Yuhan Zhang 20634834

Q1

Algorithm	**Time Complexity**	**Space Complexity**	**Complete?**	**Optimal?**
BFS	O(b^d)	O(b^d)	Y	Y
UCS	O(b^(1+C/ε))	O(b^(1+C/ε))	Y	Y
DFS	O(b^m)	O(b*m)	N	N
DLS	O(b^l)	O(b*l)	N	N
IDS	O(b^d)	O(b*d)	Y	Y
A*	O(b^d)	O(b^d)	Y	Y

ε: The minimum step cost, B = branching factor, d = depth, L = limit, C = cost

Q2

P 3

BFS: count = 5

DFS: count = 5

P 5

BFS: count = 27

DFS: count = 14

DLS: ran with limit = 17, count = 14

P 6

Use BFS:

moveToSample: count = 5

removeSample: count = 19

returnToCharger: count = 11

Use DFS:

moveToSample: count = 9

removeSample: count = 10

returnToCharger: count = 3

Use DLS:

moveToSample: count = 9

removeSample: count = 10

returnToCharger: count = 3

Q5

a) What engineering breakthroughs were responsible for the success of Deep Blue? Which are applicable to domains other than chess, and which are special to chess?

Several engineering milestones were critical in making Deep Blue's victory over Garry Kasparov possible. First, its chess chips against the general-purpose computer hardware ate financial resources like sand; for chess-specific calculations, it ran way faster and much more efficient. The system also utilized parallel processing, dividing tasks over many processors—a strategy that sent millions of positions per second cranking through to be evaluated. Deep Blue also applied advanced search algorithms, particularly alpha-beta pruning, to reduce the number of nodes in the game tree that must be examined by pruning branches that would not impact the outcome. Looking deeper into the game tree added to the predictive power of the system's performance and thus enabled it to project a move and its counter.

A number of these enhancements have applicability beyond chess, also. Custom hardware and parallel processing are some of the quintessential methods used in wide fields such as cryptography, scientific simulations, and machine learning, when large-scale computation is required. For example, most modern deep learning training heavily relies on either a GPU or a TPU to work out performance significantly. Similarly, the alpha-beta pruning-based search algorithms can be applied to decision tree problems arising in game theory, AI planning, and optimization problems. What Deep Blue does with extensive software testing and continuous improvement is also at the heart of every kind of software development.

However, many of the concrete design decisions made in Deep Blue were specific to chess. Its evaluation function was designed to take expert heuristics and build a representation of the value of chess positions from them. Although a very common technique in AI, the design of evaluation functions designed for chess does not directly

generalize to other domains without significant modification. Furthermore, the enormous chess-specific databases of grandmaster games and precomputed opening and endgame tables, which comprised such an integral part of Deep Blue's strategy, have few direct analogues to other areas.

b) Because AlphaZero defeated Stockfish while using only 1 percent of its time to do the calculation.

Its success tends to be based on different search and evaluation, as it uses only 1% of the calculation time of Stockfish. Whereas Stockfish relies on a brute-force examination of millions of positions per second, AlphaZero investigates very select lines of play using a technique called Monte Carlo Tree Search. All the computational resources are put into a smaller number of key decisions, which enables MCTS to do smart selective deepening and to successfully return with a much more efficient search. As a result, the AlphaZero search turns out to consider far fewer positions than other engines, but it's looking at the most important lines.

What makes AlphaZero particularly strong is that it applies deep neural networks to evaluate the following: a value network supplying an estimate of the probability of winning from every given position, a policy network giving a prediction of the probability of various moves. By means of self-play, AlphaZero learns to evaluate positions with high precision and hence reduces the need for exploring millions of possibilities. These neural networks guide MCTS such that AlphaZero can focus on the most promising moves—given much fewer nodes are evaluated per second. In raw numbers, Stockfish is searching about 70 million positions per second, versus about 80,000 for AlphaZero. But AlphaZero's evaluations are vastly more efficient—meaning by the time it has to search as many nodes, it can make much stronger decisions. By using neural networks to support the program's evaluations with highly accurate valuations, the system achieves superior performance for a fraction of the computational effort. The brute-force search performed by Stockfish in contrast to the selective, knowledge-driven search by AlphaZero explains how AlphaZero can outplay Stockfish despite its much lower node count.