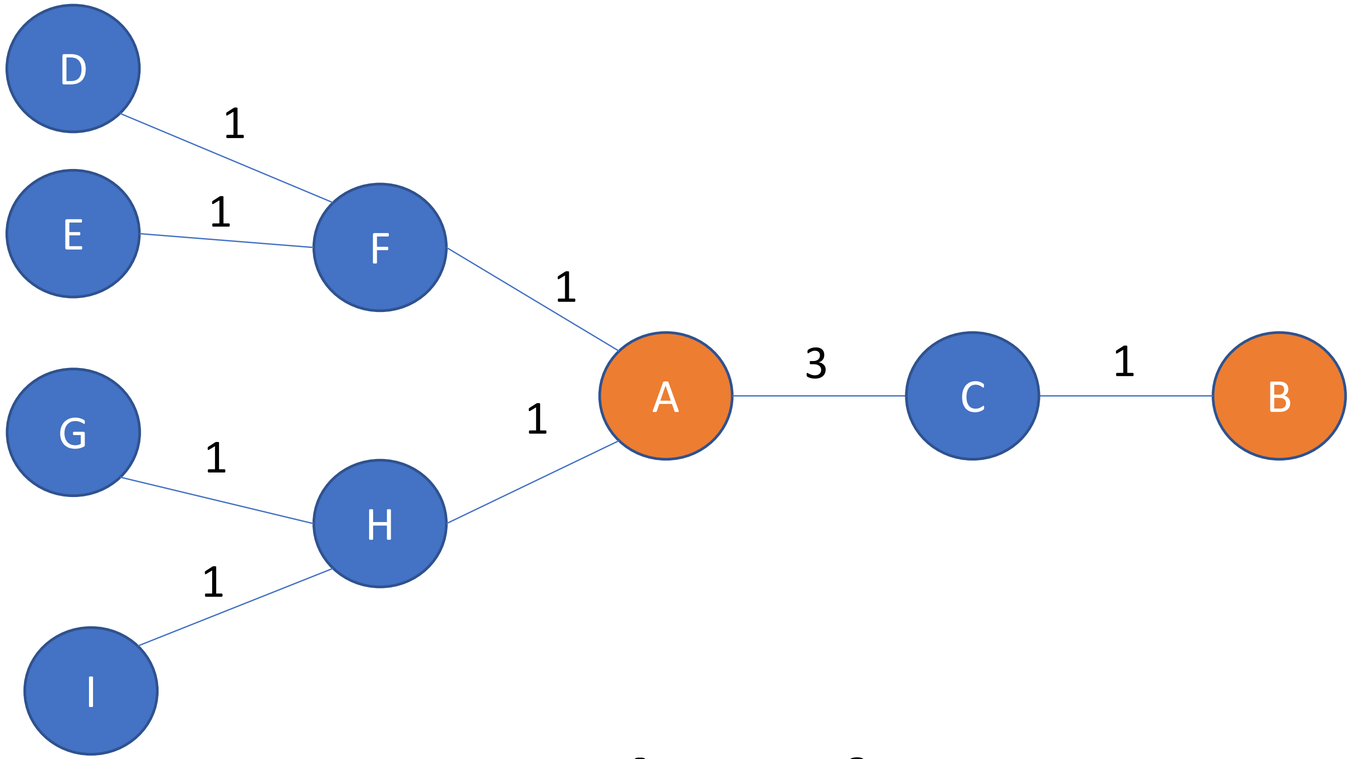
Artificial Intelligence

Lecture 04: Searching

Lecture Outline

- Best-first Search
  - Greedy Search
  - A* Search
  - Heuristics

When UCS goes wrong…



Goal: Get from A… to… B

Best-First Search

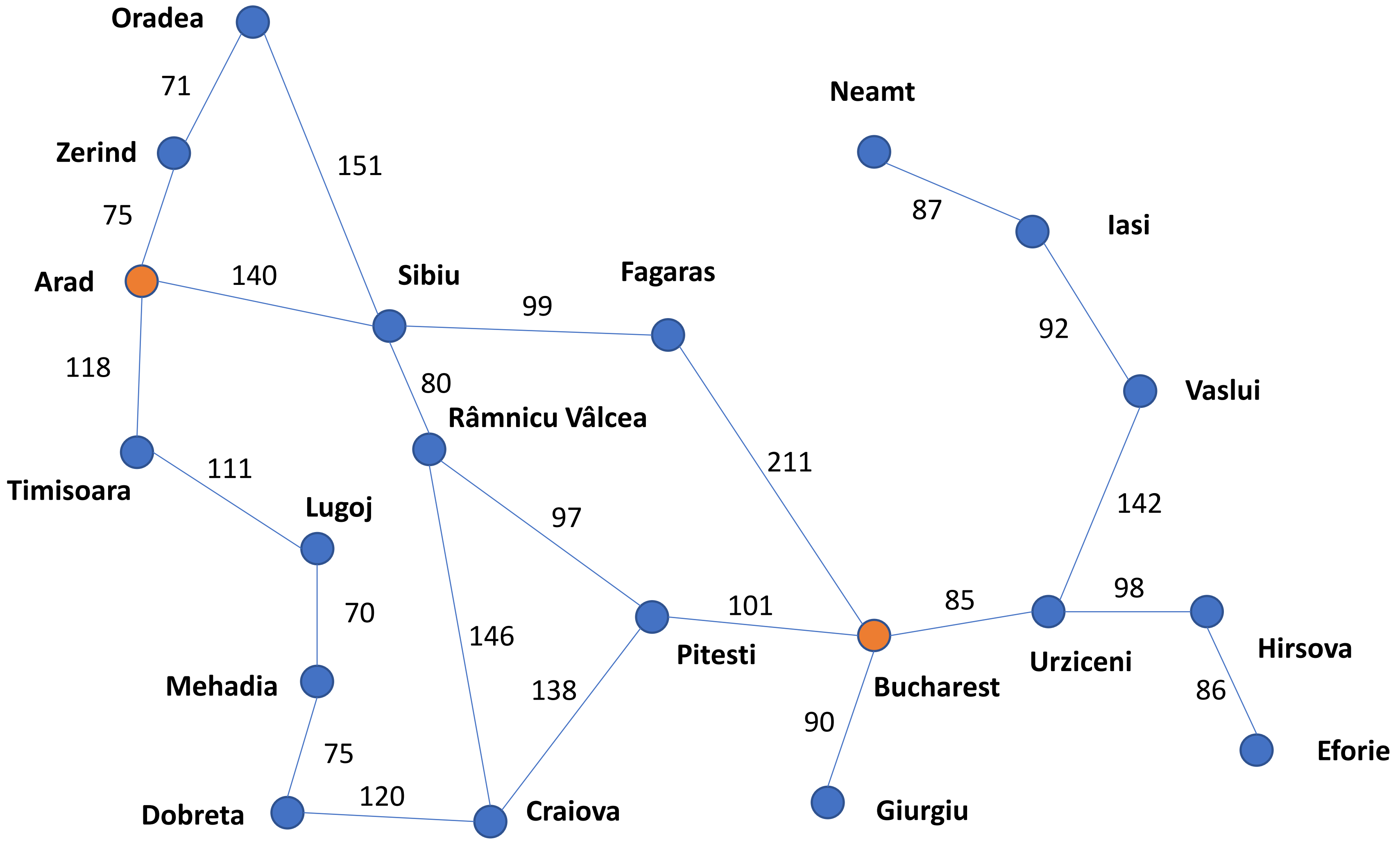
The idea:

- What if we had an ‘**evaluation function**’ which gave us an idea of the ‘**desirability**’ of each node
