Artificial Intelligence

Informed Search

Today

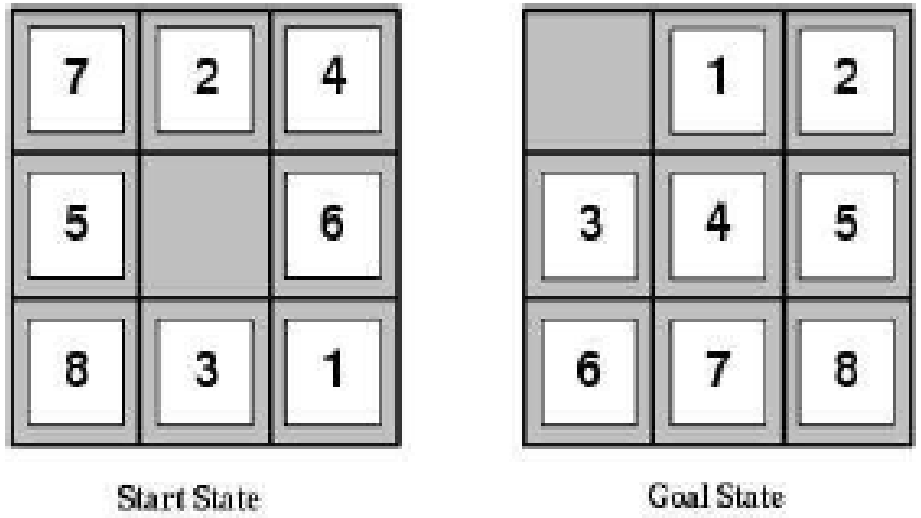
- Informed Search
  - Heuristics
  - Greedy Search
  - A* Search

Strategies that know whether one non-goal state is “more promising” than another are called informed search or heuristic search strategies

Limitations of uninformed search

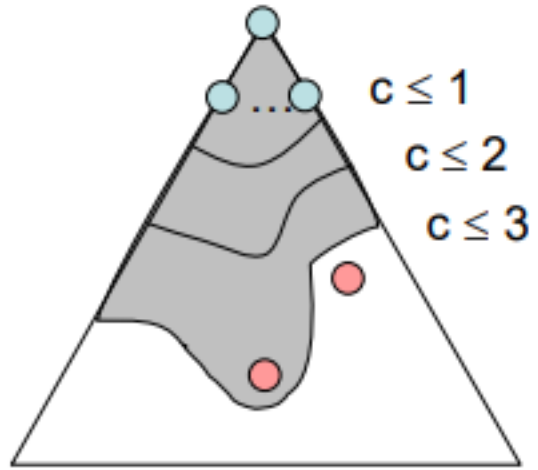
- 8-puzzle
  - Avg. solution cost is about 22 steps
  - branching factor ~ 3
  - E.g., d=12, IDS expands 3.6 million states on average

    [**24 puzzle** has $\mathbf{10^{24}}$ states (much worse)]



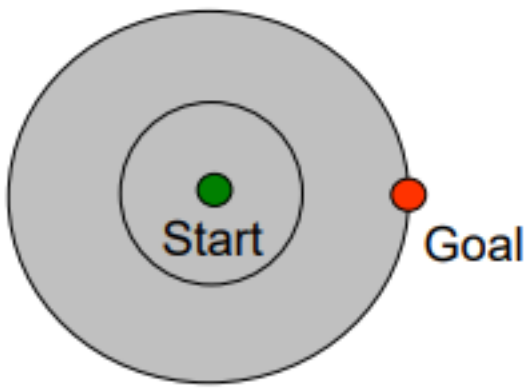
Uniform Cost Issues

- Remember: UCS explores increasing cost contours



- The good: UCS is complete and optimal!

- The bad:
  - Explores options in every “direction”
  - No information about goal location

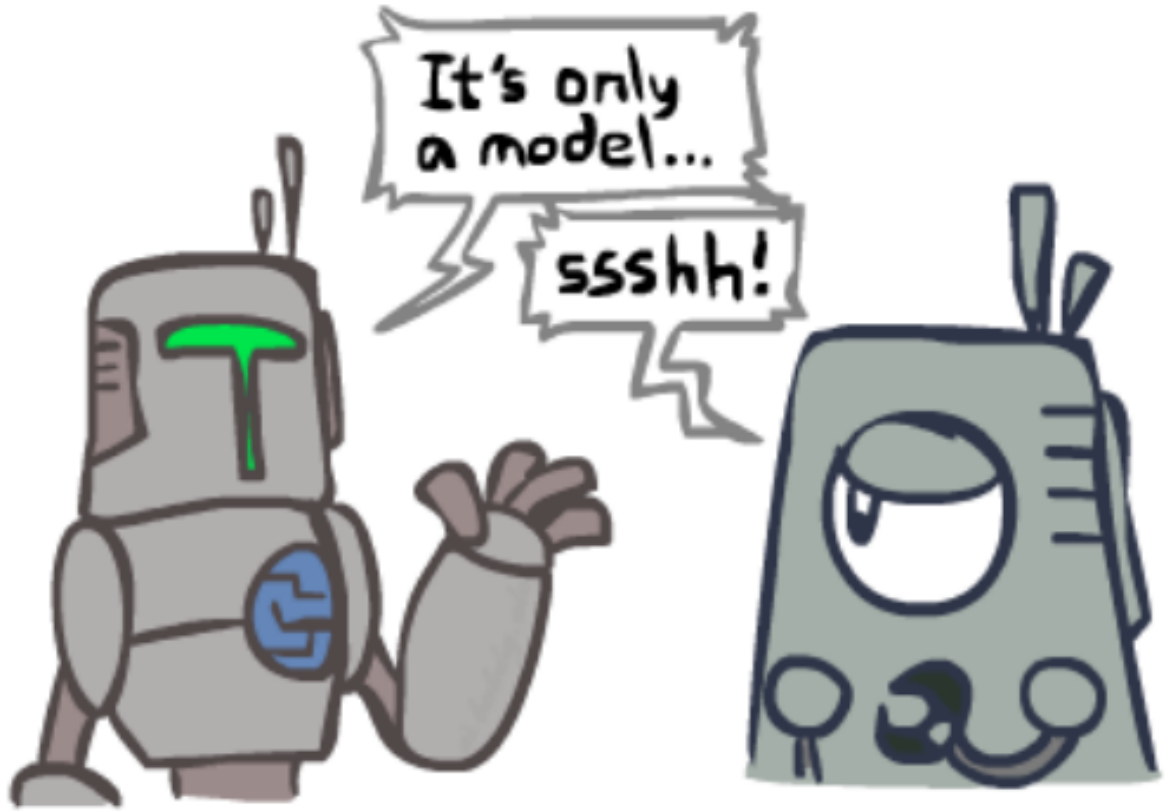


The One Queue

- All these search algorithms are the same except for fringe strategies
  - Conceptually, all fringes are priority queues (i.e. collections of nodes with attached priorities)
  - Practically, for DFS and BFS, you can avoid the log(n) overhead from an actual priority queue, by using stacks and queues
  - Can even code one implementation that takes a variable queuing object

Search and Models

- Search operates over models of the world
  - The agent doesn't actually try all the plans out in the real world!
  - Planning is all "in simulation"
  - Your search is only as good as your models…



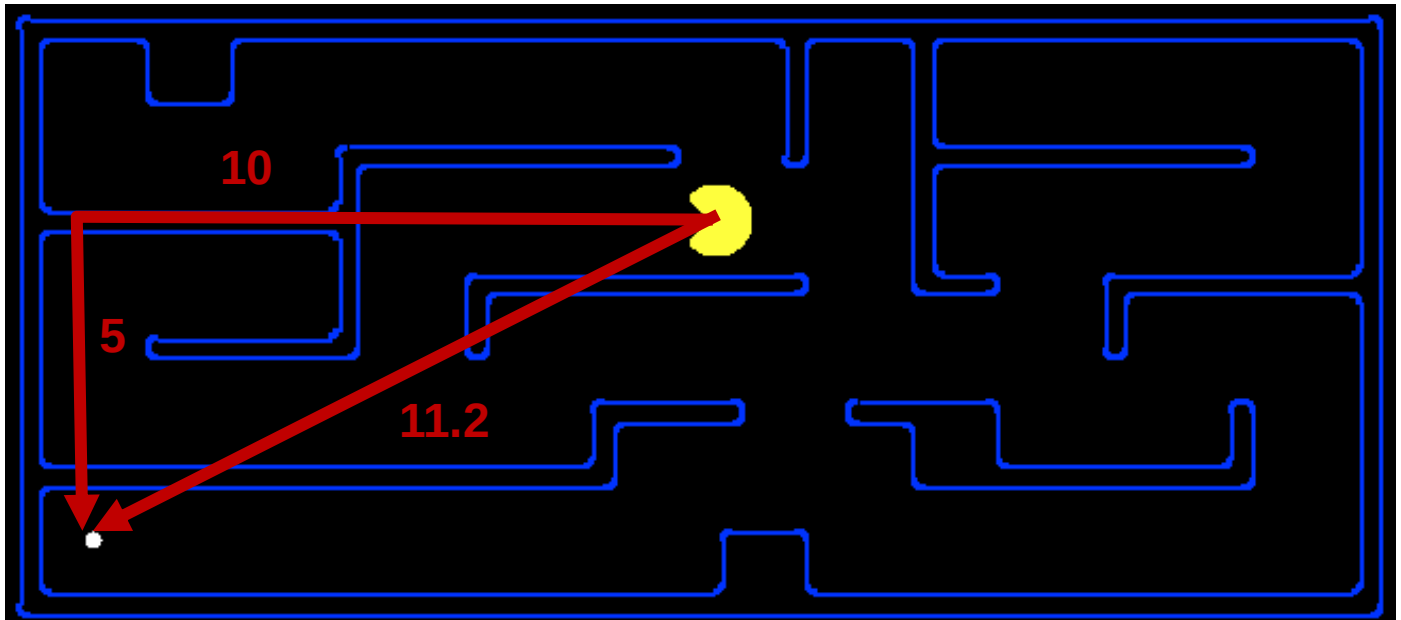
Search Gone Wrong?

