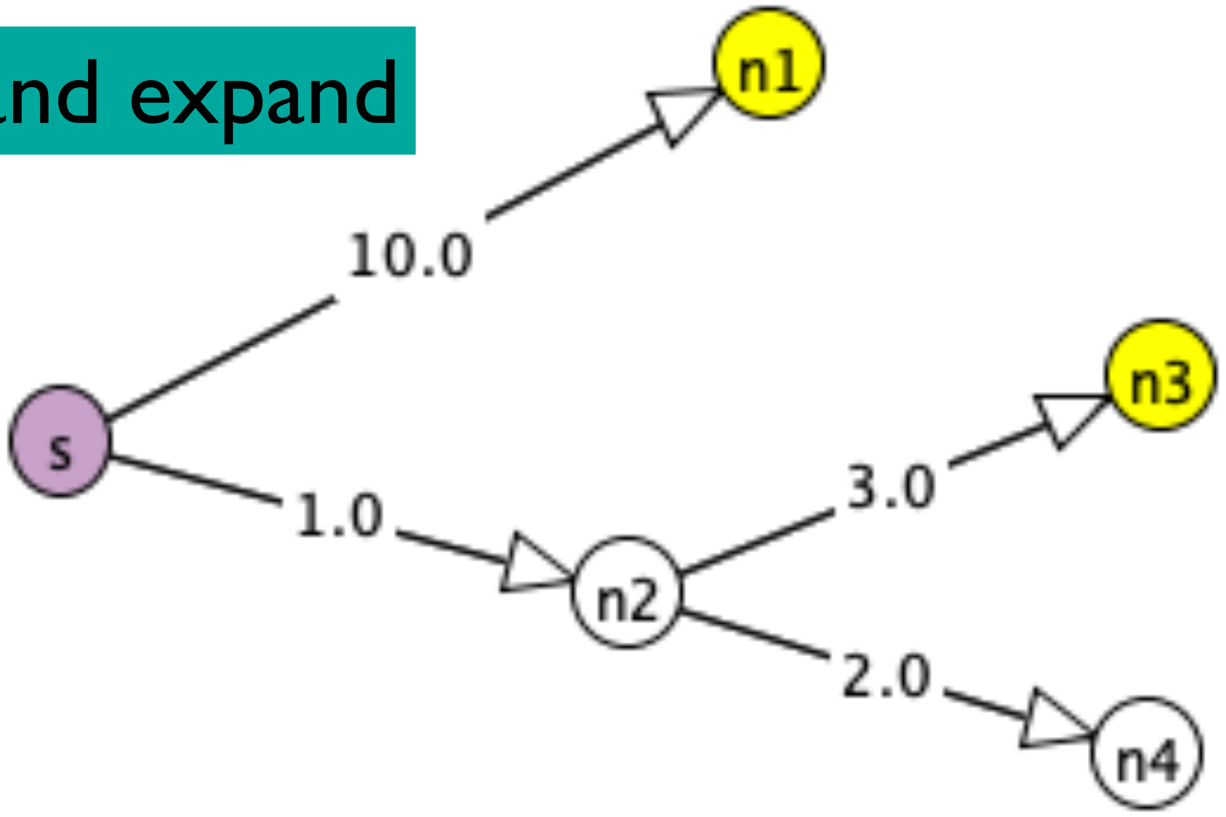
BFS vs. LCFS

BFS queue

select, remove and expand

<S>		
<S,n2>	<S,n1>	
<S,n1>	<S,n2,n4>	<S,n2,n3>

GOAL



LCFS priority queue

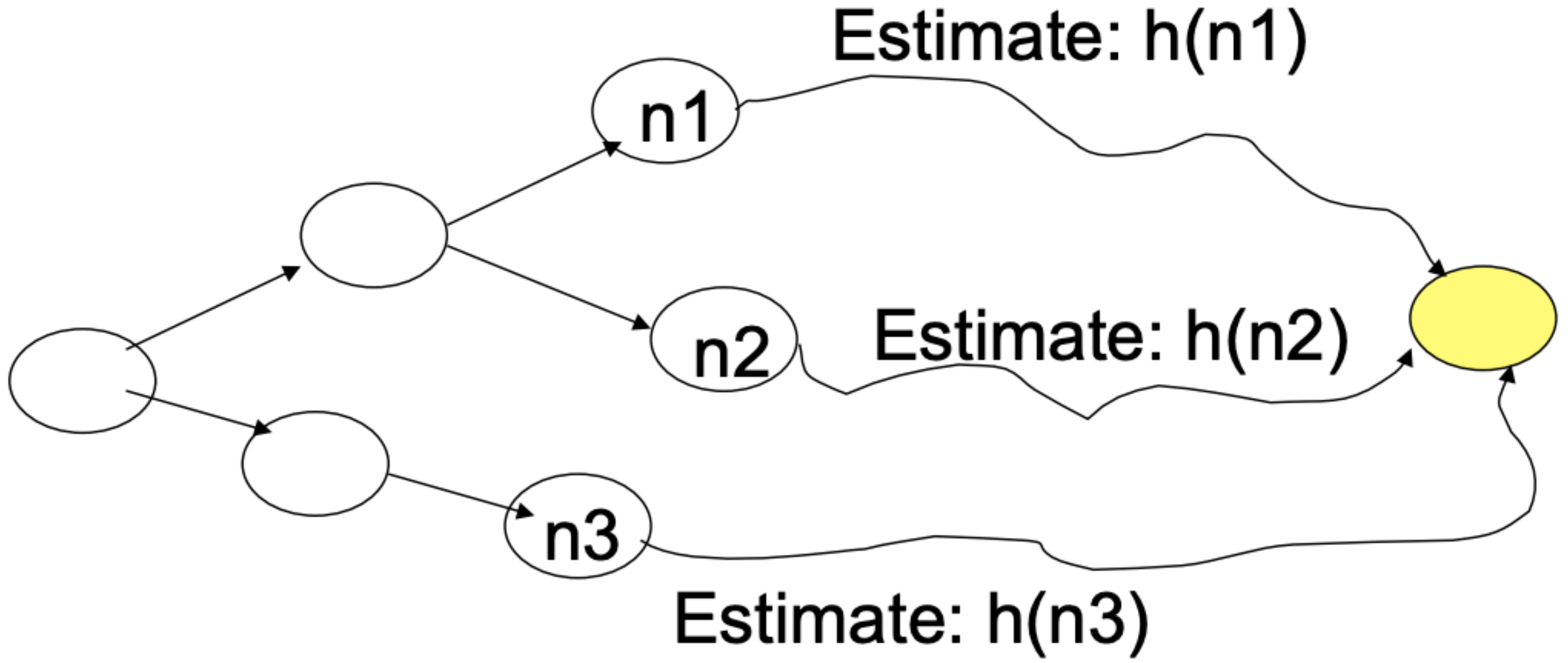
<S> (cost = 0)		
<S,n2> (cost = 1)	<S,n1> (cost = 10)	
<S,n2, n4> (cost = 3)	<S,n2,n3> (cost = 4)	<S,n1> (cost = 10)
<S,n2, n3> (cost = 4)	<S,n1> (cost = 10)	

GOAL

LCFS is the same as Dijkstra's algorithm.

Heuristic function

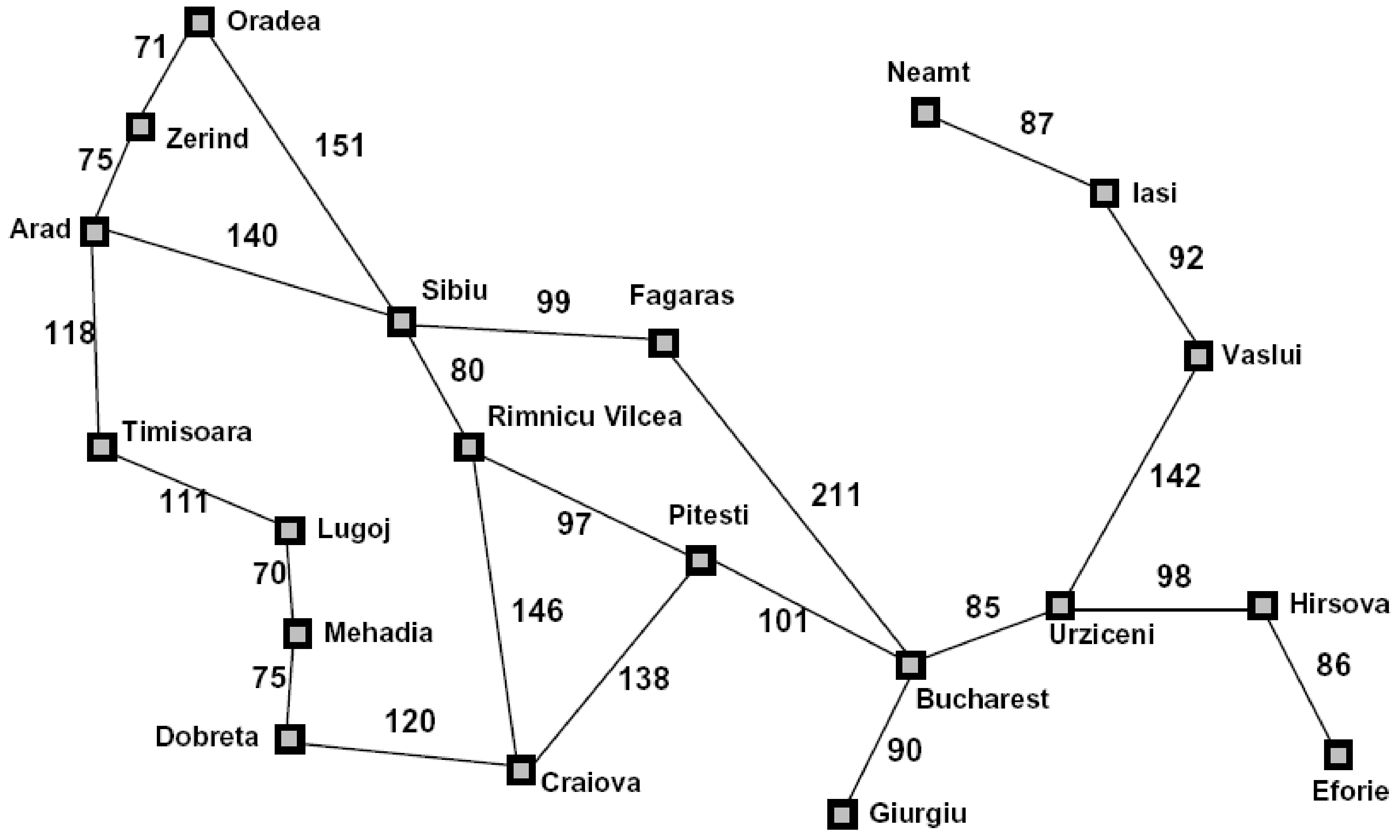
A **search heuristic** $h(n)$ is an **estimate** of the cost of the lowest-cost path from node n to a goal node.



- How to construct h(n1), h(n2), and h(n3)?
  From your intuition and analysis of the problem.

- It has to be cheap and easy to compute compared to the original problem.

Example of $h(n)$



Straight−line distance
to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Euclidean distance: straight-line distance between source and goal node

Heuristic search

What information we could use to better select paths from the frontier?

A. An estimate of the distance/cost from the last node on the path to the goal

B. An estimate of the distance from the start node to the goal node

C. An estimate of the path so far

D. Any of the above would work

Today: Learning outcomes

From this lecture, students are expected to be able to:

- Define admissible heuristic
- Construct admissible heuristics for a given problem
- Define/read/write/trace/debug Best-First search and A* search

Lecture outline

- Recap from last lecture (~10 mins)
- **Admissible heuristic (~20 mins)** 👈
- Best-first search (~10 mins)
- A* search (~30 mins)
- Summary and wrap-up (~5 mins)

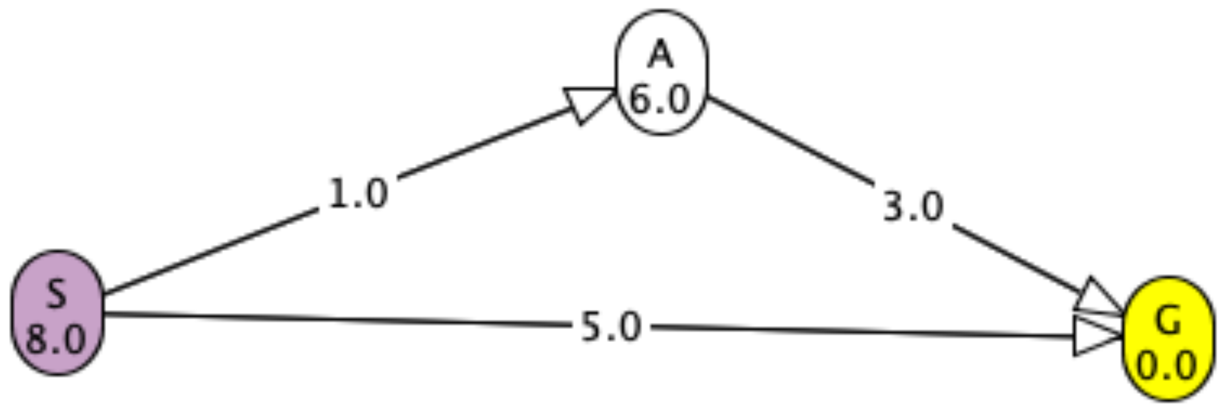
Admissible heuristic

A search heuristic is **admissible** if it never overestimates the actual cost of the cheapest path from a node to a goal.

