

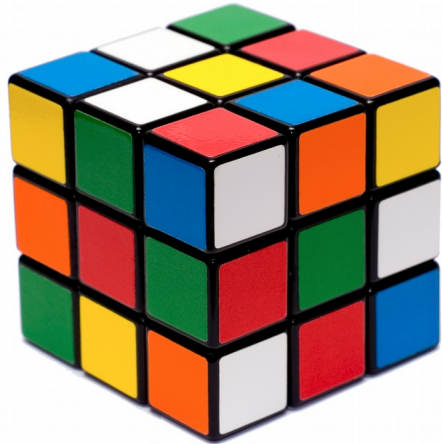
Heuristic Search

Rob Platt

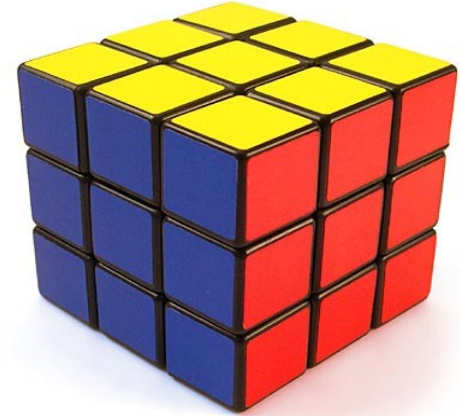
Northeastern University

Some images and slides are used from:
AIMA

Recap: What is graph search?



Start state

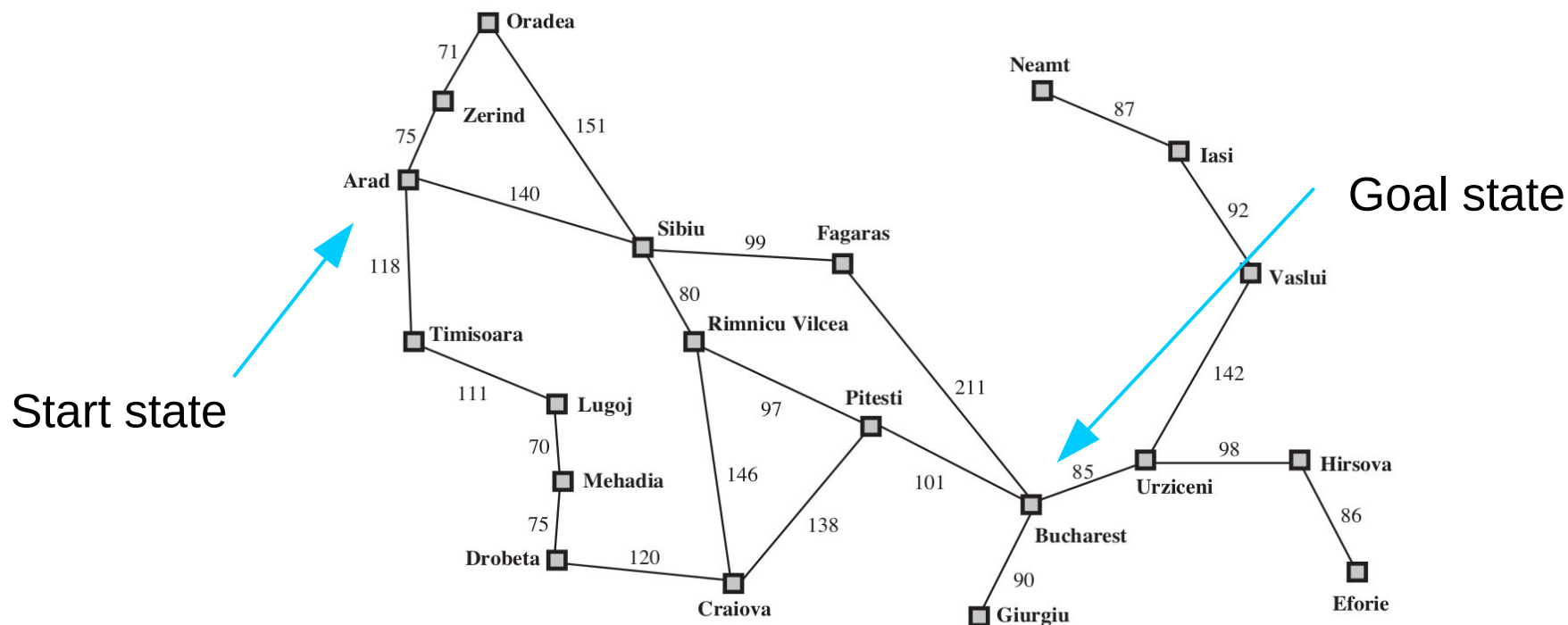


Goal state

Graph search: find a path from start to goal

- what are the states?
- what are the actions (transitions)?
- how is this a graph?

Recap: What is graph search?

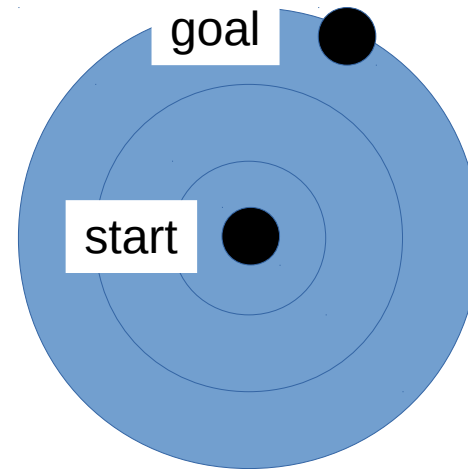


Graph search: find a path from start to goal

- what are the states?
- what are the actions (transitions)?
- how is this a graph?

Recap: BFS/UCS

It's like this



- search in all directions equally until discovering goal

Idea

Is it possible to use additional information to decide which direction to search in?