Model didn't incorporate the real world change
thats why resulted in wrong output

Search Heuristics

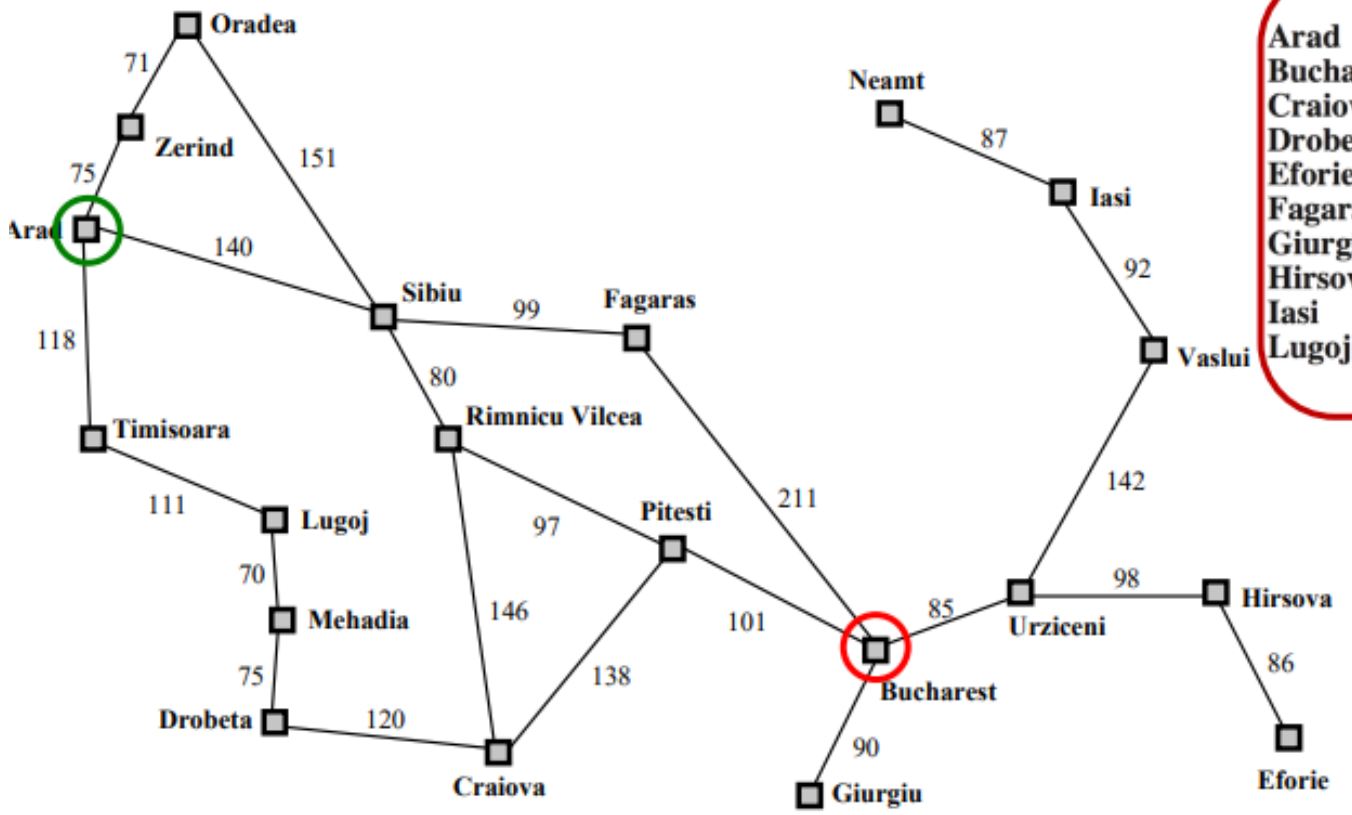
- A heuristic is:
  - A function that ***estimates*** how close a state is to a goal
  - Designed for a particular search problem

    h(n) = estimated cost of the cheapest path from the state at node n to a goal state.
- Examples: Manhattan distance, Euclidean distance for pathing



Example: Heuristic Function

Example: Euclidean distance to Bucharest



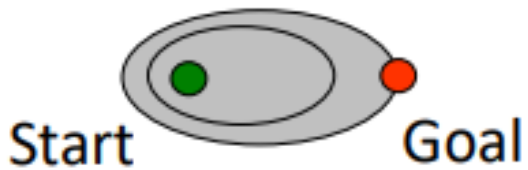
Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

h(state) → value

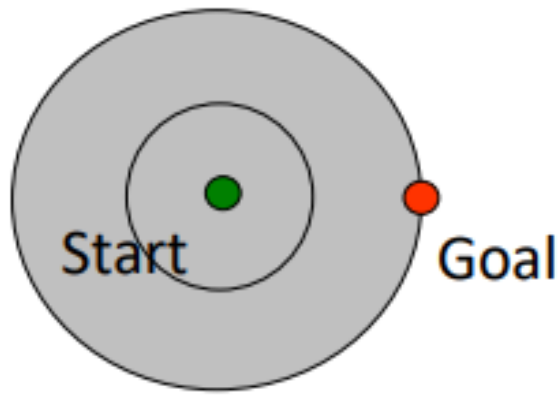
Lookup table containing distance of bucharest to all cities so in this case Lookup table is the Heuristic function

