

CE 473: Artificial Intelligence

Spring 2017

A* Search

Dieter Fox

Based on slides from Pieter Abbeel & Dan Klein
Multiple slides from Stuart Russell, Andrew Moore, Luke Zettlemoyer

Today

- A* Search
- Heuristic Design
- Graph search

Example: Pancake Problem

Action: Flip over the top n pancakes



Cost: Number of pancakes flipped

Example: Pancake Problem

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.

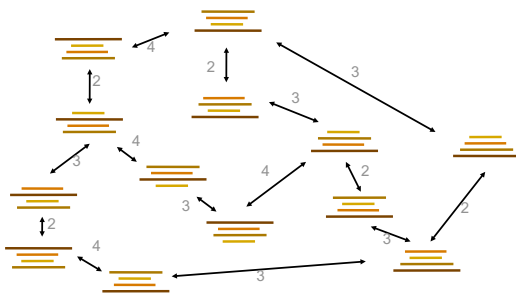
Received 18 January 1978

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 $3n/2 - 1 \leq g(n) \leq 2n + 3$.

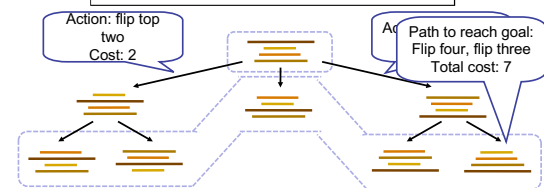
Example: Pancake Problem

State space graph with costs as weights



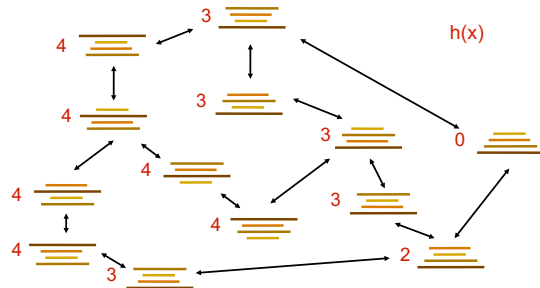
General Tree Search

```
function TREE-SEARCH(problem, strategy) returns a solution, or failure
  initialize the search tree using the initial state of problem
  loop do
    if there are no candidates for expansion then return failure
    choose a leaf node for expansion according to strategy
    if the node contains a goal state then return the corresponding solution
    else expand the node and add the resulting nodes to the search tree
  end
```



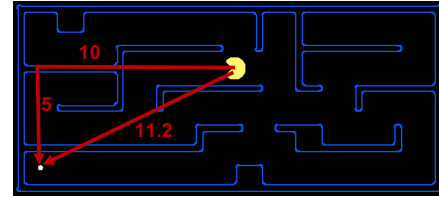
Example: Heuristic Function

Heuristic: the largest pancake that is still out of place



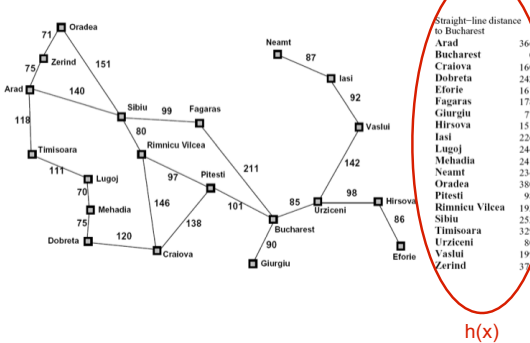
What is a *Heuristic*?

- An *estimate* of how close a state is to a goal
- Designed for a particular search problem



- Examples: Manhattan distance: $10 + 5 = 15$
Euclidean distance: 11.2

Example: Heuristic Function

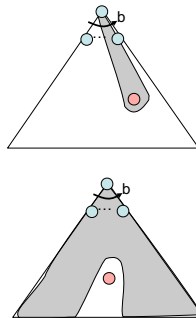


Greedy Search



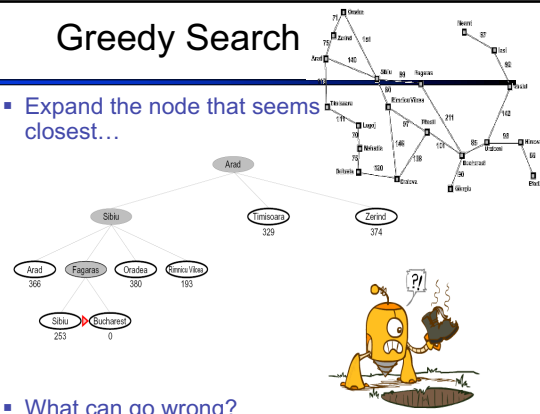
Best First (Greedy)

- Strategy: expand a node that you think is closest to a goal state
 - Heuristic: estimate of distance to nearest goal for each state
- A common case:
 - Best-first takes you straight to the (wrong) goal
- Worst-case: like a badly-guided DFS



Greedy Search

- Expand the node that seems closest...



