Agenda

- Administrivia
- What is this course all about?
- Simple problems and search
Course Website

- www.cim.mcgill.ca/~jer/courses/ai
  - course outline
  - readings (mostly from Russell & Norvig)
  - assignments
  - other course resources
Office Hours

- Wednesday 11:00-12:00 in MC 424
- text-based electronic communication is a poor substitute for face-to-face or video-mediated communication -- if you have course-related questions, please attend office hours
- however, on-line discussion encouraged …
On-line Class Communication

- we’ll be using Piazza, which supports:
  - group discussions (bulletin board)
  - collaborative (Wiki-style) Q&A
  - private communications (use this, not email, to reach me)
Evaluation

- **Assignments**: 45% of grade
  - 3 assignments, to be done individually
- **Project**: 25% of grade
  - Can be done in groups of 2-3
- **Final Exam**: 30%
Class Format

- Each class will consist of:
  - a brief review of the assigned readings
  - exercises and discussion to reinforce the concepts
  - introduction to assigned readings for next class
Exercise

- List 5 tasks that you consider to demonstrate “intelligence”
- How many of these can be done today by computer?
- How many can be done by a computer as well as or better than most humans?
What is AI?
The “Hollywood” view
What is AI?
Machines that can beat humans at intelligent tasks
What is AI?
Systems that perform complex human tasks
What is AI?

Systems that learn from and interact with humans
What is AI?
Philip K. Dick, Android
What is this course about?

- **AI fundamentals**: search, game-playing, planning
- **Machine Learning**: e.g., decision trees, temporal difference methods, induction
- **Connectionist and Evolutionary Computing**: neural networks, support vector machines, genetic algorithms
Let’s get started…
Problem Formulation

- **states**: description of “world of interest”
- **initial state**
- **successor function**: generates set of legal next states from available actions
- **goal test**: how do we know we’re done?
- **path cost**: way of choosing between multiple solutions (e.g., shortest route)
Vacuum World Problem

• **successor function:**
  • move left (L), move right (R), suck (S)

• **goal test:**
  • no dirt left in any square

• **path cost:**
  • each step costs 1
Exercise

Starting from states 1 and 2, generate the state tree, showing all possible actions and the associated successor states.

(don’t redraw existing states; use loops to indicate “no change”)
Solution

• Goal: no dirt left in any square
• Operators: move left, move right, suck
Missionaries & Cannibals

- 3 missionaries and 3 cannibals need to cross crocodile-infested river
- boat can hold 1 or 2 people
- can’t leave any missionaries outnumbered by cannibals
Missionaries & Cannibals: Formulation

• states:
• initial state:
• successor function
• goal test:
• path cost:
Missionaries & Cannibals: Formulation

• **states**: (# missionaries, # cannibals, # of boats) on left bank of river
• **initial state**: (3,3,1)
• **successor function**: move (# missionaries, # cannibals) from one bank to other
• **goal test**: (0,0,0)
• **path cost**: # of river crossings
8-Queens problem

- arrange 8 queens on a chessboard so that no two queens are on the same row, column or diagonal (i.e., attack each other)
- applications to parallel memory storage, VLSI testing, traffic control, and deadlock prevention
Naïve approach

- **state**: any arrangement of [0,8] queens on the board
- **successor function**: add a queen to any empty square

State space: $3 \times 10^{14}$
Better approach

- **state**: any arrangement of \( n=[0,8] \) queens, one per column in the leftmost \( n \) columns, with no queen attacking another
- **successor function**: add a queen to any square in the leftmost empty column such that it is not attacked by any other queen

State space: 2057
Search Methods

• use to explore state space for solution to a problem
• can be uninformed (blind) or use some reasonable knowledge (heuristics) to guide search
Uninformed Search

- **breadth-first**
  - expand shallowest nodes first (FIFO)
- **depth-first**
  - expand deepest nodes first (LIFO)
- **depth-limited search**
  - depth-first with cutoff
- **iterative-deepening**
  - combines benefits of BFS and DFS
- **bidirectional**
  - applicable when operators are reversible
Breadth-first search

- Expand *shallowest* unexpanded node
- Put successors at end of FIFO queue
Exercise:
cost to locate ‘K’ and ‘U’ using BF search
# Breadth-first search