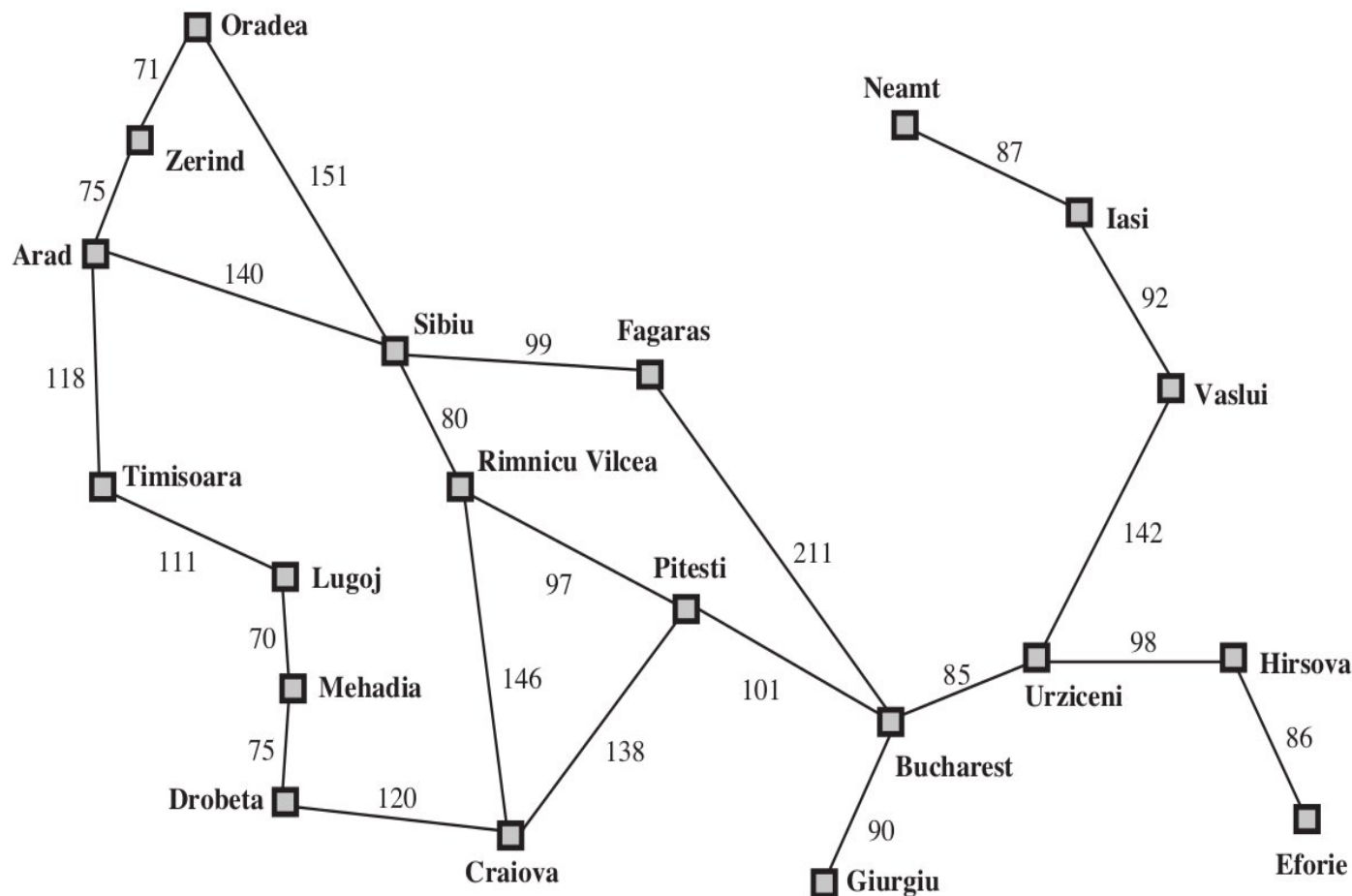
Idea

Is it possible to use additional information to decide which direction to search in?

Yes!

Instead of searching in all directions, let's bias search in the direction of the goal.

Example

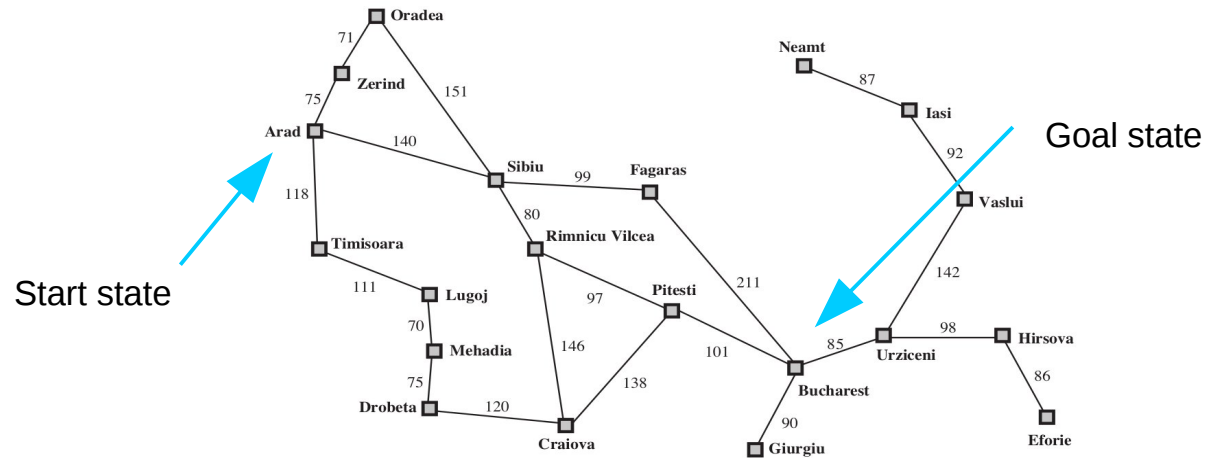


Arad	366
Bucharest	0
Craiova	160
Drobeta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	100
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374



Stright-line distances
to Bucharest

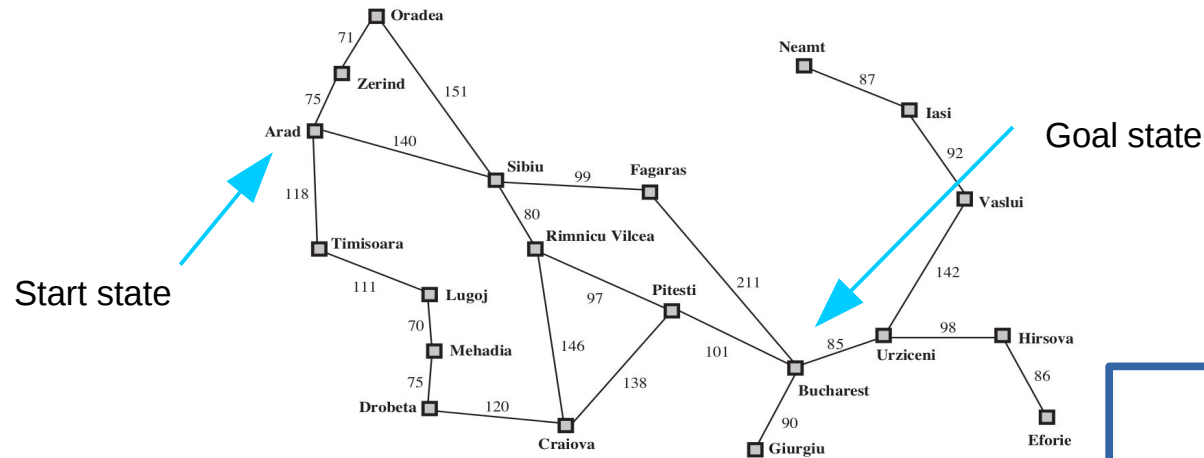
Example



Expand states in order of their distance to the goal

- for each state that you put on the fringe: calculate straight-line distance to the goal
- expand the state on the fringe closest to the goal

Example



Heuristic: $h(s)$

Expand states in order of their distance to the goal

- for each state that you put on the fringe: calculate straight-line distance to the goal
- expand the state on the fringe closest to the goal


Greedy search

Greedy Search



Greedy Search

Each time you expand a state, calculate the heuristic for each of the states that you add to the fringe.

- heuristic: $h(s)$  i.e. distance to Bucharest
- on each step, choose to expand the state with the lowest heuristic value.

Greedy Search

This is like a guess about how far the state is from the goal

Each time you expand a state, calculate the heuristic for each of the states that you add to the fringe.

- heuristic: $h(s)$ ← i.e. distance to Bucharest
- on each step, choose to expand the state with the lowest heuristic value.