- We then expand the most ‘**desirable**’ node first

Implementation:

- The **fringe** is now sorted by **desirability**

The Problem!



How to get from…
Arad to **Bucharest**!
But…

Straightline to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Faragas	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

Greedy Search

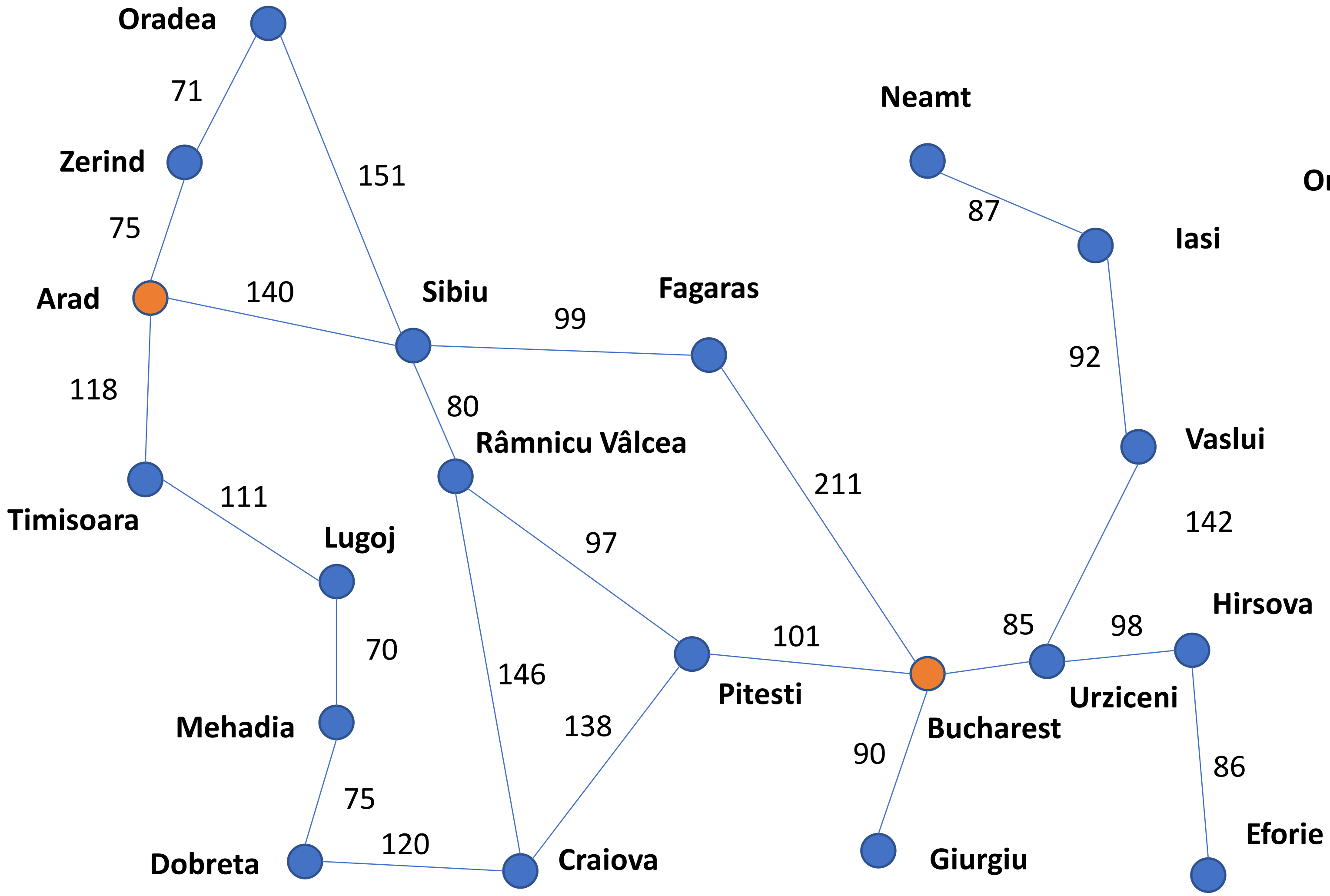
Evaluation function h(n) (h for heuristic)

