Solving Problems by Searching
Terminology

- State
- State Space
- Goal
- Action
- Cost
- State Change Function
- Problem-Solving Agent
- State-Space Search
Formal State-Space Model

Problem = \((S, s, A, f, g, c)\)

- \(S = \text{state space}\)
- \(s = \text{initial state}\)
- \(A = \text{actions}\)
- \(f = \text{state change function} \quad f: S \times A \rightarrow S\)
- \(g = \text{goal test function} \quad g: S \rightarrow \{\text{true, false}\}\)
- \(c = \text{cost function} \quad c: S \times A \times S \rightarrow R\)

- How do we define a solution?
- How about an optimal solution?
3 Coins Problem
A Very Small State Space Problem

• There are 3 (distinct) coins: coin1, coin2, coin3.

• The initial state is H H T

• The legal operations are to turn over exactly one coin.
  – 1 (flip coin1), 2 (flip coin2), 3 (flip coin3)

• There are two goal states: H H H
  T T T

What are S, s, A, f, g, c?
• What are some solutions?
• What if the problem is changed to allow only 3 actions?
Modified State-Space Problem

• How would you define a state for the new problem?

• How do you define the operations (1, 2, 3) with this new state definition?

• What do the paths to the goal states look like now?
How do we build a search tree for the modified 3 coins problem?
The 8-Puzzle Problem

1. Formalize a state as a data structure
2. Show how start and goal states are represented.
3. How many possible states are there?
4. How would you specify the state-change function?
5. What is the goal test?
6. What is the path cost function?
7. What is the complexity of the search?

one
initial
state

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

goal
state

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

B=blank
Search Tree Example:
Fragment of 8-Puzzle Problem Space
Another Example: N Queens

- **Input:**
  - Set of states
  - Operators [and costs]
  - Start state
  - Goal state (test)

- **Output**
Example: Route Planning

- Input:
  - Set of states
  - Operators [and costs]
  - Start state
  - Goal state (test)

- Output:
Search Strategies

• Blind Search (Ch 3)
  – Depth first search
  – Breadth first search
  – Depth limited search
  – Iterative deepening search

• Informed Search (Ch 4)

• Constraint Satisfaction (Ch 5)
Depth First Search

- Maintain stack of nodes to visit
- Evaluation
  - Complete?
  - Time Complexity?
    \( O(b^d) \)
  - Space?
    \( O(d) \)
Breadth First Search

- Maintain queue of nodes to visit
- Evaluation
  - Complete? Yes
  - Time Complexity? $O(b^d)$
  - Space? $O(b^d)$

The Missionaries and Cannibals Problem  
(from text problem 3.9)

• Three missionaries and three cannibals are on one side of a river, along with a boat that can hold one or two people.

• If there are ever more cannibals than missionaries on one side of the river, the cannibals will eat the missionaries. (We call this a “dead” state.)

• Find a way to get everyone to the other side, without anyone getting eaten.
Missionaries and Cannibals Problem

The chief said to the first missionary...

Death? or Bunga-Bunga?

And the missionary said...

Well, I guess nothing's worse than death. I'll take Bunga-Bunga.
Missionaries and Cannibals Problem

---

Left Bank

Right Bank

River

M M M

C C C

M M M M C C C

---

17
Missionary and Cannibals Notes

• Define your state as (M,C,S)
  – M: number of missionaries on left bank
  – C: number of cannibals on left bank
  – S: side of the river that the boat is on

• When the boat is moving, we are in between states. When it arrives, everyone gets out.
When is a state considered “DEAD”?

1. There are more cannibals than missionaries on the left bank. (Bunga-Bunga)

2. There are more cannibals than missionaries on the right bank. (Bunga-Bunga)

3. There is an ancestor state of this state that is exactly the same as this state. (Why?)
Assignment

• Implement and solve the problem with a depth-first search using a stack and/or recursion.
  – Find and print all 4 solutions. (See web page.)
  – Keep track of the total number of states searched.
  – When you get to a dead state, count it and then back up to its parent.

• You may use the computer language of your choice for this assignment.
  – Java
  – C++
Is memory a limitation in search?

• Suppose:
  • 2 GHz CPU
  • 1 GB main memory
  • 100 instructions / expansion
  • 5 bytes / node

• 200,000 expansions / sec
• Memory filled in 100 sec ... < 2 minutes
Iterative Deepening Search

• DFS with limit; incrementally grow limit

• Evaluation
  – Complete?
    Yes
  – Time Complexity?
    $O(b^d)$
  – Space Complexity?
    $O(d)$
Cost of Iterative Deepening

<table>
<thead>
<tr>
<th>b</th>
<th>ratio ID to DFS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>1.5</td>
</tr>
<tr>
<td>10</td>
<td>1.2</td>
</tr>
<tr>
<td>25</td>
<td>1.08</td>
</tr>
<tr>
<td>100</td>
<td>1.02</td>
</tr>
</tbody>
</table>
vs. Bidirectional
Problem

• All these methods are too slow for real applications (blind)

• Solution → add guidance
  
  → “informed search”