Graduate AI

Lecture 2: Search I

Teachers:
Zico Kolter
Ariel Procaccia (this time)

Search Problems

- A search problem has:
  - States (configurations)
  - Start state and goal states
  - Successor function: maps states to (action,state,cost) triples

Example: Pancakes

Discrete Mathematics 27 (1979) 47–57.
© North-Holland Publishing Company

BOUNDS FOR SORTING BY PREFIX REVERSAL

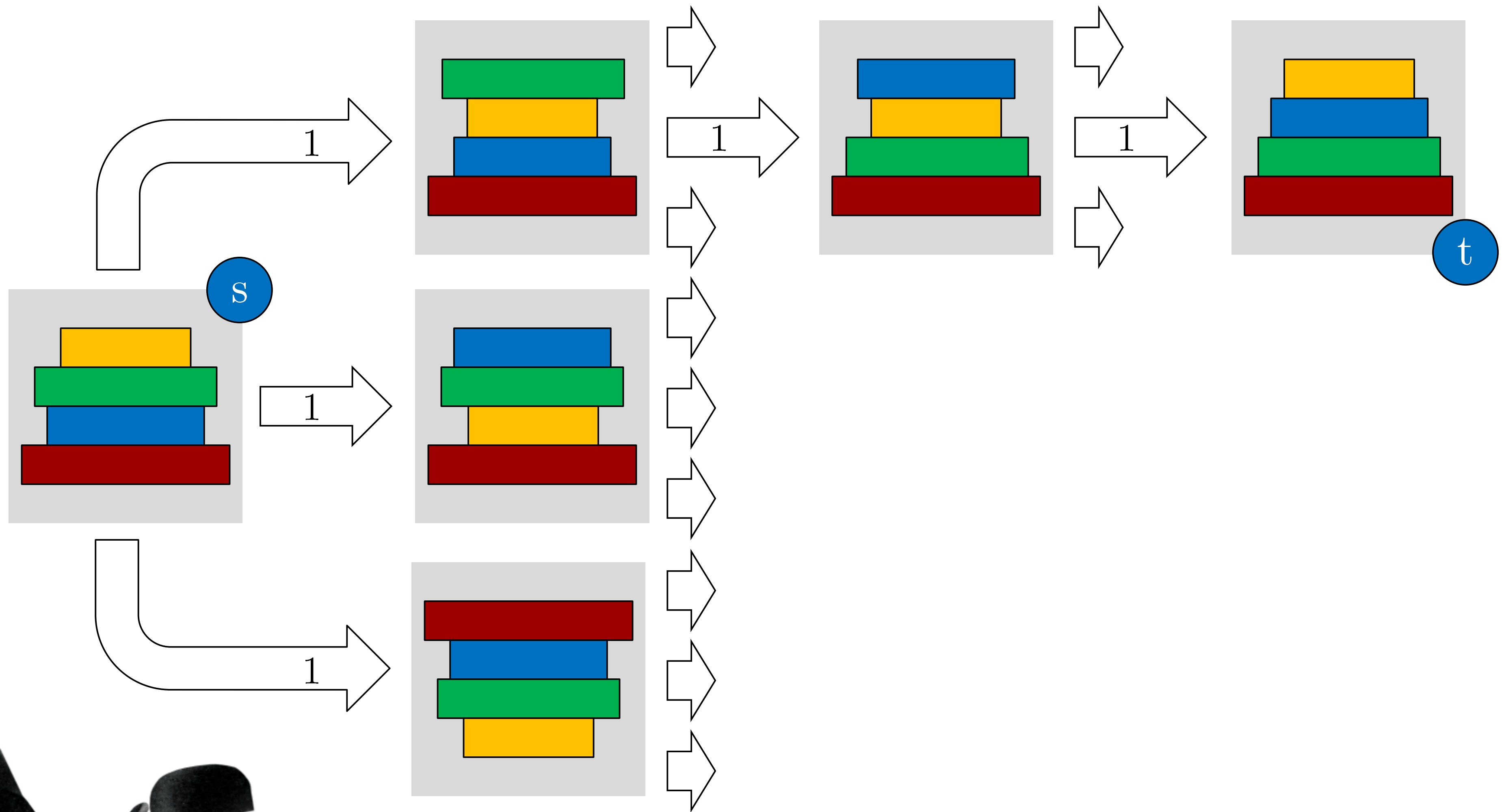
William H. GATES
Microsoft, Albuquerque, New Mexico

Christos H. PAPADIMITRIOU*†
Department of Electrical Engineering, University of California, Berkeley, CA 94720, U.S.A.

