Introduction to Artificial Intelligence

Unit # 2

Acknowledgement

- Slides of this lecture have been taken from the lecture slides of CS307 – “Introduction to Artificial Intelligence” by Dr. Sajjad Haider.

Problem Solving as Search

- Problem solving is an important aspect of Artificial Intelligence.
- A problem can be considered to consist of a goal and a set of actions that can be taken to lead to the goal.
- Search can be defined as a problem solving technique that enumerates a problem space from an initial position in search of a goal position (or solution).
- At any given time, we consider the state of the search space to represent where we have reached as a result of the actions we have applied so far.

Missionaries and Cannibals

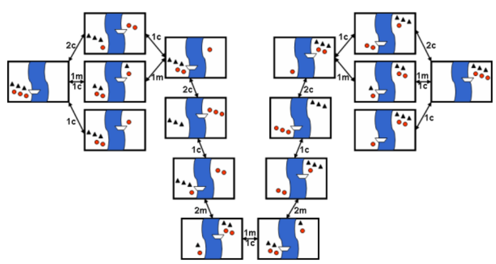
Three missionaries and three cannibals are on one side of a river, with a canoe. They all want to get to the other side of the river. The canoe can only hold one or two people at a time. At no time should there be more cannibals than missionaries on either side of the river, as this would probably result in the missionaries being eaten.

Missionaries and Cannibals (Cont’d)

- The following operators are available:
  1. Move one cannibal to the other side
  2. Move two cannibals to the other side
  3. Move one missionary to the other side
  4. Move two missionaries to the other side
  5. Move one cannibal and one missionary to the other side

  – http://www.learn4good.com/games/puzzle/boat.htm

Missionaries and Cannibals (Cont’d)



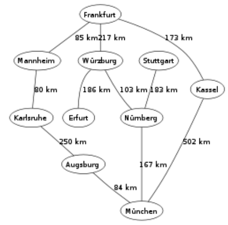
Missionaries and Cannibals (Cont’d)

- Let a state (mcb) be given by the ‘m’ numbers of missionaries, ‘c’ cannibals and ‘b’ boats on the initial bank of the river. The initial situation is represented by (331) and the goal situation by (000).
- Most AI texts that mention the problem accept this formulation and give us the solution:

331 → 310 → 321 → 300 → 311 → 110 → 221 → 020 → 031 → 010 → 021 → 000

Graph - Recap

- Graph
  - Nodes
  - Edges
  - Directed vs Undirected
  - Weighted Graph
- Tree



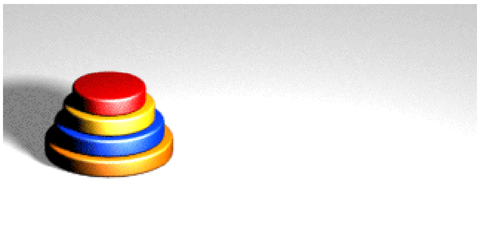
The Towers of Hanoi

We have three pegs and a number of disks of different sizes. The aim is to move from the starting state where all the disks are on the first peg, in size order (smallest at the top) to the goal state where all the pegs are on the third peg, also in size order. We are allowed to move one disk at a time, as long as there are no disks on top of it, and as long as we do not move it on top of a peg that is smaller than it.

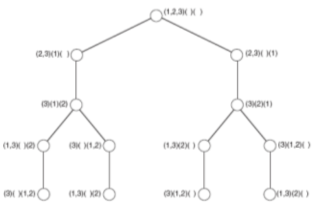
The Towers of Hanoi (From Wikipedia)

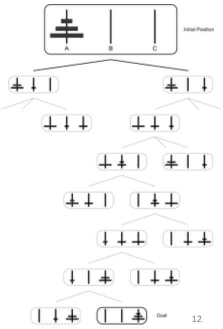
- The puzzle was invented by the French mathematician Édouard Lucas in 1883. There is a legend about an Indian temple which contains a large room with three time-worn posts in it surrounded by 64 golden disks. The priests of Brahma, acting out the command of an ancient prophecy, have been moving these disks, in accordance with the rules of the puzzle. According to the legend, when the last move of the puzzle is completed, the world will end. The puzzle is therefore also known as the Tower of Brahma puzzle.
- If the legend were true, and if the priests were able to move disks at a rate of one per second, using the smallest number of moves, it would take them 264−1 seconds or roughly 584.542 billion years [operation taking place is 2^64/60/60/24/365.25(to take into consideration leap years)/1000000000] .[1] (In context, the universe is currently about 13.7 billion years old.)