Let $c(n)$ denote **the cost of the optimal path** from node n to any goal node. A search heuristic is called admissible if $h(n) \leq c(n), \forall n$. That is for all nodes it is an underestimate of the cost to any goal.

$$0 \leq h(n) \leq c(n)$$

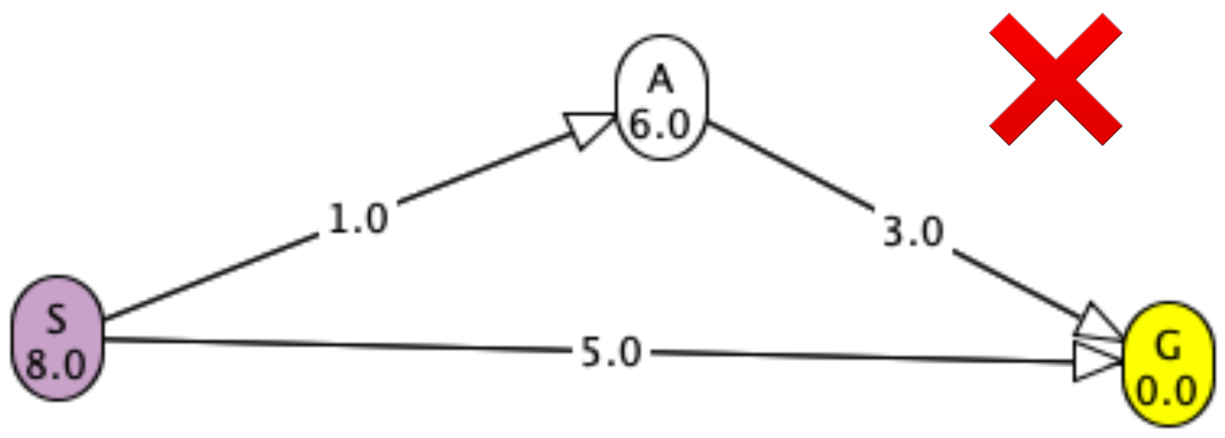
Admissible heuristic



$h(S) = 8.0$
$h(A) = 6.0$
$h(G) = 0.0$

Is this heuristic admissible?

Admissible heuristic



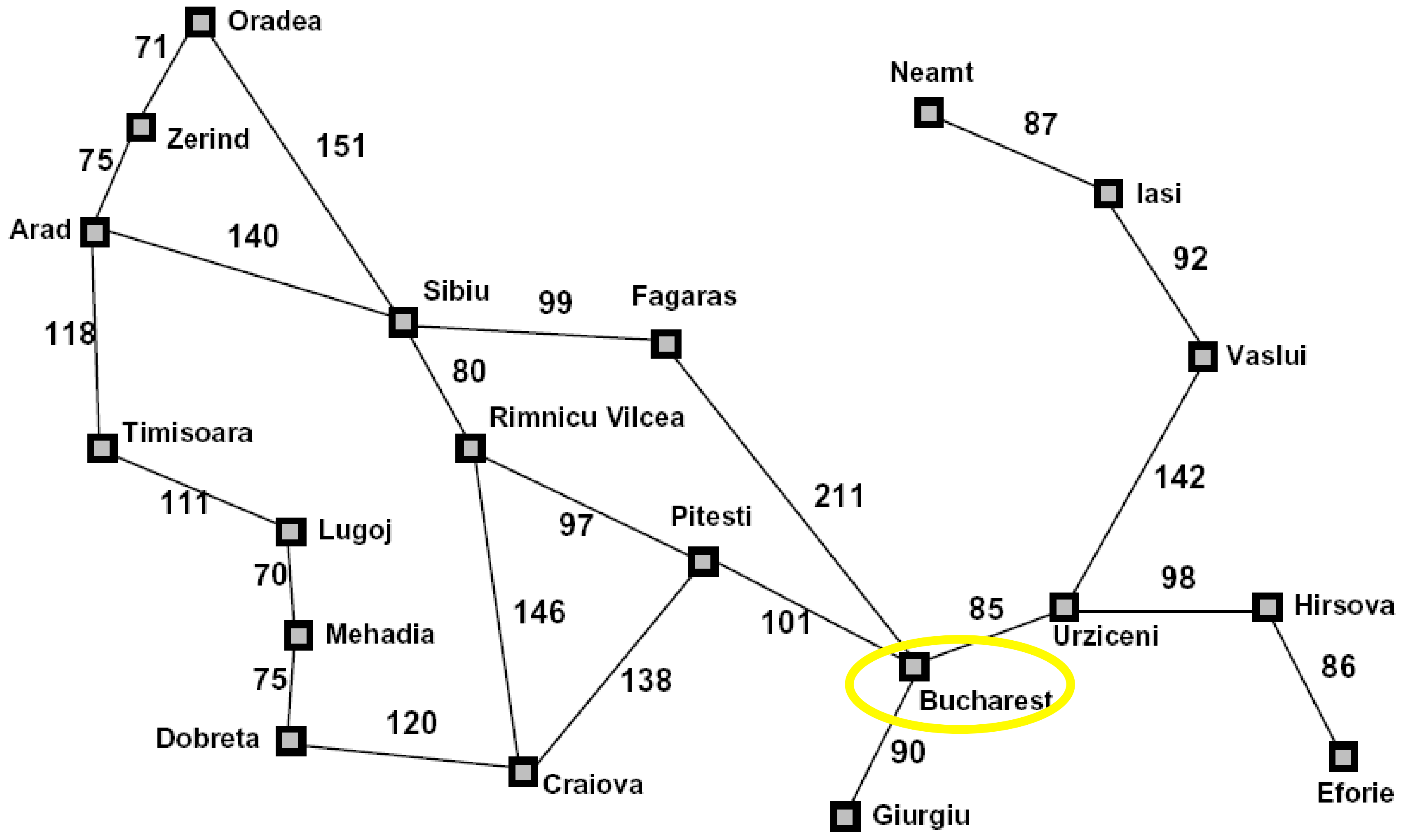
✗

h(S) = 8.0
h(A) = 6.0
h(G) = 0.0

- Is this heuristic admissible?
  No because actual cost to goal < estimated cost to goal

We need estimates to be ≤ the actual costs for a heuristic to be admissible.

Example: Travelling in Eastern Europe

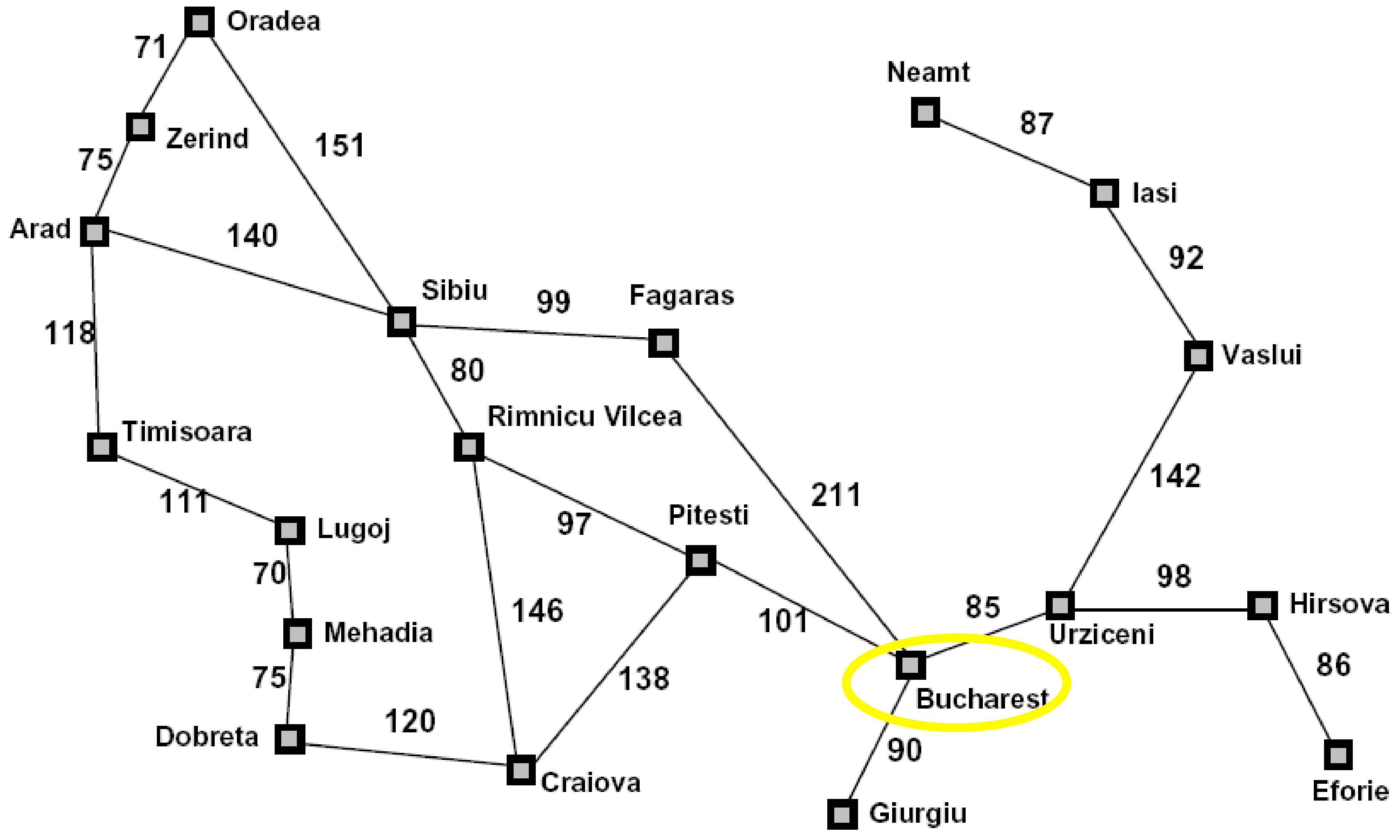


Straight−line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Is the given heuristic admissible given the goal is Bucharest?

Example: Travelling in Eastern Europe



Straight–line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Is straight line distance admissible?
Yes because any other distance will be greater than or equal to that distance.

Coming up with admissible heuristics is most of what's involved in using heuristic-based search algorithms in practice.

Creating admissible heuristics

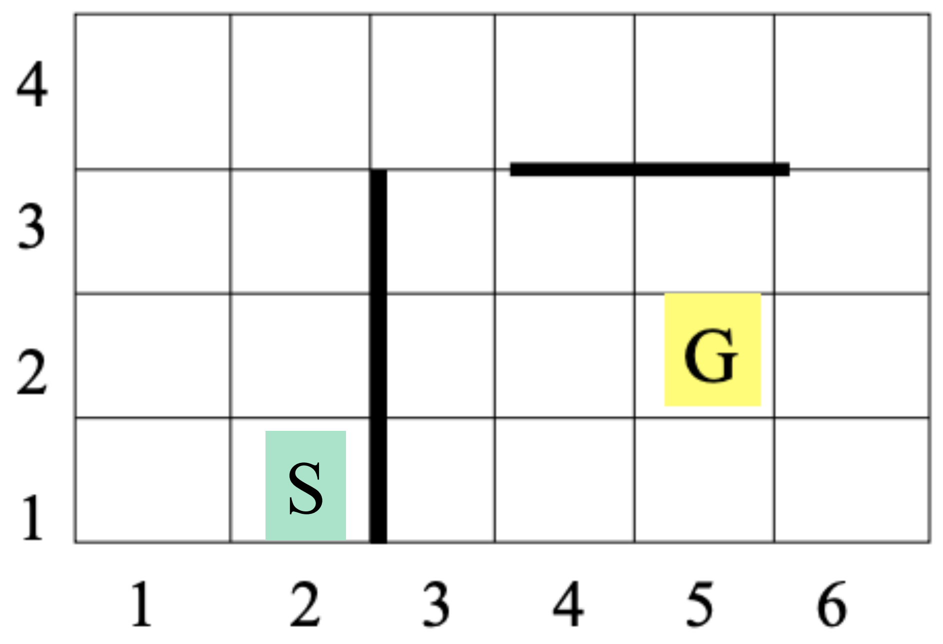
- **Calculate the solution cost for a relaxed version of the problem**
    - For example, no walls, no movement constraints
- Use (optimal) search to find the solution cost for a subproblem
    - For example, in 8-puzzle replace some numbers with blank tiles

