Lecture 3a: Heuristic Functions

CSCI 360

Introduction to Artificial Intelligence

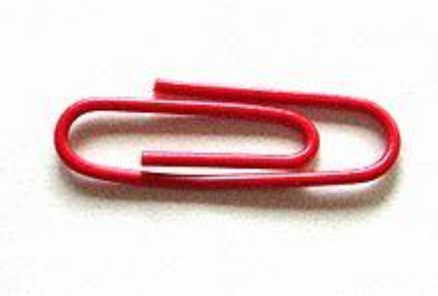
USC

Here is where we are…

Week	30000D	30282R	Topics	Chapters
1	1/7 1/9	1/8 1/10	Intelligent Agents Problem Solving and Search	[Ch 1.1-1.4 and 2.1-2.4] [Ch 3.1-3.3]
2	1/14 1/16	1/15 1/17	Uninformed Search Heuristic Search (A*)	[Ch 3.3-3.4] [Ch 3.5]
3	1/21 1/23	1/22 1/24	Heuristic Functions Local Search	[Ch 3.6] [Ch 4.1-4.2]
	1/25		Project 1 Out	
4	1/28 1/30	1/29 1/31	Adversarial Search Knowledge Based Agents	[Ch 5.1-5.3] [Ch 7.1-7.3]
5	2/4 2/6	2/5 2/7	Propositional Logic Inference First-Order Logic	[Ch 7.4-7.5] [Ch 8.1-8.4]
	2/8 2/8		Project 1 Due Homework 1 Out	
6	2/11 2/13	2/12 2/14	Rule-Based Systems Search-Based Planning	[Ch 9.3-9.4] [Ch 10.1-10.3]
	2/15		Homework 1 Due	
7	2/18 2/20	2/19 2/21	SAT-Based Planning Knowledge Representation	[Ch 10.4] [Ch 12.1-12.5]
8	2/25 2/27	2/26 2/28	Midterm Review **Midterm Exam**	

“One Red Paperclip”

It may be hard to estimate the value of...



$\neq$

“One Red Paperclip”– Real transactions made by Kyle MacDonald

- 7/14/2005, he traded the *paperclip* for a **fish-shaped pen**.
- 7/14/2005, he traded the **pen** for a hand-sculpted **doorknob**.
- 7/25/2005, he traded the **doorknob** for a Coleman **camp stove**.
- 9/24/2005, he traded the **camp stove** for a Honda **generator**.
- 11/16/2005, he traded the **generator** for an "**instant party**": an empty keg, an IOU for filling the keg with any beer, and a neon Budweiser sign.
- 12/8/2005, he traded the "**instant party**" to for a Ski-Doo **snowmobile**.
- 12/15/05, he traded the **snowmobile** for a **two-person trip to Yahk**, Canada.
- 1/7/2006, he traded the “**second spot” on the Yahk trip** for a box **truck**.
- 2/22/2006, he traded the **truck** for a **recording contract** with Metalworks.
- 4/11/2006, he traded the **contract** for a **year's rent** in Phoenix, Arizona.
- 4/26/2006, he traded the **year's rent** for **one afternoon with Alice Cooper**.
- 5/26/2006, he traded the **afternoon with Cooper** for a motorized **snow globe**.
- 6/2/2006, he traded the **snow globe** for a **role in the film** Donna on Demand.
- 7/5/2006, he traded the **role** for a *two-story farmhouse* in Kipling, Saskatchewan.

https://en.wikipedia.org/wiki/One_red_paperclip

“One Red Paperclip”

The world's largest paperclip, installed at Bell Park in Kipling in 2007

What have we learned so far?

- What is AI?
- Problem-solving agent
- Uninformed search
- Informed search (A*)

Recap: Cost Function *f(n)*

- Estimated cost of best path from initial state to goal state

total cost

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

– The **actual path cost** from the initial state to the state at node n

The **estimated cost** from the state at node n to a goal state

Recap: Greedy Best-first Search

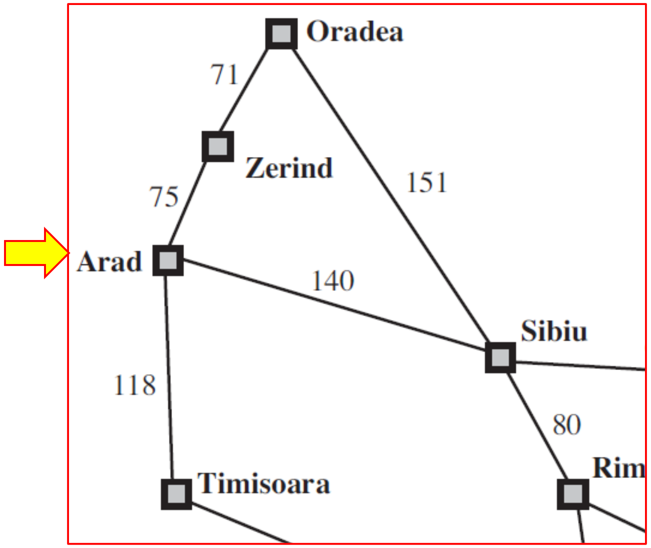
- **Intuition:** Expands node that "appears" to be **closest to the goal** since it's likely to lead a solution quickly
  - i.e., f(n) = h(n)

- **Example:** In Map of Romania, use "*straight-line distance to Bucharest*" as the heuristic function ***h(n)***

Recap: Greedy Best-first Search (example run)

- Starting at "**Arad**", there are three actions
  - *Go(Zerind)*
  - *Go(Sibiu)*
  - *Go(Timisoara)*

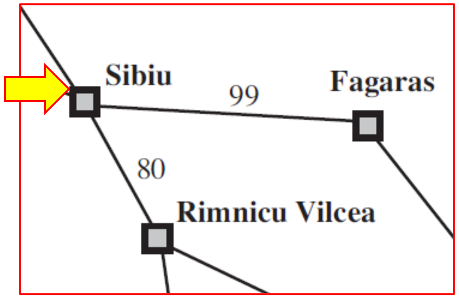
- After that, which of the new nodes should be chosen for expansion?



