Announcements

- Projects:
  - Project 1 (Search) is out, due Friday Apr 19th
  - Can talk to each other, but must write own solutions
  - Do the basic search algorithms ASAP!
Today

- A* Search
- Heuristic Design
- Graph search
Recap: Search

- **Search problem:**
  - States (configurations of the world)
  - Successor function: a function from states to lists of (state, action, cost) triples; drawn as a graph
  - Start state and goal test

- **Search tree:**
  - Nodes: represent plans for reaching states
  - Plans have costs (sum of action costs)

- **Search Algorithm:**
  - Systematically builds a search tree
  - Chooses an ordering of the fringe (unexplored nodes)
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 $\sigma$ of the integers from 1 to $n$, let $f(\sigma)$ be the smallest number of prefix reversals that will transform $\sigma$ to the identity permutation, and let $f(n)$ be the largest such $f(\sigma)$ for all $\sigma$ 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
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

Action: flip top two
Cost: 2

Path to reach goal:
Flip four, flip three
Total cost: 7

Monday, April 8, 13
Uniform Cost Search

- **Strategy**: expand lowest path cost
- **The good**: UCS is complete and optimal!
- **The bad**:
  - Explores options in every “direction”
  - No information about goal location
Example: Heuristic Function

Heuristic: the largest pancake that is still out of place
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
Example: Heuristic Function

Straight-line distance to Bucharest:
- Arad: 366
- Bucharest: 0
- Craiova: 160
- Dobrota: 242
- Eforie: 161
- Fagaras: 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

h(x)
Combining UCS and Greedy

- Uniform-cost orders by path cost, or \( \text{backward cost} \) \( f(n) = g(n) \)
- Best-first orders by goal proximity, or \( \text{forward cost} \) \( f(n) = h(n) \)
- 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 goal cost
- We need estimates to be less than actual costs!
Admissible Heuristics

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

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

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

- Examples:

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

Monday, April 8, 13
Optimality of A*: Blocking

Notation:
- $g(n) = \text{cost to node } n$
- $h(n) = \text{estimated cost from } n \text{ to the nearest goal (heuristic)}$
- $f(n) = g(n) + h(n) = \text{estimated total cost via } n$
- $G^*$: a lowest cost goal node
- $G$: another goal node
Optimality of A*: Blocking

Proof:

- What could go wrong?
- We’d have to have to pop a suboptimal goal $G$ off the fringe before $G^*$

This can’t happen:

- For all nodes $n$ on the best path to $G^*$
  - $f(n) < f(G)$
- So, $G^*$ will be popped before $G$

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

$$g(n) + h(n) \leq g(G^*)$$

$$g(G^*) < g(G)$$

$$g(G) = f(G)$$

$f(n) < f(G)$
Properties of A*

Uniform-Cost

A*

Monday, April 8, 13
UCS vs A* Contours

- Uniform-cost expanded in all directions

- A* expands mainly toward the goal, but does hedge its bets to ensure optimality
Uniform cost search (UCS):
Which Algorithm?

- A*, Manhattan Heuristic:
Which Algorithm?

- Best First / Greedy, Manhattan Heuristic:
Creating Heuristics

8-puzzle:

- 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
  - \( h(\text{start}) = 8 \)

- Is it admissible?

<table>
<thead>
<tr>
<th>Average nodes expanded when optimal path has length...</th>
<th>4 steps</th>
<th>8 steps</th>
<th>12 steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>UCS</td>
<td>112</td>
<td>6,300</td>
<td>3.6 x 10^6</td>
</tr>
<tr>
<td>TILES</td>
<td>13</td>
<td>39</td>
<td>227</td>
</tr>
</tbody>
</table>
What if we had an easier 8-puzzle where any tile could slide any direction at any time, ignoring other tiles?

Total Manhattan distance

\[ h(\text{start}) = 3 + 1 + 2 + \ldots \]

\[ = 18 \]

Admissible?

Average nodes expanded when optimal path has length…

<table>
<thead>
<tr>
<th>…4 steps</th>
<th>…8 steps</th>
<th>…12 steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>TILES</td>
<td>13</td>
<td>39</td>
</tr>
<tr>
<td>MANHATTAN</td>
<td>12</td>
<td>25</td>
</tr>
</tbody>
</table>
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!
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 (why?)
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
- ...
Failure to detect repeated states can cause exponentially more work. Why?
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 + list of expanded states (closed list)
  - Expand the search tree node-by-node, but…
  - Before expanding a node, check to make sure its state is new

- Python trick: store the closed list 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

Monday, April 8, 13
Proof:

- **Main idea:** Argue that nodes are popped with non-decreasing f-scores
  - for all n, n' with n' popped after n:
    - \( f(n') \geq f(n) \)
    - is this enough for optimality?

- **Sketch:**
  - **assume:** \( f(n') \geq f(n) \), for all edges \( (n,a,n') \) and all actions a
    - is this true?
  - **proof by induction:** (1) always pop the lowest f-score from the fringe, (2) all new nodes have larger (or equal) scores, (3) add them to the fringe, (4) repeat!
Consistency

- Wait, how do we know parents have better f-values than their successors?

- **Consistency** for all edges \((n, a, n')\):
  - \(h(n) \leq c(n, a, n') + h(n')\)

- **Proof that** \(f(n') \geq f(n)\),
  - \(f(n') = g(n') + h(n') = g(n) + c(n, a, n') + h(n') \geq g(n) + h(n) = f(n)\)
Optimality

- **Tree search:**
  - A* optimal if heuristic is admissible (and non-negative)
  - UCS is a special case (h = 0)

- **Graph search:**
  - A* optimal if heuristic is consistent
  - UCS optimal (h = 0 is consistent)

- Consistency implies admissibility

- In general, natural admissible heuristics tend to be consistent
Summary: A*

- A* uses both backward costs and (estimates of) forward costs
- A* is optimal with admissible heuristics
- Heuristic design is key: often use relaxed problems
To Do:

- Keep up with the readings
- Get started on PS1
  - it is long; start soon
  - due a week from Friday