Example: Grid world

- Search problem: Robot has to find a route from start location to goal location on a grid with obstacles

- Actions: move up, down, left, right from tile to tile

- Cost: the number of steps

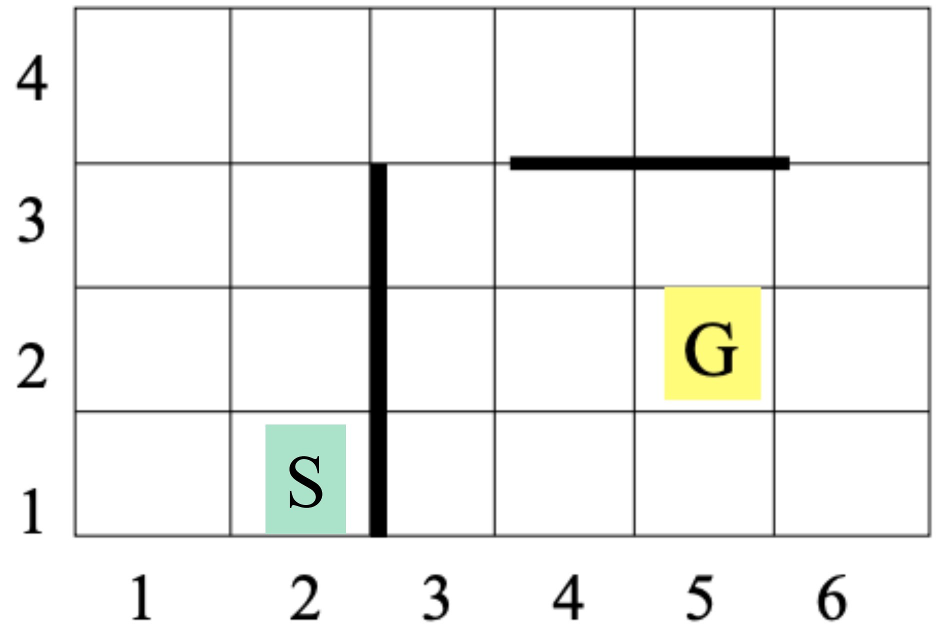


Example: Grid world

Possible $h(n)$: Manhattan distance (L_1 norm) between two points

= sum of the absolute difference between co-ordinates

$= |x_2 - x_1| + |y_2 - y_1|$



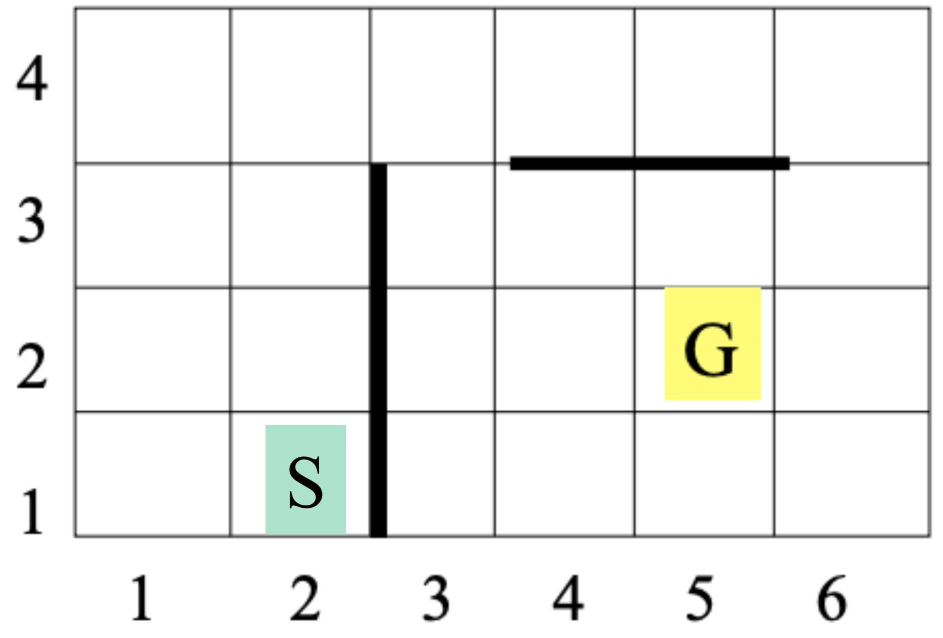
Location of S = (2,1)
Location of G = (5,2)
Manhattan distance (S, G) = ?

Example: Grid world

Possible $h(n)$: Manhattan distance (L_1 norm) between two points

= sum of the absolute difference between co-ordinates

$= |x_2 - x_1| + |y_2 - y_1|$



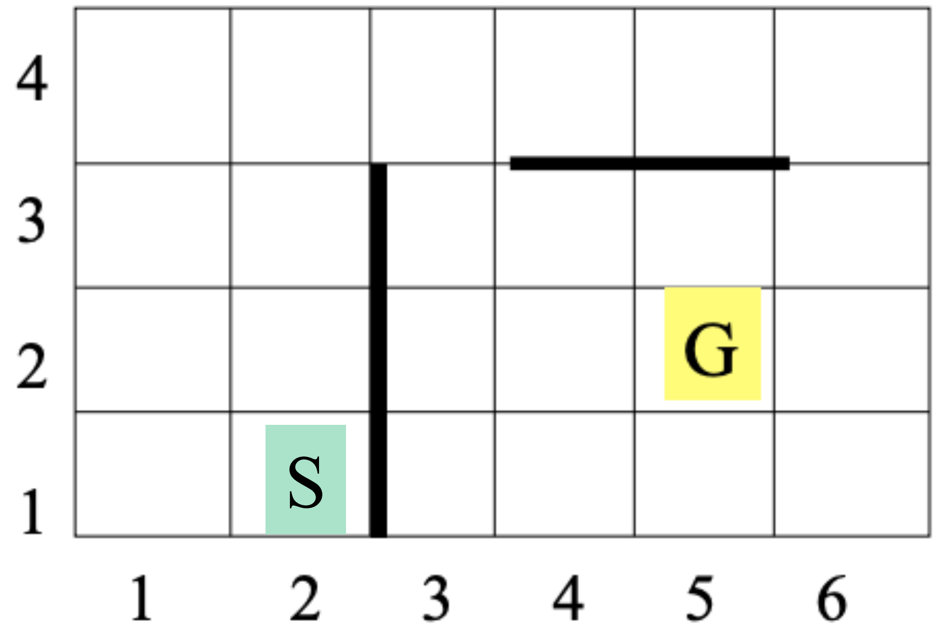
Location of S = (2,1)
Location of G = (5,2)
Manhattan distance (S, G) = (5-2) + (2-1) = 4

Example: Grid world

Possible $h(n)$: Manhattan distance (L_1 norm) between two points

= sum of the absolute difference between co-ordinates

$= |x_2 - x_1| + |y_2 - y_1|$



Manhattan distance (S, G) = 4

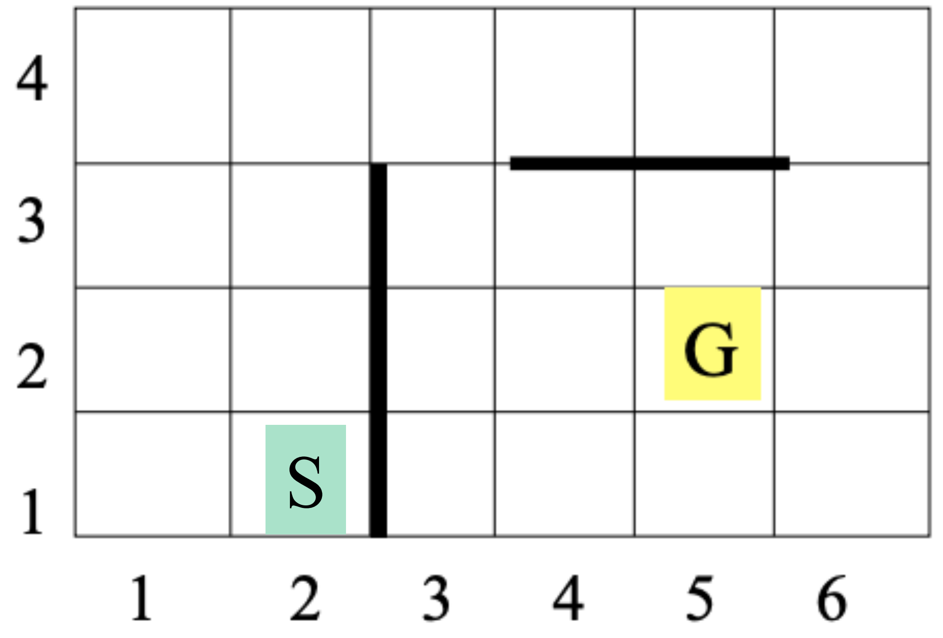
Is this an admissible heuristic?

Example: Grid world

Possible $h(n)$: Manhattan distance (L_1 norm) between two points

= sum of the absolute difference between co-ordinates

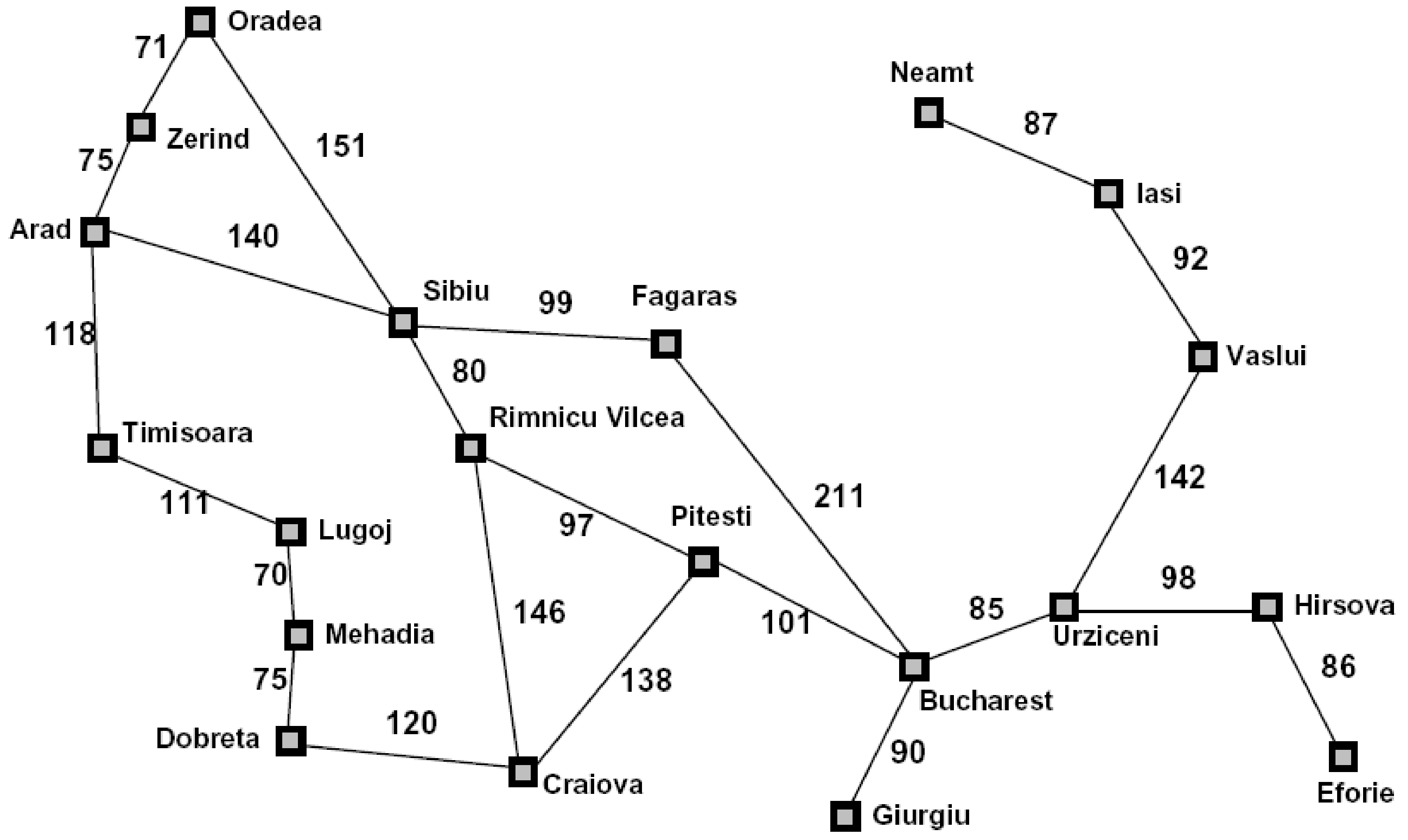
$= |x_2 - x_1| + |y_2 - y_1|$



Manhattan distance (S, G) = 4

Is this an admissible heuristic?
Yes. The cost of the optimal solution without obstacles is $\leq$ the cost when there are obstacles.