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

Recap: Greedy Best-first Search (example run)

- Starting at "**Sibiu**", there are two actions
  - *Go(Fagaras)*
  - *Go(Rimnicu Vilcea)*



- After that, which of these two new nodes should be expanded next?

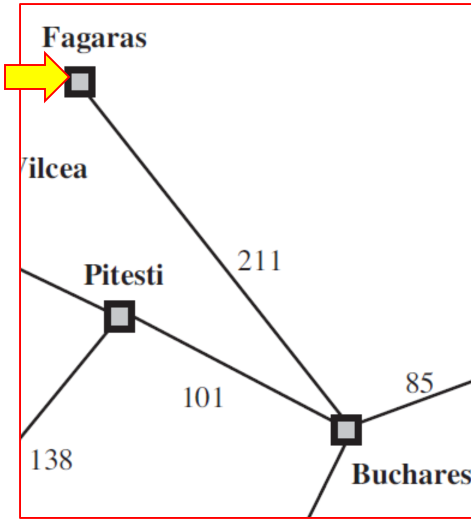
Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374

Recap: Greedy Best-first Search (example run)

- Starting at "**Fagaras**", there is only one action
  - *Go(Bucharest)*

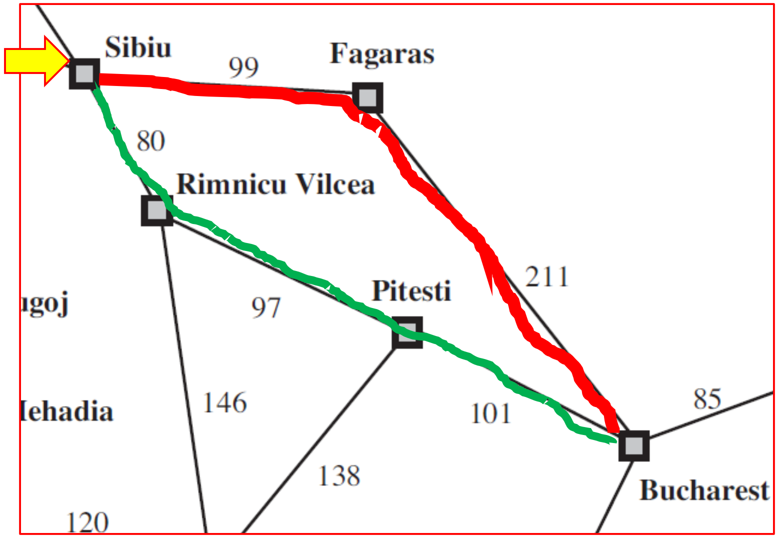
- It's a goal state!

- But, is this the **optimal** solution?



Recap: Greedy Best-first Search (optimality)

- Starting at "**Fagaras**", there is only one action
  - *Go(Bucharest)*
- It's a goal state!
- No, it is **not optimal**

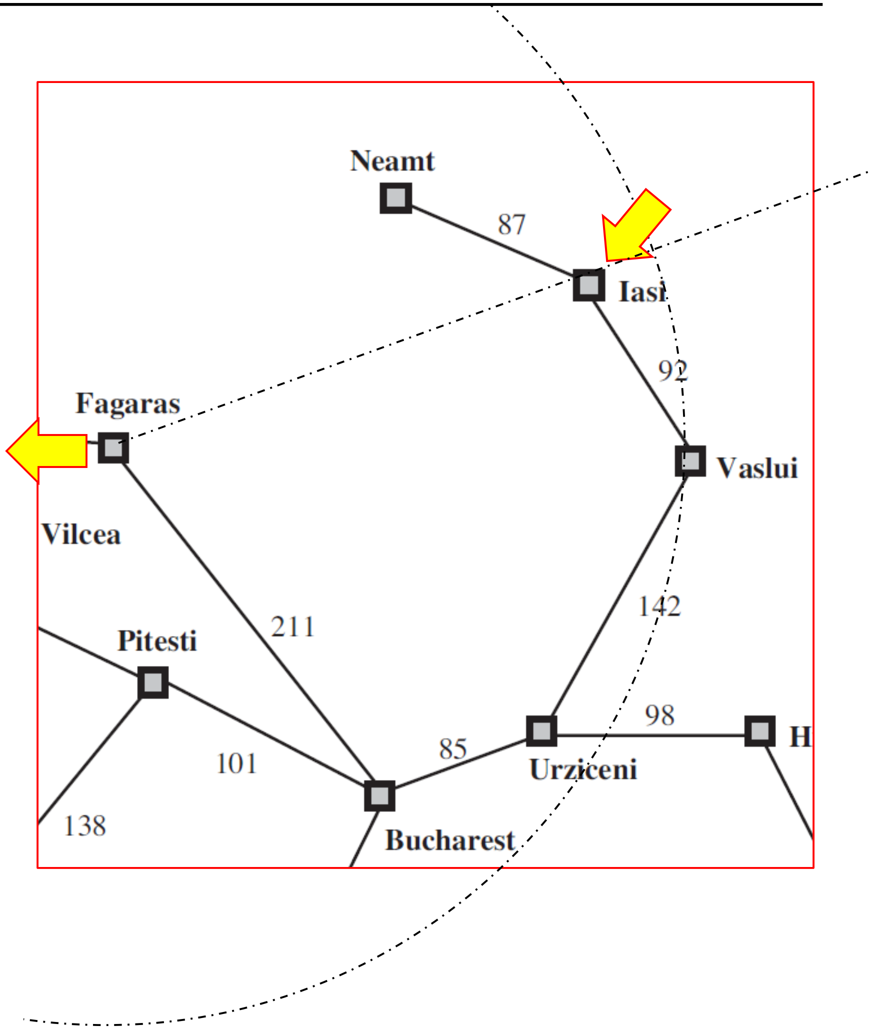


278 kilometers vs. 310 kilometers

Recap: Greedy Best-first Search (completeness)

