Question 1

Algorithm	Time Complexity	Space Complexity	Complete?	Optimal?
BFS	$O(b^{d+1})$	$O(b^{d+1})$	Yes	Yes
UCS	$O(b^{(1+C/\varepsilon)})$	$O(b^{(1+C/\varepsilon)})$	Yes	Yes
DFS	$O(b^{n})$	O(bn)	Yes on finite graph	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 when heuristic is admissible	Yes when heuristic is admissible

Question 2.6

Rover State Counts:

Move_to_sample:
BFS: 3
DFS: 3
DLS with limit of 10: 5

Remove_sample:
BFS: 15
DFS: 15
DLS with limit of 10: 15

Return_to_charger:
BFS: 23
DFS: 29
DLS with limit of 10: 29

Question 3
A* Star vs UCS State Counts

A* Star: 27
UCS: 32

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?

Deep Blue's success largely comes from the power of the chip search engine, which was able to see many moves ahead, and paired with an efficient parallel algorithm to figure out moves efficiently, it was able to successfully beat a world champion chess player. I think that parallelism is a great applicable technique that is important to use, especially when we have large tasks that may take a long time to run. It greatly increases efficiency in time and resources because you can do multiple tasks simultaneously, breaking down tasks into chunks instead of doing them one after the other. I think that the algorithm itself was more tailored towards chess because of the database being used, which saw endgame moves and is more specific to chess, but similar algorithms do follow moves the same way. This reminded me of the minimax algorithm, which I used in CS 212 for tic tac toe. It scores moves by going through every possible move, and scoring. While it may have worked for a simple game like tic tac toe, I imagine there are more efficient algorithms out there for more complex games like chess.

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.

AlphaZero is much more efficient than StockFish and Deep Blue because it uses a heuristic search rather than a brute-force approach. While revolutionary at the time, Deep Blue uses a lot of resources by modern standards. Instead of searching millions of nodes in a brute-force way, AlphaZero finds good-quality moves by searching fewer nodes and avoiding unnecessary searches. This makes it a lot more efficient and more consistent in finding the best move.