Example: Driver

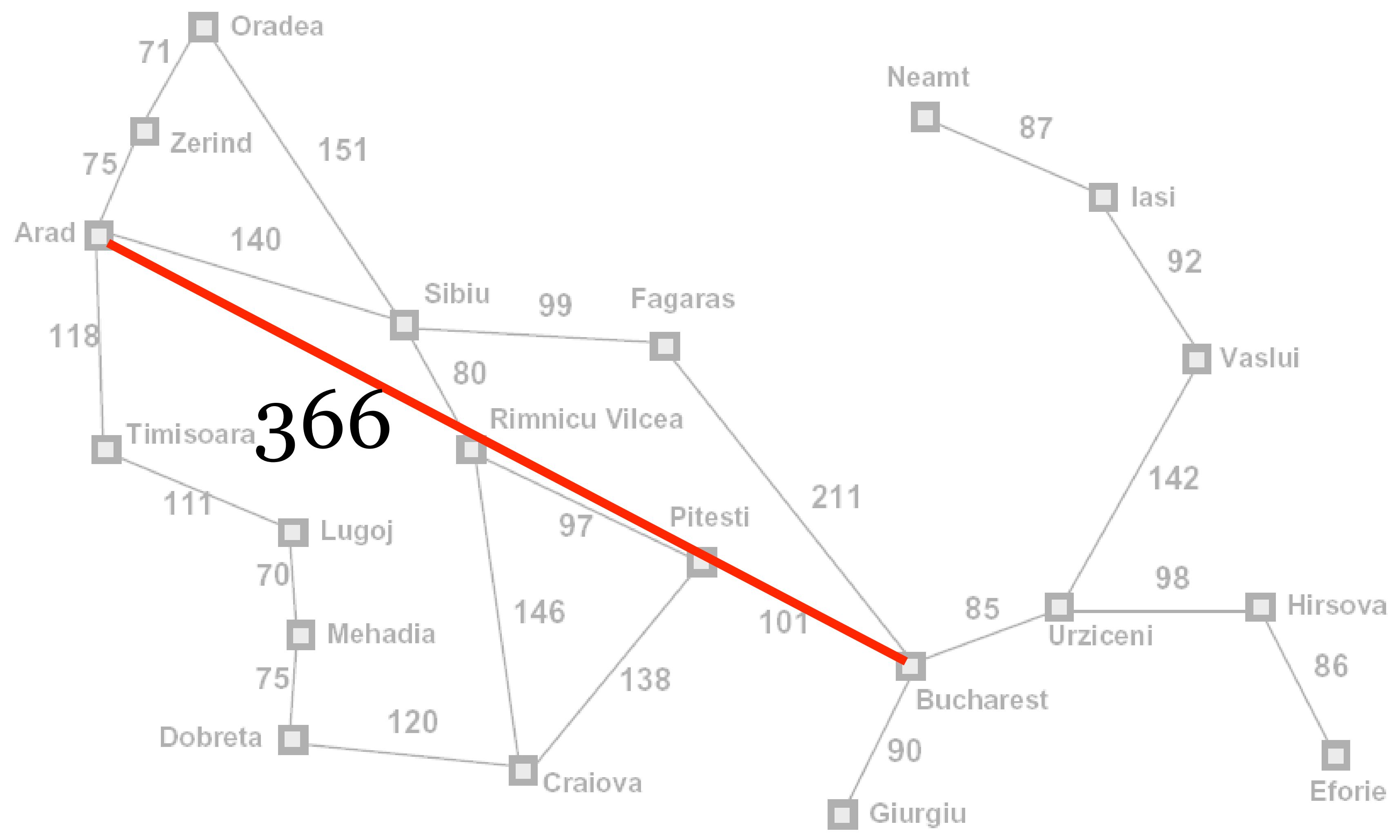


Straight–line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Similarly, if the nodes are points on a Euclidean plane and the cost is the distance, we can use the straight-line distance from n to the closest goal as the value of $h(n)$.

Example: Driver

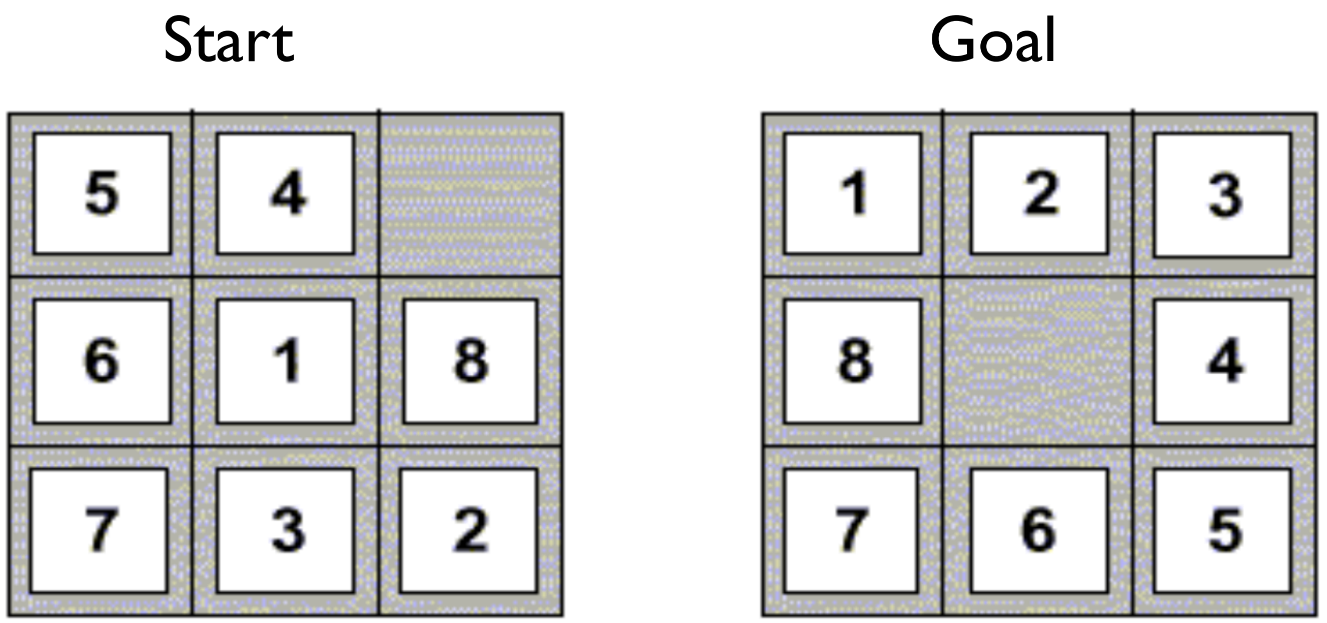


Straight–line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Similarly, If the nodes are points on a Euclidean plane and the cost is the distance, we can use the straight-line distance from n to the closest goal as the value of $h(n)$.

Pair-share: Heuristic for 8-puzzle



Come up with an admissible heuristic for this problem.

Heuristic for 8-puzzle

A reasonable admissible heuristic for 8-puzzle is

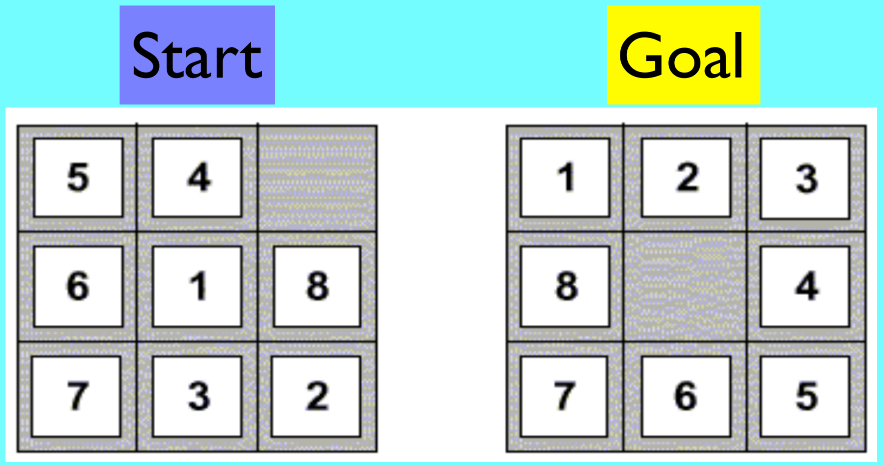
A. Number of misplaced tiles plus number of correctly placed tiles

B. Number of misplaced tiles ✅

C. Number of correctly placed tiles

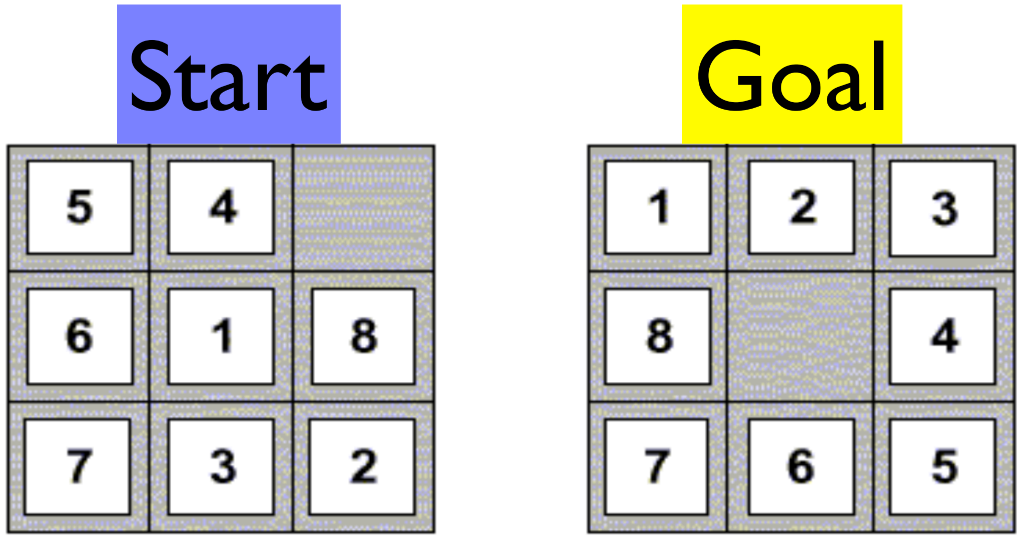
D. Number of tiles we haven't moved yet

E. None of the above



Example: 8-puzzle

- $h_1(n)$ = number of misplaced tiles
- $h_2(n)$ = total Manhattan distance (number of squares from desired location of each tile)

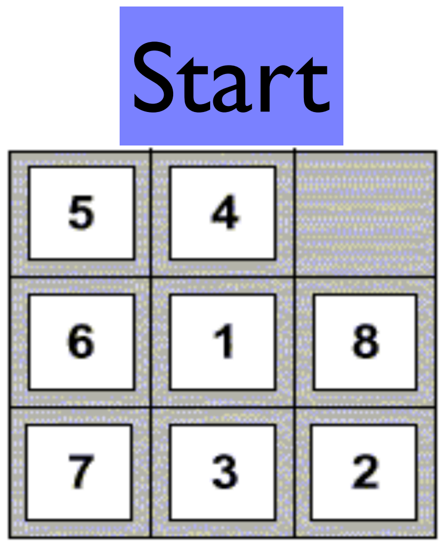


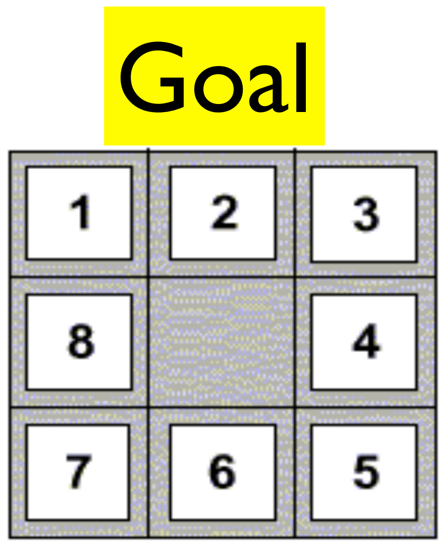
$h_1(n)$ assumes that a tile can go anywhere whether or not another tile is already at that location

$h_2(n)$ assumes a tile can go to any adjacent position whether or not another tile is already at that location

Example: 8-puzzle

- $h_1(n) =$ number of misplaced tiles
- $h_2(n) =$ total Manhattan distance (number of squares from desired location of each tile)





$h_1(start) = 8$

$h_2(start) = 4 + 2 + 2 + 2 + \ldots$

Creating admissible heuristics

- Calculate the solution cost for a relaxed version of the problem
  - For example, no walls, no movement constraints
- **Use (optimal) search to find the solution cost for a subproblem**
  - For example, in 8-puzzle replace some numbers with blank tiles

Another way to construct heuristic

Find solution cost for a subproblem.