- Greedy Tree-Search is not complete even in a finite state space

- **Example**:
  - From "**Iasi**" to "**Fagaras**"
    - *Go(Neamt)*
    - *Go(Vaslui)*
  - Which node to expand next?
    - ***Neamt***, since its "straight-line distance" to *Fagaras* is shorter
    - Will never find the solution due to infinite loop "Neamt – Iasi"

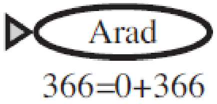


Recap: Use A* search, instead – minimizing total cost

- Most widely known form of “best-first search”
  - **Goal:** find the cheapest solution
  - **At each step**: expand node with lowest value of the total estimated solution cost: $f(n) = g(n)+h(n)$

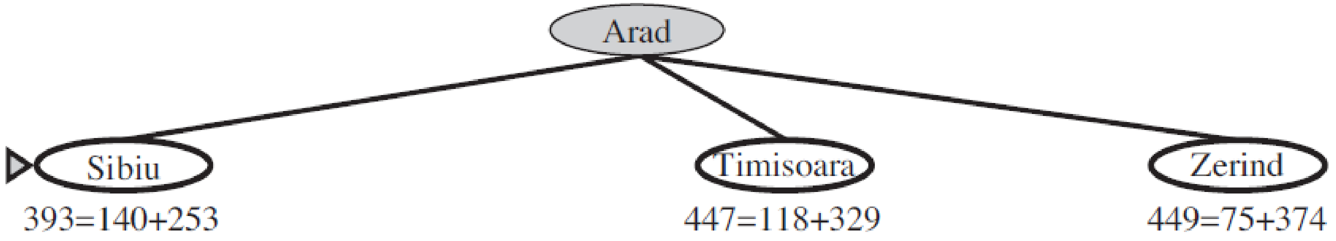
Recap: A* search (example run)

- Total estimated solution cost: $f(n) = g(n) + h(n)$



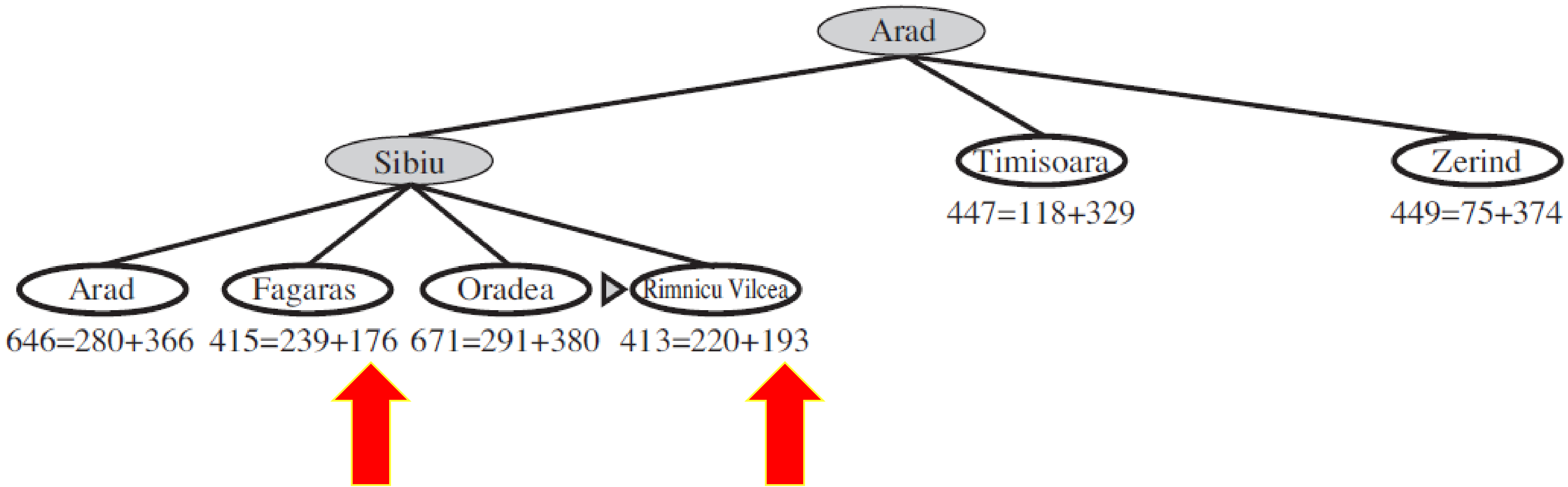
Recap: A* search (example run)

- Total estimated solution cost: *f(n) = g(n) + h(n)*



Recap: A* search (example run)