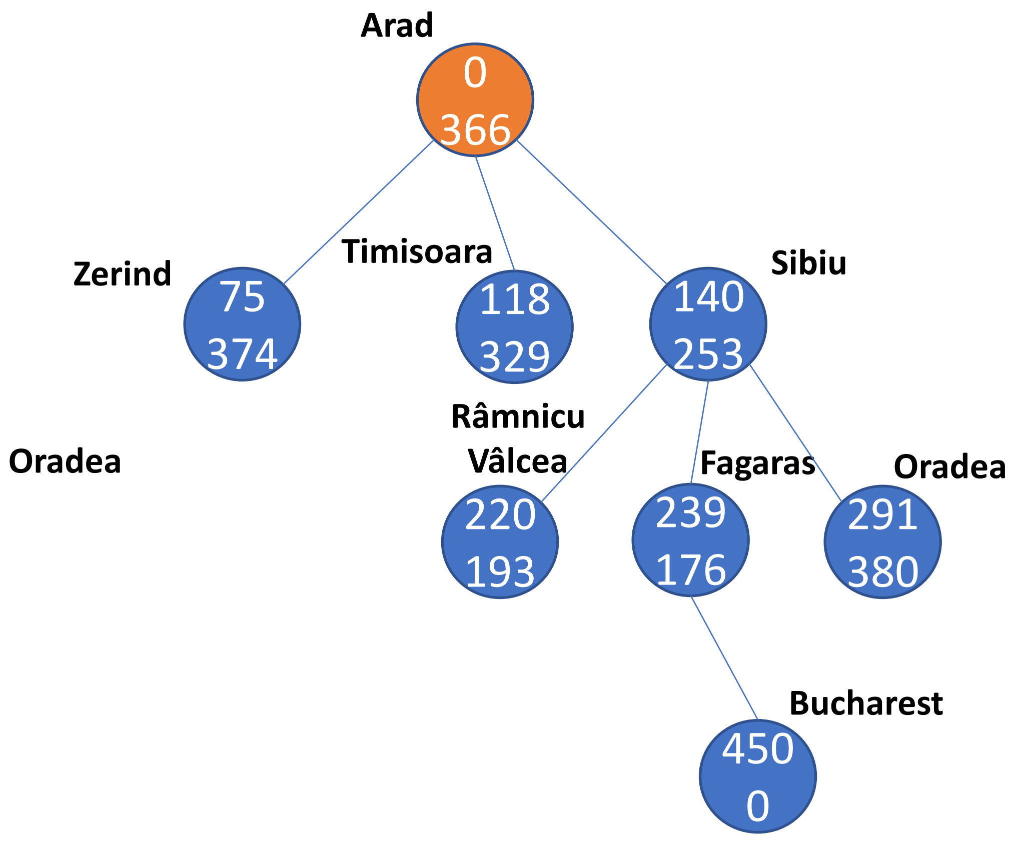
In our case, h(n) is the straight-line distance to Bucharest

Greedy search expands the node that **appears** to be the closest to the goal

Side note: this use of the term **Greedy** isn’t exactly the same as the usual use of a ‘greedy algorithm’

Greedy Search





Greedy Search Analysis

Complete?

- No (Yes if finite and state checks)

Time Complexity?