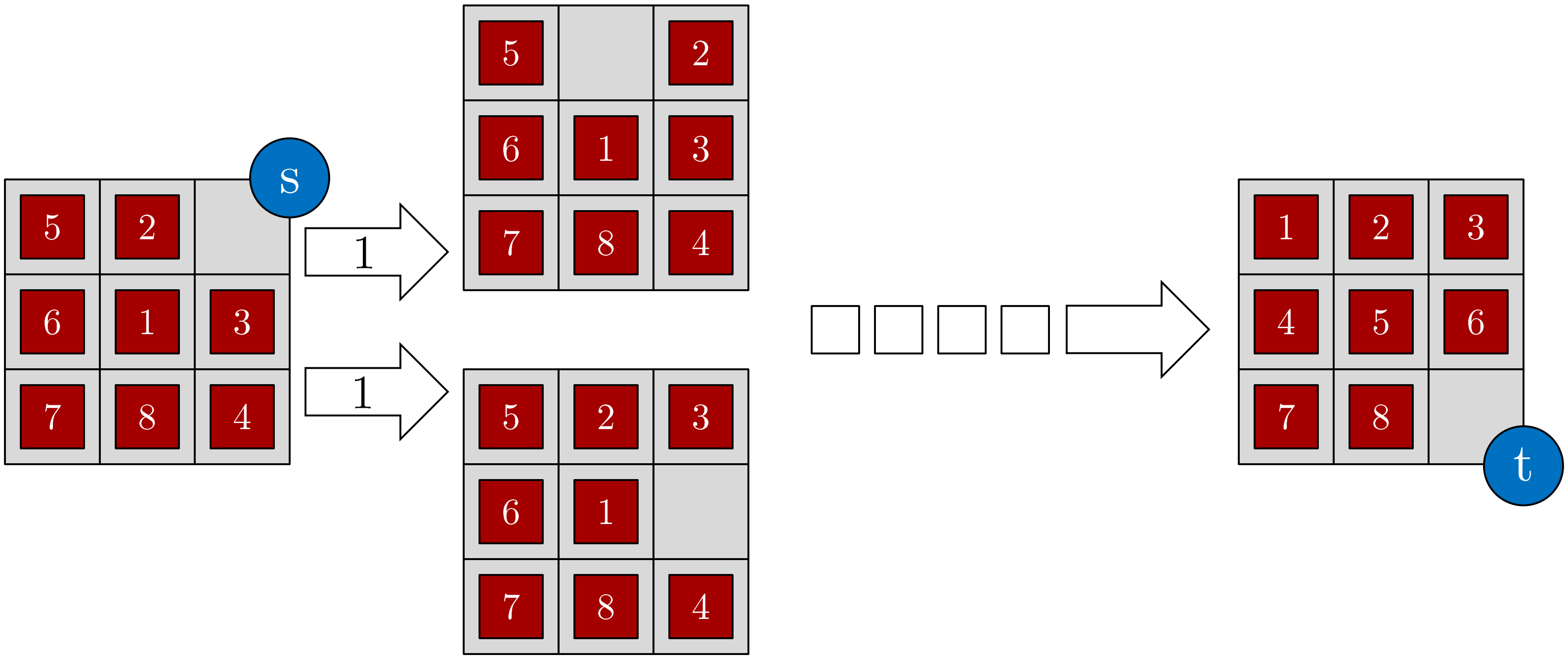
Received 18 January 1978
Revised 28 August 1978

For a permutation σ of the integers from 1 to n, let $f(\sigma)$ be the smallest number of prefix reversals that will transform σ to the identity permutation, and let $f(n)$ be the largest such $f(\sigma)$ for all σ in (the symmetric group) S_n. We show that $f(n) \leq (5n+5)/3$, and that $f(n) \geq 17n/16$ for n a multiple of 16. If, furthermore, each integer is required to participate in an even number of reversed prefixes, the corresponding function $g(n)$ is shown to obey $3n/2 - 1 \leq g(n) \leq 2n+3$.