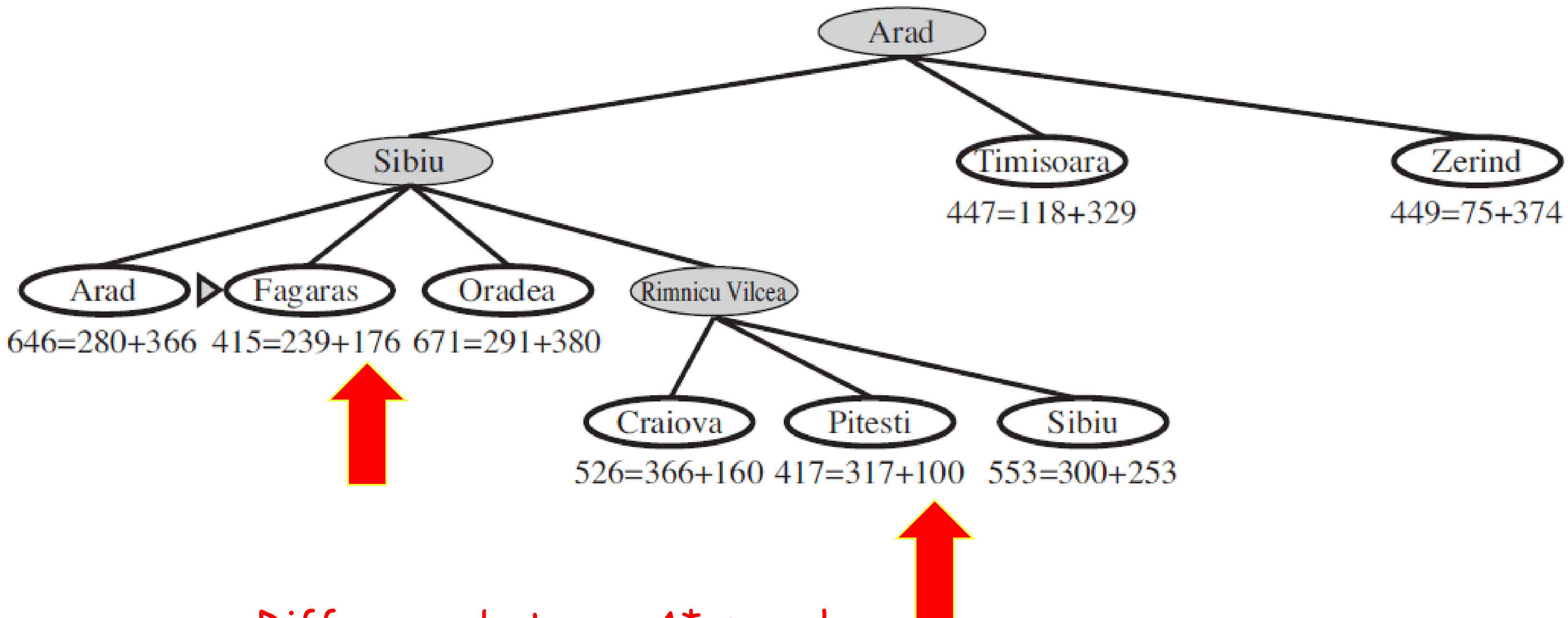
- Total estimated solution cost: *f(n) = g(n) + h(n)*



Difference between A* search
and Greedy best-first search

Recap: A* search (example run)

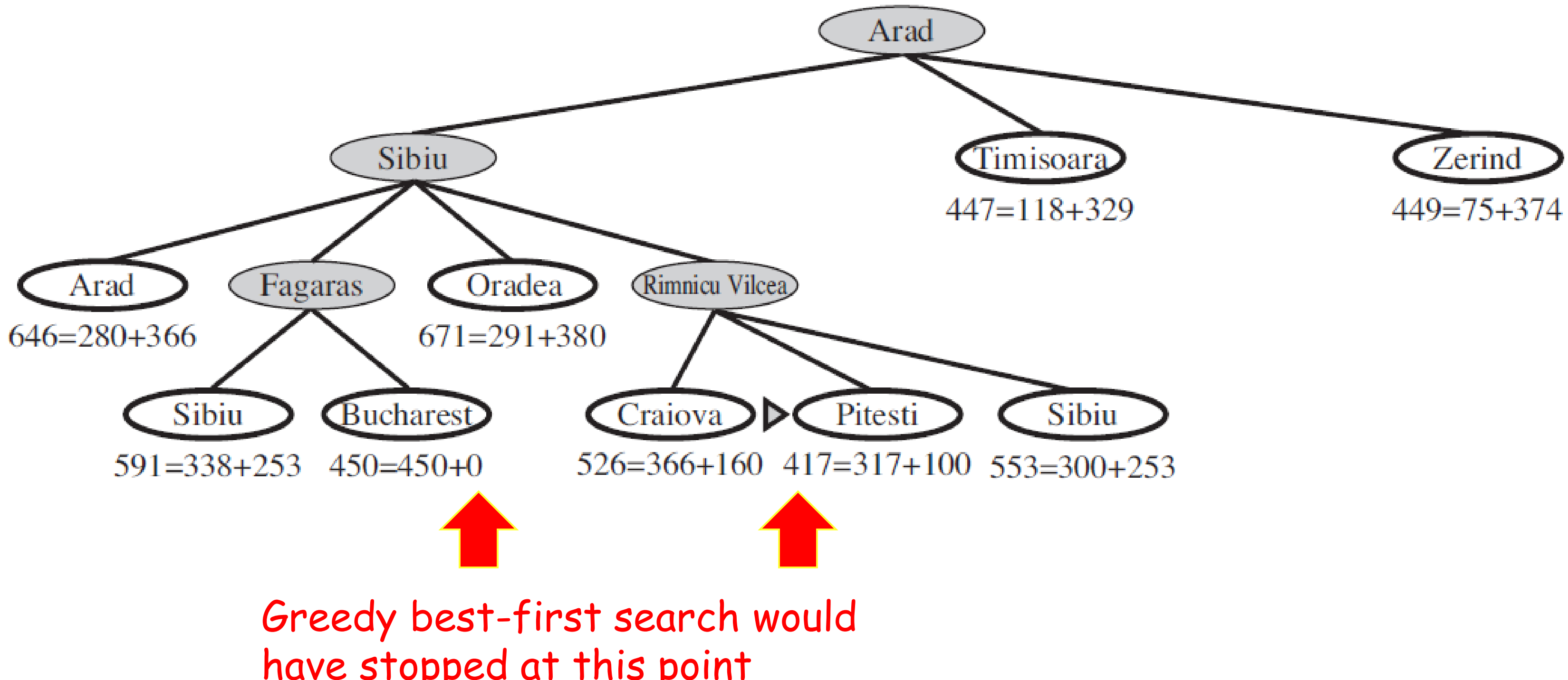
- Total estimated solution cost: $f(n) = g(n) + h(n)$