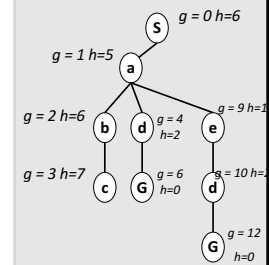
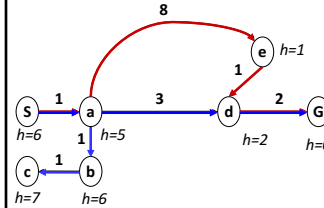
- What can go wrong?

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* $h(n)$

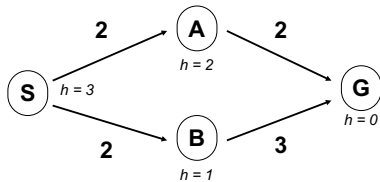


- A* Search orders by the sum: $f(n) = g(n) + h(n)$

Example: Teg Grenager

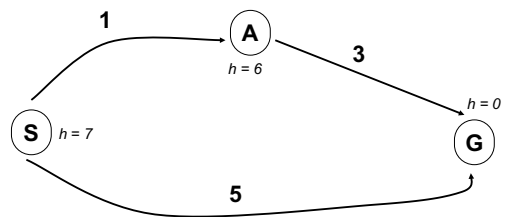
When should A* terminate?

- Should we stop when we enqueue a goal?



- No: only stop when we dequeue a goal

Is A* Optimal?



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

Admissible Heuristics

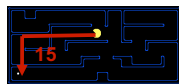
- A heuristic h is *admissible* (optimistic) if:

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

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

- Examples:

4



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

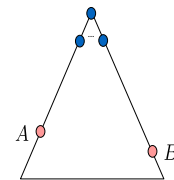
Optimality of A* Tree Search

Assume:

- A is an optimal goal node
- B is a suboptimal goal node
- h is admissible

Claim:

- A will exit the fringe before B



Optimality of A* Tree Search

Proof:

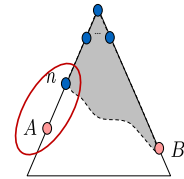
- Imagine B is on the fringe
- Some ancestor n of A is on the fringe, too (maybe A!)
- Claim: n will be expanded before B

- $f(n)$ is less or equal to $f(A)$

$$f(n) = g(n) + h(n) \quad \text{Definition of f-cost}$$

$$f(n) \leq g(A) \quad \text{Admissibility of h}$$

$$g(A) = f(A) \quad h = 0 \text{ at a goal}$$



Optimality of A* Tree Search

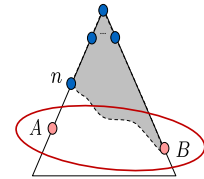
Proof:

- Imagine B is on the fringe
- Some ancestor n of A is on the fringe, too (maybe A!)
- Claim: n will be expanded before B

- $f(n)$ is less or equal to $f(A)$
- $f(A)$ is less than $f(B)$

$$g(A) < g(B) \quad \text{B is suboptimal}$$

$$f(A) < f(B) \quad h = 0 \text{ at a goal}$$



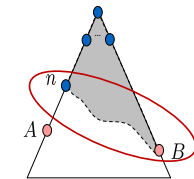
Optimality of A* Tree Search

Proof:

- Imagine B is on the fringe
- Some ancestor n of A is on the fringe, too (maybe A!)
- Claim: n will be expanded before B

- $f(n)$ is less or equal to $f(A)$
- $f(A)$ is less than $f(B)$
- n expands before B

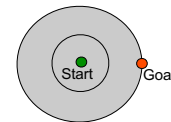
- All ancestors of A expand before B
- A expands before B
- A* search is optimal



$$f(n) \leq f(A) < f(B)$$

UCS vs A* Contours

- Uniform-cost expanded in all directions

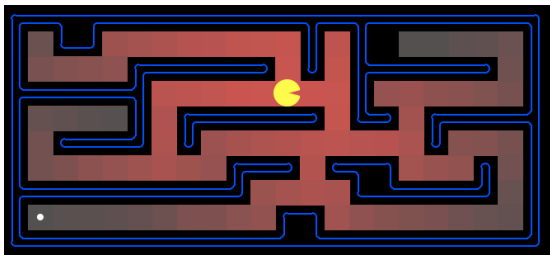


- A* expands mainly toward the goal, but hedges its bets to ensure optimality



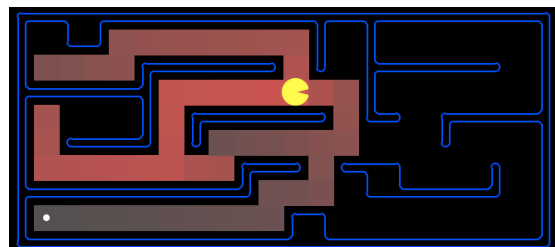
Which Algorithm?