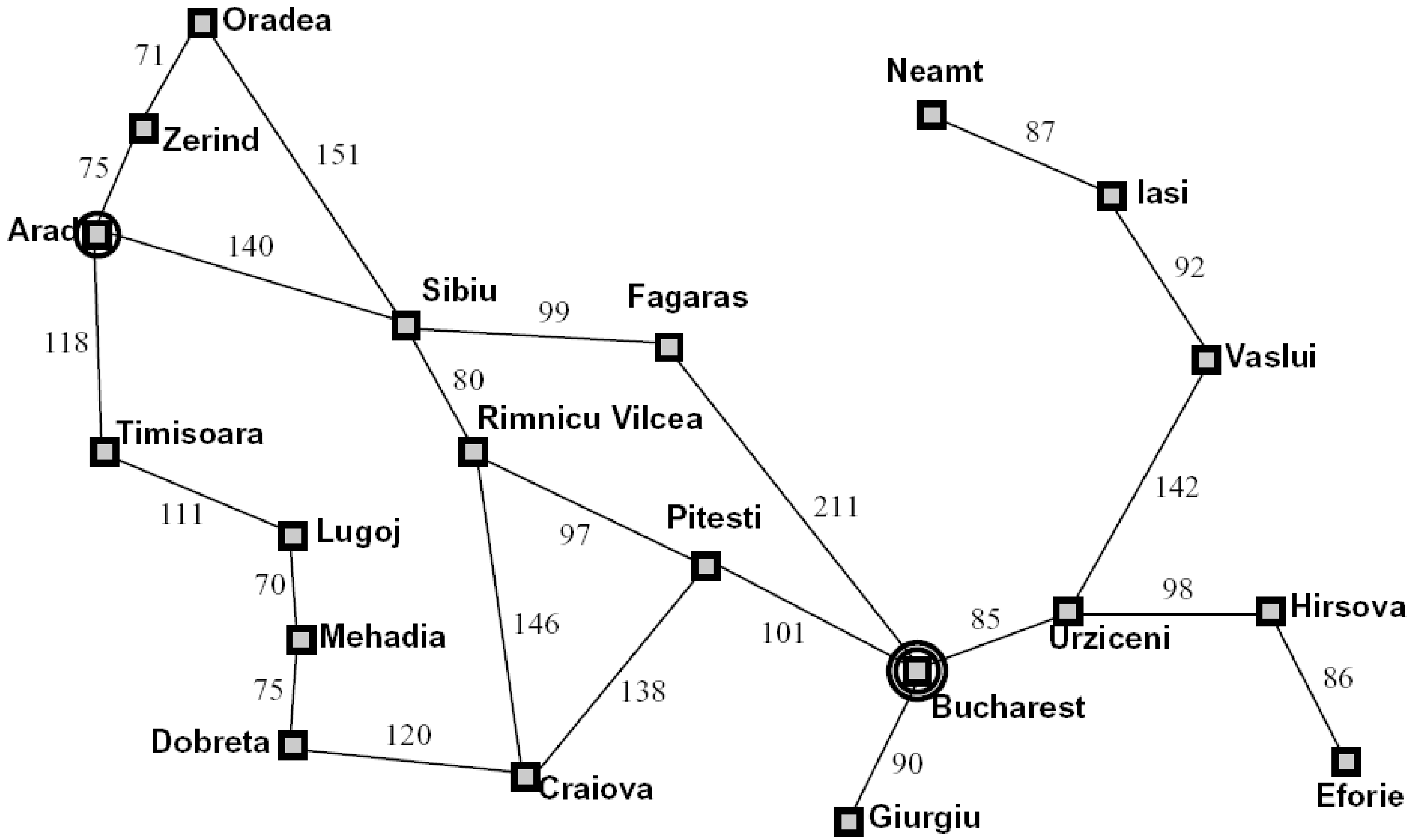
Example: Pancakes



EXAMPLE: 8-PUZZLE



Example: Pathfinding



Tree Search

function Tree-Search(problem, strategy)

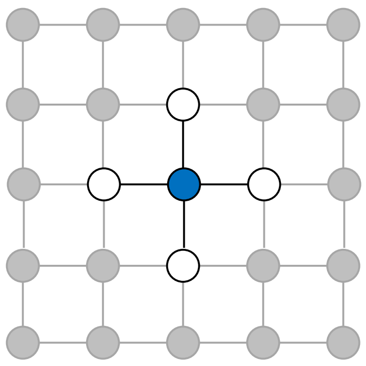
set of frontier nodes contains the start state of problem

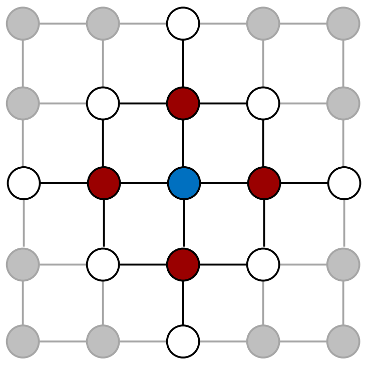
loop

- if there are no frontier nodes then return failure
- choose a frontier node for expansion using strategy
- if the node contains a goal then return the corresponding solution
- else expand the node and add the resulting nodes to the set of frontier nodes

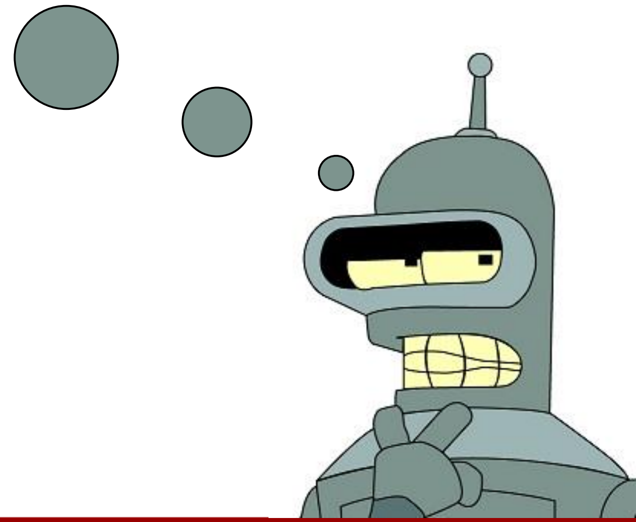
Tree Search

- Tree search can expand many nodes corresponding to the same state
- In a rectangular grid:
  - Search tree of depth d has 4^d leaves
  - There are only $4d$ states at Manhattan distance exactly d from any given state





Algorithms that forget their history are doomed to repeat it!