Effect of heuristics

Guide search ***towards the goal*** instead of ***all over the place***

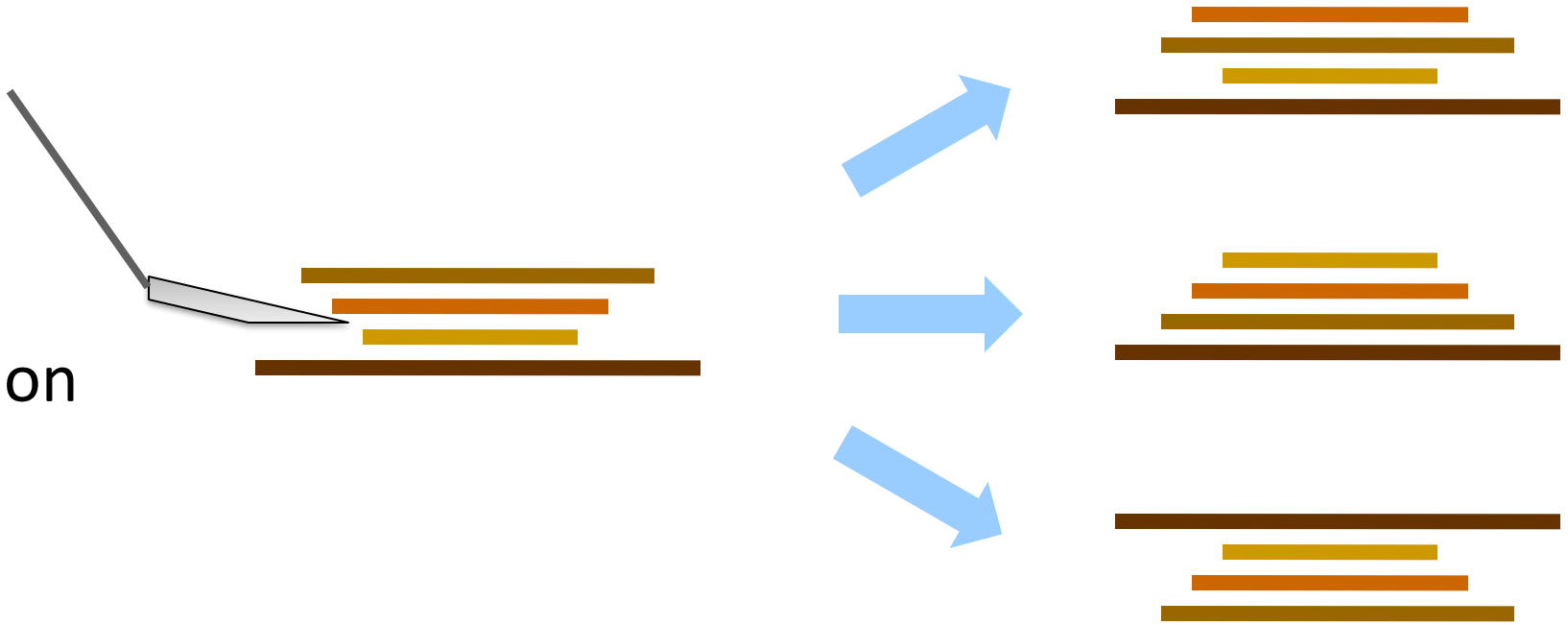


Informed



Uninformed

Example: Pancake Flipping Problem

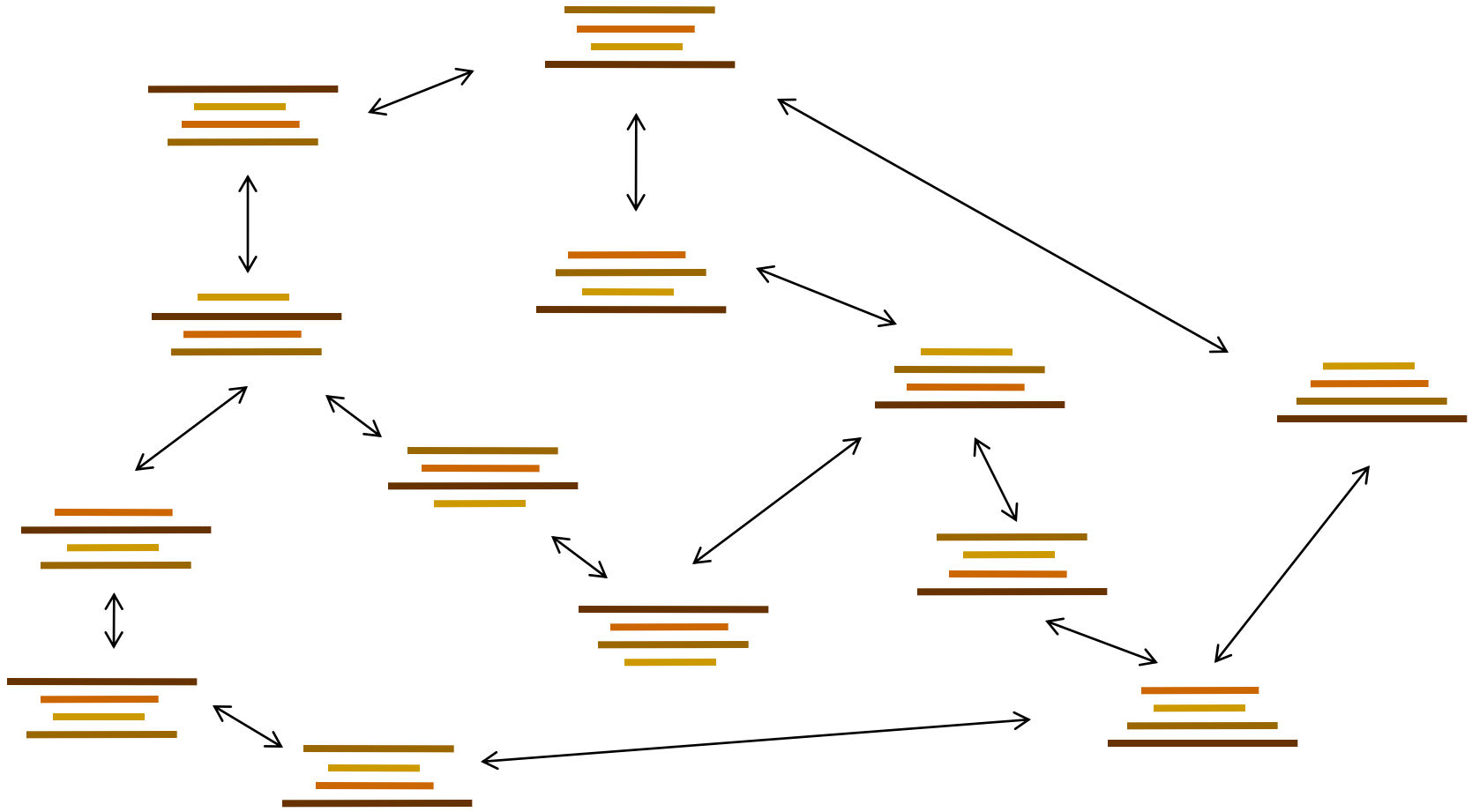


Flip Position

Cost: Number of pancakes flipped

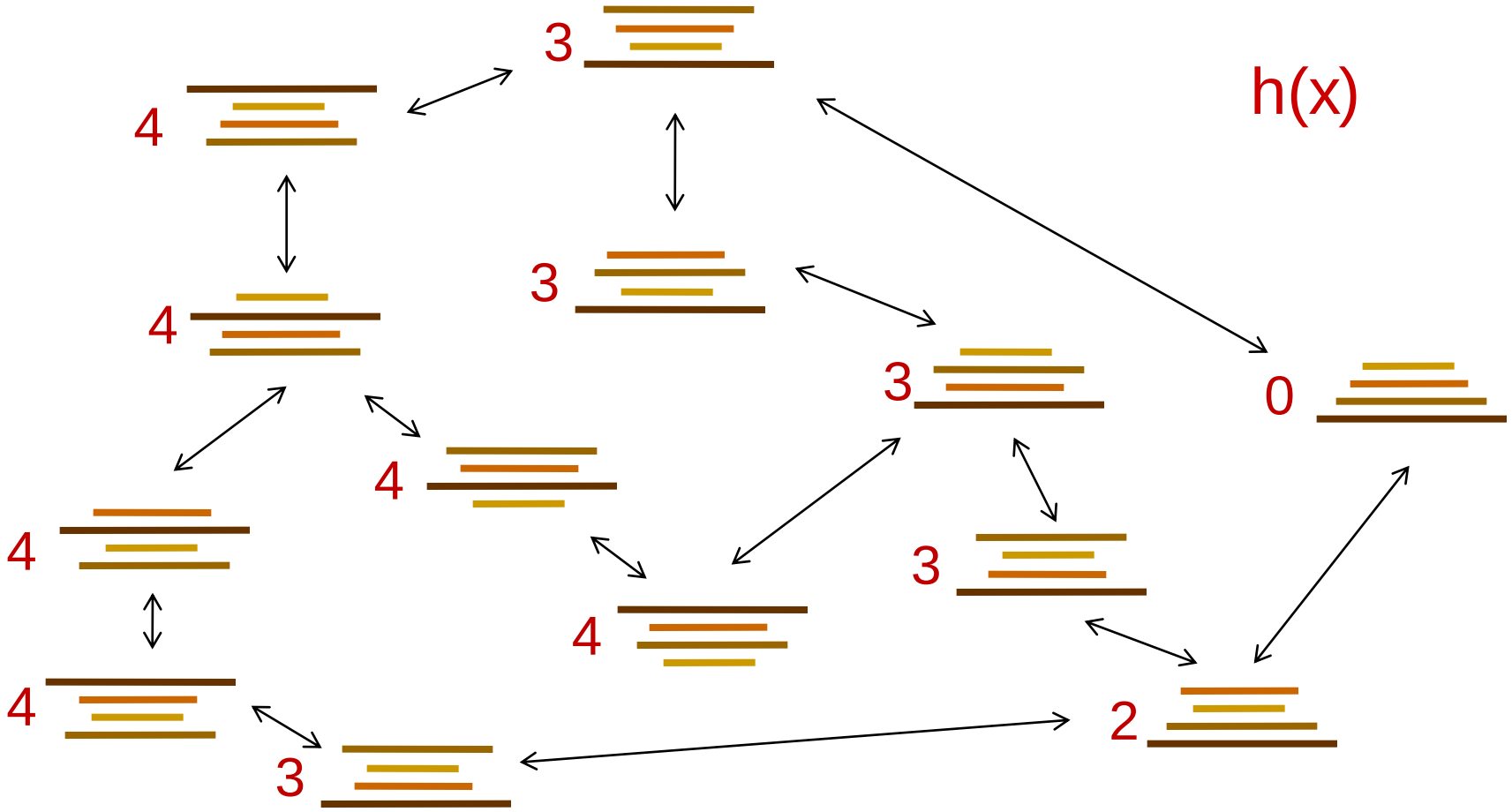
Pancake Flipping Problem: Heuristic Function?

For this state space what could be a good heuristic Function?



Pancake Flipping Problem: Heuristic Function?