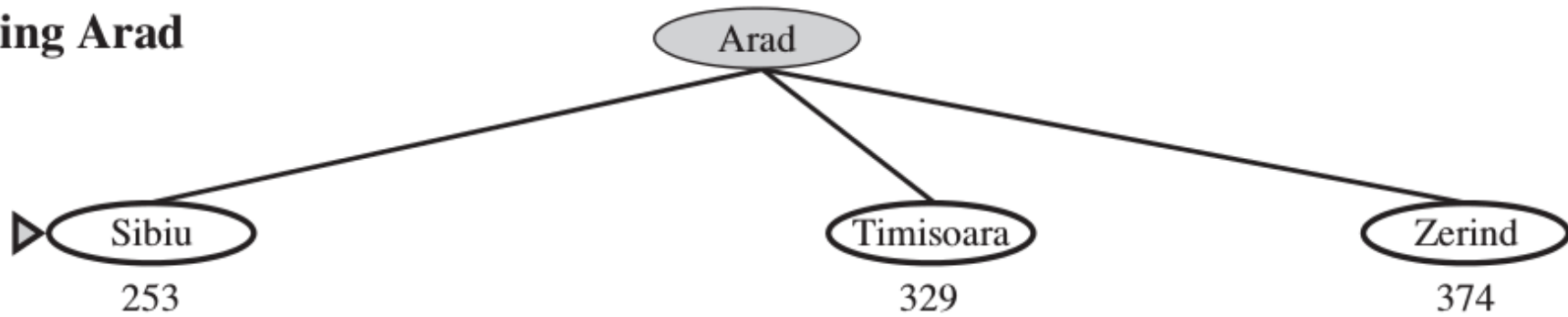
Example: Greedy Search

(a) The initial state



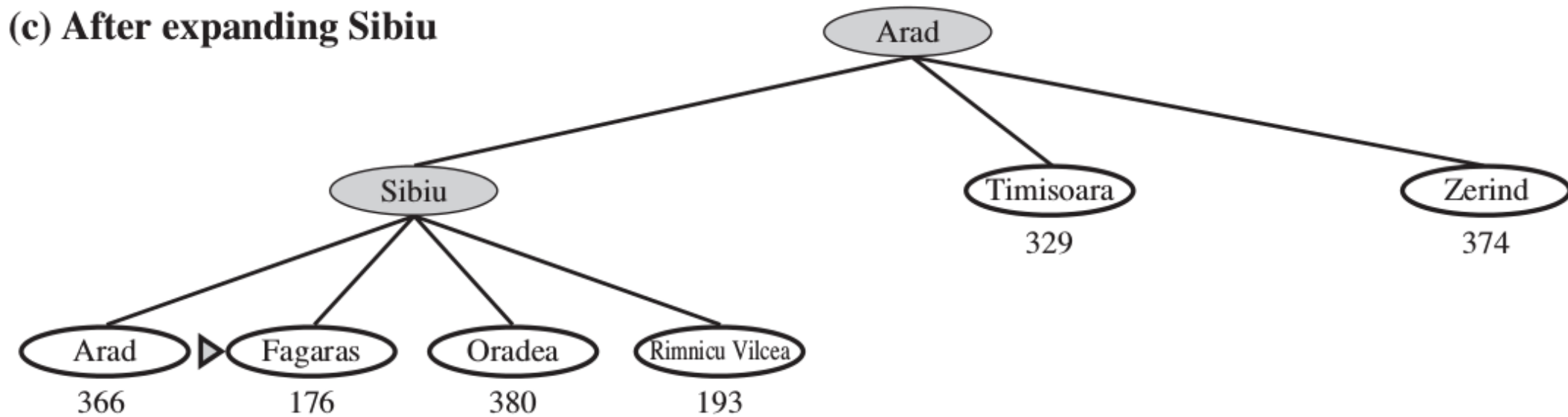
Example: Greedy Search

(b) After expanding Arad



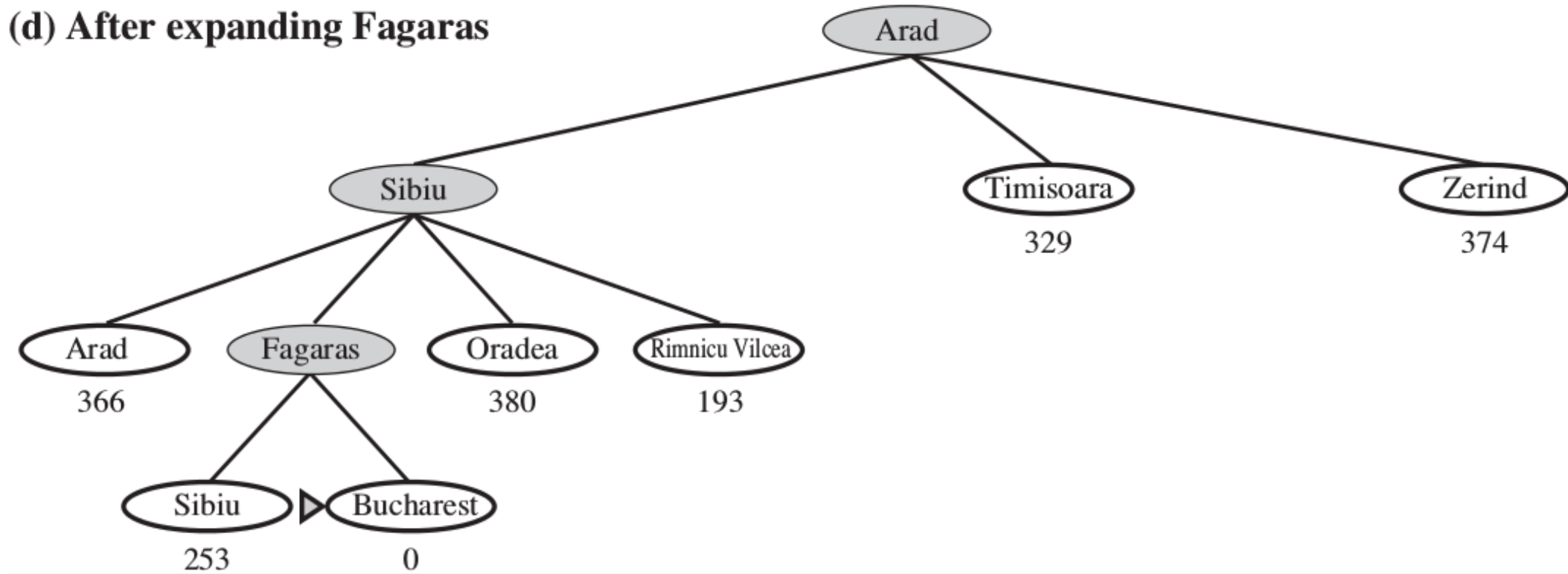
Example: Greedy Search

(c) After expanding Sibiu



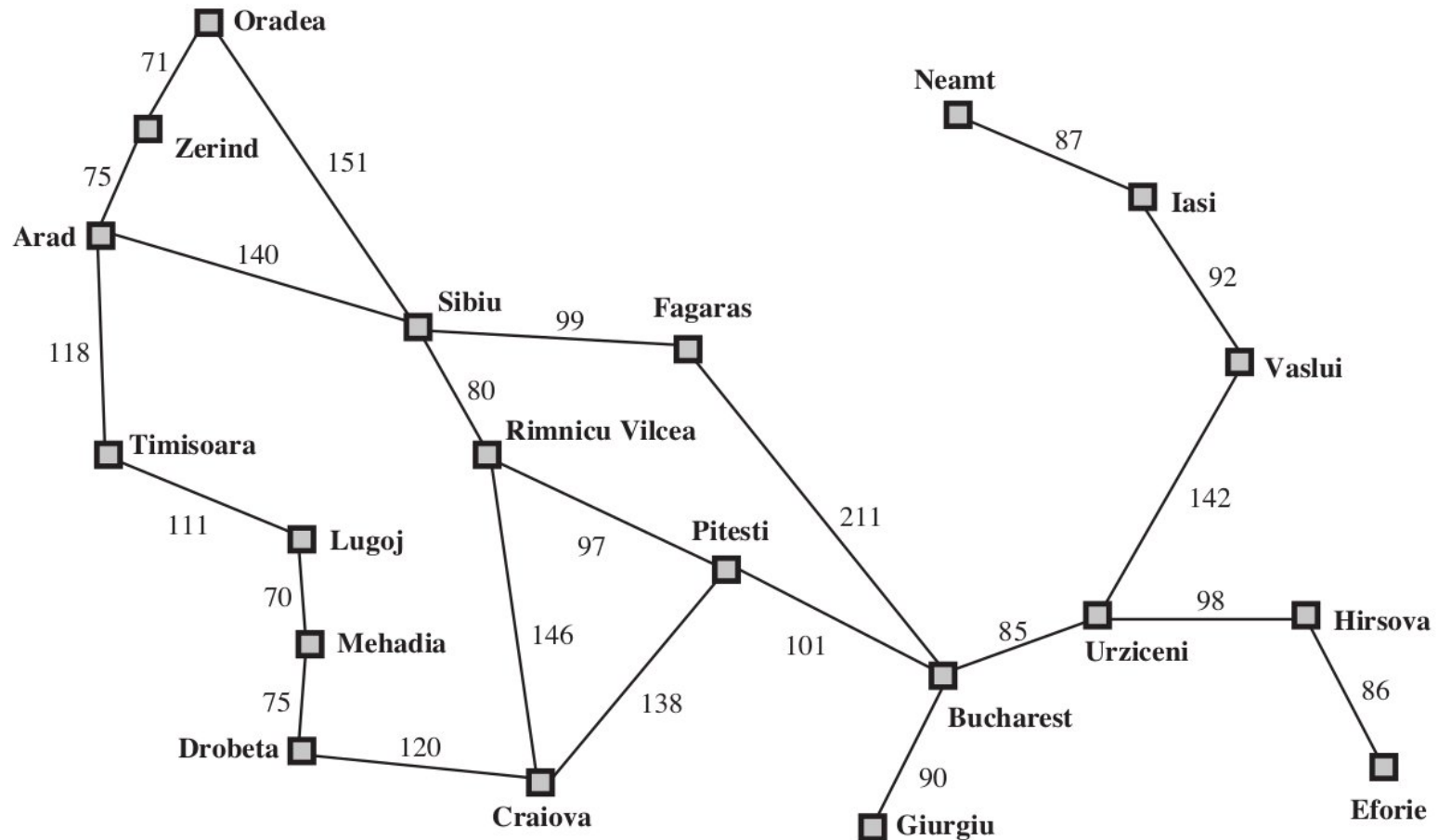
Example: Greedy Search

(d) After expanding Fagaras



Path: A-S-F-B

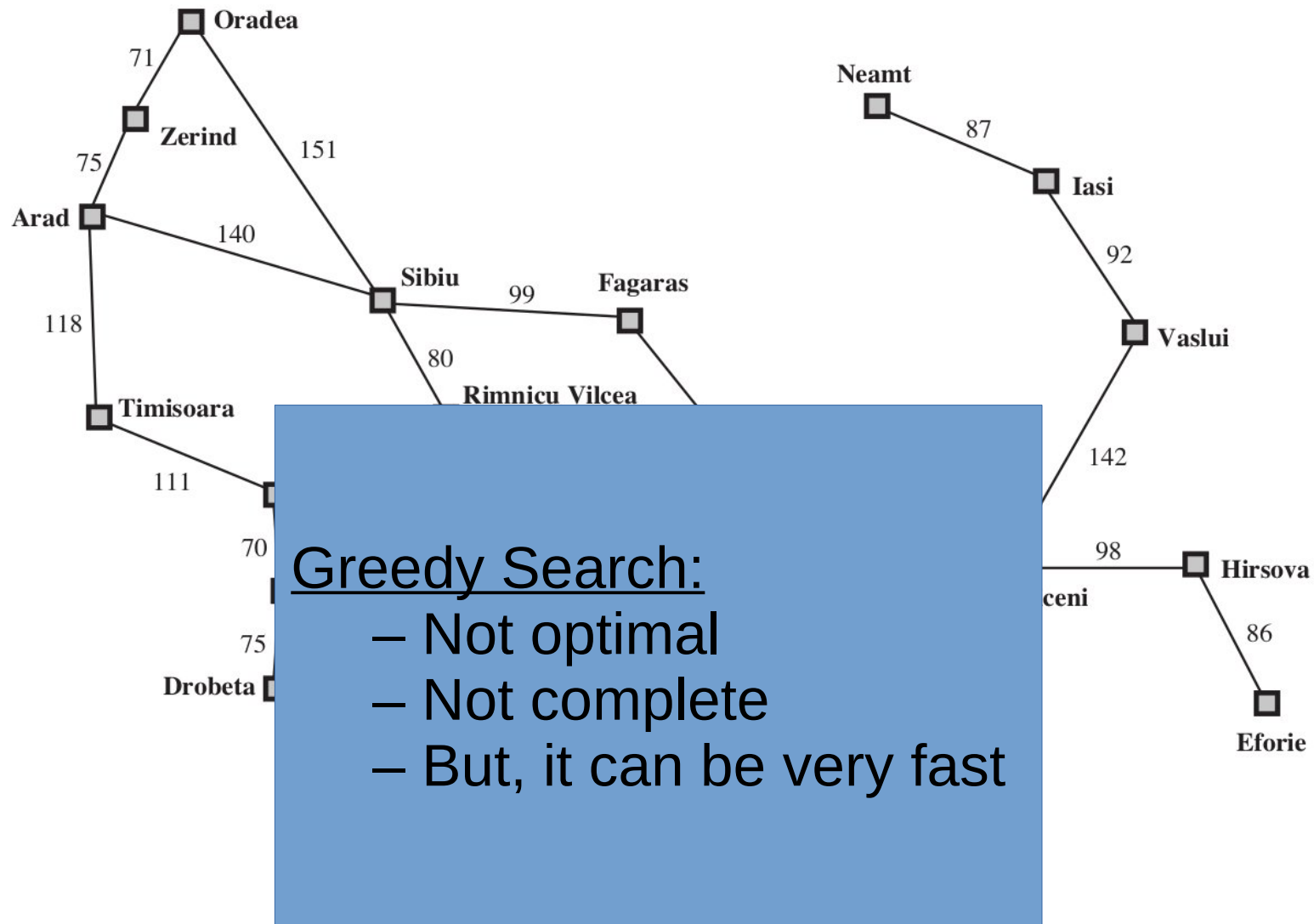
Example: Greedy Search



Path: A-S-F-B

Notice that this is not the optimal path!

Example: Greedy Search



Notice that this is not the optimal path!

Greedy vs UCS

Greedy Search:

- Not optimal
- Not complete
- But, it can be very fast

UCS:

- Optimal
- Complete
- Usually very slow

Greedy vs UCS

Greedy Search:

- Not optimal
- Not complete
- But, it can be very fast

UCS:

- Optimal
- Complete
- Usually very slow

Can we combine greedy and UCS???

Greedy vs UCS

Greedy Search:

- Not optimal
- Not complete
- But, it can be very fast

UCS:

- Optimal
- Complete
- Usually very slow

Can we combine greedy and UCS???

YES: A*

A^*



A*

s : a state

$g(s)$: minimum cost from start to

$h(s)$: heuristic at (i.e. an estimate of remaining cost-to-go)

UCS: expand states in order of $g(s)$

Greedy: expand states in order of $h(s)$

A*: expand states in order of $f(s) = g(s) + h(s)$

A^*

What is “cost-to-go”?

s : a state

$g(s)$: minimum cost from start to s

$h(s)$: heuristic at s (i.e. an estimate of remaining
cost-to-go)



UCS: expand states in order of $g(s)$

Greedy: expand states in order of $h(s)$

A^* : expand states in order of $f(s) = g(s) + h(s)$

A*

What is “cost-to-go”?

s : a state – minimum cost required
to reach a goal state

$g(s)$: minimum cost from start to s

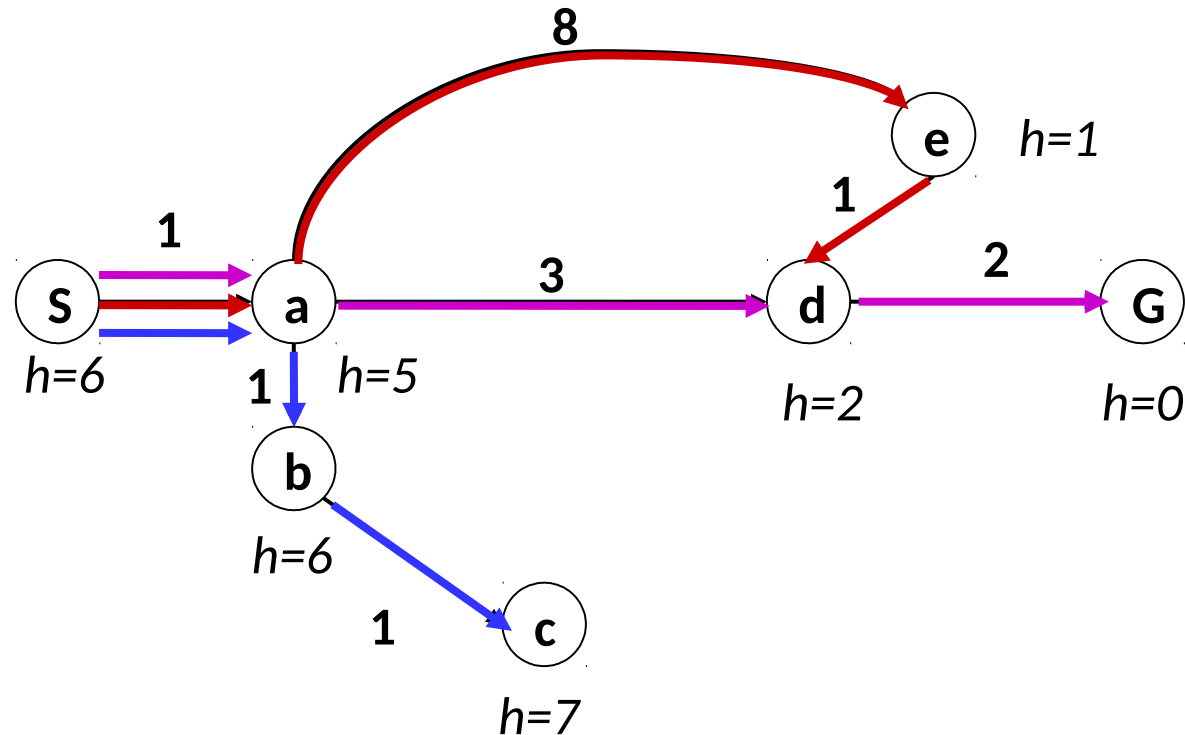
$h(s)$: heuristic at s (i.e. an estimate of remaining
cost-to-go)

