Three missionaries and three cannibals wish to cross the river. They have a small boat that will carry up to two people. Everyone can navigate the boat. If at any time the Cannibals outnumber the Missionaries on either bank of the river, they will eat the Missionaries. Find the smallest number of crossings that will allow everyone to cross the river safely.

What is the “state” of this problem (it should capture all possible valid configurations)?
Missionaries and Cannibals

Three missionaries and three cannibals wish to cross the river. They have a small boat that will carry up to two people. Everyone can navigate the boat. If at any time the Cannibals outnumber the Missionaries on either bank of the river, they will eat the Missionaries. Find the smallest number of crossings that will allow everyone to cross the river safely.

<table>
<thead>
<tr>
<th>MMMCCC B</th>
<th>MMCC</th>
<th>MC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>B MC</td>
<td>B MMCC</td>
</tr>
</tbody>
</table>

Searching for a solution

What states can we get to from here?

MMMCCC B

Code!

http://www.cs.pomona.edu/~dkauchak/classes/c30/examples/cannibals.txt
Missionaries and Cannibals Solution

<table>
<thead>
<tr>
<th>Step</th>
<th>Missionaries</th>
<th>Cannibals</th>
<th>Near side</th>
<th>Far side</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>MMMCCC</td>
<td>B</td>
<td></td>
<td>B</td>
</tr>
<tr>
<td>1</td>
<td>Two cannibals cross over:</td>
<td>MMMC</td>
<td>B</td>
<td>CC</td>
</tr>
<tr>
<td>2</td>
<td>One comes back:</td>
<td>MMNC</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>3</td>
<td>Two cannibals go over again:</td>
<td>MMH</td>
<td>B</td>
<td>CCC</td>
</tr>
<tr>
<td>4</td>
<td>One comes back:</td>
<td>MMHC</td>
<td>B</td>
<td>CC</td>
</tr>
<tr>
<td>5</td>
<td>Two missionaries cross:</td>
<td>MC</td>
<td>B</td>
<td>MMCC</td>
</tr>
<tr>
<td>6</td>
<td>A missionary &amp; cannibal return:</td>
<td>MMCC</td>
<td>B</td>
<td>MC</td>
</tr>
<tr>
<td>7</td>
<td>Two missionaries cross again:</td>
<td>CC</td>
<td>B</td>
<td>MMMC</td>
</tr>
<tr>
<td>8</td>
<td>A cannibal returns:</td>
<td>CCC</td>
<td>B</td>
<td>MMMC</td>
</tr>
<tr>
<td>9</td>
<td>Two cannibals cross:</td>
<td>C</td>
<td>B</td>
<td>MMMCC</td>
</tr>
<tr>
<td>10</td>
<td>One returns:</td>
<td>CC</td>
<td>B</td>
<td>MMMC</td>
</tr>
<tr>
<td>11</td>
<td>And brings over the third:</td>
<td>B</td>
<td>B</td>
<td>MMMCCC</td>
</tr>
</tbody>
</table>

How is this solution different than the n-queens problem?

One other problem

What would happen if we ran DFS here?

One other problem

If we always go left first, will continue forever!
One other problem

DFS vs. BFS

Why do we use DFS then, and not BFS?

Consider a search problem where each state has two states you can reach.
Assume the goal state involves 20 actions, i.e. moving between ~20 states.

How big can the queue get for BFS?

At any point, need to remember roughly a "row".
Consider a search problem where each state has two states you can reach. Assume the goal state involves 20 actions, i.e. moving between ~20 states.

How big does this get?

Doubles every level we have to go deeper. For 20 actions that is $2^{20} = \approx 1$ million states!

How many states would DFS keep on the stack?

Only one path through the tree, roughly 20 states.
One other problem

If we always go left first, will continue forever!

Solution?

DFS avoiding repeats

```python
def dfs(state, visited):
    # note that we've visited this state
    visited[str(state)] = True
    if state.is_goal():
        return [state]
    else:
        result = []
        for s in state.next_states():
            # check if we've visited a state already
            if not(s in visited):
                result += dfs(s, visited)
        return result
```

Other search problems

What problems have you seen that could be posed as search problems?

What is the state?