Difference between A* search
and Greedy best-first search

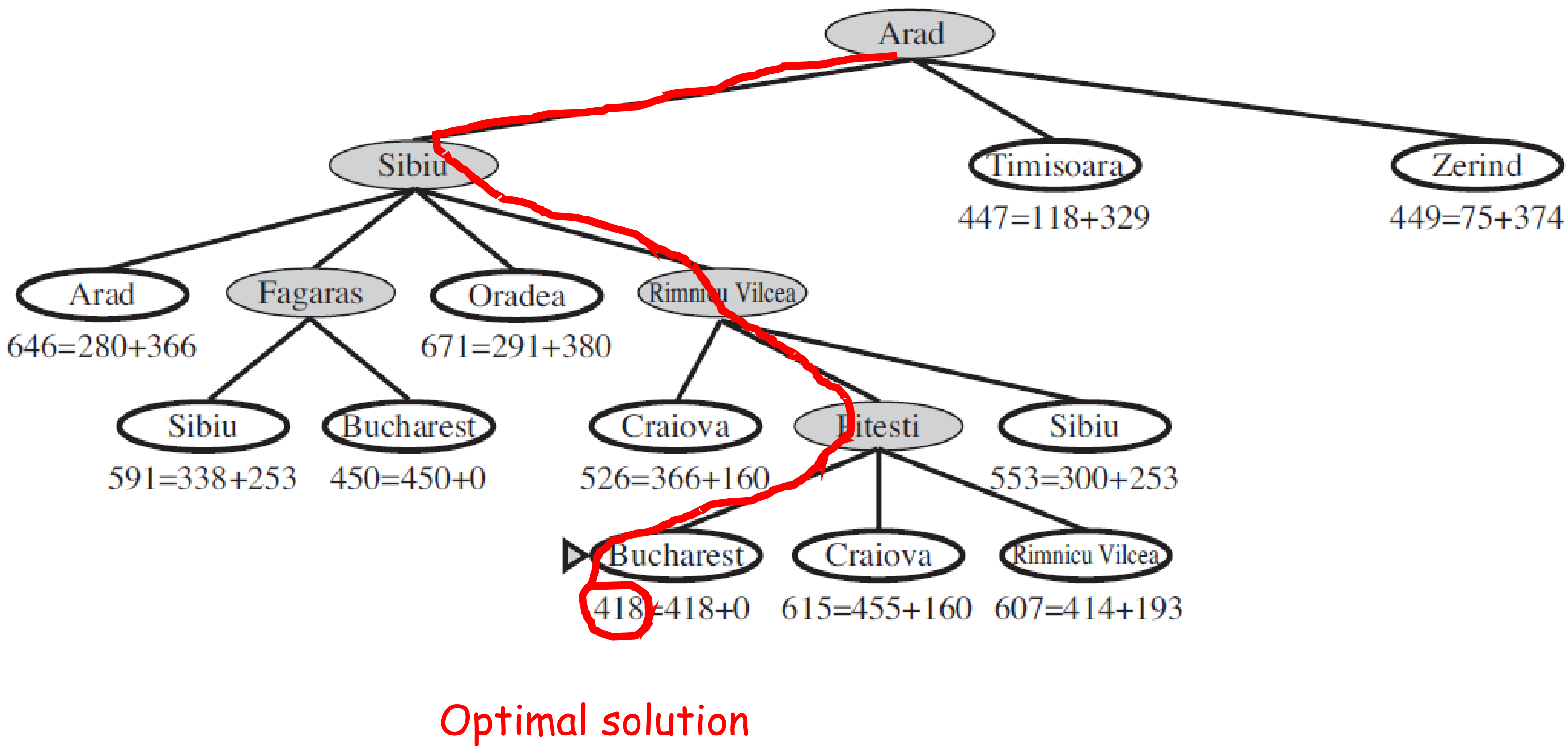
Recap: A* search (example run)

- Total estimated solution cost: $f(n) = g(n) + h(n)$



Recap: A* search (example run)

- Total estimated solution cost: *f(n) = g(n) + h(n)*



Recap: Condition for optimality (A*)

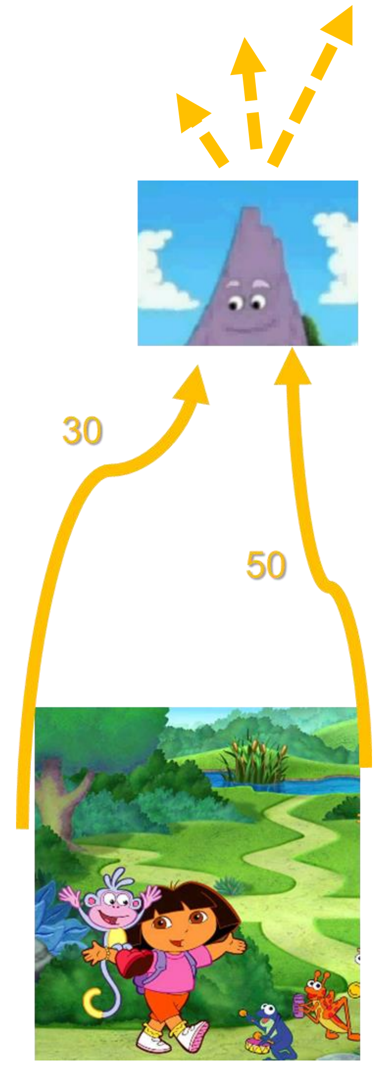
- Heuristic function ***h(n)*** must be **admissible**
- An admissible heuristic *never overestimates* the true cost to reach the goal

Feel free to under-estimate!