UCS: expand states in order of $g(s)$

Greedy: expand states in order of $h(s)$

A*: expand states in order of $f(s) = g(s) + h(s)$

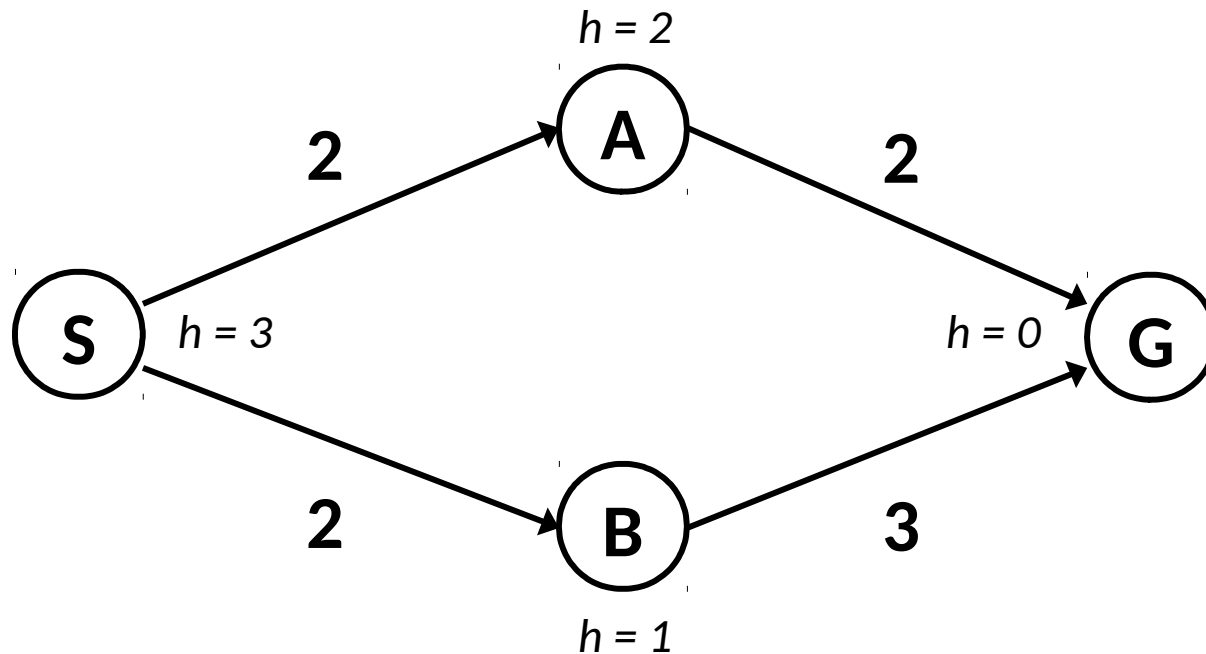
A^*



- **Uniform-cost** orders by path cost from Start: $g(n)$
- **Greedy** orders by estimated cost to goal: $h(n)$
- **A^*** orders by the sum: $f(n) = g(n) + h(n)$

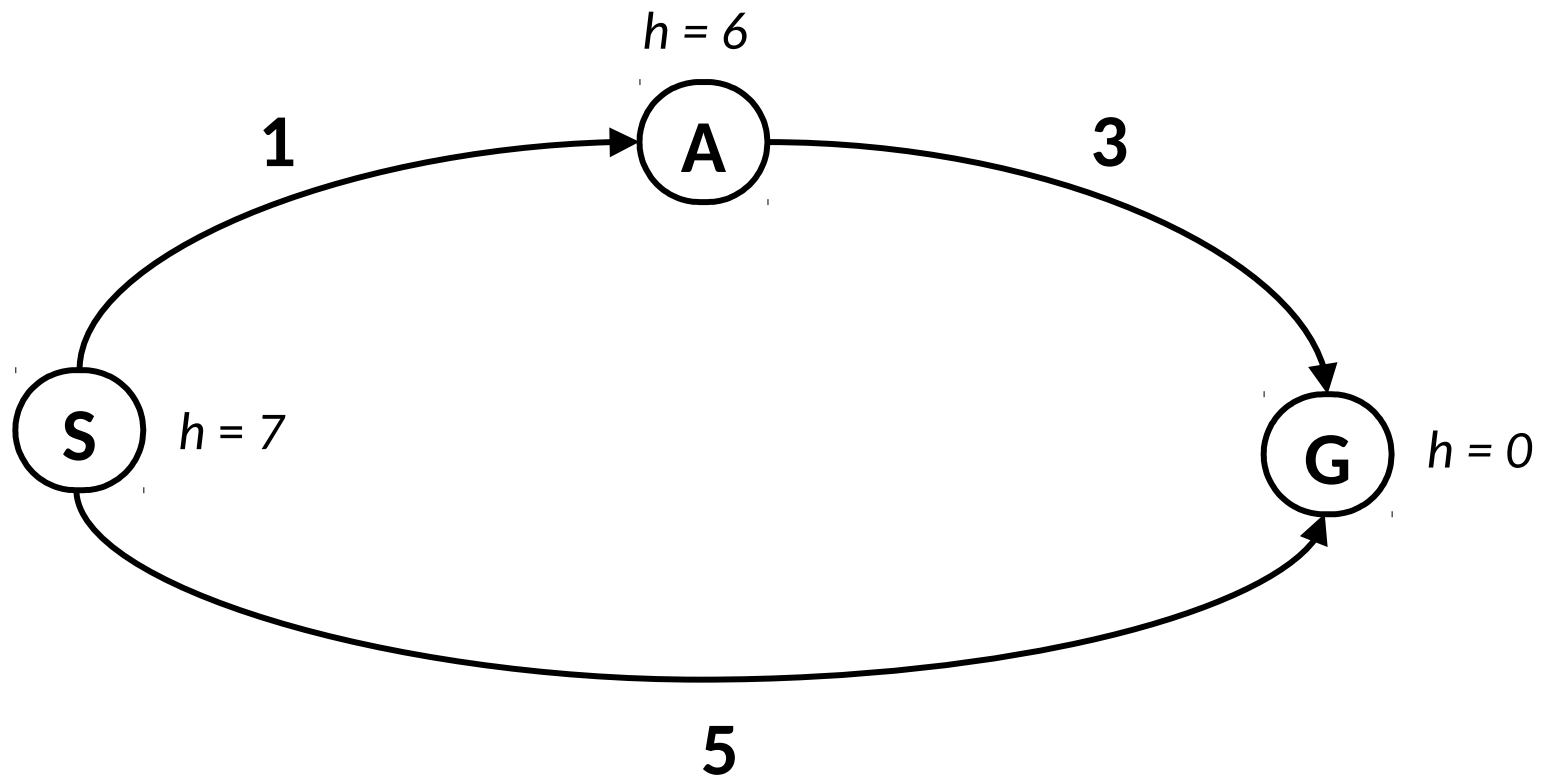
When should A* terminate?

Should we stop when we enqueue a goal?



No: only stop when we dequeue a goal

Is A^* optimal?



What went wrong here?

When is A^* optimal?

It depends on whether we are using the tree search or the graph search version of the algorithm.