Graph Search

function Graph-Search(problem, strategy)

set of frontier nodes contains the start state of problem

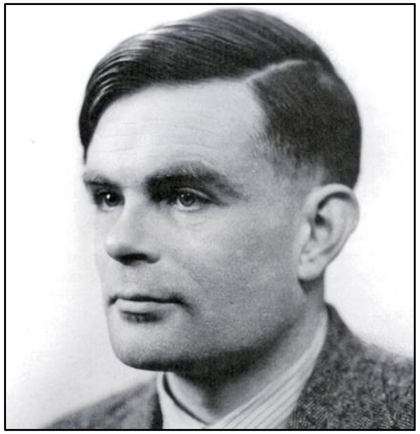
loop

- if there are no unexpanded frontier nodes then return failure
- choose an unexpanded frontier node for expansion using strategy, and add it to the expanded set
- if the node contains a goal then return the corresponding solution
- else expand the node and add the resulting nodes to the set of frontier nodes, only if not in the expanded set

Uninformed vs. Informed

Uninformed

Can only generate successors and distinguish goals from non-goals

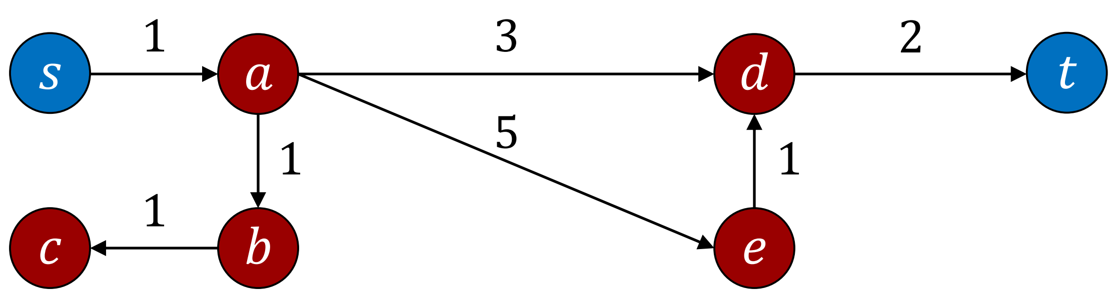


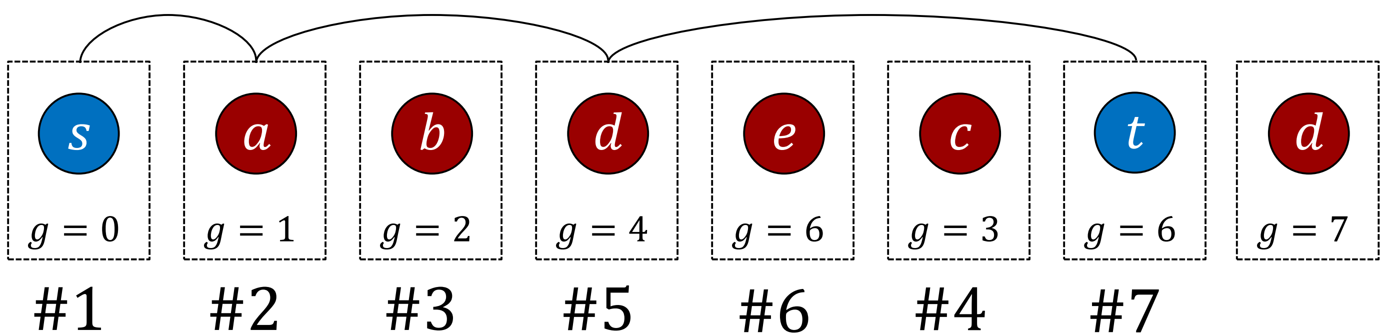
Informed

Strategies that know whether one non-goal is more promising than another

Uniform Cost Search

- **Strategy:** Expand by $g(x)$ = work done so far

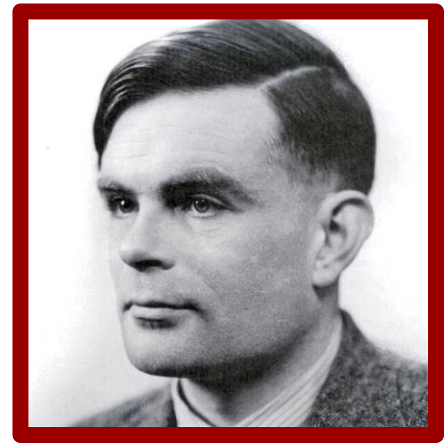




Uninformed vs. Informed

Uninformed

Can only generate successors and distinguish goals from non-goals

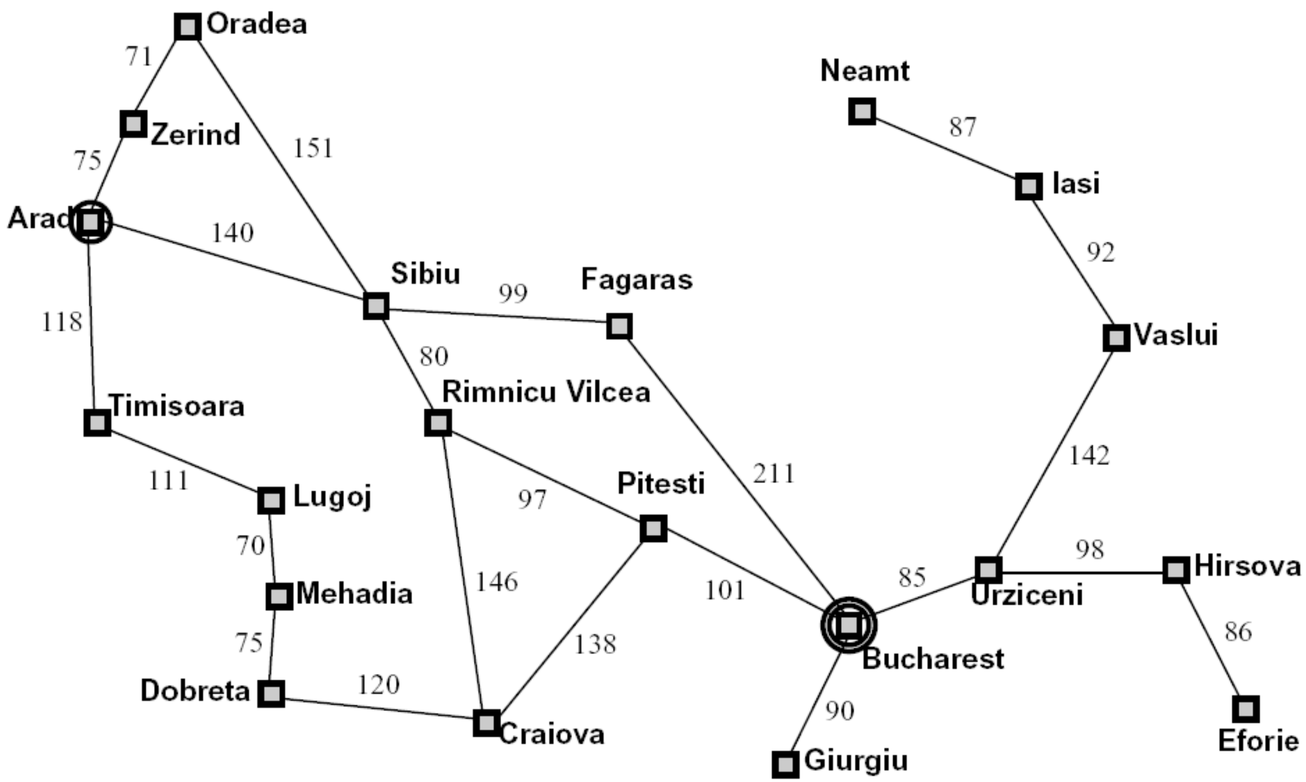


Informed

Strategies that know whether one non-goal is more promising than another

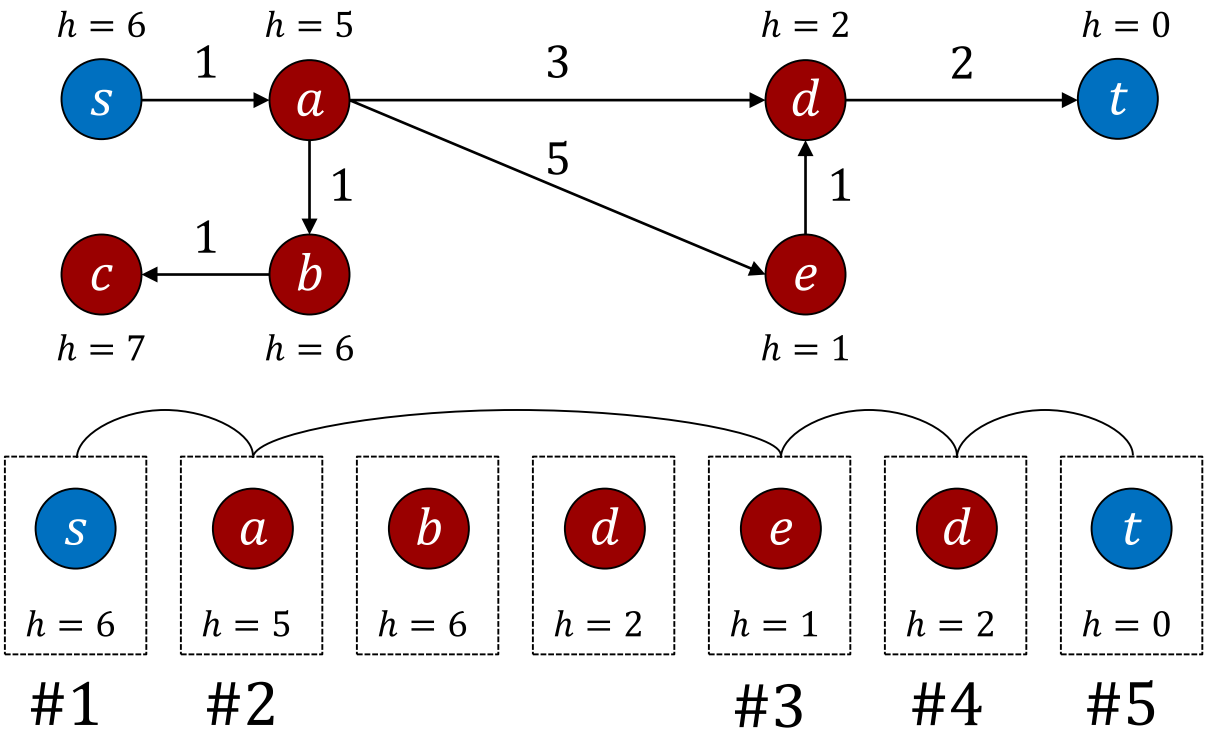
Example: Heuristic

City	Aerial dist
Arad	366
Sibiu	253
Rimnicu Vilcea	193