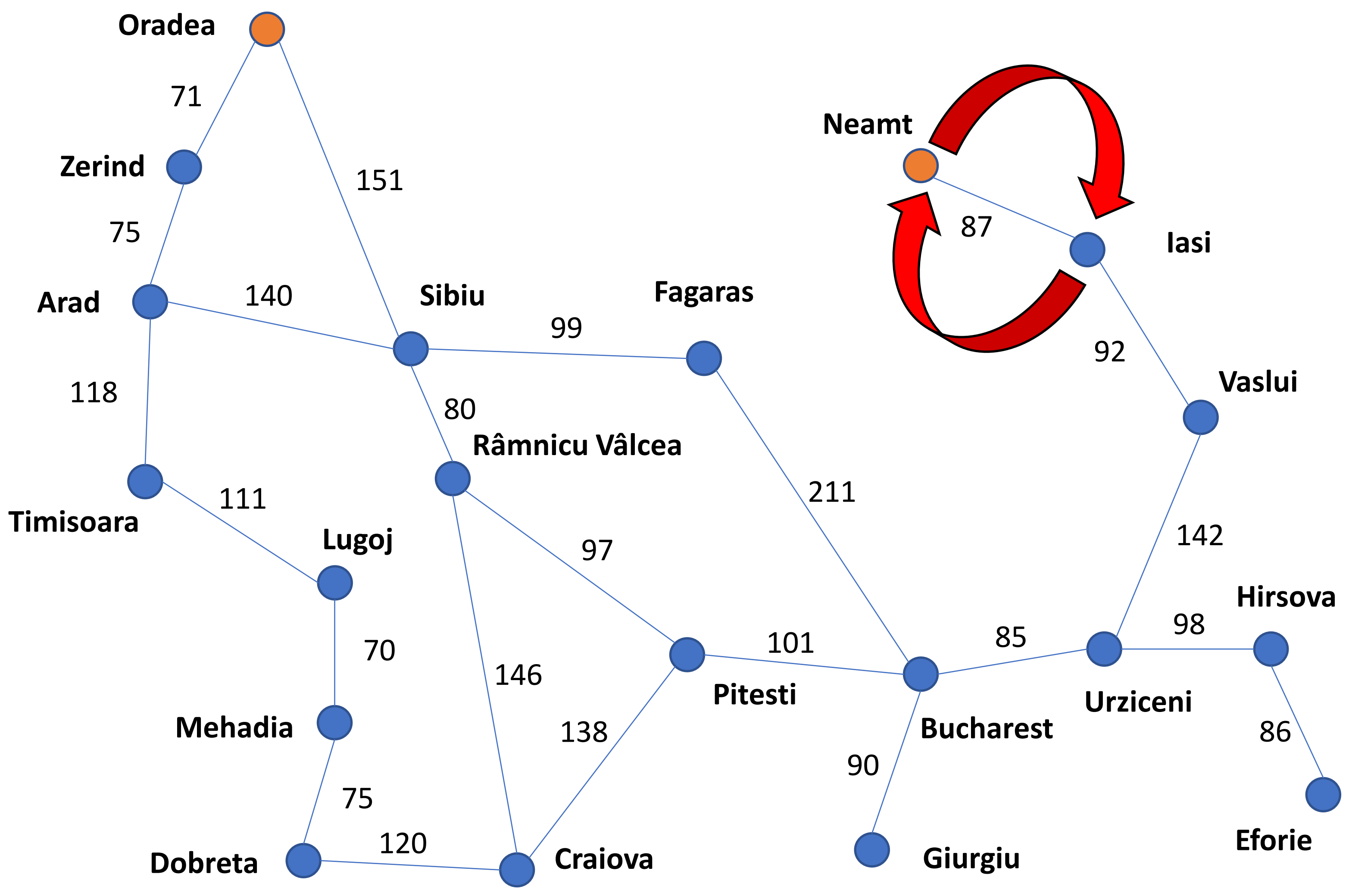
- $O(b^m)$
- Dependent on heuristic

Space Complexity?

- $O(b^m)$
- Keeps all nodes in memory

Optimal?

- No…



A* Search

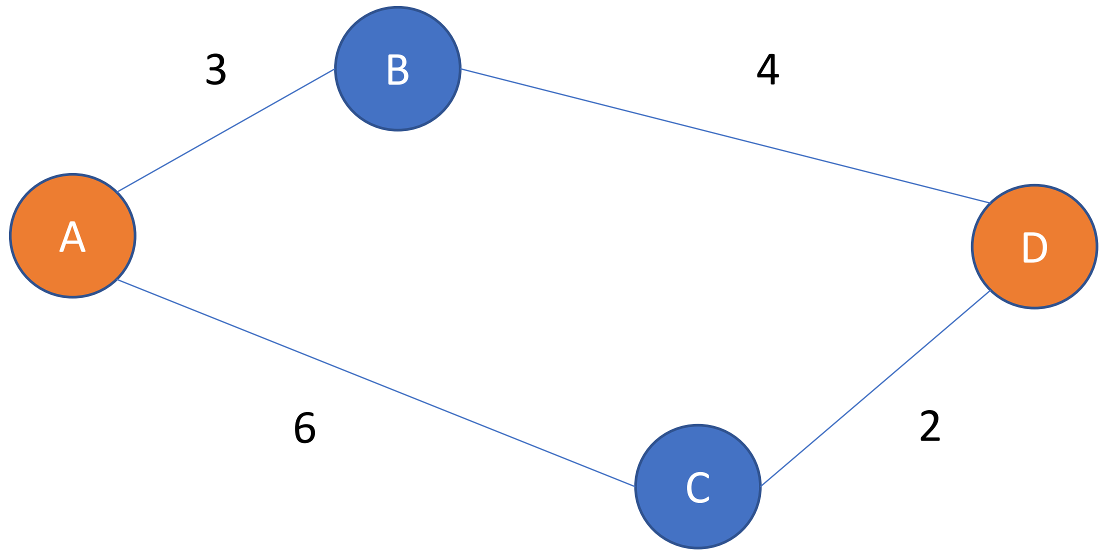
Goal is A => D

Greedy Search would expand C before B… because C is closer to D

But… , C has already cost me 6 plus an extra 2 is 8

Not at as good as B which is

3 plus an extra 4 is 7



A* Search

A subtle variation on greedy search

Two main components:

#1: What is the current path cost already?

