CS 427

HOMEWORK 2

BAKER, ALEX

Problem 1

Part A

Data: Mechanical parts in an automobile
Goal: Problem Part

This would benefit form a data driven search because it is unknown which mechanical parts are causing problems and therefore there is no clear goal state without searching.

Part B

Data: Common ancestor
Goal: Me

This would benefit from a goal driven search, because the branching factor could potentially be smaller going backwards since each state will only have two parents.

Part C

Data: Me
Goal: Distant Cousin

This would benefit from a bidirectional search in order to find the common ancestor.

Part D

Data: A claim in plane geometry
Goal: A theorem proving the claim

This would benefit from a bidirectional search. Determining if goal or data driven search is better for the proof largely depends on the claim and the type of proof being used, therefore bidirectional search will be able to take into account both types of searches depending on the problem.

Part E

Data: Sonar reading
Goal: Identified Object

This would benefit from data driven search because the start is clearly defined from the sonar reading, which could then be matched to a potential very larger number of objects.

Part F

Data: plant
Goal: classification of the plant

This would benefit from data driven search because the start state is clearly defined by the plant, which could then be matched to a larger number of species, genus, etc…

Problem 2

b: Branching factor
d: depth of optimal solution
m: maximum depth of the tree

BFS

Time Complexity

Time complexity will be the amount of time it takes DFS to search through the tree to find the optimal solution. Let DFS spend 1-unit time per node in the tree. Then the time complexity of DFS is b^d

Space Complexity

Because time complexity is based on each node in the tree BFS searches, the space complexity will be equivalent, needing 1-unit space for each node visited. Then the space complexity of DFS is b^d

Completeness

BFS is complete because it will always find the optimal solution given that $b \neq \infty$

Optimality

BFS is optimal because it searches all of the nodes on level n before moving to level n+1, so it will always find the optimal solution.

DFS

Time Complexity

DFS will always search to the bottom of the tree before moving onto the next branch, so it will have a time complexity of b^m

Space Complexity

DFS will have the same space complexity as time complexity of b^m

Completeness

DFS is not complete since it is possible to have $m \neq \infty$

Optimality

DFS is not guaranteed to find the optimal solution since it will search to the bottom of the tree before moving on.

Problem 3

Part A

Depth first search would work well for solving games such as chess, since there is potentially a high branching factor, and it is known there are a finite number of states to the tree does not have an infinite depth.

N: A representation of the current state of the board
A: choosing a valid move through the game rules given the current state
S: Start state of the game
GD: State of the game such that the computer wins based on game rules

Part B

BFS would work well in a binary search given the low branching factor.

N: A state with a total ordering
A: choose left child if goal is less than the current state, right child if goal is greater than current state
S: root of the tree
GD: current state matches the goal state

Part C

Best first search works well for finding local maxima, such as finding Sandia peak on a topographical map

N: Lon/Lat coordinates on a map of Albuquerque
A: Moving along the topographical map in the direction of increasing elevation
S: UNM
GD: the highest point on the topographical map