Recall:

- in tree search, we do not track the explored set
- in graph search, we do

Recall: Breadth first search (BFS)

```
function BREADTH-FIRST-SEARCH(problem) returns a solution, or failure  
  node  $\leftarrow$  a node with STATE = problem.INITIAL-STATE, PATH-COST = 0  
  if problem.GOAL-TEST(node.STATE) then return SOLUTION(node)  
  frontier  $\leftarrow$  a FIFO queue with node as the only element  
  explored  $\leftarrow$  an empty set  
  loop do  
    if EMPTY?(frontier) then return failure  
    node  $\leftarrow$  POP(frontier) /* chooses the shallowest node in frontier */  
    add node.STATE to explored  
    for each action in problem.ACTIONS(node.STATE) do  
      child  $\leftarrow$  CHILD-NODE(problem, node, action)  
      if child.STATE is not in explored or frontier then  
        if problem.GOAL-TEST(child.STATE) then return SOLUTION(child)  
        frontier  $\leftarrow$  INSERT(child, frontier)
```

Figure 3.11 Breadth-first search on a graph.

What is the purpose of the *explored* set?

When is A* optimal?

It depends on whether we are using the tree search or the graph search version of the algorithm.

Optimal if h is consistent



Optimal if h is admissible

When is A* optimal?

It depends on whether we are using the tree search or the graph search version of the algorithm.

Optimal if h is consistent

– $h(s)$ is an underestimate of the cost of each arc.

Optimal if h is admissible

– $h(s)$ is an underestimate of the true cost-to-go.

When is A* optimal?

It depends on whether we are using the tree search or the graph search version of the algorithm.

Optimal if h is consistent

– $h(s)$ is an underestimate of the cost of each arc.

Optimal if h is admissible

– $h(s)$ is an underestimate of the true cost-to-go.

What is “cost-to-go”?

– minimum cost required to reach a goal state

When is A* optimal?

It depends on whether we are using the tree search or the graph search version of the algorithm.

Optimal if h is consistent

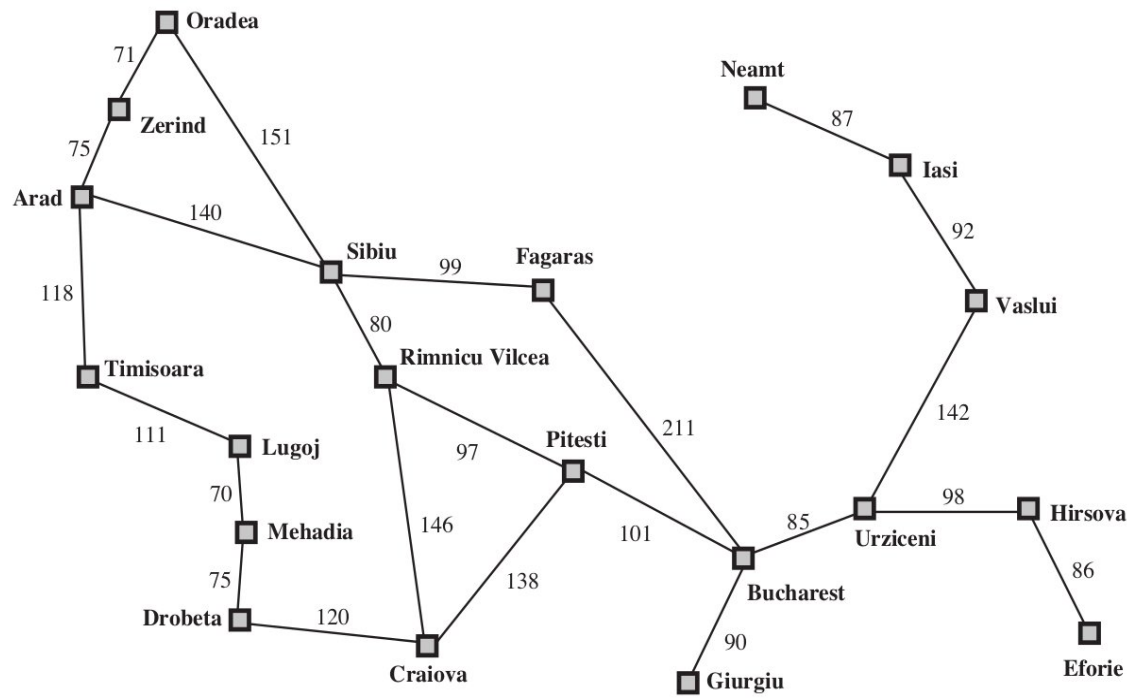
– $h(s)$ is an underestimate of the cost of each arc.

Optimal if h is admissible

– $h(s)$ is an underestimate of the true cost-to-go.

More on this later...

Admissibility: Example



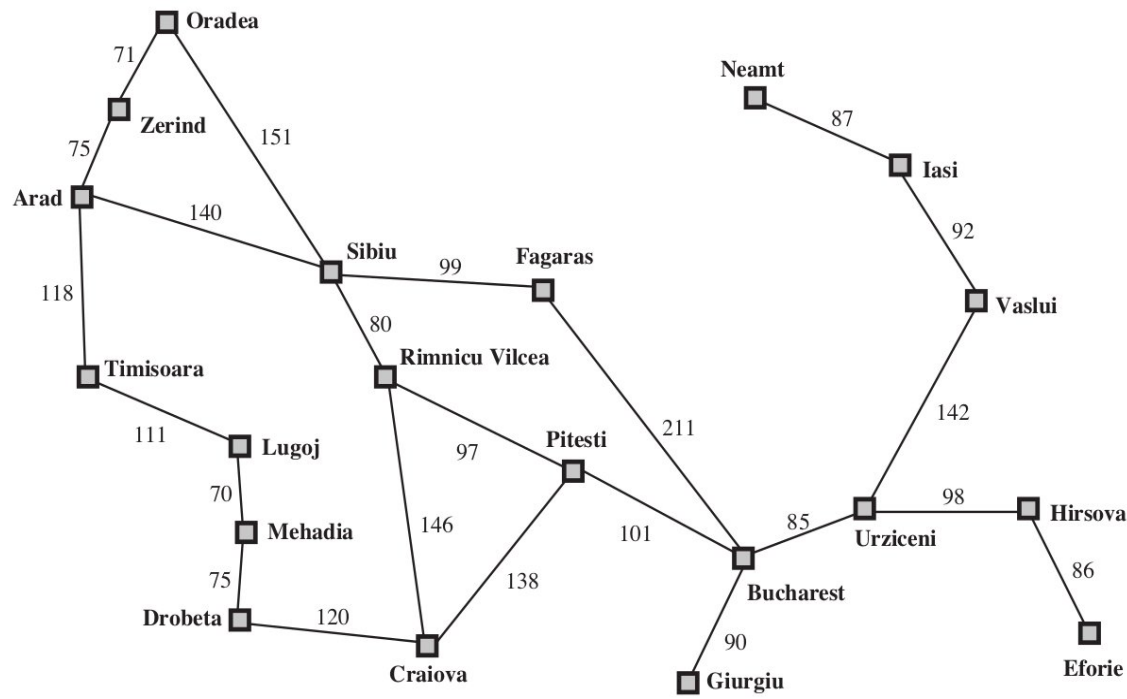
Arad	366
Bucharest	0
Craiova	160
Drobeta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	100
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374



Straight-line distances
to Bucharest

$h(s)$ = straight-line distance to goal state (Bucharest)

Admissibility



Arad	366
Bucharest	0
Craiova	160
Drobeta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	100
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