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complete?</td>
<td>Yes (if b is finite)</td>
</tr>
<tr>
<td>Time complexity</td>
<td>$1 + b + b^2 + b^3 + \ldots + b^d = O(b^d)$</td>
</tr>
<tr>
<td>Space complexity</td>
<td>$O(b^d)$ (every node kept in memory)</td>
</tr>
<tr>
<td>Optimal?</td>
<td>Yes (if cost = 1 per step)</td>
</tr>
</tbody>
</table>

- **b**: maximum branching factor of search tree
- **d**: depth of the least cost solution

Exponential time/memory requirements make breadth-first search unsuitable for large problems.
Depth-first search

- Expand **deepest** unexpanded node
- Put successors at front of LIFO queue
Exercise:
cost to locate ‘K’ and ‘U’ using DF search
# Depth-first search

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complete?</td>
<td>No (fails in infinite-depth spaces or spaces with loops)</td>
</tr>
<tr>
<td>Time complexity</td>
<td>$O(b^m)$ (bad if $m \gg d$)</td>
</tr>
<tr>
<td>Space complexity</td>
<td>$O(bm)$ (linear in space)</td>
</tr>
<tr>
<td>Optimal?</td>
<td>No</td>
</tr>
</tbody>
</table>

b: maximum branching factor of search tree  
d: depth of the least cost solution  
m: maximum depth of state space
How to get best of both worlds?

- i.e., how to combine completeness of breadth first & space complexity of depth-first search?
- start with depth-limited search
  - solves the infinite depth problem
Depth-limited search

- depth-first search with depth limit $\ell$

Complete? only if $\ell > d$

Time complexity $O(b^\ell)$

Space complexity $O(b\ell)$

Optimal? only if $\ell = d$
Iterative-deepening search

- use depth-limited search as subroutine with increasing $l$
- is this efficient?

<table>
<thead>
<tr>
<th>Complete?</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time complexity</td>
<td>$d + (d-1)b + (d-2)b^2 + \ldots + b^d = O(b^d)$</td>
</tr>
<tr>
<td>Space complexity</td>
<td>$O(bd)$</td>
</tr>
<tr>
<td>Optimal?</td>
<td>Yes (if cost = 1 per step)</td>
</tr>
</tbody>
</table>
8-squares problem
What’s a good state description and successor function?

initial state

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

goal state

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

- minimize estimated cost to goal, $h(n)$
- start by expanding minimal cost node
Informed Search: A* search

- minimize estimated cost to goal: \( f(n) = g(n) + h(n) \)
  - \( g(n) \) = cost of solution from start to \( n \) and \( h(n) \) is estimated cost of cheapest solution from \( n \) to goal
  - \( h(n) \) is admissible if it never over-estimates true cost to reach goal
- \( A^* \) uses a best-first search: chooses least-cost path from initial state to goal state
Optimality of A* search

• If h(n) is admissible (valid), A* is optimal:
  • no optimal algorithm employing the same heuristic will expand fewer nodes than A*

• Exercise: why?
Exercise

• for the 8-squares problem, which of the following are valid (admissible) heuristics?
  \[ h(n) = \text{number of displaced tiles} \quad \text{or} \quad h(n) = \text{sum of Manhattan distances of displaced tiles} \]

• which heuristic is better?
Exercise

• for the 8-squares problem, which of the following are valid (admissible) heuristics?
  h(n) = number of displaced tiles or
  h(n) = sum of Manhattan distances of the displaced tiles

  • ANSWER: BOTH
    h(n) is valid if it never overestimates actual cost to reach goal

• which heuristic is better?

  • ANSWER: Second one
    it provides a higher lower bound on the estimate
Homework: the 8-squares problem

Generate the first two levels of the state space for this problem by drawing a labelled state tree, using the Manhattan distance heuristic to assign an A* value to each node. What are the first three moves you would make?
Recap

- Office hours: Wednesday 11:00-12:00 in MC 424
- Piazza for course communication
- Course website: www.cim.mcgill.ca/~jer/courses/ai
- Considered bases of AI: systems that think/act like a human or think/act rationally
- Looked at how to formulate a problem in AI terms
Readings

- From course page:
  www.cim.mcgill.ca/~jer/courses/ai
  - Russell & Norvig Ch. 1.1, 3, 5-5.4
Next Class Agenda

- Two-player games
- e.g., NIM
  - last-player to pick-up stick loses)