Start state

Goal state

State-space/transition between states

8-puzzle

Start State

Goal State
8-puzzle

goal

state representation?

start state?

state-space/transitions?

8-puzzle

state:
- all 3 x 3 configurations of the tiles on the board

transitions between states:
- Move Blank Square Left, Right, Up or Down.
- This is a more efficient encoding than moving each of the 8 distinct tiles

Cryptarithmetic

Find an assignment of digits (0, ..., 9) to letters so that a given arithmetic expression is true.
examples:

\[
\begin{align*}
\text{SEND} + \text{MORE} &= \text{MONEY} \\
\text{FORTY} + \text{TEN} &= \text{SIXTY}
\end{align*}
\]

F=2, O=9, R=7, etc.
Remove 5 Sticks

Given the following configuration of sticks, remove exactly 5 sticks in such a way that the remaining configuration forms exactly 3 squares.

Water Jug Problem

Given a full 5-gallon jug and a full 2-gallon jug, fill the 2-gallon jug with exactly one gallon of water.

Water Jug Problem

State = \((x,y)\), where \(x\) is the number of gallons of water in the 5-gallon jug and \(y\) is the number of gallons in the 2-gallon jug.

Initial State = \((5,2)\)

Goal State = \((*,1)\), where * means any amount.

Operator table

<table>
<thead>
<tr>
<th>Name</th>
<th>Cond.</th>
<th>Transition</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Empty5</td>
<td>–</td>
<td>((x,y))→(0,y)</td>
<td>Empty 5-gal. jug</td>
</tr>
<tr>
<td>Empty2</td>
<td>–</td>
<td>((x,y))→(x,0)</td>
<td>Empty 2-gal. jug</td>
</tr>
<tr>
<td>2to5</td>
<td>(x \leq 3)</td>
<td>((x,2))→(x+2,0)</td>
<td>Pour 2-gal. into 5-gal.</td>
</tr>
<tr>
<td>5to2</td>
<td>(x \geq 2)</td>
<td>((x,0))→(x-2,2)</td>
<td>Pour 5-gal. into 2-gal.</td>
</tr>
<tr>
<td>5to2part</td>
<td>(y &lt; 2)</td>
<td>((1,y))→(0,y+1)</td>
<td>Pour partial 5-gal. into 2-gal.</td>
</tr>
</tbody>
</table>

8-puzzle revisited

How hard is this problem?

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>2</td>
</tr>
</tbody>
</table>
8-puzzle revisited

The average depth of a solution for an 8-puzzle is 22 moves.

An exhaustive search requires searching $\sim 3^{22} = 3.1 \times 10^{10}$ states:
- BFS: 10 terabytes of memory
- DFS: 8 hours (assuming one million nodes/second)

Can we do better?

Is DFS and BFS intelligent?

from: Claremont to: Rowland Heights

What would the search algorithms do?

from: Claremont to: Rowland Heights

How do you think google maps does it?
from: Claremont to: Rowland Heights

BFS

from: Claremont to: Rowland Heights

Ideas?

from: Claremont to: Rowland Heights

We’d like to bias search towards the actual solution

Informed search

Order to visit based on some knowledge of the world that estimates how “good” a state is

- \( h(n) \) is called an evaluation function

Best-first search

- rank to visit based on \( h(n) \)
- take the most desirable state in to_visit first
- different approaches depending on how we define \( h(n) \)
**Heuristic**

Heuristic function: $h(n)$

<table>
<thead>
<tr>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Map path finding?</td>
</tr>
<tr>
<td>8-puzzle?</td>
</tr>
<tr>
<td>Missionaries and cannibals?</td>
</tr>
</tbody>
</table>

An estimate of how close the node is to a goal

Uses domain-specific knowledge!

Merriam-Webster's Online Dictionary

Heuristic (pron. 'hyu-ˈris-tik): adj. [from Greek *heuriskein* to discover.] involving or serving as an aid to learning, discovery, or problem-solving by experimental and especially trial-and-error methods.

The Free On-line Dictionary of Computing (2/19/13)

heuristic 1. Of or relating to a usually speculative formulation serving as a guide in the investigation or solution of a problem: "The historian discovers the past by the judicious use of such a heuristic device as the 'ideal type'" (Karl J. Weintraub).