Fagaras	176
Pitesti	100



Greedy Search

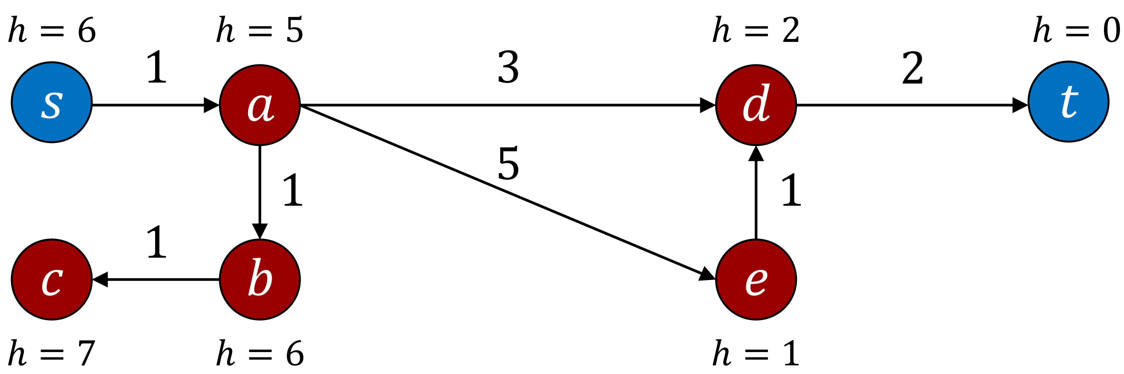
- Strategy: Expand by $h(x)$ = heuristic evaluation of cost from x to goal



A* SEARCH

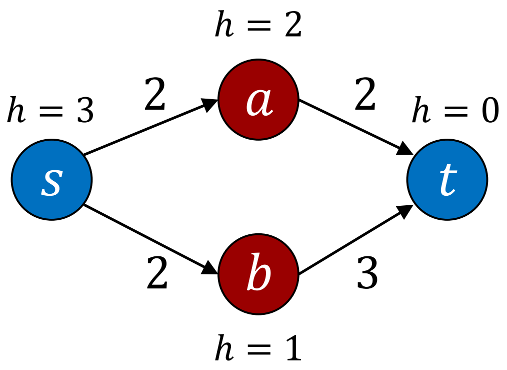
- **Strategy:** Expand by $f(x) = h(x) + g(x)$
- **Poll 1:** Which node is expanded fourth?

1. d
2. e
3. t
4. c



A* Search

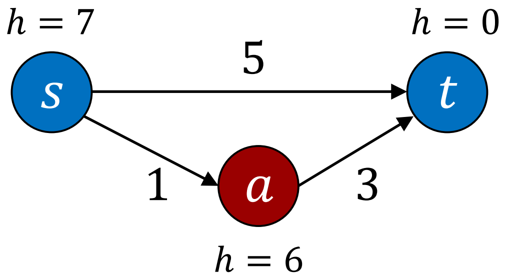
- Should we stop when we discover a goal?



- No: Only stop when we expand a goal

A* Search

- Is A* optimal?



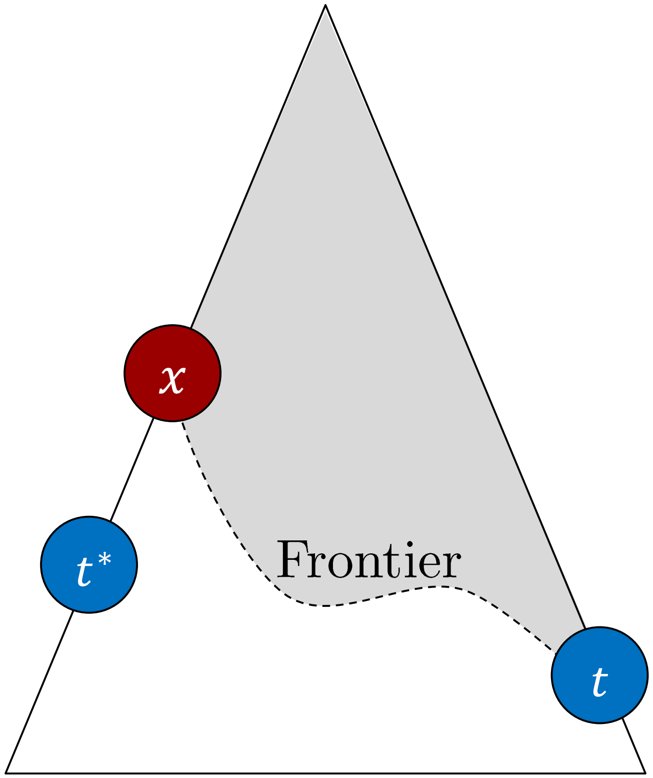
- Good path has pessimistic estimate
- Circumvent this issue by being optimistic!