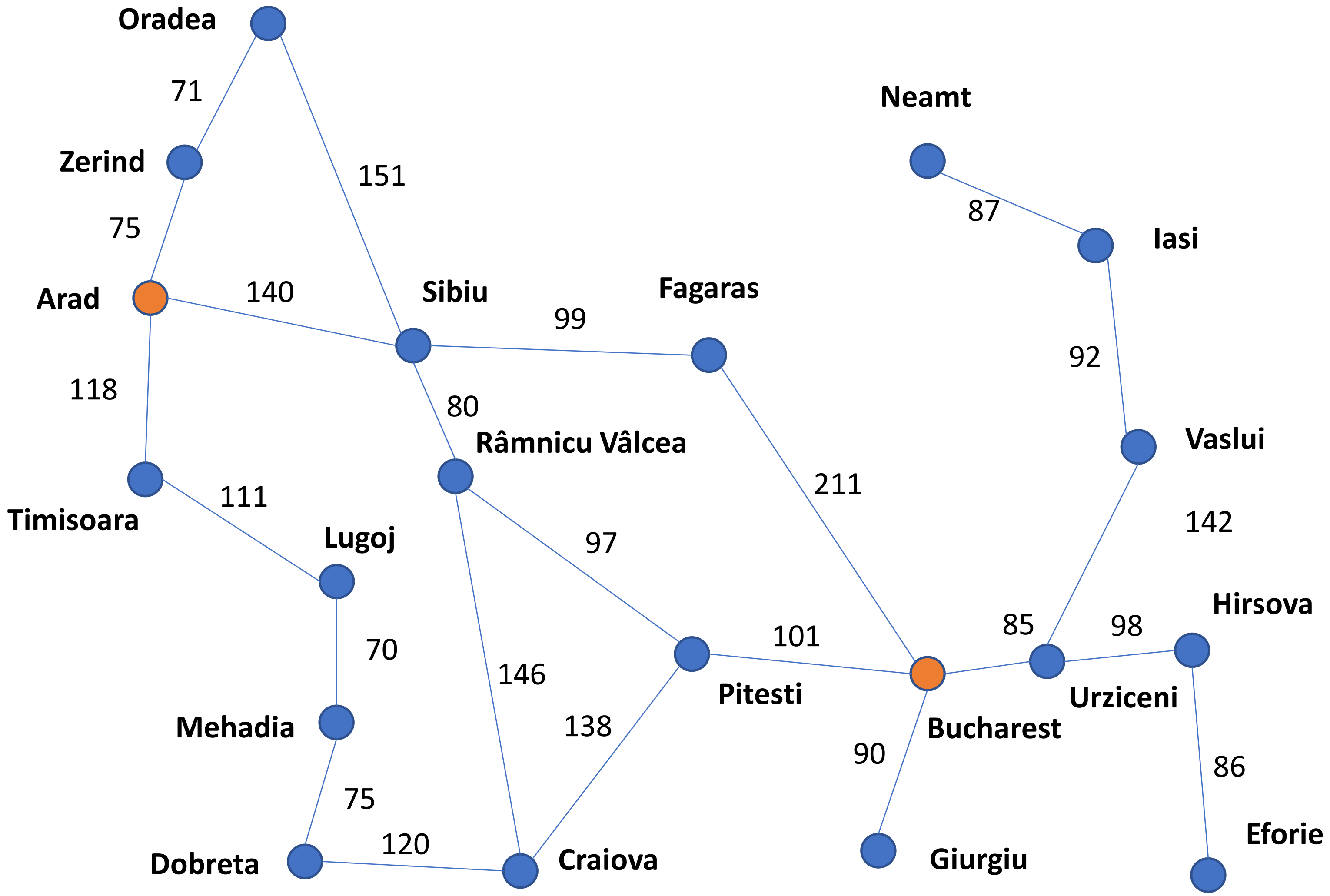
#2: What is the estimate of the remaining path?