Under-estimation is fine (and is encouraged)

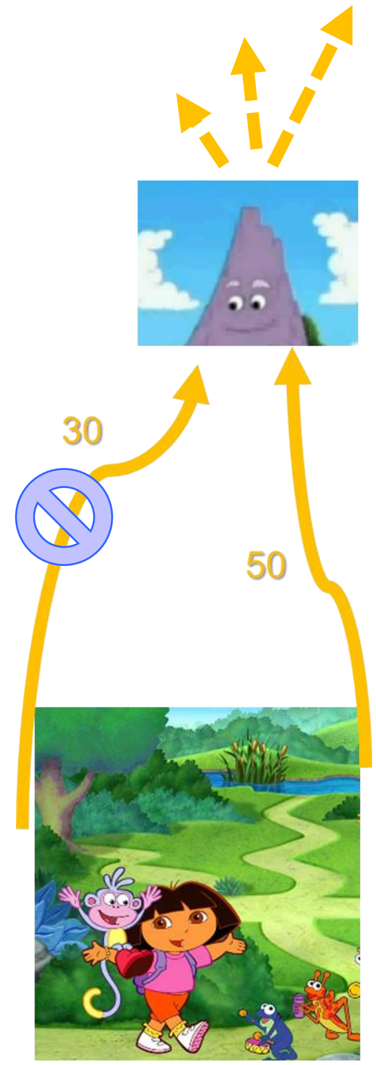
- If we underestimate the distance
  - **30** ➔ 10
  - **50** ➔ 5

- A* search is still “optimal”
  - Step 1. Dora and Boots travel though the path labeled “5” (50) and reach the “tall mountain”
  - Step 2. Since the actual cost is 50, they explore the other path labeled “10” (30)
  - Step 3. Only after the optimal path is found, they move forward to explore from the “tall mountain”



Over-estimation is problematic

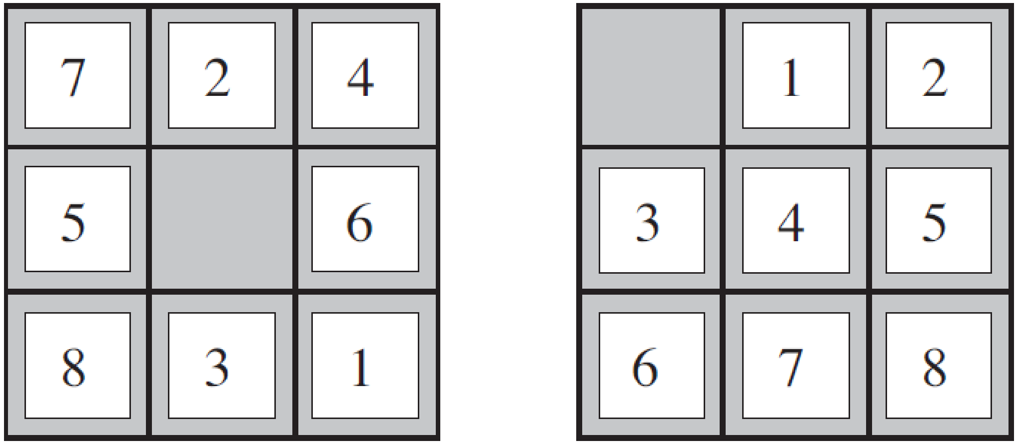
- If we over-estimate the distance
  - **30** → 100
  - **50** → 55
- A* search loses the "optimality"
  - Step 1. Dora and Boots travel through the path labeled "55" (or 50) and reach the tall mountain
  - Step 2. Since the other path is labeled "100" (which is larger than 50), they will move forward from the tall mountain
  - The optimal path labeled "100" (30) is never explored, due to the over-estimated cost



Outline for Today

- What is AI?
- Problem-solving agent
- Uninformed search
- **Informed search (A*)**
- **Heuristic functions**
  - How to design “admissible” heuristics

A* Search for 8-Puzzle



Start State   Goal State

A* Search for 8-Puzzle

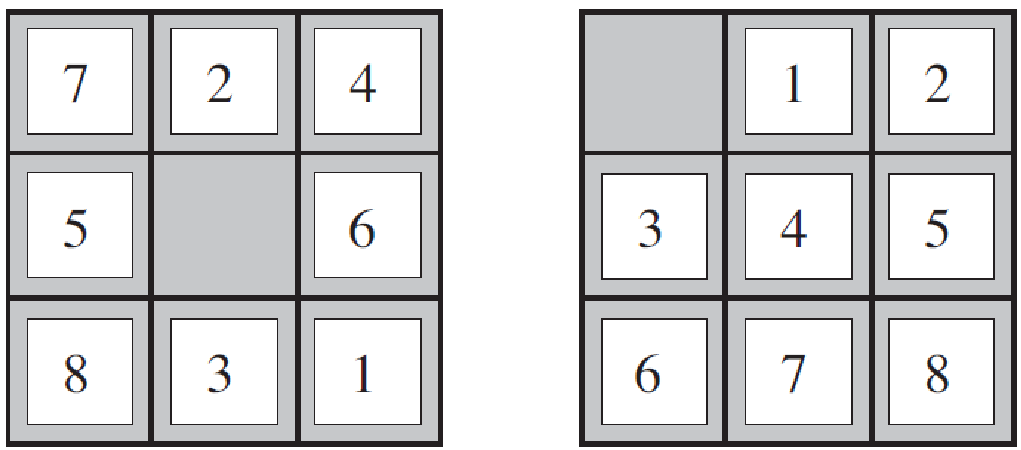
- Branching factor < 3
- Average solution cost is 22 steps
  - Exhaustive tree search to depth 22 ➔ 3^{22} states
  - Graph search ➔ 181,440 distinct states

Heuristic functions for 8-puzzle

- Two candidates
  - h1(n) = number of misplaced tiles
  - h2(n) = sum of the distances of the titles from their goal positions
    - Manhattan distance: The sum of the horizontal and vertical distances

h1 = ??

h2 = ??