Original problem

	1	3
8	2	5
7	6	4

$\longrightarrow$

Subproblem

	1	3
@	2	@
@	@	4

Solution

1	2	3
8		4
7	6	5

1	2	3
@		4
@	@	@

Summary: Constructing admissible heuristic

- Identify relaxed version of the problem where one or more constraints have been dropped
  - Problem with fewer restrictions on the actions
    - Grid world: The agent can move through walls
    - Travelling in Eastern Europe: The agent can move straight
    - 8-puzzle: Tiles can move anywhere or tiles can move to any adjacent square

Summary: Constructing admissible heuristic

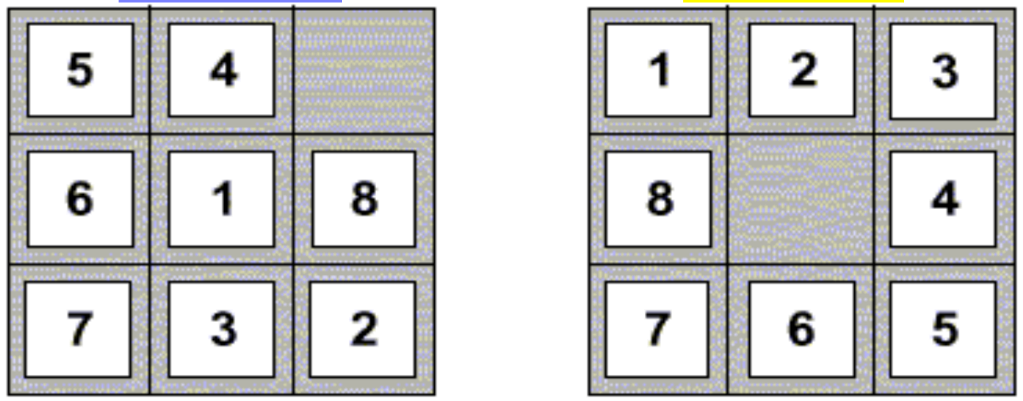
- Identify constraints which when dropped make the problem extremely easy to solve.
- This is important because heuristics are not useful if they are as hard to solve as the original problem.
- This leads to admissible heuristics because problem only gets simpler.

Example: 8-puzzle

- $h_1(n) =$ number of misplaced tiles
- $h_2(n) =$ total Manhattan distance (number of squares from desired location of each tile)

Start

Goal



$h_1(start) = 8$

$h_2(start) = 4 + 2 + 2 + 2 + \ldots$

Which one is better?

Heuristic dominance

Consider two admissible heuristics h_1 and h_2. If $h_2(n) \geq h_1(n)$ for every state n then h_2 dominates h_1.

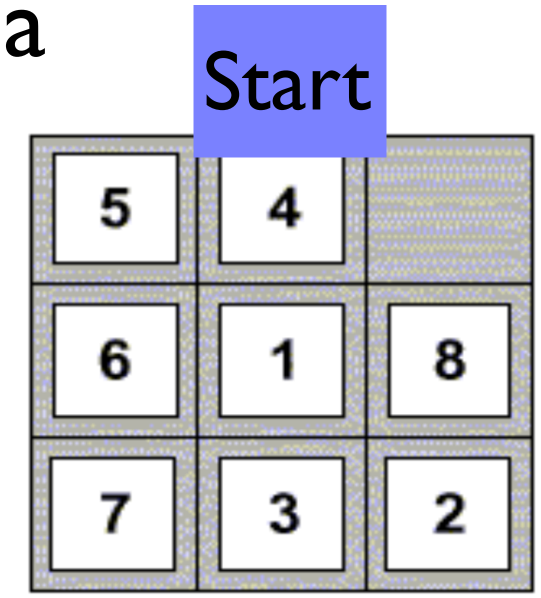
Example: Heuristics dominance

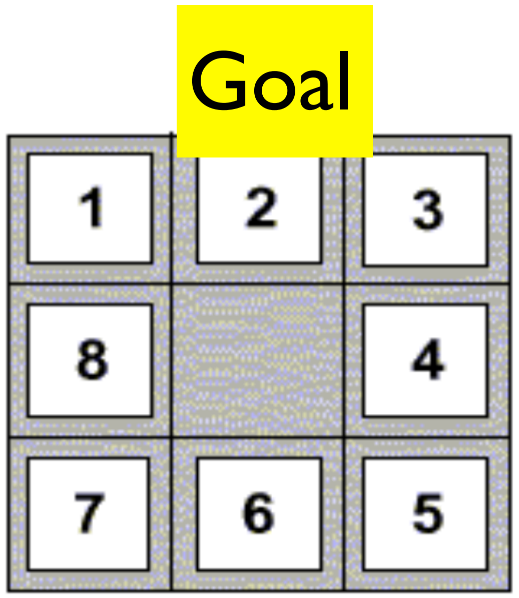
Consider the 8-puzzle problem, where you are allowed to move a tile to an adjacent square if it is empty.

Consider two heuristics for this problem.

$h_1(n) =$ tiles can move anywhere

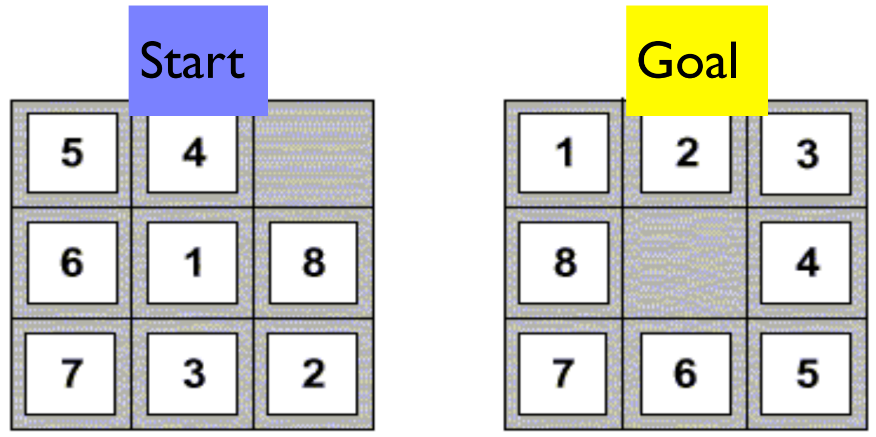
$h_2(n) =$ tiles can move to any adjacent square





Example: Heuristics dominance

Consider the 8-puzzle problem, where you are allowed to move a tile to an adjacent square if it is empty.



Consider two heuristics for this problem.

$h_1(n) =$ tiles can move anywhere

$h_2(n) =$ tiles can move to any adjacent square

	$h_1(n)$	$h_2(n)$
If tile in correct position	0	0
If tile one move from correct position	1	1
Otherwise	1	> 1

$$\boxed{h_2(n) \geq h_1(n)}$$

Heuristic dominance

If $h_2(n) \geq h_1(n)$ for every state n (both admissible heuristics) then h_2 dominates h_1. Which one is better for search?

A. h_1

B. h_2 ✅

C. It depends

D. 64

Heuristics dominance

Search costs for the 8-puzzle (average number of paths expanded). Averaged over 100 instances of the 8-puzzle, for various solutions.

$$\boxed{h_2(n) \geq h_1(n)}$$

	$d = 12$	$d = 24$
IDS	3,644,035 paths	too many paths
$A^*(h_1)$	227 paths	39,135 paths
$A^*(h_2)$	73 paths	1,641paths

Quality of estimate and work per node

- How about using the actual costs as a heuristic?
  - Would it be admissible
  - Would we save on the number of nodes expanded?
  - What's wrong with it?

Quality of estimate and work per node

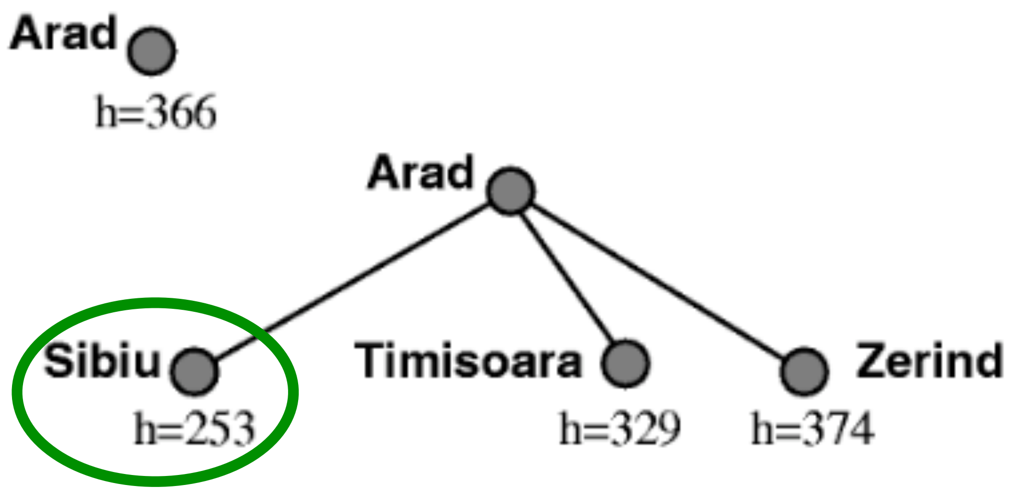
- How about using the actual costs as a heuristic?
  - Would it be admissible
  - Would we save on the number of nodes expanded?
  - What's wrong with it?

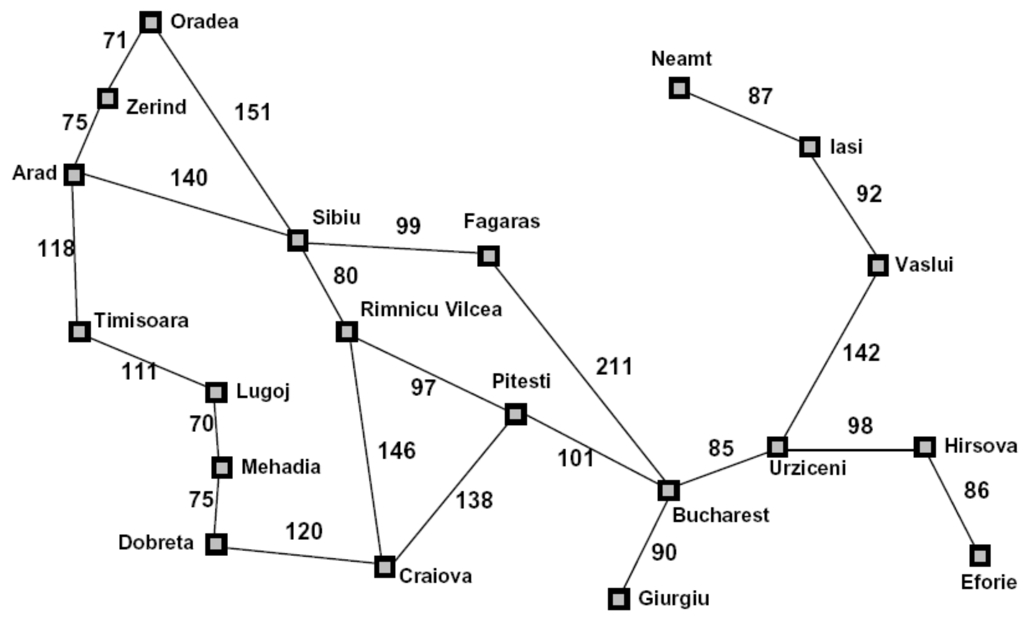
As heuristics gets closer to the true cost, you will expand fewer nodes but usually do more work per node to compute the heuristic itself.

Best-First Search (BestFS)

- Select the path whose end is closest to a goal according to the heuristic function.
- Best-First search selects a path on the frontier with minimum h-value (for the end node).
- It treats the frontier as a priority queue ordered by h.
- This is a **greedy** approach; it always takes the path which appears locally best.