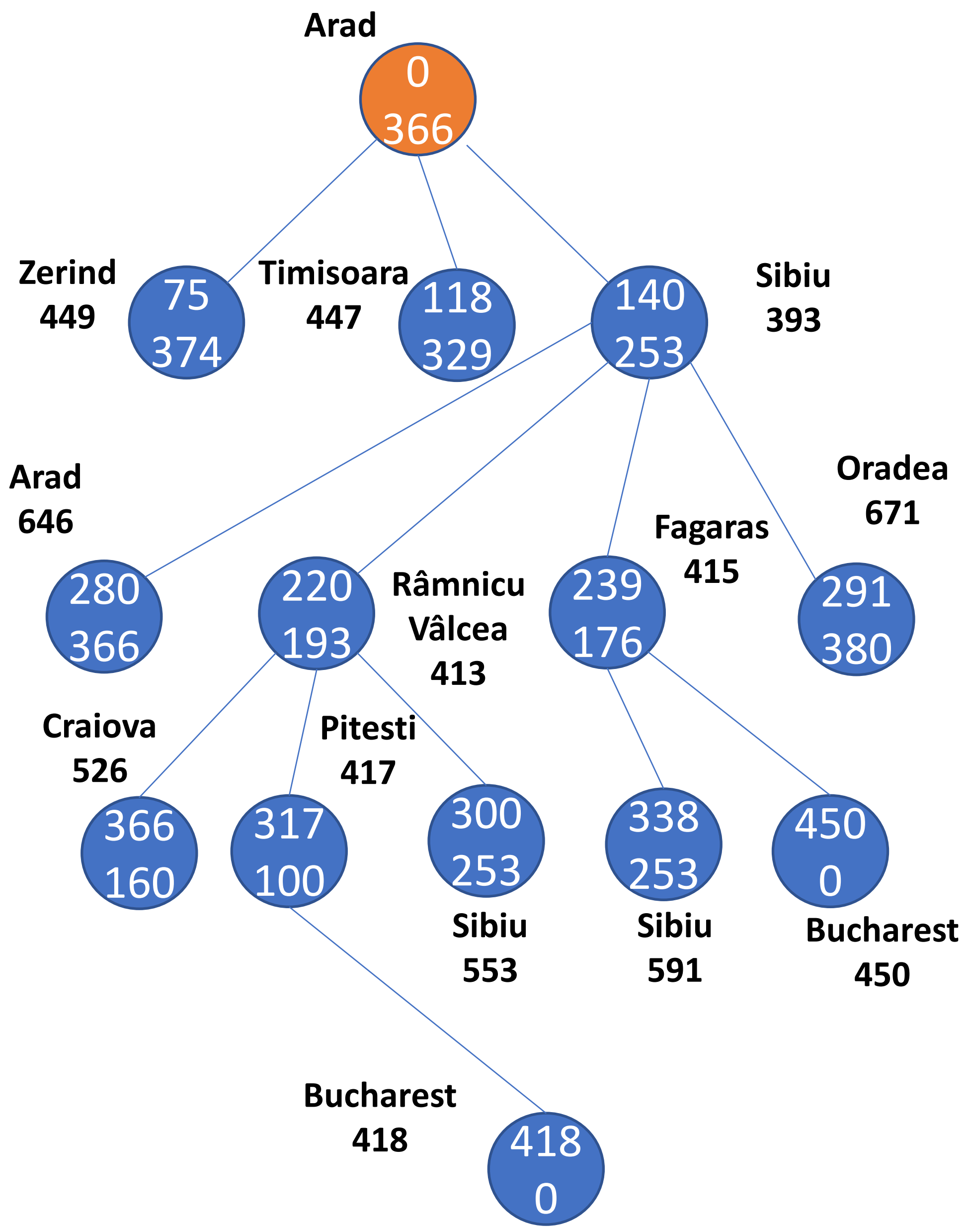
#1: $g(n)$, #2: $h(n)$

Evaluation Function

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

A* Search


Oradea
71
Zerind
151
75
Arad
140
Sibiu
99
Fagaras
Neamt
87
Iasi
92
118
80
Râmnicu Vâlcea
211
Vaslui
Timisoara
111
Lugoj
97
142
70
146
101
85
98
Hirsova
Pitesti
Urziceni
Mehadia
138
Bucharest
90
86
75
120
Dobreta
Craiova
Giurgiu
Eforie


Arad
0
366
Zerind
449
75
374
Timisoara
447
118
329
140
253
Sibiu
393
Arad
646
280
366
220
193
Râmnicu
Vâlcea
413
239
176
Fagaras
415
Oradea
671
291
380
Craiova
526
366
160
Pitesti
417
317
100
300
253
Sibiu
553
338
253
Sibiu
591
450
0
Bucharest
450
Bucharest
418
418
0

How does A* work?

A* only works if the heuristic function h(n) underestimates the remaining distance

i.e., $h(n) \leq h^*(n)$ (h*(n) is the true cost)

For example…

$h(G) = 0$ (G is the goal node)

For our road-map version, the straight-line distance is always shorter than the actual road distance (can be equal).

Is it really optimal?

If we have generated a bad path… G_2

Let us assume ‘n’ is on the real shortest path

$\Rightarrow f(G_2) = g(G_2)$ (as $h(G_2) = 0$)

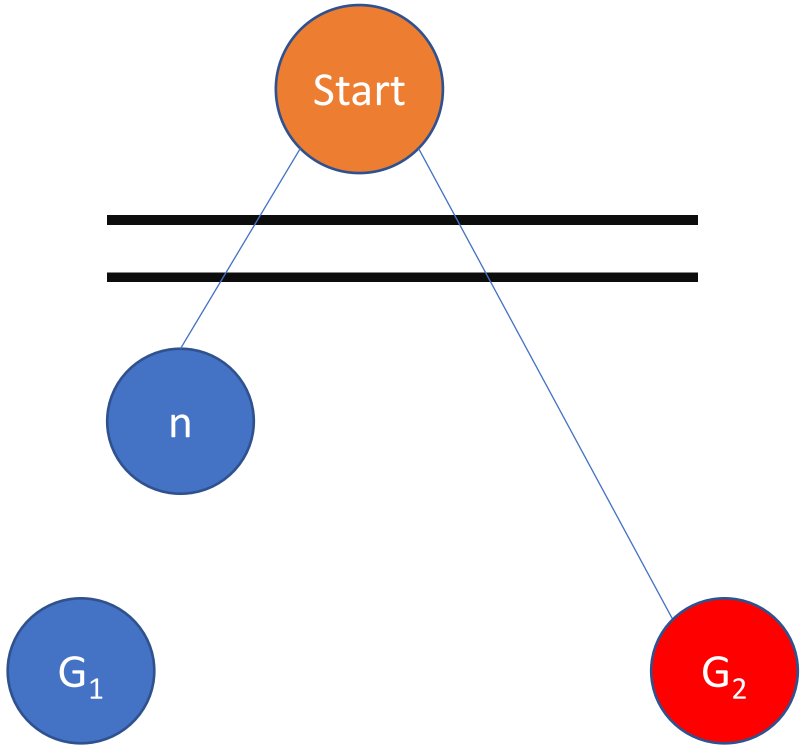
$\Rightarrow g(G_2) > g(G_1)$ (we know G_2 is not optimal)

