Questions?

- Epsilon in UCS
- Informed search “feels like” AI
# Uniform Cost Search

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Complete</th>
<th>Optimal</th>
<th>Time</th>
<th>Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFS w/ Path Checking</td>
<td>Y</td>
<td>N</td>
<td>$O(b^{m+1})$</td>
<td>$O(b^m)$</td>
</tr>
<tr>
<td>BFS</td>
<td>Y</td>
<td>N</td>
<td>$O(b^{s+1})$</td>
<td>$O(b^s)$</td>
</tr>
<tr>
<td>UCS</td>
<td>Y*</td>
<td>Y</td>
<td>$O(b^{C*/\epsilon})$</td>
<td>$O(b^{C*/\epsilon})$</td>
</tr>
</tbody>
</table>

* UCS can fail if actions can get arbitrarily cheap
Uniform Cost Issues

- Remember: explores increasing cost contours
- The good: UCS is complete and optimal!
- The bad:
  - Explores options in every “direction”
  - No information about goal location
Search Heuristics

- Any estimate of how close a state is to a goal
- Designed for a particular search problem
- Examples: Manhattan distance, Euclidean distance
Heuristics

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
Best First / Greedy Search

- Expand the node that seems closest...

- What could possibly go wrong?
Best First / Greedy Search

- **A common case:**
  - Best-first takes you straight to the (wrong) goal

- **Worst-case:** like a badly-guided DFS in the worst case
  - Can explore everything
  - Can get stuck in loops if no cycle checking

- **Like DFS in completeness** (finite states w/ cycle checking)
Search Gone Wrong?
Heuristic Design

- Optimal Heuristic for finding goal in pacman maze?
- Heuristic for getting to dinner?
Combining UCS and Greedy

- **Uniform-cost** orders by path cost, or *backward cost* \( g(n) \)
- **Best-first** 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 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.

- Example:

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

Notation:

- $g(n) = \text{cost to node } n$
- $h(n) = \text{estimated cost from } n \text{ to the nearest goal (heuristic)}$
- $f(n) = g(n) + h(n) = \text{estimated total cost via } n$
- $G^* = \text{a lowest cost goal node}$
- $G = \text{another goal node}$
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:
  - Imagine a suboptimal goal $G$ is on the queue
  - Some node $n$ which is a subpath of $G^*$ must also be on the fringe (why?)
  - $n$ 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*
UCS vs A* Contours

- Uniform-cost expanded in all directions

- A* expands mainly toward the goal, but does hedge its bets to ensure optimality
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?)
Example: 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

- Why is it admissible?

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

- This is a relaxed-problem heuristic

### 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>UCS</td>
<td>112</td>
<td>6,300</td>
</tr>
<tr>
<td>TILES</td>
<td>13</td>
<td>39</td>
</tr>
</tbody>
</table>
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
- Why admissible?
- \( h(\text{start}) = 3 + 1 + 2 + \ldots \)
  \[ = 18 \]

![Start State]

![Goal State]

<table>
<thead>
<tr>
<th></th>
<th>7</th>
<th>2</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>

Average nodes expanded when optimal path has length...

<table>
<thead>
<tr>
<th></th>
<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>
<td>227</td>
</tr>
<tr>
<td>MANHATTAN</td>
<td>12</td>
<td>25</td>
<td>73</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!
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
Other A* Applications

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