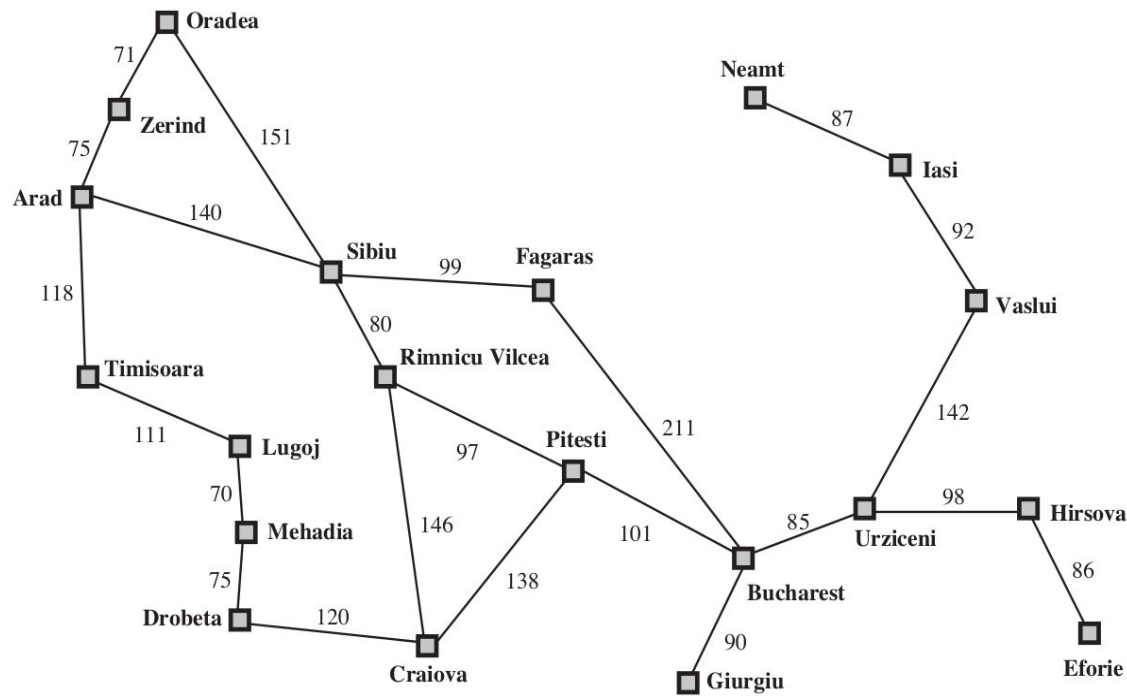


Straight-line distances
to Bucharest

$h(s)$ = straight-line distance to goal state (Bucharest)

Is this heuristic admissible???

Admissibility



Arad	366
Bucharest	0
Craiova	160
Drobeta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	100
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374



Straight-line distances
to Bucharest

$h(s)$ = straight-line distance to goal state (Bucharest)

Is this heuristic admissible???

YES! Why?

Admissibility: Example

7	2	4
5		6
8	3	1

Start state



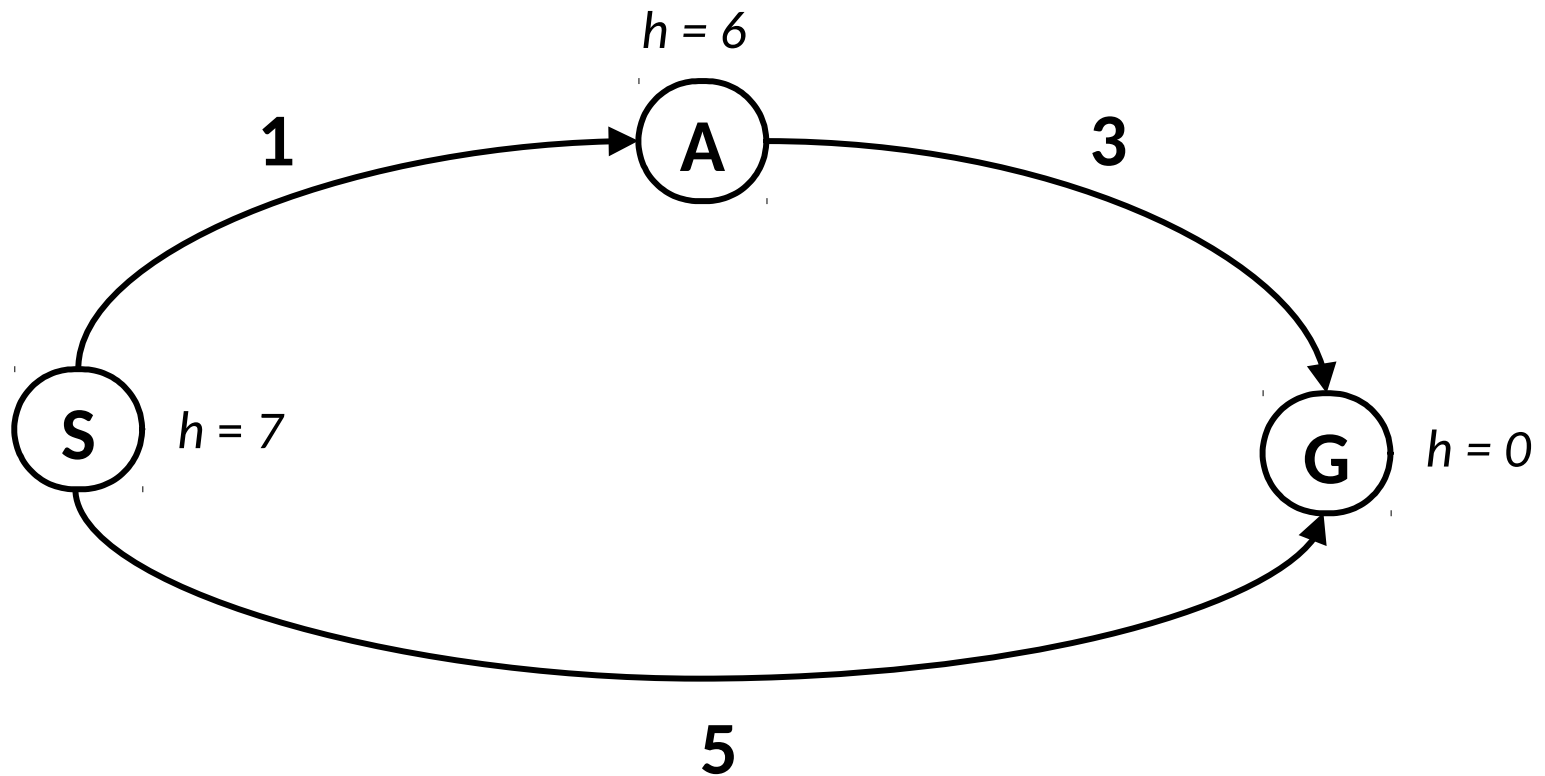
	1	2
3	4	5
6	7	8

Goal state

$$h(s) = ?$$

Can you think of an admissible heuristic for this problem?

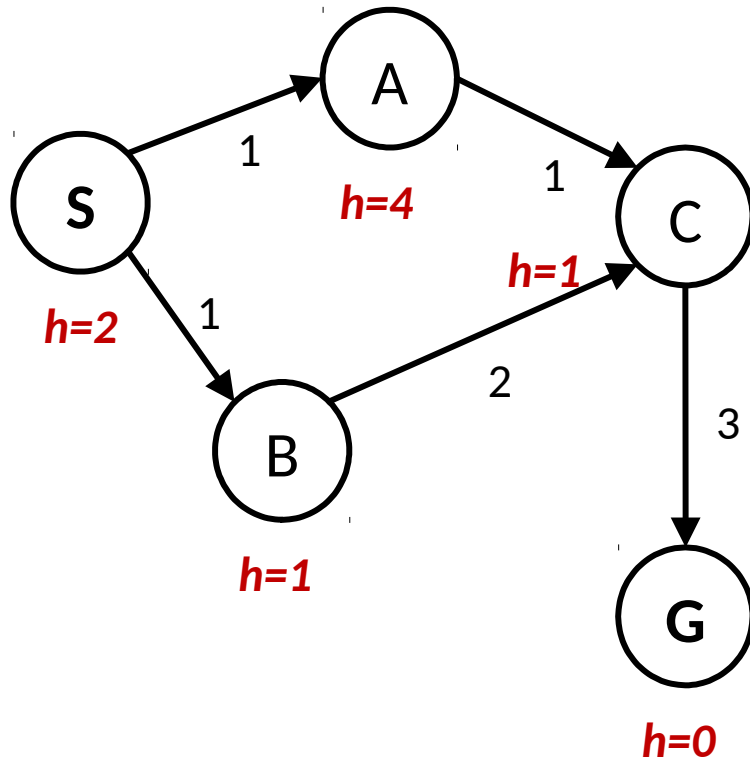
Admissibility



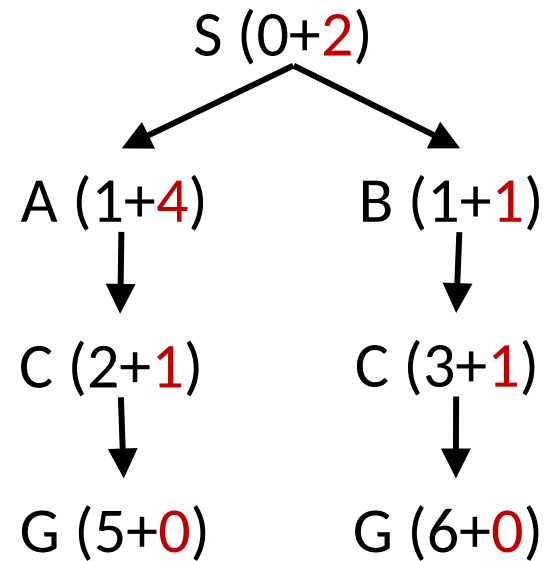
Why isn't this heuristic admissible?