& $g(G_1) \geq f(n)$

As h is admissible we know:

$g(G_2) > f(n)$

So we’ll always expand ‘n’ before G_2



Note:
Admissible means that the heuristic **never overestimates** the remaining distance to the target

Analysing A*

Complete?

- Yes (unless there are infinite ‘closer nodes’)

Time Complexity

- Exponential in [relative error in h * length of solution]

Space Complexity

- Keeps all nodes in memory

Optimal?

- Yes

Some further observations about H*

What happens when h(n) is always set to zero.

I.e., the estimate is always zero.

Is this admissible?

What does the algorithm look like?

What happens when h(n) = h*(n)

I.e., the heuristic is magically correct?

Is this admissible?

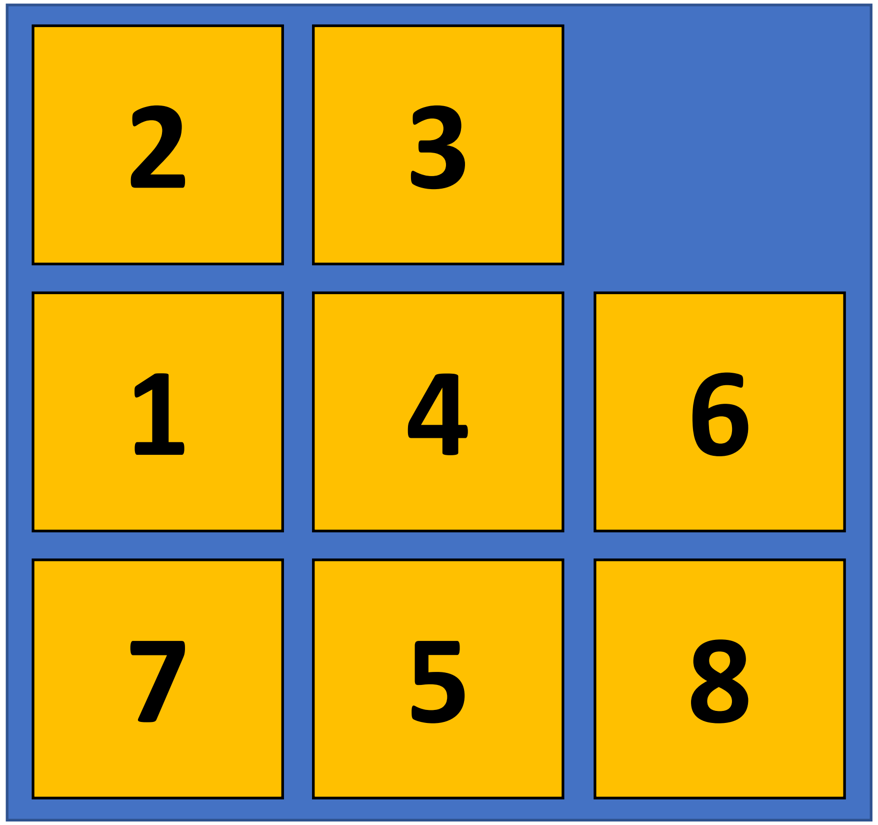
What does this algorithm look like?