The Towers of Hanoi (Cont’d)

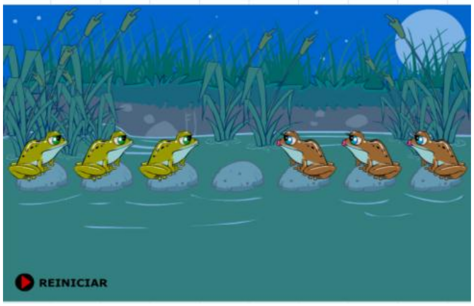


The Towers of Hanoi (Partial Search Tree)





Frog Problem



http://www.hellam.net/maths2000/frogs.html

Assignment # 1
Due before the next class

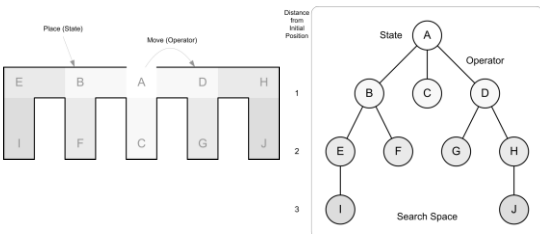
Draw complete search space of the 'Frog Problem' in the form of a tree and identify all possible solutions from the tree.

Representation:
Each node of the tree will be represented in the form of a 7-char string, say 1110222 where:

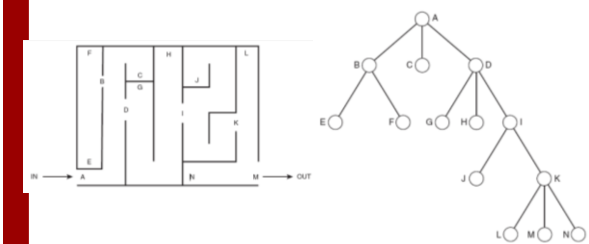
- 1 represents green frog
- 2 represents red frog
- And 0 represents empty block

The root node of the tree will be 1110222 and the goal node is 2220111.

Representing Physical Space as Tree



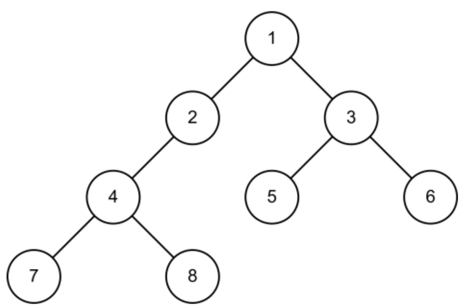
Traversing a Maze



Breadth-first Search

- In breadth-first search (BFS), we search the graph from the root node in order of the distance from the root.
- Rather than digging deep down into the graph, progressing further and further from the root (as is the case with DFS), BFS checks each node nearest the root before descending to the next level.
- The implementation of BFS uses a FIFO (first-in first-out) queue.

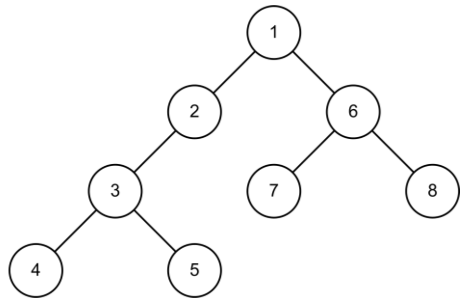
Search Order of the BFS



Depth-first Search

- The Depth-first search (DFS) algorithm is a technique for searching a graph that begins at the root node, and exhaustively searches each branch to its greatest depth before backtracking to previously unexplored branches.
- The implementation of DFS uses a LIFO (last-in first-out) queue.

Search Order of the DFS



BFS and DFS Algorithms

BFS

- Let L be a list containing the initial state
- Loop
  - If L is empty return failure
  - Node ← remove-first(L)
  - If Node is a goal
    - Return the path from initial state of Node
  - Else
    - Generate all successors of Node
    - Add generated nodes to the ***back*** of L
- End Loop

DFS

- Let L be a list containing the initial state
- Loop
  - If L is empty return failure
  - Node ← remove-first(L)
  - If Node is a goal
    - Return the path from initial state of Node
  - Else
    - Generate all successors of Node
    - Add generated nodes to the ***front*** of L
- End Loop

Depth-first vs. Breadth-first

Scenario	Depth first	Breadth first
Some paths are extremely long, or even infinite	Performs badly	Performs well
All paths are of similar length	Performs well	Performs well
All paths are of similar length, and all paths lead to a goal state	Performs well	Wasteful of time and memory
High branching factor	Performance depends on other factors	Performs poorly