Heuristic: the number (id) of the largest pancake that is still out of place

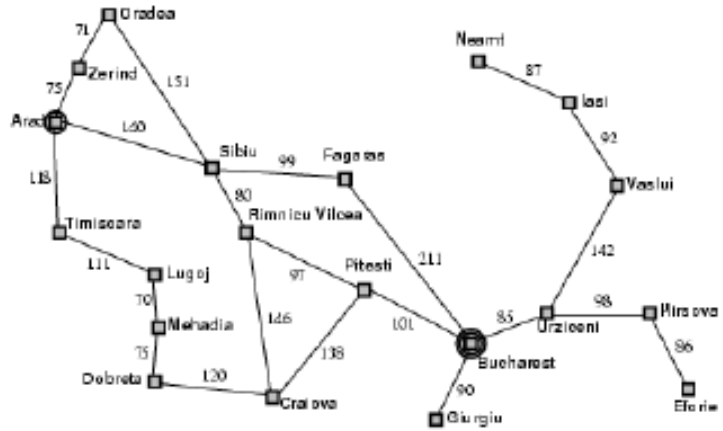


GREEDY SEARCH

Greedy best-first search example

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

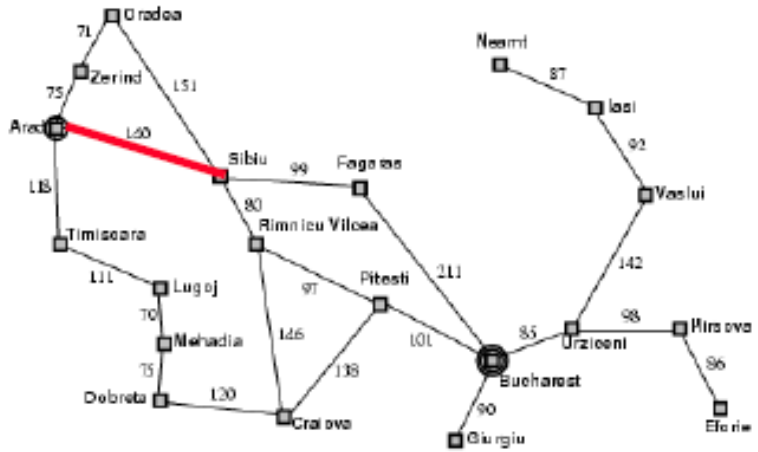
h(state) → value



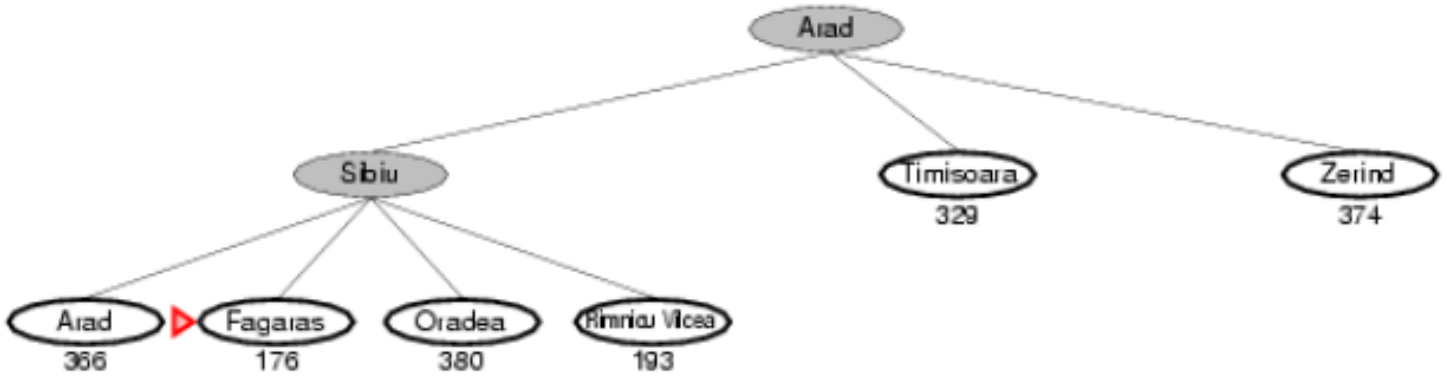
Greedy best-first search example

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

h(state) → value



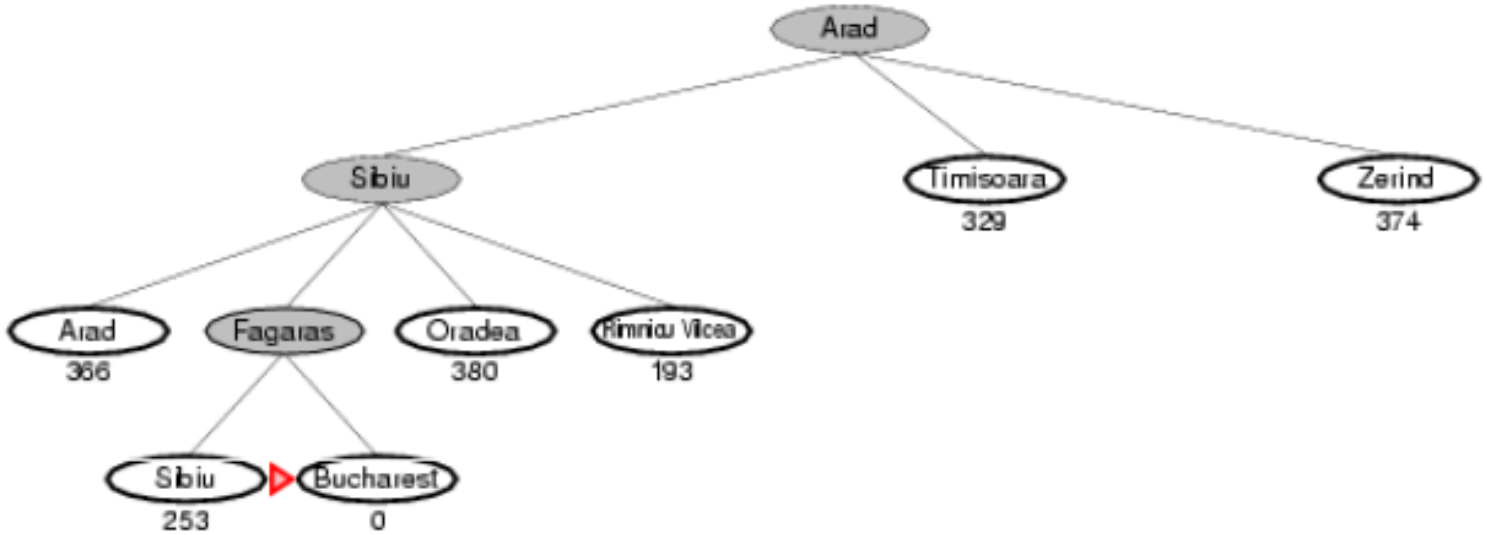
Greedy best-first search example



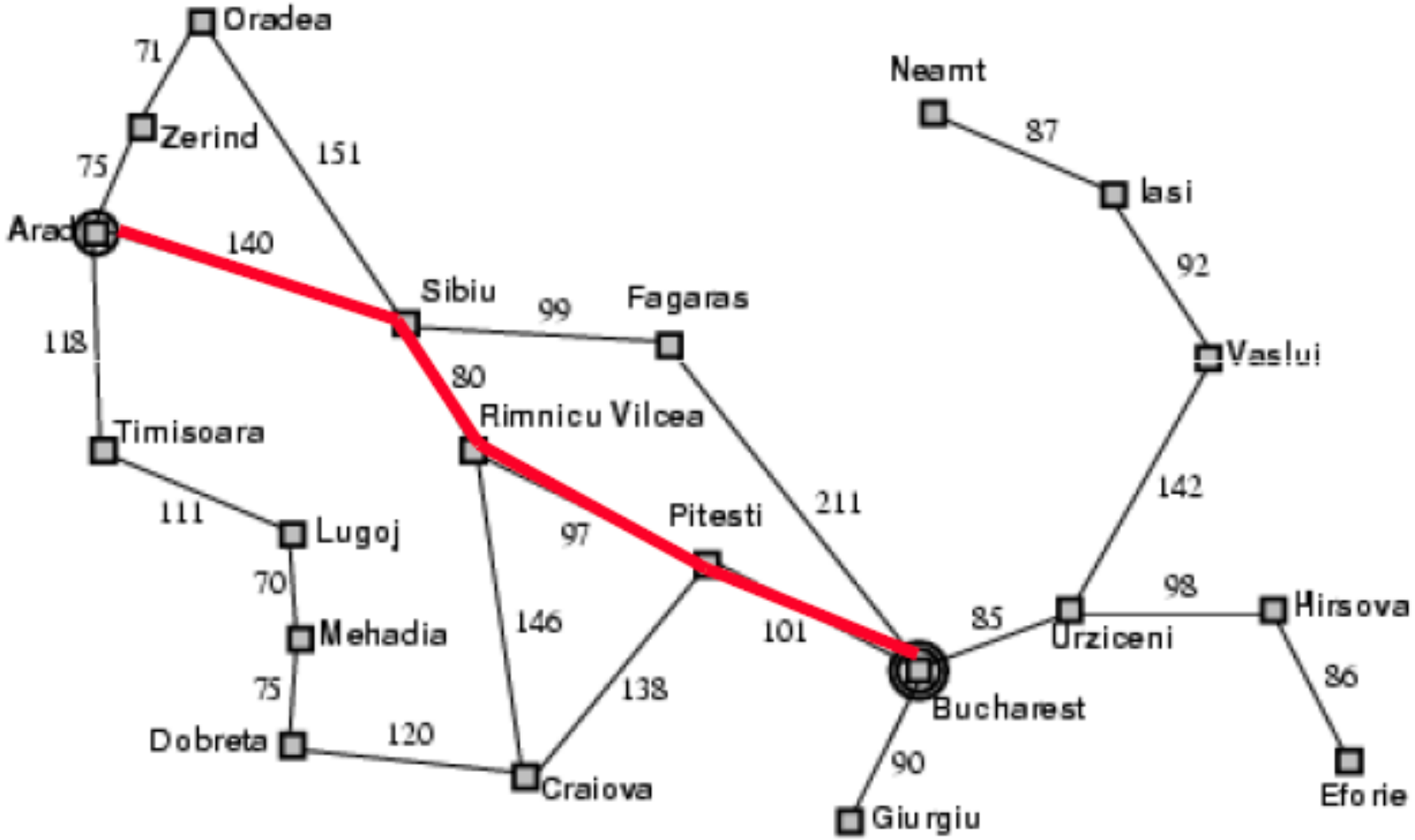
Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374