Making an Admissible heuristic

$h_1(n)$ = number of misplaced tiles

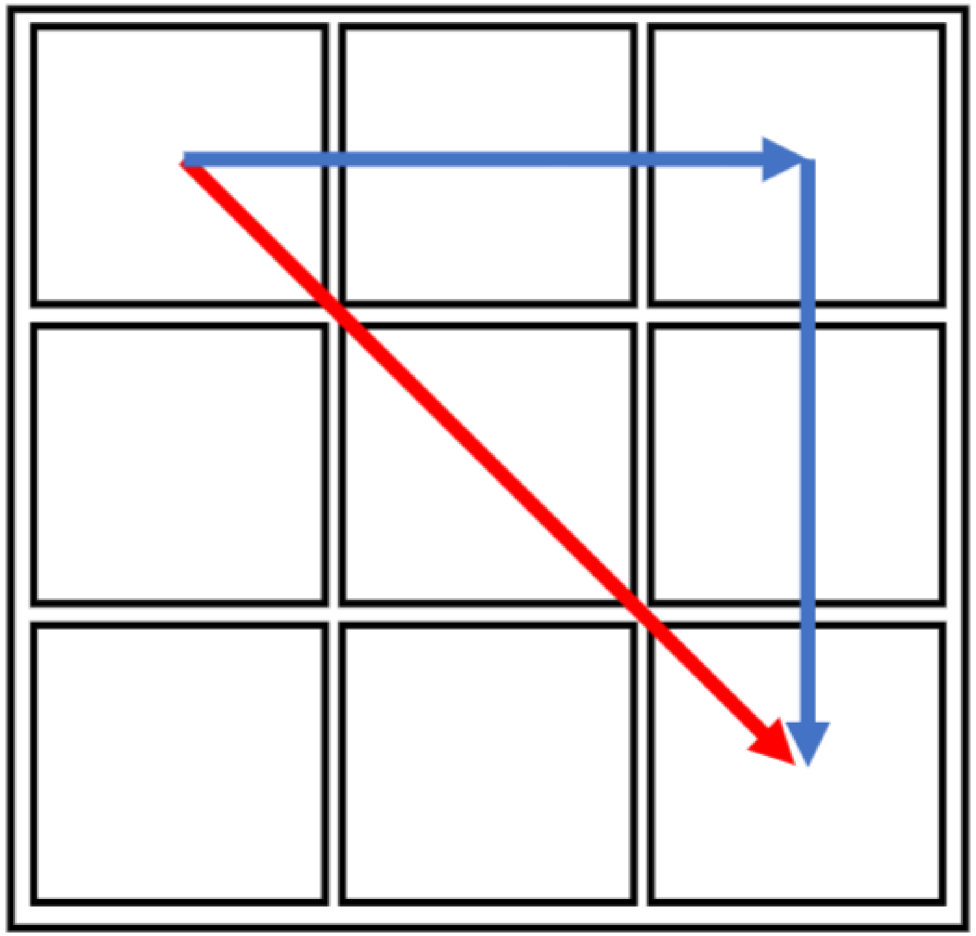
$h_2(n)$ = total **Manhattan** distance

- (total number of squares from desired location no diagonals)



Manhattan Distance

Euclidean Distance (straight line distance)
= sqrt(2) * 2



Manhattan Distance (horizontal + vertical distance)
= 2 + 2

Making an Admissible heuristic

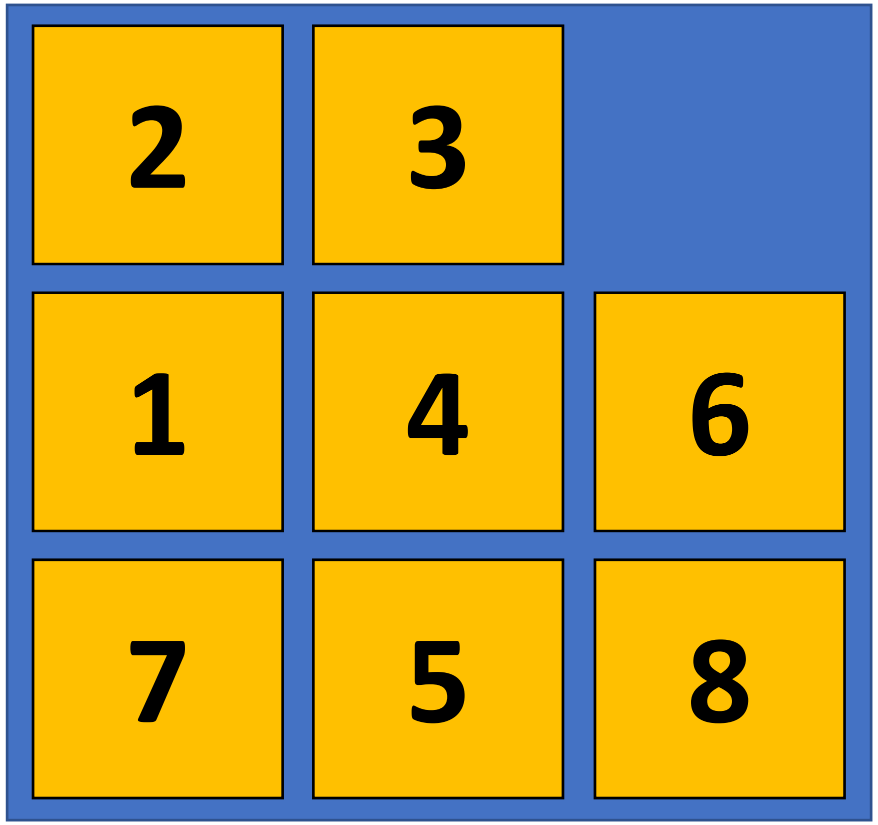
$h_1(n)$ = number of misplaced tiles

$h_2(n)$ = total **Manhattan** distance

- (total number of squares from desired location no diagonals)

$h_1(n) = 6$

$h_2(n) = 1 + 1 + 1 + 1 + 0 + 0 + 1 + 1$
$= 6$



Are these heuristics admissible?

$h_1(n)$ = number of misplaced tiles

$h_2(n)$ = total **Manhattan** distance

- (total number of squares from desired location no diagonals)

$h_1(n) = 6$

$h_2(n) = 1 + 1 + 1 + 1 + 0 + 0 + 1 + 1$

$= 6$

h_1 ?

h_2 ?

Better to ask, how many moves would be necessary as a minimum?

Which is better?

Intuitions about heuristics

What happens when your heuristic… is always zero?

What happens when your heuristic… is always ‘correct’ (i.e., magic)?

What happens when you don’t take existing path into your calculation?