ADMISSIBLE HEURISTICS

- h is admissible if for all nodes x,
$$h(x) \leq h^*(x),$$
where h^* is the cost of the optimal path to a goal
- Example: Aerial distance in the pathfinding example
- Example: $h \equiv 0$

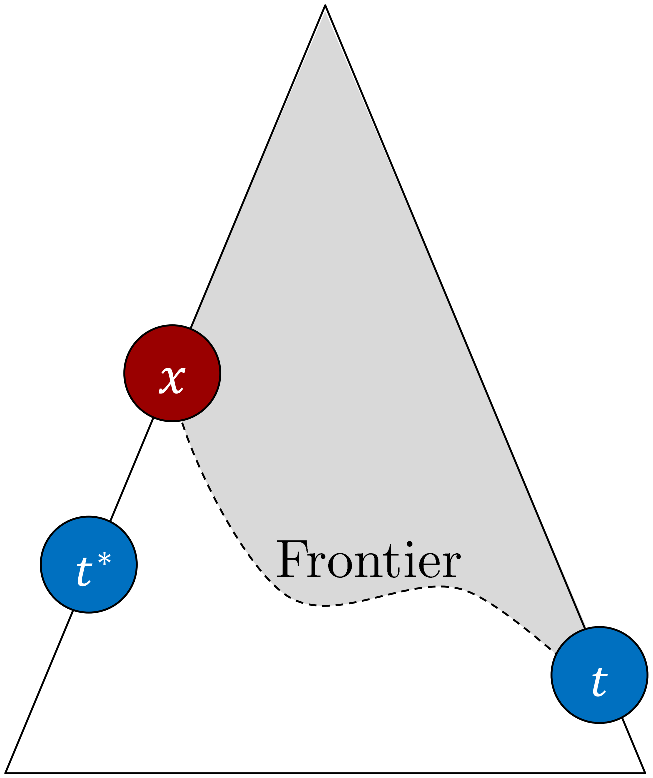
Optimality of A*

- Theorem: A* tree search with an admissible heuristic returns an optimal solution
- Proof:
  - Assume suboptimal goal t is expanded before optimal goal t^*



Optimality of A*

- Proof (cont.):
  - There is a node x on the optimal path to t^* that has been discovered but not expanded
  - $f(x) = g(x) + h(x)$
    $\leq g(x) + h^*(x)$
    $= g(t^*) < g(t) = f(t)$
  - x should have been expanded before t! ■



8-PUZZLE HEURISTICS

- h_1: #tiles in wrong position
- h_2: sum of Manhattan distances of tiles from goal
- Poll 2: Which heuristic is admissible?
  1. Only h_1
  2. Only h_2
  3. Both h_1 and h_2
  4. Neither one

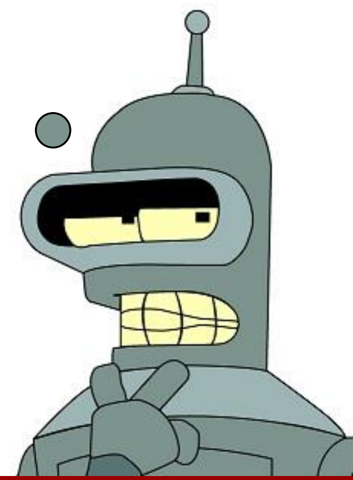
5	2	
6	1	3
7	8	4

Example state

1	2	3
4	5	6
7	8	

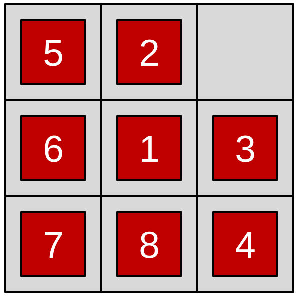
Goal state

Heuristic for designing admissible heuristics: relax the problem!

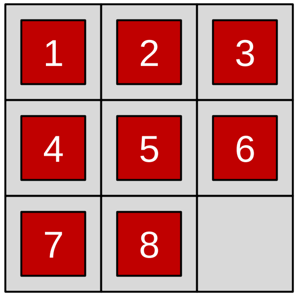


8-puzzle Heuristics

- h_1: #tiles in wrong position
- h_2: sum of Manhattan distances of tiles from goal
- h dominates h' iff $\forall x, h(x) \geq h'(x)$
- Poll 3: What is the dominance relation between h_1 and h_2?
  1. h_1 dominates h_2
  2. h_2 dominates h_1
  3. h_1 and h_2 are incomparable



Example state



Goal state

8-PUZZLE HEURISTICS

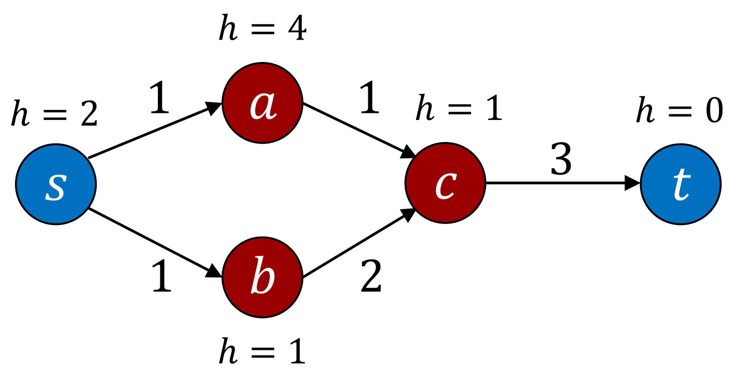
- The following table gives the number of nodes expanded by A* with the two heuristics, averaged over random 8-puzzles, for various solution lengths

Length	$A^*(h_1)$	$A^*(h_2)$
16	1301	211
18	3056	363
20	7276	676
22	18094	1219
24	39135	1641

- Moral: Good heuristics are crucial!