BestFS example

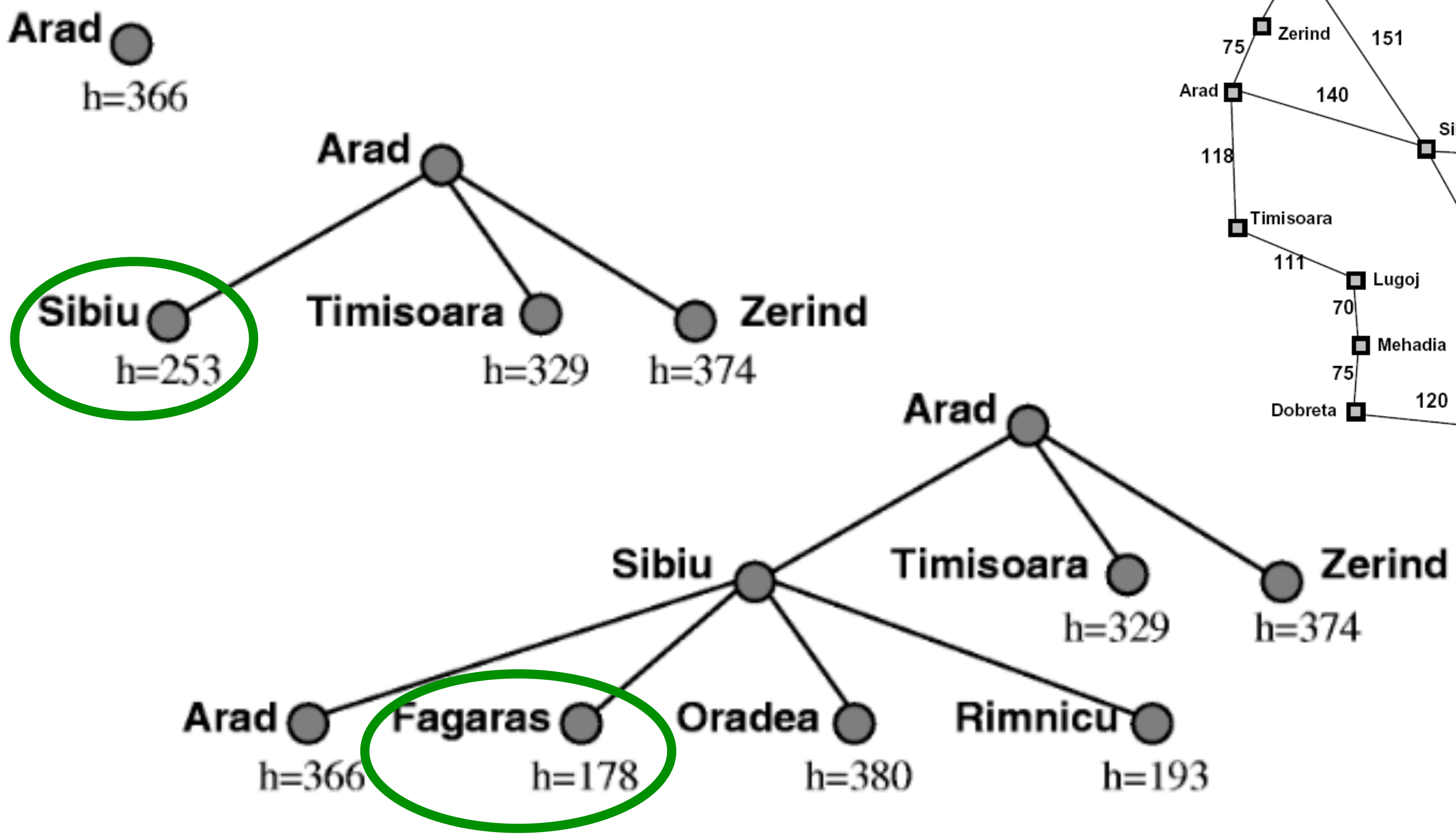


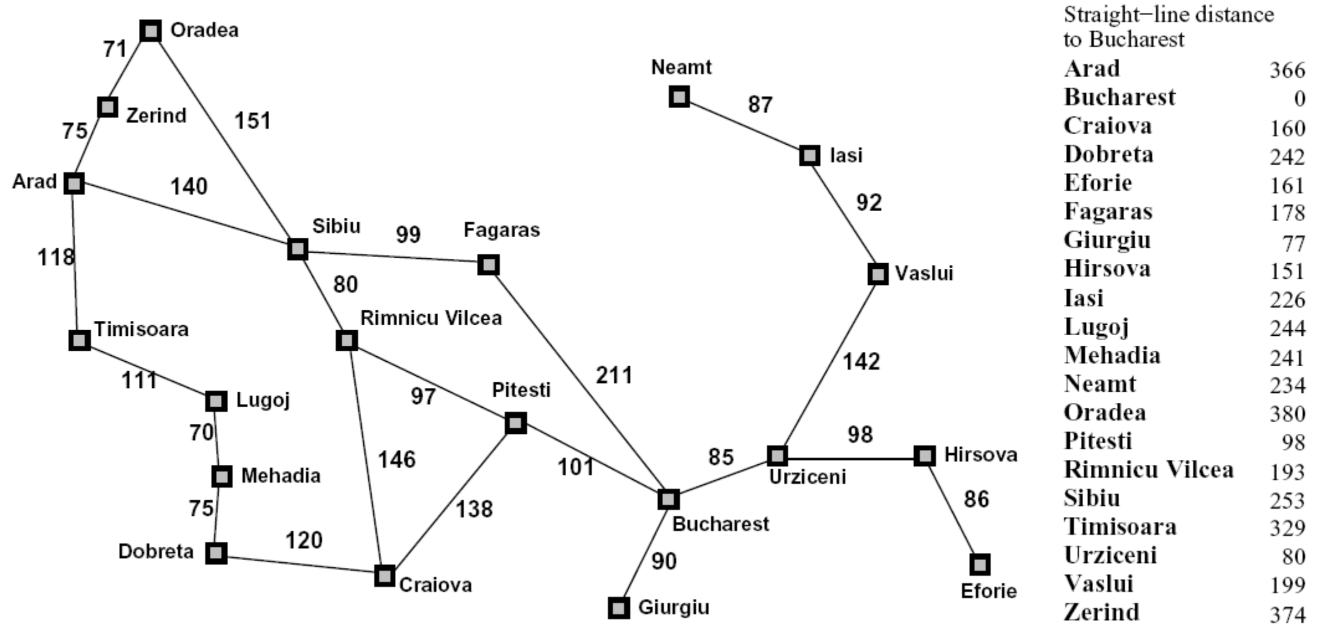


Straight−line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

BestFS example

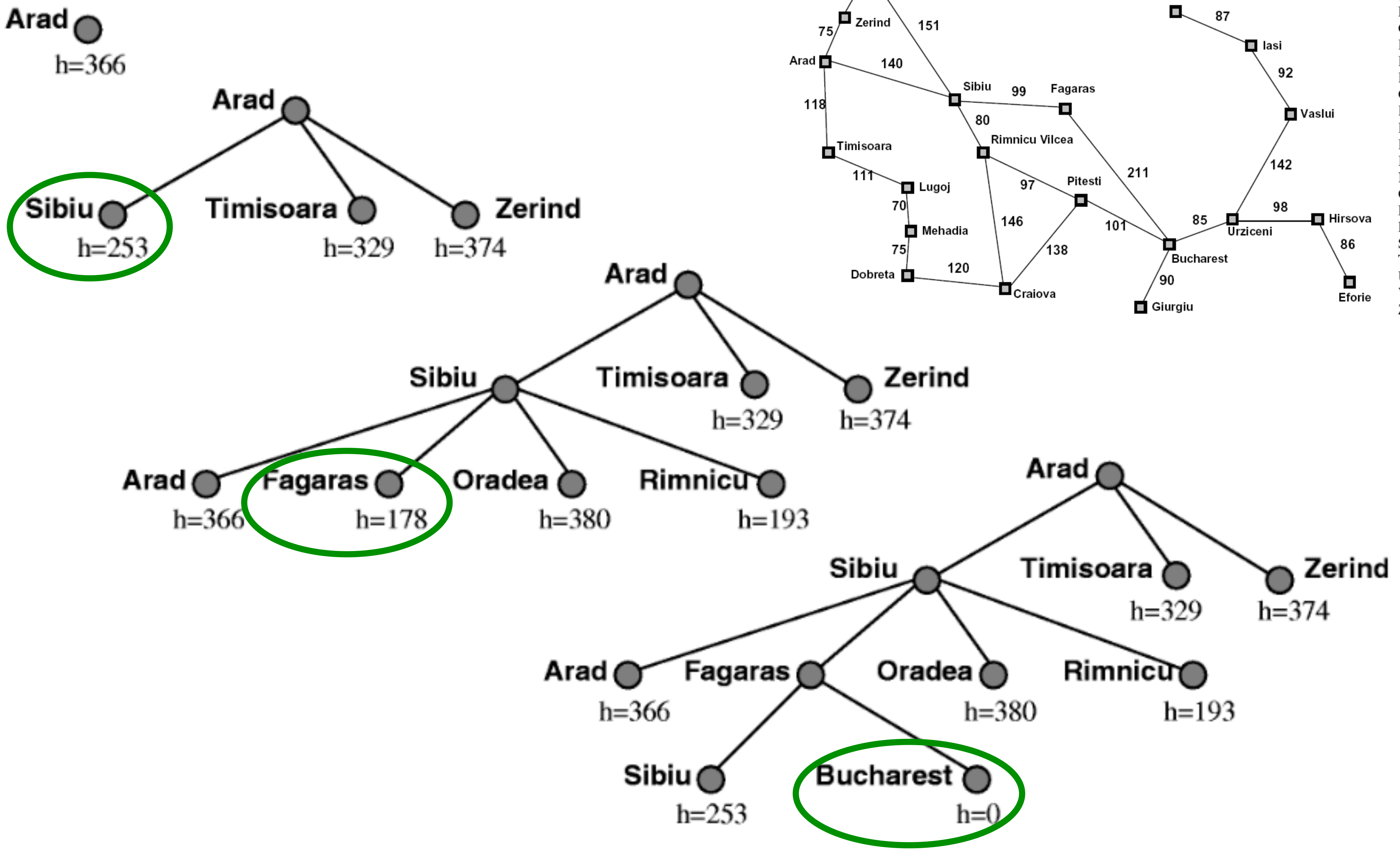


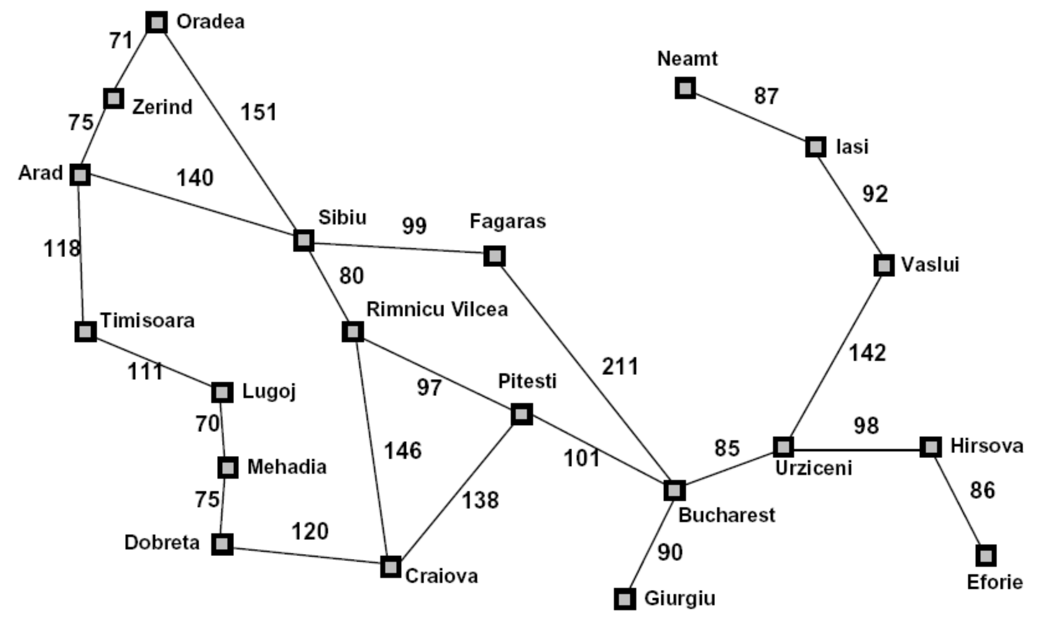


Straight−line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

BestFS example

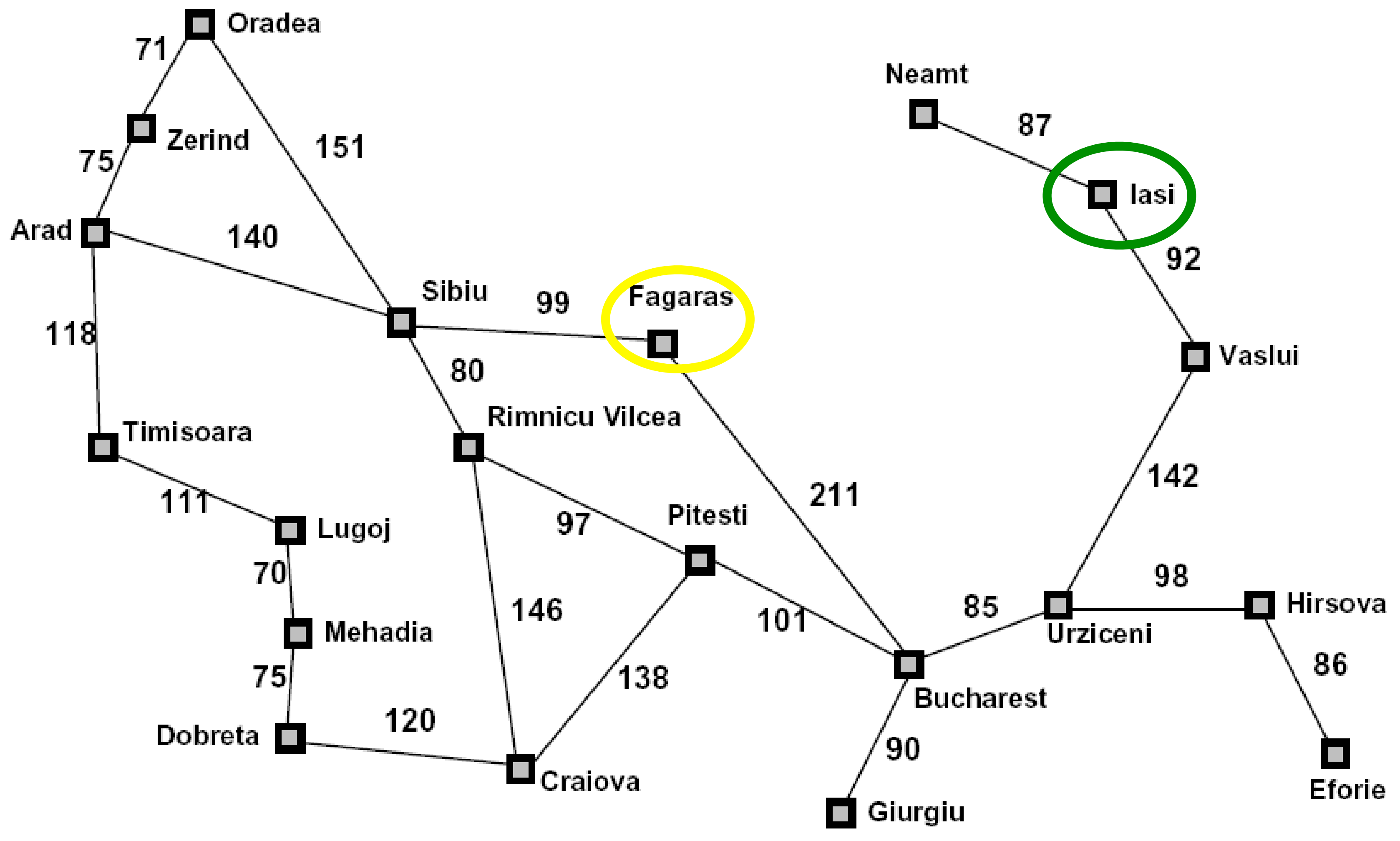




Straight−line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Is BestFS complete?



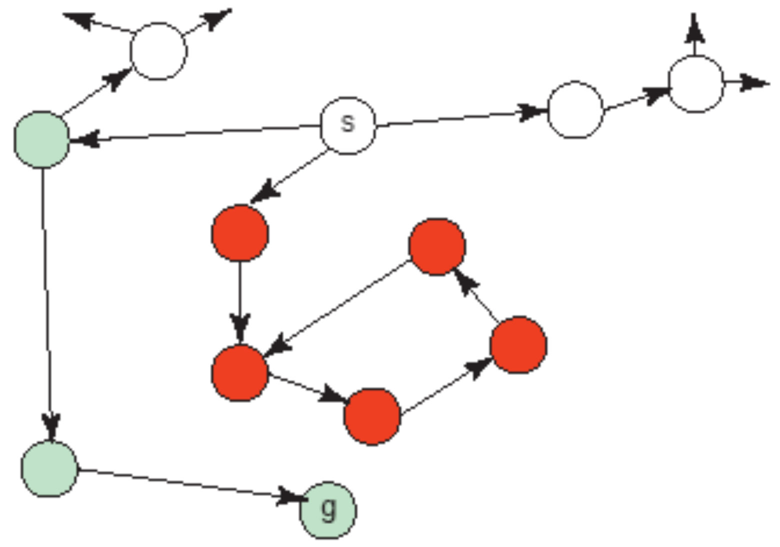
Straight−line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Consider the problem of getting from Iasi to Fagaras. The heuristic to expand Neamt which is a dead end.

Analysis of Best-First Search

- Completeness
  - A low heuristic value in a cycle can mean the cycle gets followed forever.



- Optimality
  - No (why not?)
- Worst case time and space complexity
