Announcements

- **Homework 1: Search**
  - On edX - online, instant grading, submit as often as you like.

- **Project 1: Search**
  - Start early and ask questions. It’s longer than most!
CS 188: Artificial Intelligence

Informed Search

Instructor: Marco Alvarez

University of Rhode Island

[These slides were created by Dan Klein and Pieter Abbeel for CS188 Intro to AI at UC Berkeley (ai.berkeley.edu)]
Today

- Informed Search
  - Heuristics
  - Greedy Search
  - A* Search
Recap: Search

- **Search problem:**
  - States (configurations of the world)
  - Actions and costs
  - Successor function (world dynamics)
  - 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)
  - Optimal: finds least-cost plans
Uninformed Search
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

[Demo: contours UCS empty (L3D1)]
[Demo: contours UCS pacman small maze (L3D3)]
Video of Demo Contours UCS Empty
Video of Demo Contours UCS Empty
Video of Demo Contours UCS Empty
Video of Demo Contours UCS Pacman Small Maze

SCORE: 0
Video of Demo Contours UCS Pacman Small Maze
Video of Demo Contours UCS Pacman Small Maze

SCORE: 0
Informed Search
Search Heuristics
Search Heuristics

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

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

- A heuristic is:
  - A function that estimates how close a state is to a goal
  - Designed for a particular search problem
A heuristic is:
- A function that estimates how close a state is to a goal
- Designed for a particular search problem
A heuristic is:
- A function that *estimates* how close a state is to a goal
- Designed for a particular search problem
- Examples: Manhattan distance, Euclidean distance for pathing
Search Heuristics

- A heuristic is:
  - A function that \textit{estimates} how close a state is to a goal
  - Designed for a particular search problem
  - Examples: Manhattan distance, Euclidean distance for pathing
A heuristic is:

- A function that *estimates* how close a state is to a goal
- Designed for a particular search problem
- Examples: Manhattan distance, Euclidean distance for pathing
Example: Heuristic Function

$h(x)$
Effect of heuristics

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

- **Informed**
- **Uninformed**
Greedy Search
Greedy Search
Greedy Search

- Expand the node that seems closest...(order frontier by \( h \))
- What can possibly go wrong?
Greedy Search

- Expand the node that seems closest...(order frontier by h)
- What can possibly go wrong?
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

- Expand the node that seems closest...(order frontier by h)
- What can possibly go wrong?

\[ \text{Sibiu-Fagaras-Bucharest} = 99 + 211 = 310 \]
\[ \text{Sibiu-Rimnicu Vilcea-Pitesti-Bucharest} = 80 + 97 + 101 = 278 \]
Greedy Search

- **Strategy:** expand a node that *seems* closest to a goal state
  - Heuristic: estimate of distance to nearest goal for each state

[Demo: contours greedy empty (L3D1)]
[Demo: contours greedy pacman small maze (L3D4)]
Greedy Search

- **Strategy**: expand a node that *seems* 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

[Demo: contours greedy empty (L3D1)]
[Demo: contours greedy pacman small maze (L3D4)]
Greedy Search

- **Strategy:** expand a node that *seems* 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

[Demo: contours greedy empty (L3D1)]
[Demo: contours greedy pacman small maze (L3D4)]
Greedy Search

- **Strategy:** expand a node that **seems** 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

[Demo: contours greedy empty (L3D1)]
[Demo: contours greedy pacman small maze (L3D4)]
Video of Demo Contours Greedy (Empty)
Video of Demo Contours Greedy (Empty)
Video of Demo Contours Greedy (Empty)
Video of Demo Contours Greedy (Pacman Small Maze)
Video of Demo Contours Greedy (Pacman Small Maze)
Video of Demo Contours Greedy (Pacman Small Maze)
A* Search
A* Search
A* Search
A* Search

UCS
A* Search

UCS

Greedy
A* Search

UCS

Greedy

A*
Combining UCS and Greedy

Example: Teg Grenager
Combining UCS and Greedy

- **Uniform-cost** orders by path cost, or *backward cost* $g(n)$

---

**Example:** Teg Grenager
Combining UCS and Greedy

- Uniform-cost orders by path cost, or \textit{backward cost} $g(n)$
Combining UCS and Greedy

- **Uniform-cost** orders by path cost, or **backward cost**  $g(n)$
Combining UCS and Greedy

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

Example: Teg Grenager
Combining UCS and Greedy

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

Example: Teg Grenager
Combining UCS and Greedy

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

Example: Teg Grenager
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
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?
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?
Is A* Optimal?