Start State Goal State

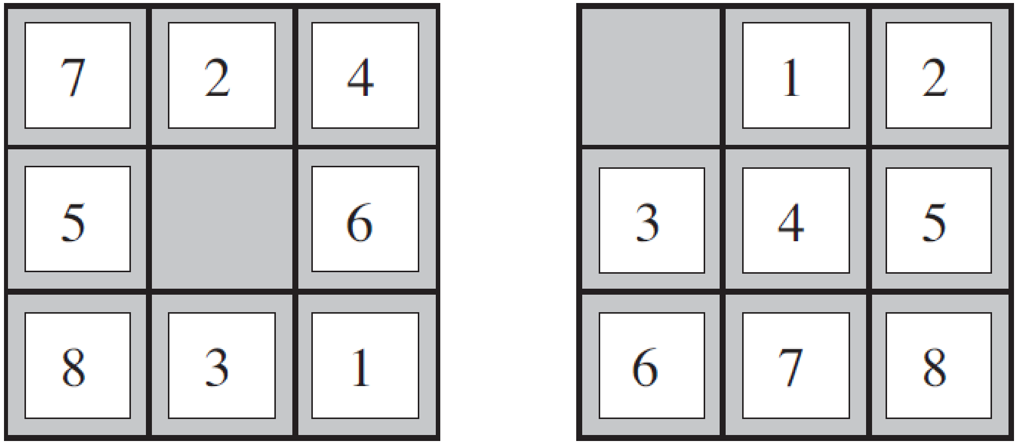
Heuristic functions for 8-puzzle

- Two candidates
  - h1(n) = number of misplaced tiles
  - h2(n) = sum of the distances of the titles from their goal positions
    - Manhattan distance: The sum of the horizontal and vertical distances

h1 = 8

h2 = 18

How do you know which one is better?



Start State

Goal State

Experimental results: A* with h1 and h2

- Solving 1200 random 8-puzzle problems

	Search Cost (nodes generated)		
d	IDS	$\mathrm{A}^*(h_1)$	$\mathrm{A}^*(h_2)$
2	10	6	6
4	112	13	12
6	680	20	18
8	6384	39	25
10	47127	93	39
12	3644035	227	73
14	–	539	113
16	–	1301	211
18	–	3056	363
20	–	7276	676
22	–	18094	1219
24	–	39135	1641

The effect of heuristic accuracy

- Effective branching factor b*
  - Assume the heuristic is perfect, A* would explore a total of (*N*) nodes and the solution depth is *d*

N and *d* are from measurement

$$N+1 = 1+b^*+(b^*)^2 + \ldots + (b^*)^d$$

Q: Let (N=52) and (d=5), what would be the value of (b*)?

Answer: b*=1.92

Experimental results: A* with h1 and h2

- Solving 1200 random 8-puzzle problems

	Search Cost (nodes generated)			Effective Branching Factor		
d	IDS	$A^*(h_1)$	$A^*(h_2)$	IDS	$A^*(h_1)$	$A^*(h_2)$
2	10	6	6	2.45	1.79	1.79
4	112	13	12	2.87	1.48	1.45
6	680	20	18	2.73	1.34	1.30
8	6384	39	25	2.80	1.33	1.24
10	47127	93	39	2.79	1.38	1.22
12	3644035	227	73	2.78	1.42	1.24
14	–	539	113	–	1.44	1.23
16	–	1301	211	–	1.45	1.25
18	–	3056	363	–	1.46	1.26
20	–	7276	676	–	1.47	1.27
22	–	18094	1219	–	1.48	1.28
24	–	39135	1641	–	1.48	1.26

d

N

b*

Why Heuristic h2(n) is better than h1(n)?

- By their definitions, for any node ***n***, we have ***h1(n) ≤ h2(n)***
  - We say that "h2 dominates h1"

- **Domination** translates into **efficiency:** A* using h2 will never expand more nodes than A* using h1

$$f(n) < C^*$$

$$g(n) + h(n) < C^*$$

$$g(n) < C^* - h(n)$$

- Two candidates
  - h1(n) = number of misplaced tiles
  - h2(n) = sum of the distances of the titles from their goal positions

Why Heuristic h2(n) is better than h1(n)?

- By their definitions, for any node ***n***, we have ***h1(n) ≤ h2(n)***
  - We say that "h2 dominates h1"

- **Domination translates into efficiency**: A* using h2 will never expand more nodes than A* using h1

$$f(n) < C^*$$

$$g(n) + h(n) < C^*$$

$$\boxed{g(n) < C^* - h(n)}$$

larger h(n) ➔ smaller g(n)

When h(n)=0, as in the Uniform-cost Search, g(n) is only bounded by C* (i.e., no guidance from goal at all)

How to generate heuristic functions?