  - $O(b^m)$ but an be reduced using a good heuristic function

A* search

- Combining LCFS and Best-First Search
- A* search takes into account both
  - the cost of the path to a node $cost(p)$
  - the heuristic value of that path $h(p)$
- Let $f(p) = cost(p) + h(p)$

$$\underbrace{\underbrace{\text{start} \xrightarrow{\text{actual}}}_{\text{cost}(p)} n \underbrace{\xrightarrow{\text{estimate}} \text{goal}}_{h(p)}}_{f(p)}$$

A* search

A* is a mix of the following two algorithms.

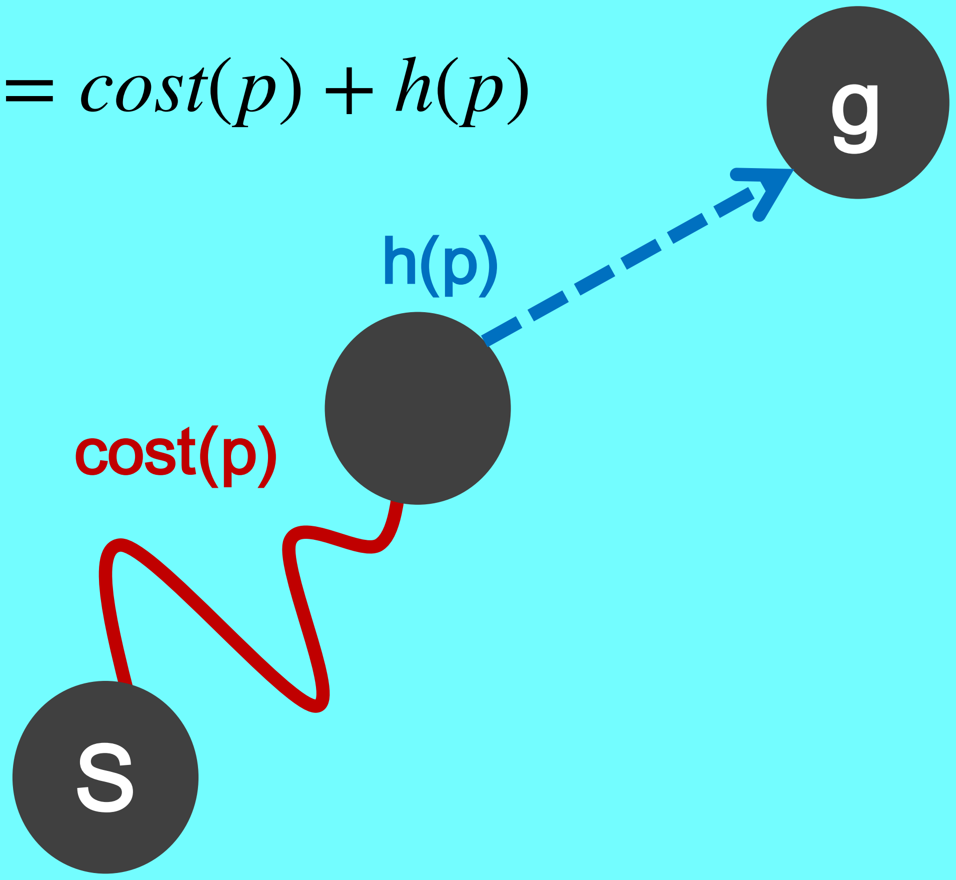
A. BFS and BestFS

B. LCFS and BestFS ✅

C. BFS and LCFS

D. None of the above

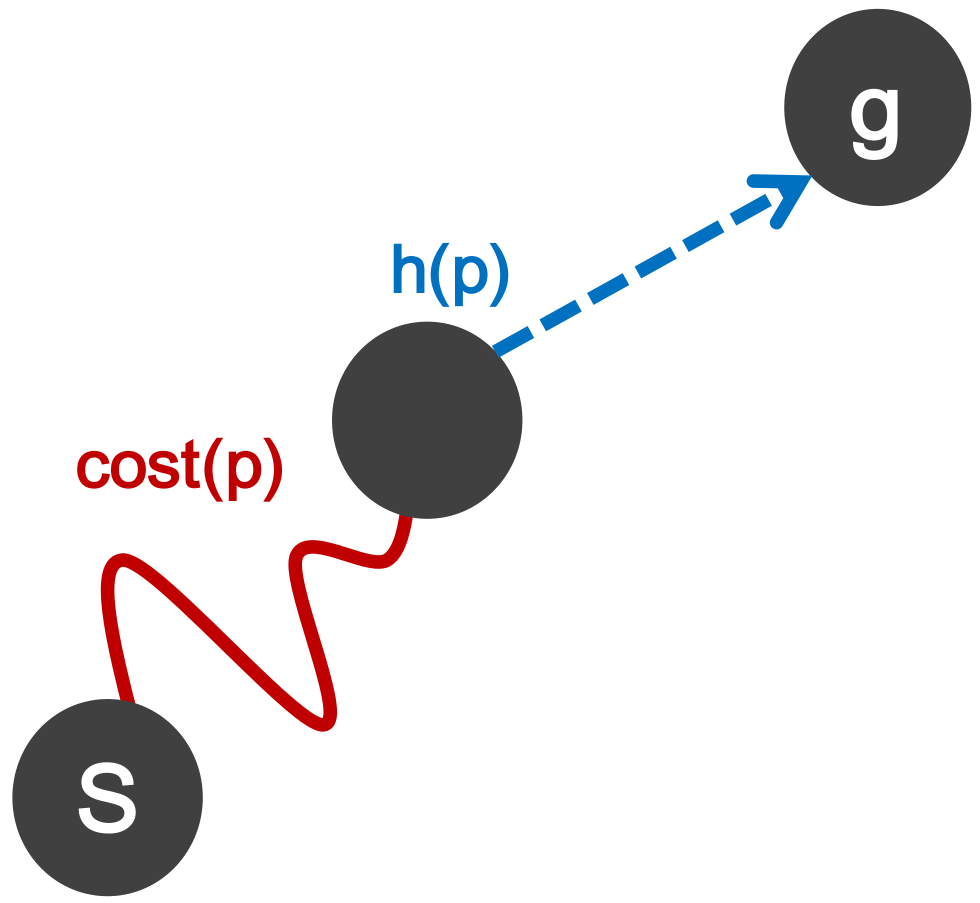
$$f(p) = cost(p) + h(p)$$



A* search

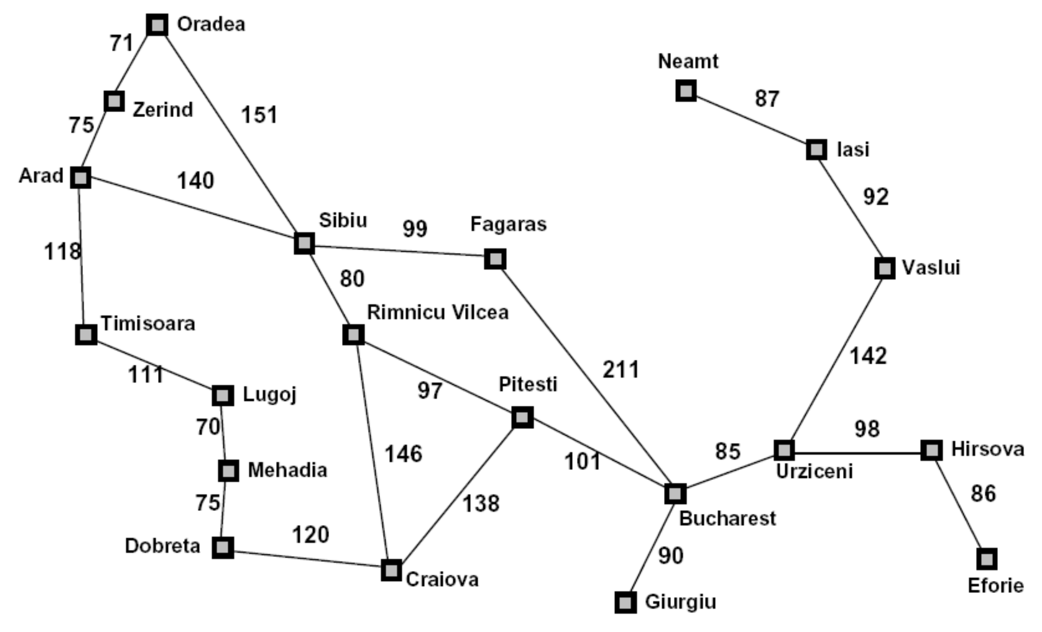
- A* treats the frontier as a priority queue ordered by $f(p) = cost(p) + h(p)$

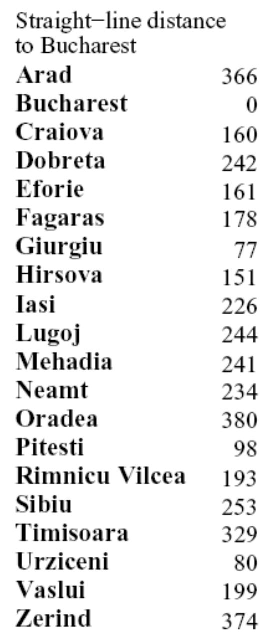
- It always selects the path on the frontier with the lowest estimated total distance to a goal.