- Uniform cost search (UCS):



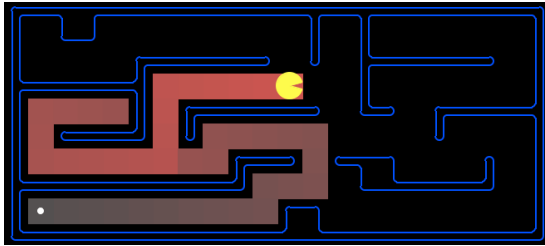
Which Algorithm?

- A*, Manhattan Heuristic:



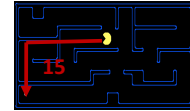
Which Algorithm?

- Best First / Greedy, Manhattan Heuristic:



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



- Inadmissible heuristics are often useful too

Creating Heuristics

8-puzzle:

7	2	4
5		6
8	3	1

Start State

	1	2
3	4	5
6	7	8

Goal State

- What are the states?
- How many states?
- What are the actions?
- What states can I reach from the start state?
- What should the costs be?

8 Puzzle I

- Heuristic: Number of tiles misplaced

7	2	4
5		6
8	3	1

Start State

	1	2
3	4	5
6	7	8

Goal State

- $h(\text{start}) = 8$
- Is it admissible?

	Average nodes expanded when optimal path has length...		
	...4 steps	...8 steps	...12 steps
UCS	112	6,300	3.6×10^6
TILES	13	39	227

8 Puzzle II

- What if we had an easier 8-puzzle where any tile could slide any direction at any time, ignoring other tiles?
- Total *Manhattan distance*
- $h(\text{start}) = 3 + 1 + 2 + \dots$

$$= 18$$

7	2	4
5		6
8	3	1

Start State

	1	2
3	4	5
6	7	8

Goal State

- Admissible?

	Average nodes expanded when optimal path has length...		
	...4 steps	...8 steps	...12 steps
TILES	13	39	227
MANHATTAN	12	25	73

8 Puzzle III

- How about using the *actual cost* as a heuristic?
 - Would it be admissible?
 - Would we save on nodes expanded?
- What's wrong with it?
- With A*: a trade-off between quality of estimate and work per node!

Trivial Heuristics, Dominance

- Dominance: $h_a \geq h_c$ if

$$\forall n : h_a(n) \geq h_c(n)$$

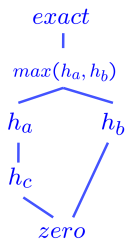
- Heuristics form a semi-lattice:

- Max of admissible heuristics is admissible

$$h(n) = \max(h_a(n), h_b(n))$$

- Trivial heuristics

- Bottom of lattice is the zero heuristic (what does this give us?)
- Top of lattice is the exact heuristic

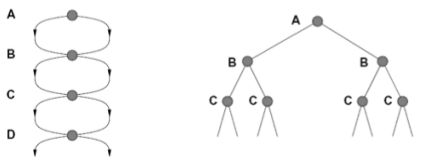


A* Applications

- Pathing / routing problems
- Resource planning problems
- Robot motion planning
- Language analysis
- Machine translation
- Speech recognition
- ...

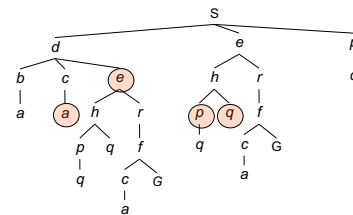
Tree Search: Extra Work!

- Failure to detect repeated states can cause exponentially more work. Why?



Graph Search

- In BFS, for example, we shouldn't bother expanding some nodes (which, and why?)

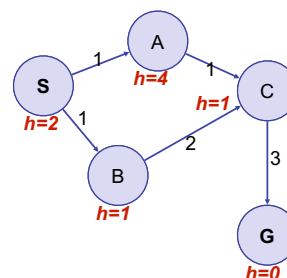


Graph Search

- Idea: never **expand** a state twice
- How to implement:
 - Tree search + set of expanded states ("closed set")
 - Expand the search tree node-by-node, but...
 - Before expanding a node, check to make sure its state has never been expanded before
 - If not new, skip it, if new add to closed set
- Hint: in python, store the closed set as a set, not a list
- Can graph search wreck completeness? Why/why not?
- How about optimality?

A* Graph Search Gone Wrong

State space graph



Search tree