What went wrong?
- Actual bad goal cost < estimated good goal cost
Is A* Optimal?

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

Inadmissible (pessimistic) heuristics break optimality by trapping good plans on the fringe

Admissible (optimistic) heuristics slow down bad plans but never outweigh true costs
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
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

- Example:
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

Example:
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.

**Example:**

![Image of a maze with a path highlighted]

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

[Demo: contours UCS / greedy / A* empty (L3D1)]
[Demo: contours A* pacman small maze (L3D5)]
Video of Demo Contours (Empty) -- UCS
Video of Demo Contours (Empty) -- UCS
Video of Demo Contours (Empty) -- UCS
Video of Demo Contours (Empty) -- Greedy
Video of Demo Contours (Empty) -- Greedy
Video of Demo Contours (Empty) -- Greedy
Video of Demo Contours (Empty) - A*
Video of Demo Contours (Empty) - A*
Video of Demo Contours (Empty) - A*
Video of Demo Contours (Pacman Small Maze) - A*
Video of Demo Contours (Pacman Small Maze) - A*
Video of Demo Contours (Pacman Small Maze) - A*
Comparison

Greedy  Uniform Cost  A*
Video of Demo Pacman (Tiny Maze) - UCS / A*
Video of Demo Pacman (Tiny Maze) - UCS / A*
Video of Demo Empty Water Shallow/Deep - Guess Algorithm
Video of Demo Empty Water Shallow/Deep - Guess Algorithm
Creating Heuristics
Creating Admissible Heuristics

- Most of the work in solving hard search problems optimally is in coming up with admissible heuristics.
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.
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.
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.
Example: 8 Puzzle

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

- Heuristic: Number of tiles misplaced

![Start State](image1)

![Goal State](image2)
8 Puzzle I

- Heuristic: Number of tiles misplaced
- Why is it admissible?
8 Puzzle I

- Heuristic: Number of tiles misplaced
- Why is it admissible?
- $h(\text{start}) =$
8 Puzzle I

- Heuristic: Number of tiles misplaced
- Why is it admissible?
- \( h(\text{start}) = 8 \)
8 Puzzle I

- Heuristic: Number of tiles misplaced
- Why is it admissible?
- \( h(\text{start}) = 8 \)

Start State

Goal State

<table>
<thead>
<tr>
<th>Average nodes expanded when the optimal path has...</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>

Statistics from Andrew Moore
8 Puzzle I

- Heuristic: Number of tiles misplaced
- Why is it admissible?
- $h(\text{start}) = 8$
- This is a relaxed-problem heuristic

![Start State and Goal State](image)

<table>
<thead>
<tr>
<th>Average nodes expanded when the optimal path has...</th>
</tr>
</thead>
<tbody>
<tr>
<td>...4 steps</td>
</tr>
<tr>
<td>UCS</td>
</tr>
<tr>
<td>TILES</td>
</tr>
</tbody>
</table>

Statistics from Andrew Moore
8 Puzzle

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

Start State

Goal State

<table>
<thead>
<tr>
<th>Average nodes expanded when the optimal path has...</th>
</tr>
</thead>
<tbody>
<tr>
<td>...4 steps</td>
</tr>
<tr>
<td>------------</td>
</tr>
<tr>
<td>UCS</td>
</tr>
<tr>
<td>TILES</td>
</tr>
</tbody>
</table>

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

Total *Manhattan* distance
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 is it admissible?
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 is it admissible?

- \( h(\text{start}) = \)
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 is it admissible?

- $h(\text{start}) = 3 + 1 + 2 + \ldots = 18$
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 is it admissible?

- \( h(\text{start}) = 3 + 1 + 2 + \ldots = 18 \)

### Average nodes expanded when the optimal path has...

<table>
<thead>
<tr>
<th></th>
<th>...4 steps</th>
<th>...8 steps</th>
<th>...12 steps</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TILES</strong></td>
<td>13</td>
<td>39</td>
<td>227</td>
</tr>
<tr>
<td><strong>MANHATTAN</strong></td>
<td>12</td>
<td>25</td>
<td>73</td>
</tr>
</tbody>
</table>
How about using the *actual cost* as a heuristic?
- Would it be admissible?
- Would we save on nodes expanded?
- What’s wrong with it?
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
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
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
- As heuristics get closer to the true cost, you will expand fewer nodes but usually do more work per node to compute the heuristic itself