h(state) → value

Greedy best-first search example



BUT Optimal Path



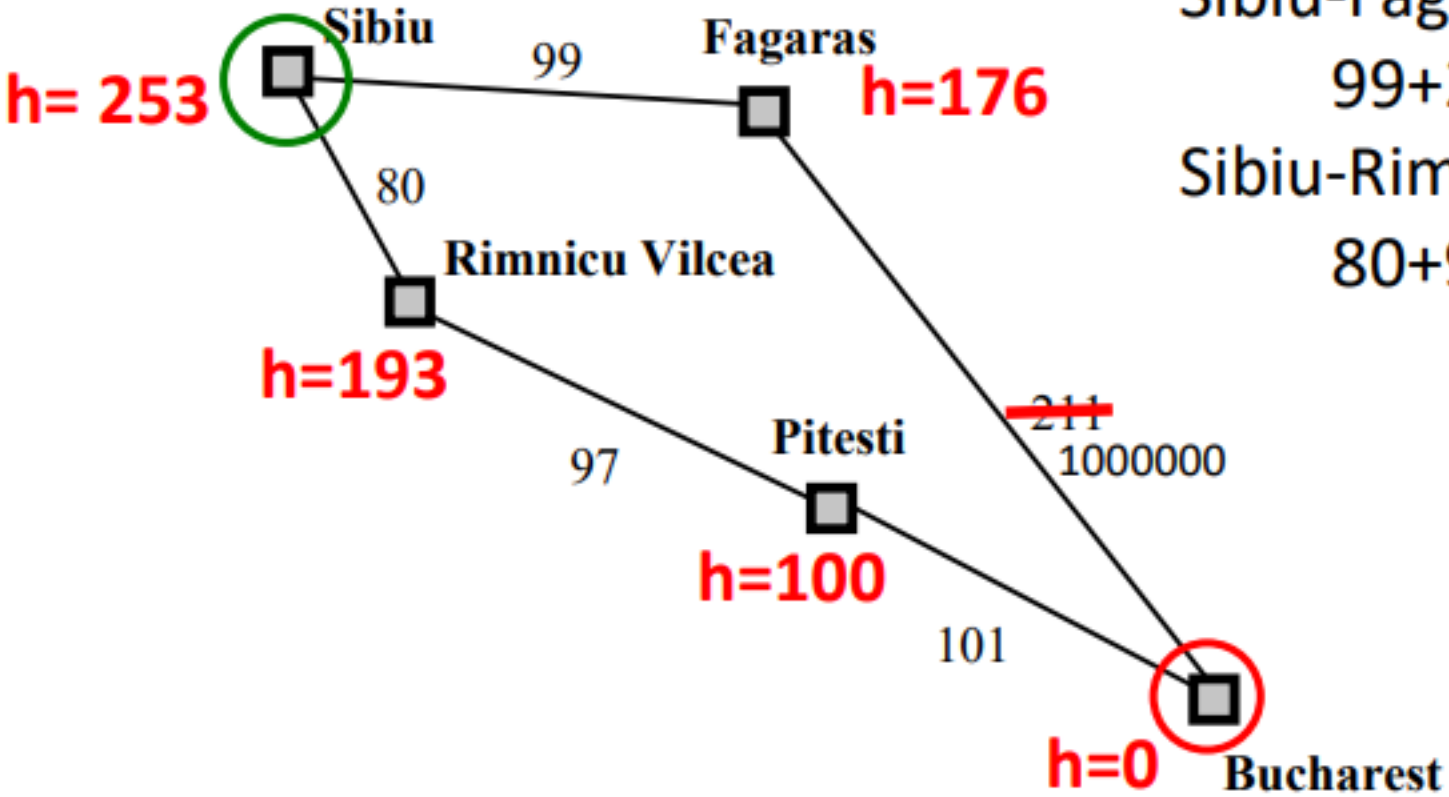
Properties of greedy best-first search

- Complete?
  - Not unless it keeps track of all states visited
    - Otherwise can get stuck in loops (just like DFS)
- Optimal?
  - No – we just saw a counter-example
- Time?
  - $O(b^m)$, can generate all nodes at depth m before finding solution
  - Can be much better with a good heuristic
- Space?
  - $O(b^m)$ – again, worst case, can generate all nodes at depth m before finding solution

Greedy Search

Expand the node that seems closest…(order frontier by h)

What can possibly go wrong?



Sibiu-Fagaras-Bucharest =

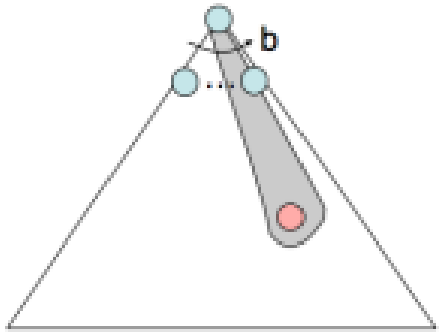
99+211 = **310**

Sibiu-Rimnicu Vilcea-Pitesti-Bucharest =

80+97+101 = **278**

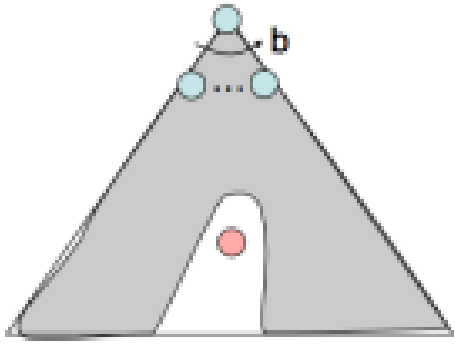
Greedy Search

- Strategy: expand a node that you think is closest to a goal state
  - Heuristic: estimate of distance to nearest goal for each state



(a) Greedy search on a good day :)

- A common case:
  - Best-first takes you straight to the (wrong) goal as seen in example.



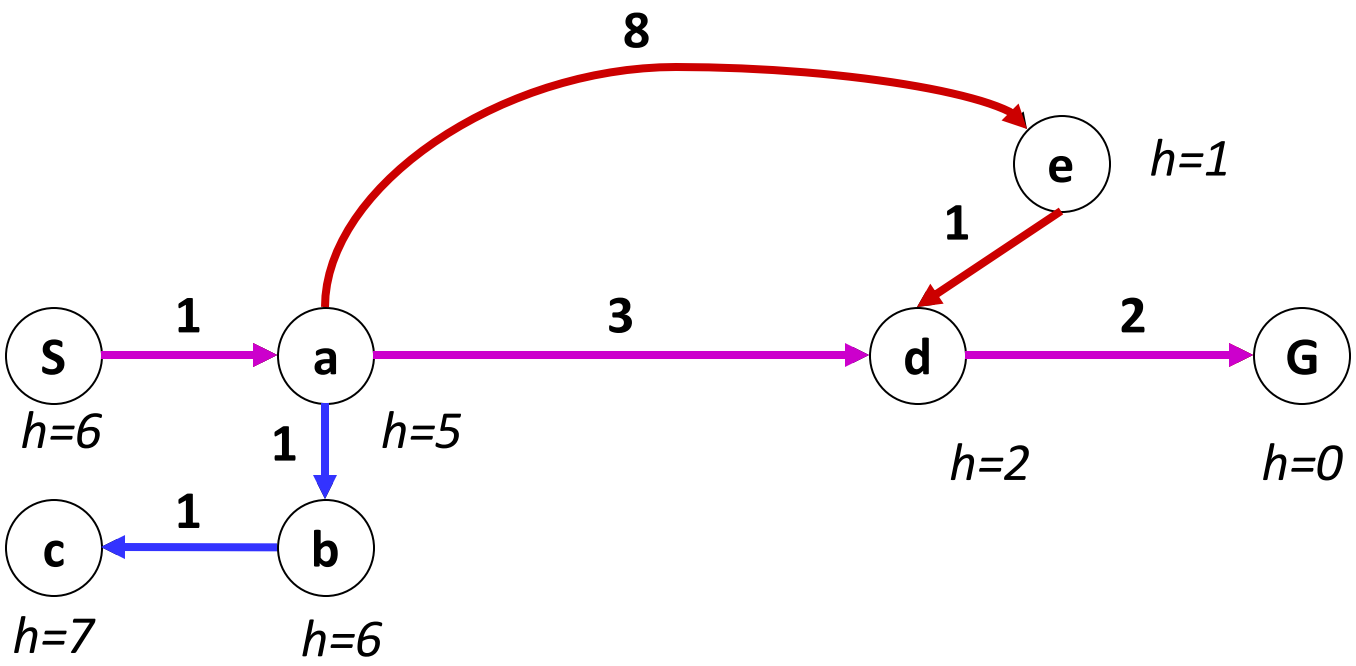
(b) Greedy search on a bad day :(

- Worst-case: like a badly-guided DFS

A* SEARCH

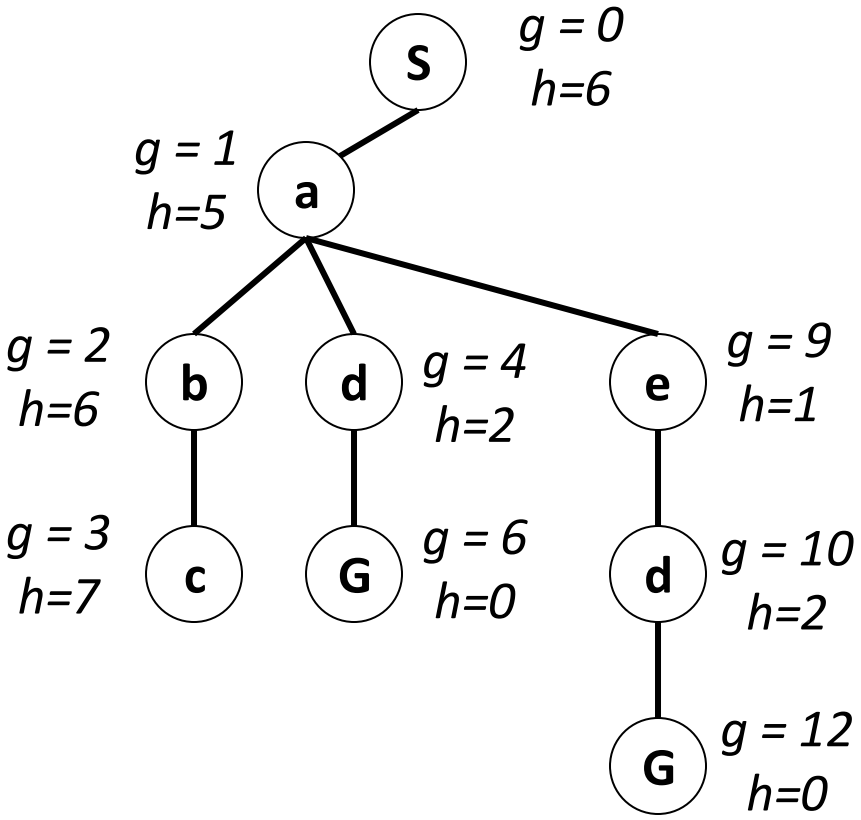
Combining UCS and Greedy

- **Uniform-cost** orders by path cost, or ***backward cost*** **g(n)**
- **Greedy** orders by goal proximity, or ***forward cost*** **h(n)**



