Eva DeThomas

Assignment 2 Results:

Question 1:

Algorithm	Time Complexity	Space Complexity	Complete?	Optimal?
BFS	O(V + E)	O(|V|)	Yes, guaranteed to find solution	Yes, assuming all actions have a uniform cost
UCS	O((V+E)logV)	O(V)	Yes, guaranteed to find a solution	Yes, guaranteed to find correct solution
DFS	O(b^n), where n is the depth of the search tree.	(O(bn)) Linear	Complete on a finite graph, if space is infinite or we can't search for repeated states then no.	No guarantee the first solution is the best
DLS	O(b^L), where L is the pre-set limit of the search tree.	(O(L))	No, the limit allows for efficiency but not all nodes are explored	No, the first solution found will be return regardless of if it's the best or not
IDS	O(b^d)	O(bd)	Yes, will not stop until solution is found	Yes, guaranteed to find correct solution
A*	At worst: O(b^d)	O(b^d)	Depends on conditions, can be complete and optimal	Depends on conditions, can be complete and optimal

Question 2:

(All answers are post adding tool functionality)

Mission complete:

Final mission complete DFS result count: 19

```
        Final mission complete BFS result count:  23
        Final mission complete LDFS result count:  9 (limit is 6)
```

Partitioned tasks:

```
        <function go_to_sample at 0x105854ca0> Final DFS result count:  12
        <function go_to_sample at 0x105854ca0> Final BFS result count:  17
        <function go_to_sample at 0x105854ca0> Final LDFS result count:  12 limit is 8

        <function remove_sample at 0x105854d30> Final DFS result count:  7
        <function remove_sample at 0x105854d30> Final BFS result count:  4
        <function remove_sample at 0x105854d30> Final LDFS result count:  7 limit is 8

        <function return_to_charger at 0x105854dc0> Final DFS result count:  11
        <function return_to_charger at 0x105854dc0> Final BFS result count:  11
        <function return_to_charger at 0x105854dc0> Final LDFS result count:  11 limit is 8
```

Even though we would expect to see less states visited in partitioned tasks, there are sometimes more because it had to do some extra searching to get to the proper state (backtracking). (Can run the program for more specific state data)

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?

The hardware and processes of deep blue could check millions of moves and cater the amount of moves checked to every position in chess. This was in part because the hardware and software could handle deeper searches, in turn meeting the needs for chess. This algorithm they chose was extremely efficient, avoiding re-searching and glossing over less relevant details (like checkmate). The evaluation function of the algorithm was able to evaluate an extreme amount of moves, so much so the author states that the nuances are too great to explain in one paper. This algorithm and the hardware that could handle it ultimately lead to deep blue's success.

However, new search behaviors could not be introduced, only changed slightly; which I assume would mean that it would be hard to transfer this to games with entirely different rules. Likely on something similar to chess like checkers, the search could still be compatible.

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.

Because it uses a neural network. Specifically it states “It replaces the handcrafted knowledge and domainspecific augmentations used in traditional game-playing programs with deep neural networks and a tabula rasa reinforcement learning algorithm.” The reinforcement training allowed it to train and evaluate less nodes to get to the same answer that stockfish did. That is why on the graph they provide, stock fish remains the same whereas alphaZero continues to start poorly and remember patterns to get better with training. Specifically “In chess, AlphaZero outperformed Stockfish after just 4 hours (300k steps)”. Alphazero needs to evaluate less nodes.