Assignment 2: Search

Question 1:

Algorithm	Time Complexity	Space Complexity	Complete?	Optimal?
BFS	O(b^d)	O(b^d)	Yes	Yes
UCS	O(b^1 + C*/ε)	O(b^1 + C*/ε)	Yes	Yes
DFS	O(b^m)	O(bm)	No	No
DLS	O(b^ℓ)	O(bℓ)	No	No
IDS	O(b^d)	O(bd)	Yes	Yes
A*	O(b^d)	O(b^d)	Yes	Yes

Legend:

- ε: The minimum step cost
- b: The branching factor
- d: The depth of the shallowest solution
- m: The maximum depth of the search tree
- ℓ: The depth limit in Depth-Limited Search (DLS)
- C*: The cost of the optimal solution

Question 2:

Part 1: Algorithm State Data

We ran the three algorithms (Breadth First Search, Depth First Search, and Depth-Limited Search) on the Mars Rover problem. The rover had to move from the station to the sample, extract the sample, bring the sample back to the station, and then go to the battery to recharge. Below is the organized data for each algorithm.

Breadth First Search (BFS):

State #	Location	Sample Extracted?	Holding Sample?	Charged?	Holding Tool?
1	station	False	False	True	False
2	sample	False	False	True	False

3	battery	False	False	True	False
4	station	False	False	True	True
5	sample	True	False	True	False
6	sample	False	False	True	True
7	battery	False	False	True	True
8	sample	True	True	True	False
9	battery	True	False	True	False
Number of States Generated: 9					

Depth First Search (DFS):

State #	Location	Sample Extracted?	Holding Sample?	Charged?	Holding Tool?
1	station	False	False	True	False
2	station	False	False	True	True
3	battery	False	False	True	True
4	sample	False	False	True	True
5	sample	True	False	True	True
6	station	True	False	True	True
7	station	True	False	True	False
8	battery	True	False	True	False
Number of States Generated: 8					

Depth-Limited Search (DLS) (Limit = 6):

State #	Location	Sample Extracted?	Holding Sample?	Charged?	Holding Tool?
1	station	False	False	True	False
2	station	False	False	True	True
3	station	False	False	True	False
4	battery	False	False	True	True
5	sample	False	False	True	True
6	sample	True	False	True	True
7	station	True	False	True	True
8	station	True	False	True	False
9	battery	True	False	True	False
Number of States Generated: 9					

Part 2: Problem Decomposition

The Mars Rover problem was decomposed into three subproblems:

1. **Move to Sample**: The rover moves from the station to the sample site.
2. **Remove Sample**: The rover extracts the sample and picks it up.
3. **Return to Charger**: The rover moves back to the station, drops off the sample, and goes to the battery to recharge.

Effect: By decomposing the problem into smaller tasks, we focus the search on relevant actions for each subproblem. This improves efficiency and reduces unnecessary exploration. The total number of states generated across all subproblems remains comparable, but each individual subproblem search becomes more manageable.

- **Move to Sample**: 4 states generated.
- **Remove Sample**: 3 states generated.
- **Return to Charger**: 2 states generated.

Conclusion: Problem decomposition did not reduce the total number of states generated across the entire problem but made each step more focused and improved search scalability.

Question 5:

a) What were the engineering advances that led to Deep Blue's success? Which of them can be transferred to other problems, and which are specific to chess?

Addressing Deep Blue's advances, these model developments included the use of specialized multiprocessor chess chips, an advanced evaluation function capable of analyzing millions of chess piece positions per second, improved search algorithms with deeper lookahead, and a comprehensive database of Grandmaster gameplay for strategic insights. Of these, I would argue that the search algorithms and hardware parallelism have broader influence on today's technology, especially in solving complex combinatorial problems. Meanwhile, the chess specific evaluation functions and dedicated chess chips are moreso tailored specifically to the game of chess and I presume have far less transferability to other technical domains.

b) AlphaZero is compared to a number of modern game-playing programs, such as StockFish, which work similarly to Deep Blue. The paper shows that AlphaZero is able to defeat StockFish even when it is given only 1/100 of the computing time. Why is that? Please frame your answer in terms of search and the number of nodes evaluated.
Assessing AlphaZero's ability to outperform StockFish comes down to its use of the Monte Carlo Tree Search algorithm in collaboration with neural networks. Unlike StockFish, which evaluates millions of nodes per second via alpha-beta pruning, AlphaZero focuses on fewer nodes (around 80,000 per sec.) – focusing attention toward promising moves. With that, this selective search approach allows AlphaZero to outperform traditional engines (like StockFish), which are more reliant on brute force search methods, making it more efficient with less computation.