A* Graph Search

- Recall: Graph search is the same as tree search, but never expand a node twice
- Is optimality of A* under admissible heuristics preserved? No!

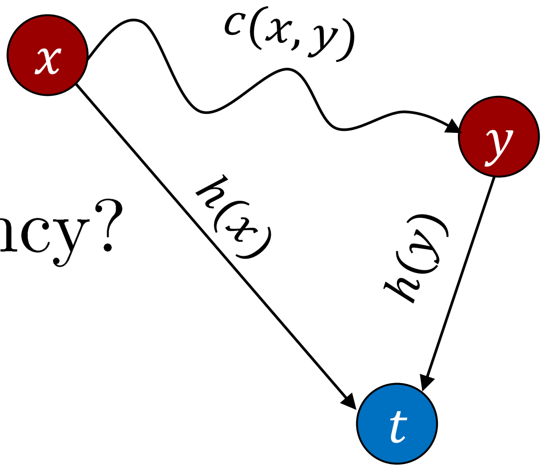


Consistent Heuristics

- $c(x, y)$ = cost of cheapest path between x and y
- h is consistent if for every two nodes x, y,

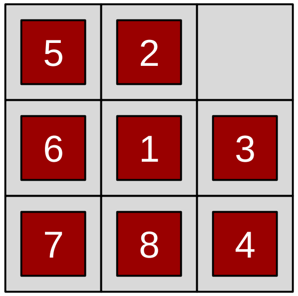
$$h(x) \leq c(x, y) + h(y)$$

- Assume $h(t) = 0$ for each goal t
- Poll 4: What is the relation between admissibility and consistency?
  1. Admissible ⇒ consistent
  2. Consistent ⇒ admissible
  3. They are equivalent
  4. They are incomparable

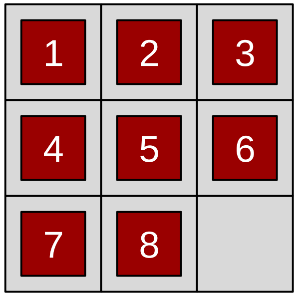


8-puzzle Heuristics, Revisited

- h_1: #tiles in wrong position
- h_2: sum of Manhattan distances of tiles from goal
- Poll 5: Which heuristic is consistent?
  1. Only h_1
  2. Only h_2
  3. Both h_1 and h_2
  4. Neither one

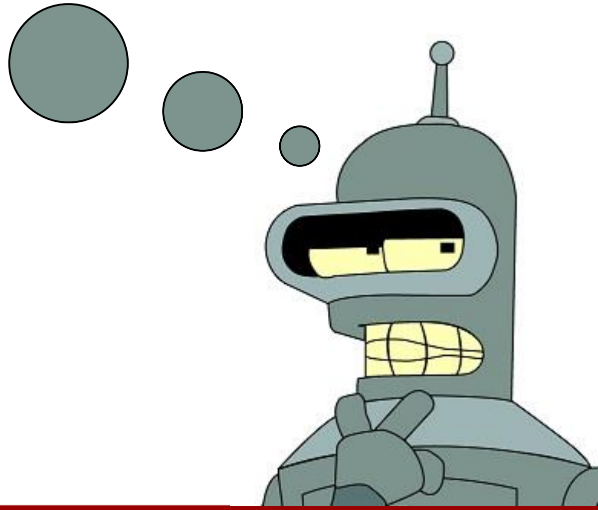


Example state



Goal state

Heuristic for designing consistent heuristics: design an admissible heuristic!



Optimality of A*, Revisited

- Theorem: A* graph search with a consistent heuristic returns an optimal solution
- Proof sketch:*
  - Assume $h(x) \leq c(x,y) + h(y)$
  - Values of $f(x)$ on a path are nondecreasing: if y is the successor of x, $f(x) = g(x) + h(x) \leq g(x) + c(x,y) + h(y) = g(y) + h(y) = f(y)$
  - When A* selects x for expansion, the optimal path to x has been found: otherwise there is a frontier node y on optimal path to x that should be expanded first
  - Nodes expanded in nondecreasing $f(x)$
  - First goal state that is expanded must be optimal ■

* Just for fun

Summary

- Terminology and algorithms:
  - Search problems
  - Tree search, graph search, uniform cost search, greedy, A*
  - Admissible and consistent heuristics
- Theorems:
  - A* tree search is optimal with admissible h
  - A* graph search is optimal with consistent h
- Big ideas:
  - Don't be too pessimistic!