Algorithm	Time Complexity	Space Complexity	Complete?	Optimal?
BFS	O(n + e) (nodes plus edges)	O(n + e + queue)	Yes	Yes
UCS	O(b^1+[C*/e])	O(b^1+[C*/e])	yes	yes
DFS	O(b^n)	O(b * n)	no	no
DLS	O(bℓ)	O(bℓ)	no	no
IDS	O(b^d) or O(b^m)	O(bd) or O(bm)	yes	yes
A*	O(b^d)	O(b^d)	yes	yes

A* EXPLANATION (Used GPT-4o):
For time complexity:

- b is the branching factor (the average number of children each node has).
- d being the depth of the optimal solution

For space complexity:

- b is the branching factor
- d is the depth of the solution.

QUESTION 2 NUMBER 6

Without user_tool, pickup_tool, and drop_tool in action list:
BFS count: 2
DFS count: 2
DLS Count: 2

With full action list:
BFS count: 23
DFS count: 29
DLS Count: 29

By changing the states action list to:
result = breadth_first_search(s, [charge, drop_tool, pick_up_sample, drop_sample, move_to_battery, move_to_sample, move_to_station], mission_complete)
The BFS count drops drastically:
BFS count: 2
DFS count: 29

```
DLS Count:  29

State data with a full action list:
BFS count:  23
DFS count:  29
DLS Count:  29
```

QUESTION #4

```
BFS count:  23
DFS count:  29
DLS count:  29
-----------------------------
A_STAR SEARCH WITH SLD
Goal found
(1,1)
(1,2)
(1,3)
(1,4)
(1,5)
(1,6)
(1,7)
(2,7)
(3,7)
(3,6)
(3,5)
(3,4)
(4,4)
(5,4)
(5,5)
(6,5)
(6,6)
(7,6)
(8,6)
(8,7)
(8,8)
None
A* count:  32
A_STAR SEARCH WITH H1
Goal found
(1,1)
```

```
(1,2)
(1,3)
(1,4)
(1,5)
(1,6)
(1,7)
(2,7)
(3,7)
(3,6)
(3,5)
(3,4)
(4,4)
(5,4)
(5,5)
(6,5)
(6,6)
(7,6)
(8,6)
(8,7)
(8,8)
None
A* count:  32
-----------------------------
at_1: f2
at_2: f1
at_3: f3
at_4: f3
at_5: f2
at_6: f2
at_7: f1
at_8: f3
at_9: f1
```

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?

Some engineering successes of Deep Blue are the chess chip, which allowed the chip to generate checking and check evasion moves, as well as generating different kinds of attack moves. Software search, where black or white have an unstoppable winning threat, sacrificing pieces until the win is discovered. Endgame databases, a database where Deep Blue contains all chess positions with five or fewer pieces on the board. Parallel search, which utilizes the chess chips to simultaneously search, utilizing parallelization and synchronization. And hardware search, a chess chip that carries out a fixed-depth null-window search, and it can use multiple chess chips simultaneously to speed up this operation.

Some things like parallel processing and the hardware search can be utilized in other applications. These engineering advancements can be used on search algorithms and search based problems to speed up their processes. Endgame databases can be used to solve problems that need to be solved quickly without computation. However, the hard-coded patterns in endgame search are not applicable to other applications. As well as the "Rooks on files" table, the extended book, and the opening book, which cannot be translated very well to applications outside of 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 able to defeat StockFish at 1/100 of the computing time because of its deep neural network. This neural network takes the board positions as input and outputs a vector of move probabilities, estimated the expected outcome from a position. AlphaZero then learns these move probabilities and value estimates to guide its search. AlphaZero uses a general purpose MCTS (Monte-Carlo Tree Search) algorithm. Each search is a series of simulated games that traverse from a position root to a leaf. Each simulation is a selected state *s* (position) with a move *x* with a low visit count, high probability and high value all according to the neural network. The search then returns a vector, $\boldsymbol{\pi}$, that represents a probability distribution of moves with respect to the visit count at the root state. Alpha zero, therefore, is able to have an evaluation speed of 80k (positions / second) vs StockFish's (70,000K). This is because AlphaZero evaluates positions using a non-linear function approximation based on the neural network. Versus the linear function approximation used in most chess programs. However, this may produce approximation errors. But, MCTS averages over these approximation errors, which cancels them out when evaluating a large subtree. Overall, AlphaZero utilizes a neural network, and through a more efficient search algorithm, it smashes StockFish.