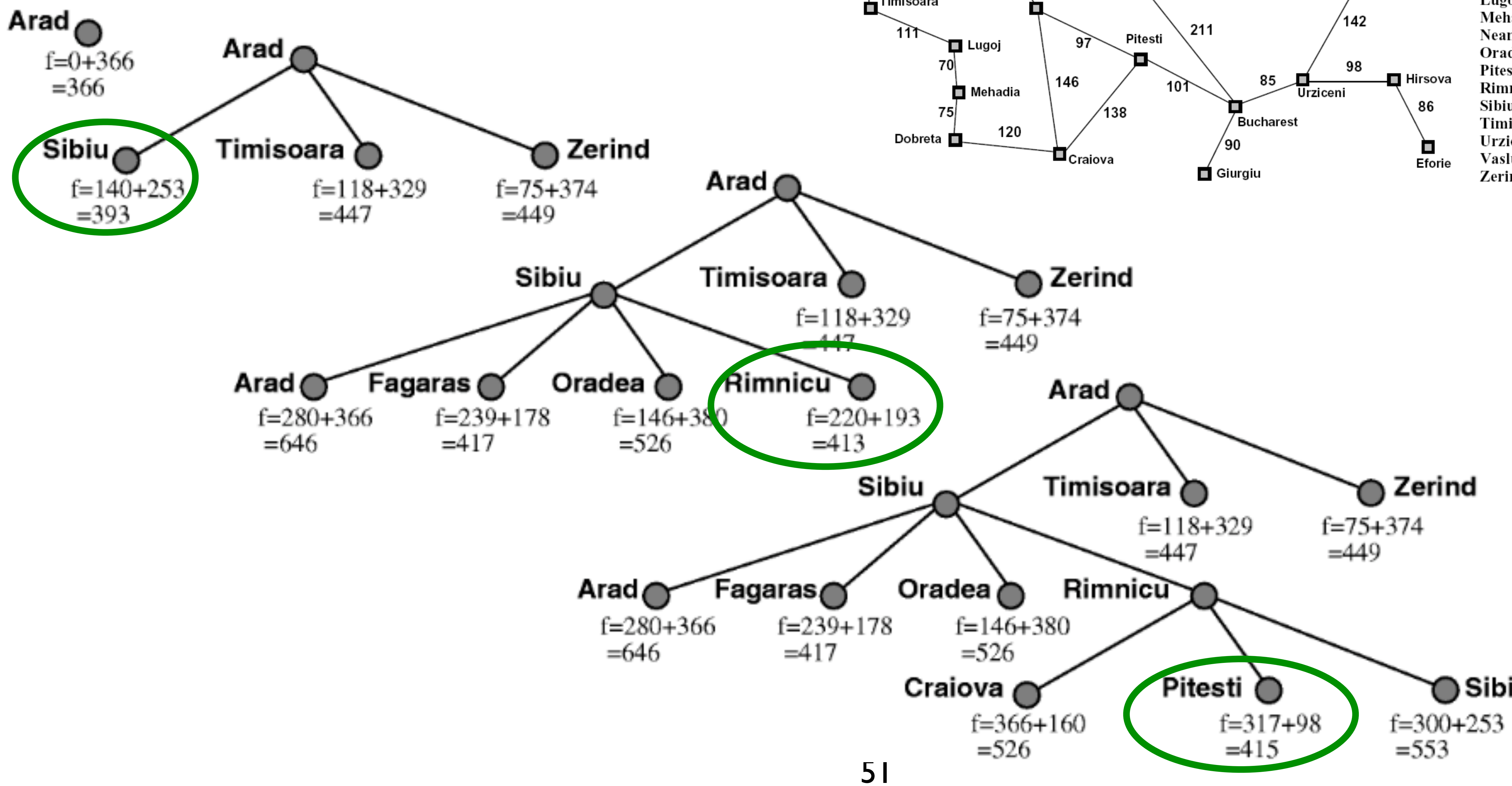
A* example: Travelling in Eastern Europe

$$f(p) = cost(p) + h(p)$$

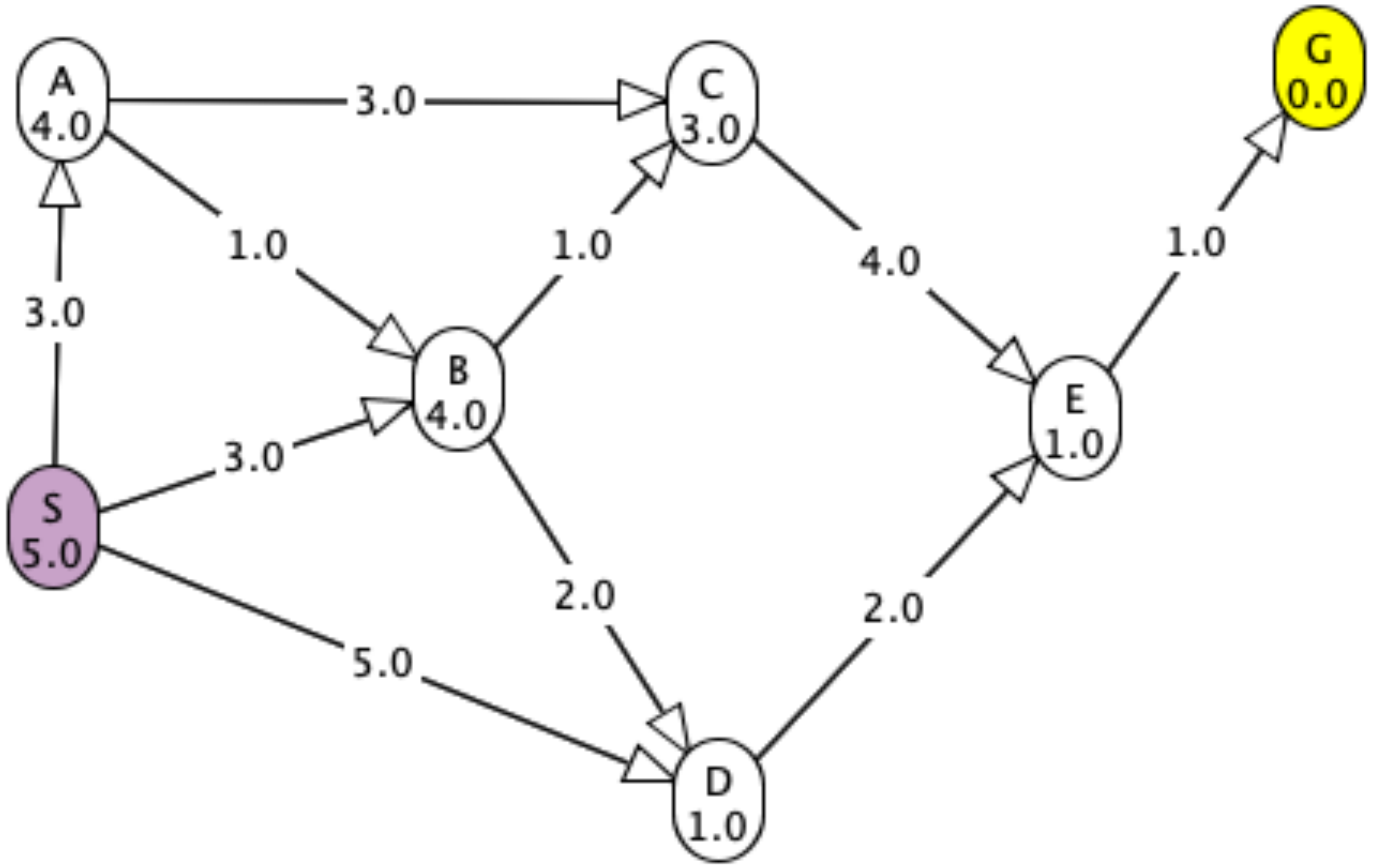




Straight−line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374



Computing *f*-values

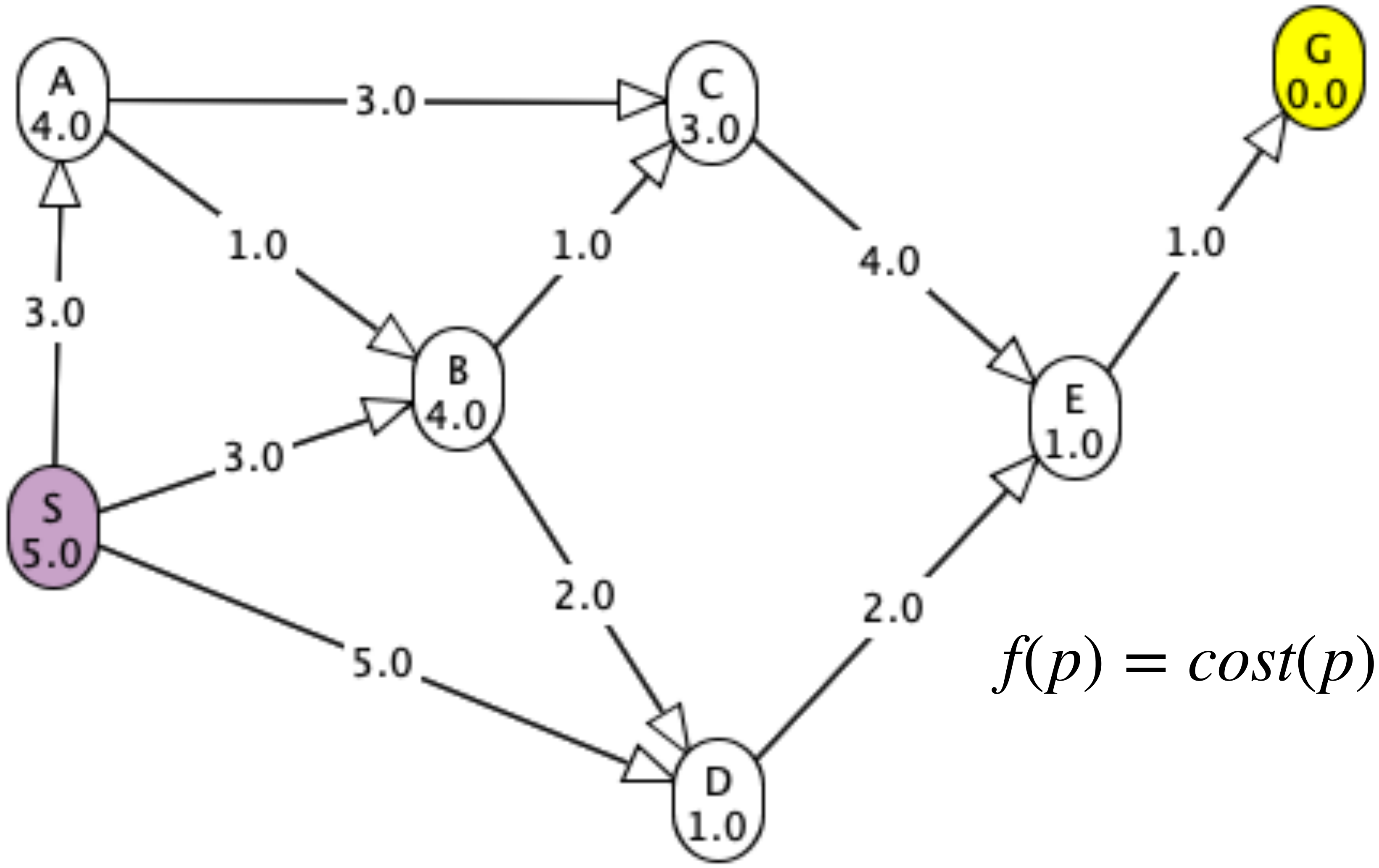


What is *f*-value of $s \to A \to B \to D$?

$cost(s \to A \to B \to D) + h(D)$

$= (3 + 1 + 2) + 1 = 7$

A* sample problem: group activity



$$f(p) = cost(p) + h(p)$$

- Trace through A* on this graph.
- Break ties alphabetically.
- What's the order of visited nodes
- What's the solution path and what is its cost?

A* search

If the heuristic is completely uninformative (e.g., $h = 0$ for all nodes) and the edge costs are all the same, A$*$ is equivalent to

A. LCFS

B. BFS ✅

C. DFS

D. A and B

Review: Learning outcomes

From this lecture, students are expected to be able to:

- Define admissible heuristic
- Construct admissible heuristics for a given problem
- Define/read/write/trace/debug Best-First search and A* search

Coming up

- Analysis of A*
- Optimality of A* search
- Branch and bound (3.8.1)