- **Admissible** heuristics can be derived from the exact solution cost of a "**relaxed**" version of the problem

- By "**relaxed**" we mean it has an "**over-approximation**" of the transition model of the original problem
  - More edges between states, to provide short cuts
  - To get "under-approximation" of the cost

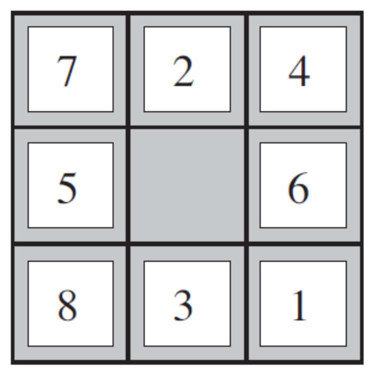
(1) Relaxed problems for 8-puzzle

- Original transition model

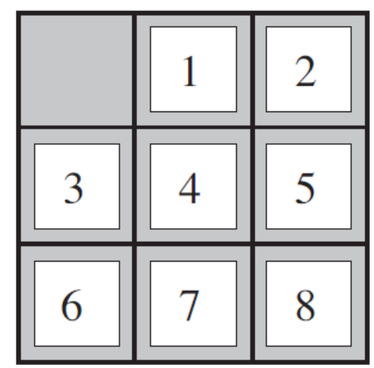
A tile can move from square A to square B *if*

A is horizontally or vertically adjacent to B, and B is blank

- Relaxed transition model
  - (a) A tile can move from square A to square B *if A is adjacent to B*
  - (b) A tile can move from square A to square B *if B is blank*
  - (c) A tile can move from square A to square B.



Start State



Goal State

Relaxed problems for 8-puzzle

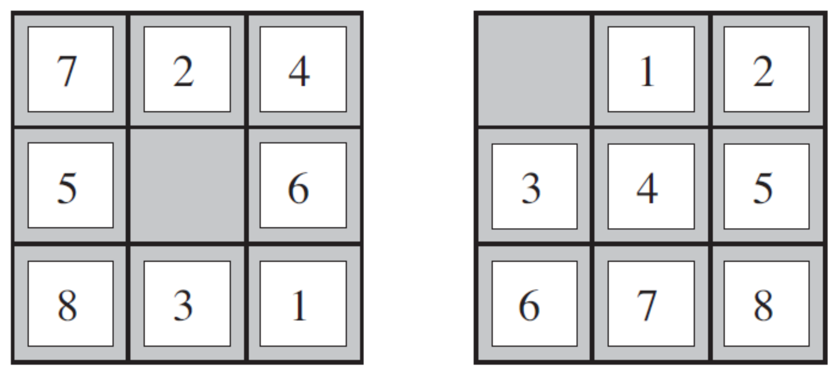
- Original transition model

A tile can move from square A to square B if
A is horizontally or vertically adjacent to B, and B is blank

- Relaxed transition model
  - (a) A tile can move from square A to square B *if A is adjacent to B*
  - (b) A tile can move from square A to square B *if B is blank*
  - (c) A tile can move from square A to square B.

Number of misplaced tiles
$h1(n) = 8$

Manhattan distance
$h2(n) = 18$



Start State   Goal State

Combining heuristic functions

- Given a **collection** of *admissible* heuristics, the **composite** heuristic (uses whichever function that is the most accurate on the node in question) is also *admissible*

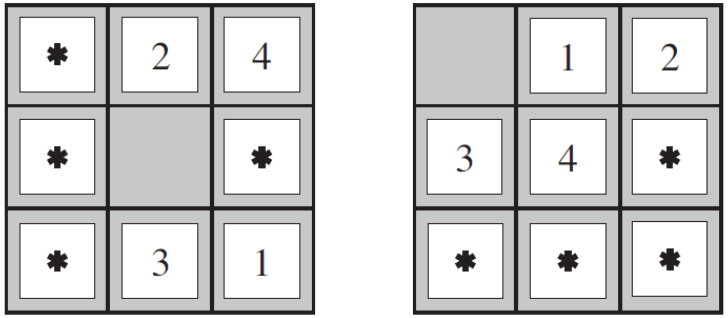
$$\boxed{h(n) = \max\{h_1(n), \ldots, h_m(n)\}}$$

- Furthermore, ***h(n)*** dominates all of these component heuristics

"better than"

(2) Generating heuristics from subproblems

- **Pattern Database:** stores exact solution cost for every possible subproblem instance
  - Every possible configuration of the first four tiles and the blank
  - Locations of the other four tiles are irrelevant (but moving them still counts toward the solution cost)



Start State Goal State

- The database can be constructed by *searching back from the goal*, and recording the cost of each pattern encountered

(3) Learning heuristic functions from experience

- Inductive learning of a heuristic function ***h(n)*** in terms of features $x_1(n)$ and $x_2(n)$

$$\boxed{h(n) = c_1 x_1(n) + c_2 x_2(n)}$$

- Candidate features
  - "number of misplaced tiles"
  - "number of pairs of adjacent tiles that are not adjacent in goal state

(3) Learning heuristic functions from experience

- Inductive learning of a heuristic function ***h(n)*** in terms of features $x_1(n)$ and $x_2(n)$

$$\boxed{h(n) = c_1 x_1(n) + c_2 x_2(n)}$$

- Candidate features
  - "number of misplaced tiles"
  - "number of pairs of adjacent tiles that are not adjacent in goal state

Input:

$$\left.\begin{array}{l}(x1=\ldots,\ x2=\ldots,\ h=\ldots)\\(x1=\ldots,\ x2=\ldots,\ h=\ldots)\\\ldots\\(x1=\ldots,\ x2=\ldots,\ h=\ldots)\end{array}\right\}\quad\begin{array}{l}c1 = ??\\c2 = ??\end{array}$$

Outline for Today

- What is AI?
- Problem-solving agent
- Uninformed search
- Informed search
- **Heuristic functions**
  - How to design them
  - How to evaluate them

Summary

- Informed search needs heuristic function, ***h(n)***, which estimates the cost of a solution from ***n***
  - **Greedy best-first search** expands nodes with minimal h(n). It is not optimal but is often fast
  - **A* search** expands nodes with minimal f(n)=g(n)+h(n). It is complete and optimal, provided that h(n) is admissible (or consistent). The space complexity of A* is still high.
  - **RBFS** and **SMA*** are robust, optimal, and use limited memory

- Performance of heuristic search depends on the quality of the heuristic function
  - Good heuristics can be constructed by (1) relaxing the problem definition, (2) storing precomputed solution costs for subproblems in a pattern database, or (3) learning from experience