Consistency

State space graph



Search tree



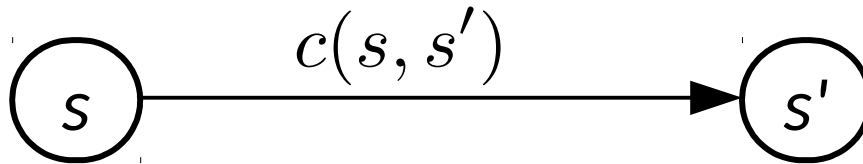
What went wrong?

Consistency

$$h(s) \leq c(s, s') + h(s')$$



Cost of going from s to s'



Consistency

$$h(s) \leq c(s, s') + h(s')$$

$$h(s) - h(s') \leq c(s, s') \quad \leftarrow \text{Rearrange terms}$$

Consistency

$$h(s) \leq c(s, s') + h(s')$$

$$\underbrace{h(s) - h(s')} \leq c(s, s')$$

Cost of going from s to s'
implied by heuristic

Actual cost of
going from s to s'



Consistency

$$h(s) \leq c(s, s') + h(s')$$

$$\underbrace{h(s) - h(s')} \leq c(s, s')$$

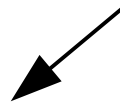
Cost of going from s to s'
implied by heuristic

Actual cost of
going from s to s'



Consistency

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



Consistency implies that the “f-cost” never decreases along any path to a goal state.

- the optimal path gives a goal state its lowest f-cost.

A* expands states in order of their f-cost.

Guarantees that A* expands states that reach the goal state optimally before expanding states that reach the goal state suboptimally.

Consistency implies admissibility

Suppose: $\forall s_t, s_{t+1} : h(s_t) \leq c(s_t, s_{t+1}) + h(s_{t+1})$

Then: $h(s_{T-1}) \leq c(s_{T-1}, s_T) + h(s_T)$


Consistency implies admissibility

Suppose: $\forall s_t, s_{t+1} : h(s_t) \leq c(s_t, s_{t+1}) + h(s_{t+1})$

Then: $h(s_{T-1}) \leq c(s_{T-1}, s_T)$

Consistency implies admissibility

Suppose: $\forall s_t, s_{t+1} : h(s_t) \leq c(s_t, s_{t+1}) + h(s_{t+1})$

Then: $h(s_{T-1}) \leq c(s_{T-1}, s_T)$  admissible

Consistency implies admissibility

Suppose: $\forall s_t, s_{t+1} : h(s_t) \leq c(s_t, s_{t+1}) + h(s_{t+1})$

Then: $h(s_{T-1}) \leq c(s_{T-1}, s_T)$

$$h(s_{T-2}) \leq c(s_{T-2}, s_{T-1}) + h(s_{T-1})$$

Consistency implies admissibility

Suppose: $\forall s_t, s_{t+1} : h(s_t) \leq c(s_t, s_{t+1}) + h(s_{t+1})$

Then: $h(s_{T-1}) \leq c(s_{T-1}, s_T)$

$h(s_{T-2}) \leq c(s_{T-2}, s_{T-1}) + h(s_{T-1})$



admissible

Consistency implies admissibility

Suppose: $\forall s_t, s_{t+1} : h(s_t) \leq c(s_t, s_{t+1}) + h(s_{t+1})$

Then: $h(s_{T-1}) \leq c(s_{T-1}, s_T)$

$$h(s_{T-2}) \leq c(s_{T-2}, s_{T-1}) + h(s_{T-1})$$



admissible



admissible

Consistency implies admissibility

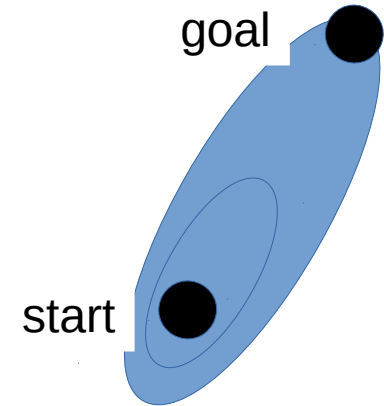
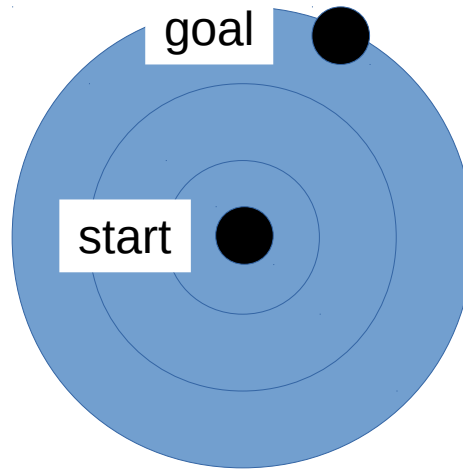
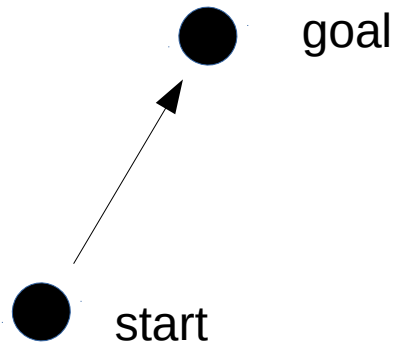
Suppose: $\forall s_t, s_{t+1} : h(s_t) \leq c(s_t, s_{t+1}) + h(s_{t+1})$

Then: $h(s_{T-1}) \leq c(s_{T-1}, s_T)$

$$h(s_{T-2}) \leq c(s_{T-2}, s_{T-1}) + h(s_{T-1})$$

...

A* vs UCS



Greedy



UCS



A*

Choosing a heuristic

The right heuristic is often problem-specific.

But it is very important to select a good heuristic!

Choosing a heuristic

Consider the 8-puzzle:

h_1 : number of misplaced tiles

h_2 : sum of manhattan distances
between each tile and its goal.

How much better is h_2 ?

	1	2
3	4	5
6	7	8

Choosing a heuristic

Consider the 8-puzzle:

h_1 : number of misplaced tiles

h_2 : sum of manhattan distances
between each tile and its goal.

	1	2
3	4	5
6	7	8

Average # states expanded on a random depth-24 puzzle:

$$A^*(h_1) = 39k$$

$$A^*(h_2) = 1.6k$$

$$IDS = 3.6M \quad (\text{by depth } 12)$$

Choosing a heuristic

Consider the 8-puzzle:

h_1 : number of misplaced tiles

h_2 : sum of manhattan distances
between each tile and its goal.

	1	2
3	4	5
6	7	8

So, getting the heuristic right can speed things
up by multiple orders of magnitude!

$IDS = 3.6M$ (by depth 12)

Choosing a heuristic

Consider the 8-puzzle:

h_1 : number of misplaced tiles

h_2 : sum of manhattan distances
between each tile and its goal.

	1	2
3	4	5
6	7	8

Why not use the actual cost to goal as a heuristic?

How to choose a heuristic?

Nobody has an answer that always works.

A couple of best-practices:

- solve a relaxed version of the problem
- solve a subproblem