CS427/527
An Introduction to Artifical Intelligence
Homework 2 (Chapters 3 & 4)

Jacob Hurst

February 3, 2019

1 Determine whether goal-driven or data-driven search would be preferable for solving each of the following problems. Justify your answer. Do not forget to tell what is the goal and what is the data in your problem definition.

1. Diagnosing mechanical problems in an automobile.
   Goal-driven search would be preferable in this case as the goal is to identify the cause for the mechanical problems, the data we have relates to the effects or problems experienced as a result of the root cause.

2. You have met a person who claims to be your distant cousin, with a common ancestor named John Doe. You would like to verify her claim.
   Goal-driven would be preferable in this case, the goal should be the shared ancestor of John Doe. The data we have is in the parents of ourselves and the supposed cousin. We can conduct the search once for the ancestor John Doe from ourselves and, if needed, another time fir the ancestor John Doe from the supposed cousin.

3. Another person claims to be your distant cousin. He does not know the common ancestors name but knows that it was no more than eight generations back. You would like to either find this ancestor or determine that she did not exist.

Goal-driven would be preferable in this case, the goal should be any shared ancestors at a max depth of eight generations back. The data we have is, again, in the parents. We will need to conduct the search twice, once from ourselves and another time from the supposed cousin.

4. A theorem prover for plane geometry.
   Goal-driven would be preferable in this case, the goal should be the proposed theorem and we should work our way back towards the basic rules/facts which lead this theorem. The basic rules/facts can be considered our data we'd like to work towards.

5. A program for examining sonar readings and interpreting them, such as telling a large submarine from a small submarine from a whale from a school of fish.
   Data-driven would be preferable in this case, the data should be the current sonar reading and the goal is to identify the current sonar reading. As the reading will not all be exactly the same it'd be best to work our way towards an approximate match. The goal is an approximate match (identification).

6. An expert system that will help a human classify plants by species, genus, etc.
   Data-driven would be preferable in this case. Similar to the complexity of sonar readings, plants will not have an exact form, rather, they will have attributes common to some species of plants. It's better to consider these attributes of the current plant as our data and work our way towards the goal of identifying an approximate match in plant species.

2 Recall the 4 rubrics for evaluating search. For each of these rubrics, give analysis to justify their values for BFS and DFS.

b is the branching factor.
d is the depth to a solution.
m is the max depth of the tree.

1. BFS

   (a) Completeness: Yes, every node will be checked until a solution is found. If there is no solution, then there will be no more nodes

to check and no solution found.

(b) Optimality: Yes, as the first solution found will contain the minimum number of steps to that solution.

(c) Time Complexity: $O(b^d)$ since the tree will branch, at most, d times before finding a solution - each branch containing b nodes $b * b * ... * b$ (d times) $= b^d$. In the worst case, the solution is the last node visited in the tree.

(d) Space Complexity: $O(b^d)$ since, at the deepest level, we must hold b^d nodes in the queue.

2. DFS

(a) Completeness: Similar to BFS, yes, every node will be checked until a solution is found. If there is no solution, then there will be no more nodes to check and no solution found.

(b) Optimality: No, since DFS follows a path all the way to its max depth, it could potentially find a solution far down the tree and return that solution when one of the earlier branches that weren't yet explored could've contained a shallower and optimal solution.

(c) Time Complexity: $O(b^d)$ since the tree will branch, at most, d times before finding a solution - each branch containing b nodes $b * b * ... * b$ (d times) $= b^d$. Similar to BFS, in the worst case, the solution is the last node visited in the tree.

(d) Space Complexity: $O(bm)$ since the longest branch has depth m, for each branch you will need to store its nodes down the path.

3 Define a search problem (e.g., in terms of problem setup, search tree shape, search tree structure, start, and/or goal location) in which each of these search algorithms will work well.

1. DFS

Search problem: A game of chess. Start: Initial board configuration. Goal: Configuration that puts opponent in checkmate. This tree will have a large branching factor as each piece has its own set of valid moves at any given moment - since the size of each level gets very large very quickly, it's better to use a depth-first search (with some modifications - like pruning).

2. BFS
   Search problem: Finding the shortest unweighted path among a set of nodes. Start: Any node in the graph (our start position). Goal: Any node in the graph (our goal position). If we are representing this as a tree rather than a graph, then each node will represent a location and its children will be the nodes directly adjacent to it. BFS would be preferable in this case as it would find the specified node at a shallowest depth (least number of hops).

3. Best-First Search
   Search problem: Finding the shortest weighted path among a set of nodes. Start: Any node in the graph (our start position). Goal: Any node in the graph (our goal position). Heuristic function: Check the cheapest node to visit first. We can represent this as a tree rather than a graph as above. Each iteration, the search algorithm would check the cheapest node to reach.