- **A* Search** orders by the sum: **f(n) = g(n) + h(n)**

Estimation of future steps what would be the cost of them

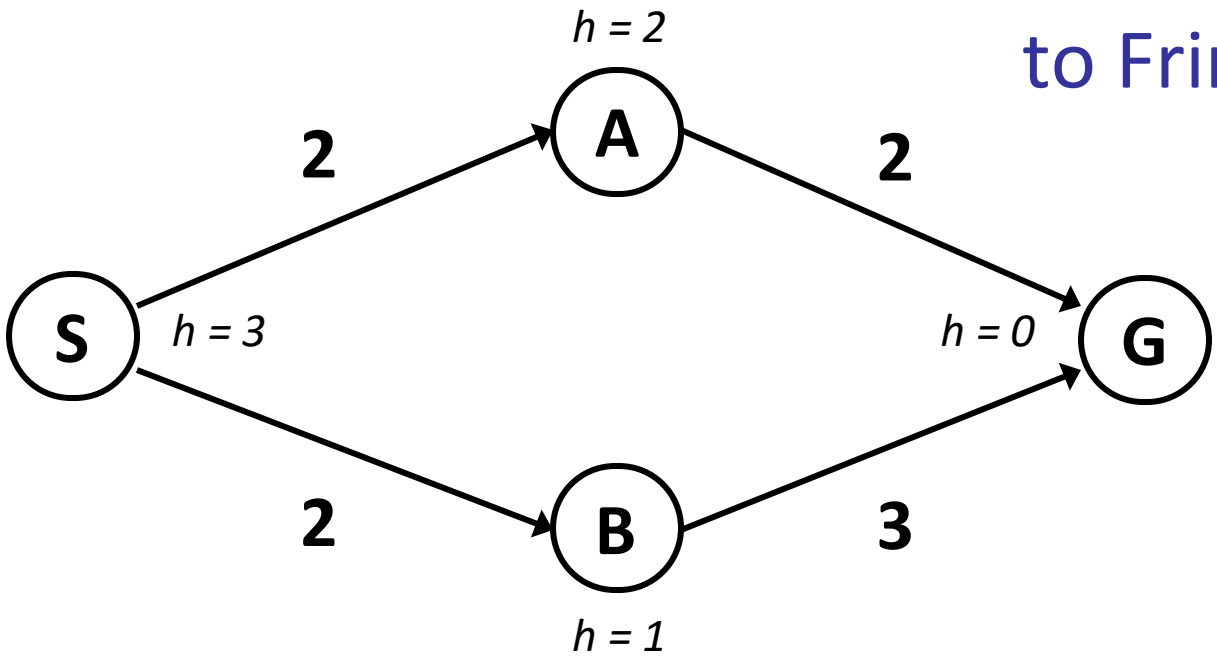


Example: Teg Grenager

When should A* terminate?

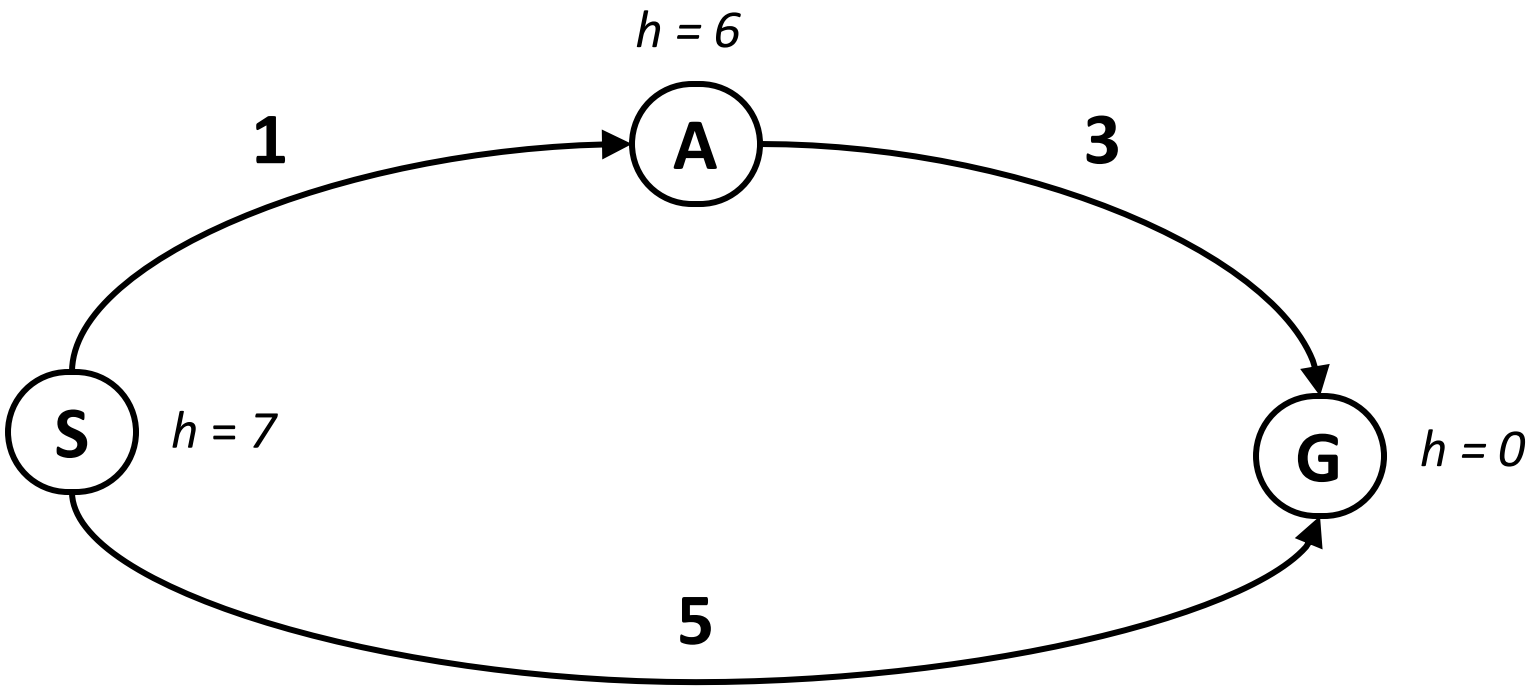
- Should we stop when we enqueue a goal?

Adding Goal to Fringe



- No: only stop when we dequeue a goal

Is A* Optimal?



- What went wrong?
  - Actual goal cost < estimated goal cost
- We need estimates to be less than actual costs!

Admissible Heuristics

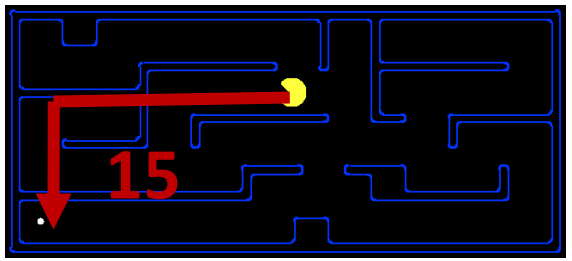
- A heuristic h is *admissible* (optimistic) if:if h(n) is never larger than h*(n), namely h(n) is always less or equal to true cheapest cost from n to the goal.

$$\forall n \quad 0 \leq h(n) \leq h^*(n)$$

h(n) = estimated cost of the cheapest path from the state at node n to a goal state.

where $h^*(n)$ is the true cost to a nearest goal

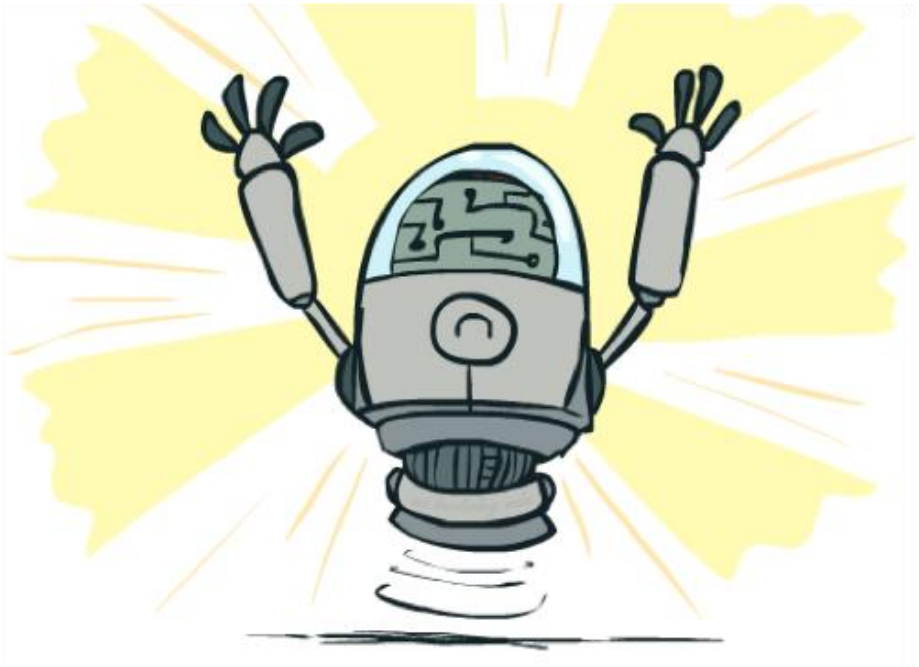
- Examples:



Is Manhattan distance here admissible

4

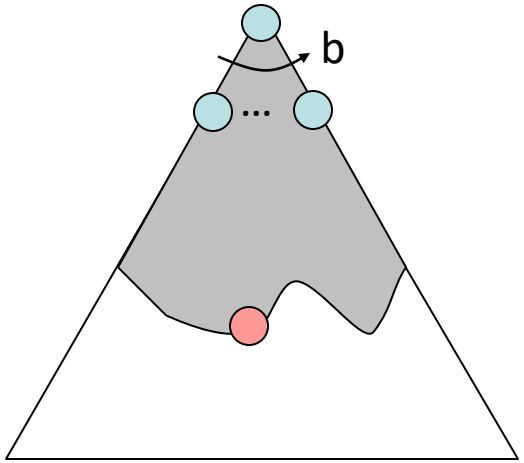
- Coming up with admissible heuristics is most of what's involved in using A* in practice.



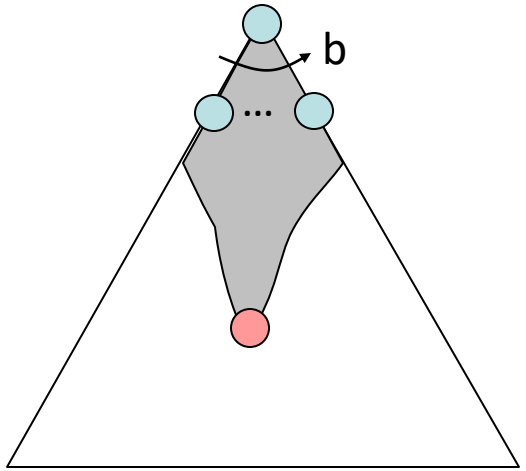
A* is optimal if *h* is admissible

Properties of A*

Uniform-Cost

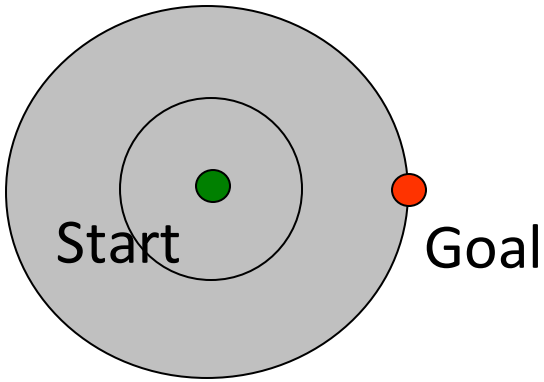


A*

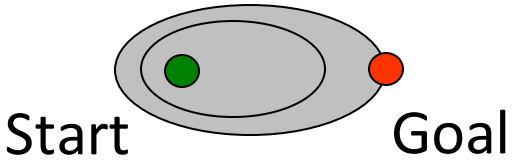


UCS vs A* Contours

- Uniform-cost expands equally in all "directions".



- A* expands mainly toward the goal, but does hedge its bets to ensure optimality.



How to Construct an Admissible Heuristic

You identify relaxed version of the problem:

- where one or more constraints have been dropped
- problem with fewer restrictions on the actions

Robot: the agent can move through walls

Driver: the agent can move straight

8puzzle: (1) tiles can move anywhere

(2) tiles can move to any adjacent square

Result: The cost of an optimal solution in the relaxed problem is an admissible heuristic for the original problem

How to Construct an admissible Heuristic (cont.)

You should identify constraints which, when dropped, make the problem extremely easy to solve

- this is important because heuristics are not useful if they're as hard to solve as the original problem!

This was the case in our examples

Robot: ***allowing*** the agent to move through walls. Optimal solution to this relaxed problem is Manhattan distance

Driver: ***allowing*** the agent to move straight. Optimal solution to this relaxed problem is straight-line distance

8puzzle: (1) tiles **can move anywhere** Optimal solution to this relaxed problem is number of misplaced tiles

(2) tiles can move to **any adjacent square…**.

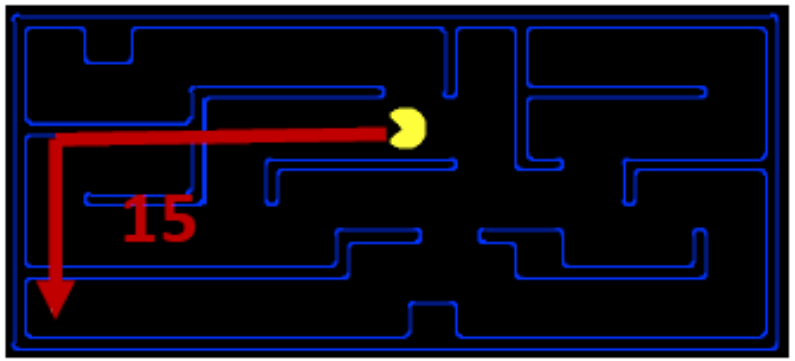
Creating Admissible Heuristics

- Most of the work in solving hard search problems optimally is in coming up with admissible heuristics.

- Often, admissible heuristics are solutions to *relaxed problems,* where new actions are available or included in relaxed problems

- Originally you have to go by road now in relaxed you can go by Air.

366

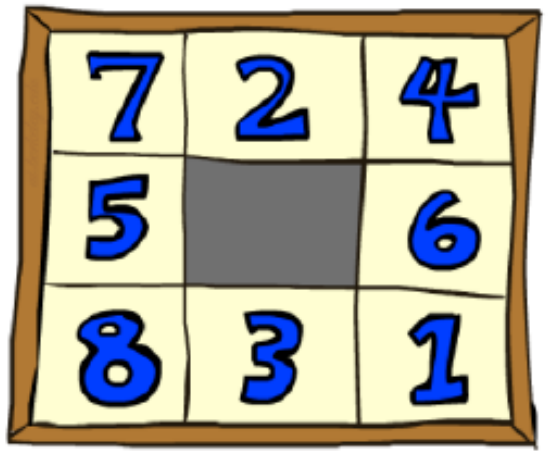


for Manhattan distance New Action in this??

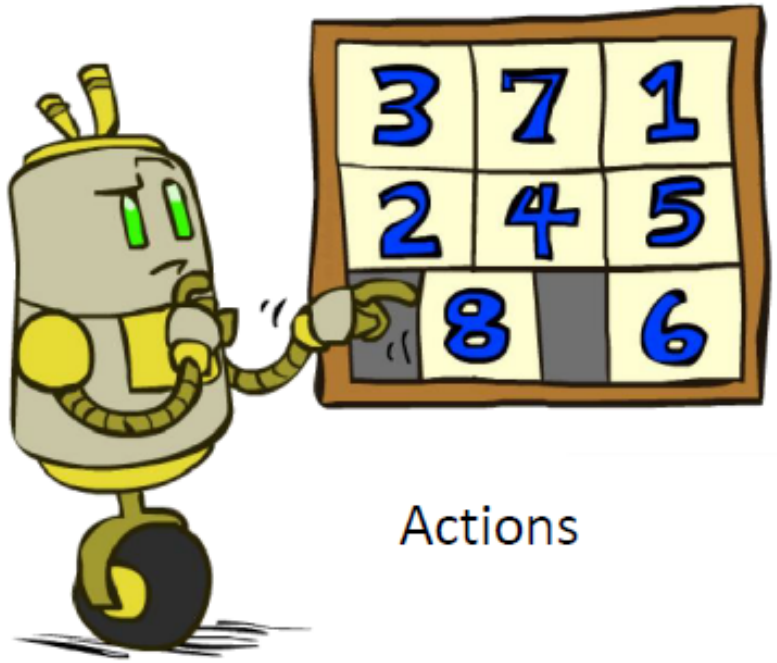
- Inadmissible heuristics are often useful too.

for Euclidean distance New Action in this??

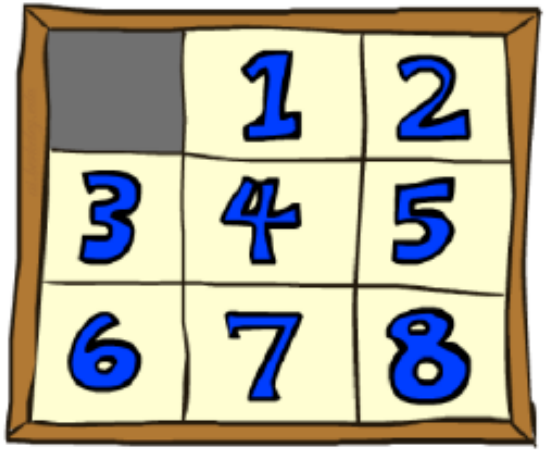
Example: 8 Puzzle



Start State



Actions



Goal State

- What are the states?
- How many states?
- What are the actions?
- How many successors from the start state?
- What should the costs be?

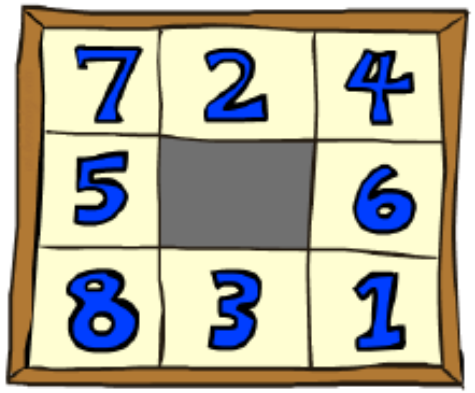
Heuristics Function ??

Every time we move a Tile cost is one

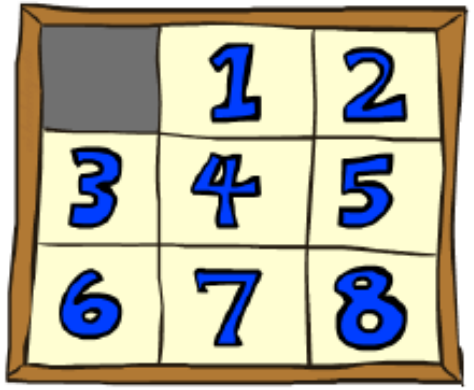
how far from final state .. And how we calculate this heuristics

8 Puzzle I

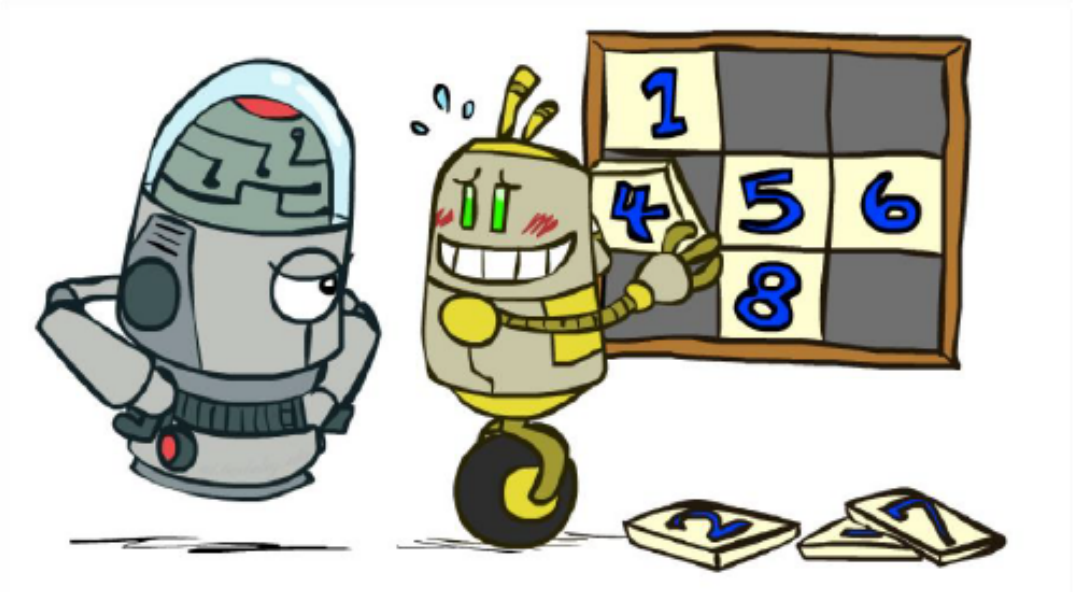
- Heuristic: Number of tiles misplaced
- Why is it admissible?
- h(start) = 8
- This is a *relaxed-problem* heuristic



Start State



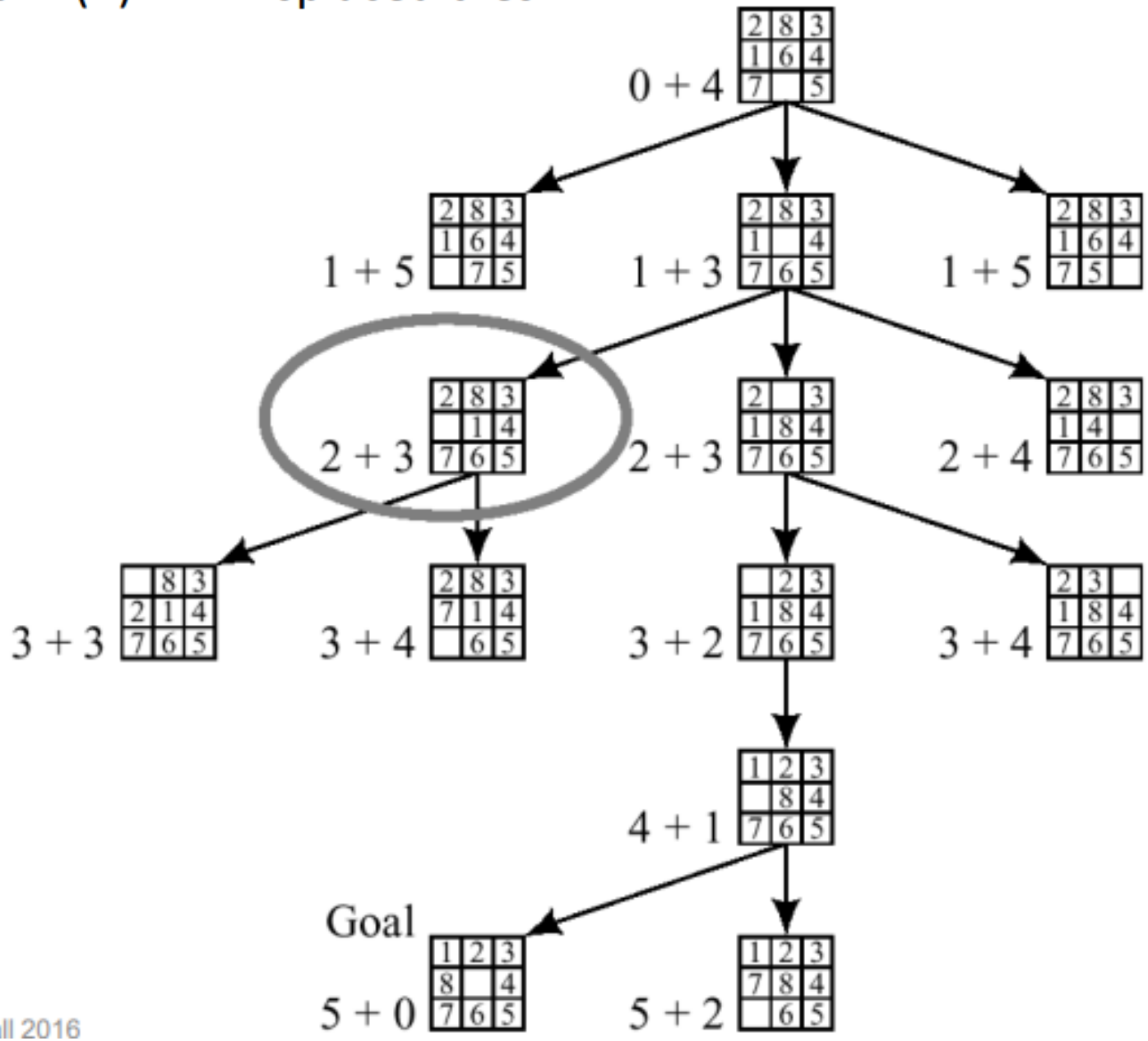
Goal State



	Average nodes expanded when the optimal path has...		
	...4 steps	...8 steps	...12 steps
UCS	112	6,300	3.6×10^6
A* (tiles)	13	39	227

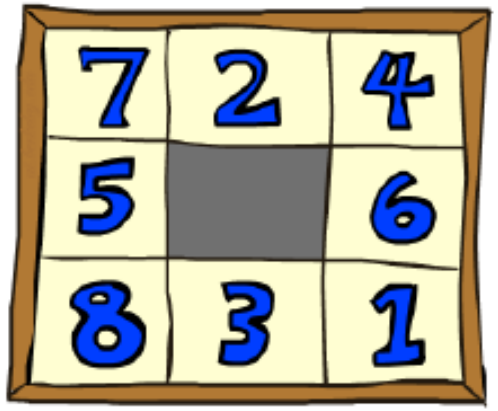
Move tile from one place it in destination

A* on 8-Puzzle with h(n) = # misplaced tiles

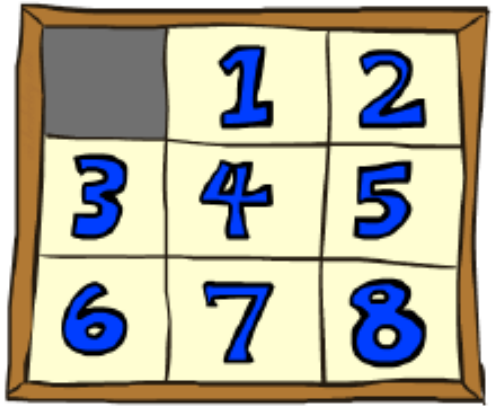


8 Puzzle II

- What if we had an easier 8-puzzle where any tile could slide any direction at any time, ignoring other tiles?



Start State



Goal State

- Total *Manhattan* distance
- Why is it admissible?
- h(start) = 3 + 1 + 2 + ... = 18

	Average nodes expanded when the optimal path has...		
	...4 steps	...8 steps	...12 steps
TILES	13	39	227
A* (Manhatten)	12	25	73

distance from current position to final position

A* Applications

- Video games
- Pathing / routing problems
- Resource planning problems
- Robot motion planning
- Language analysis
- Machine translation
- Speech recognition
- …