Daniel Koontz
CS480 Sokoban

This program was designed to solve Sokoban puzzles. I utilized BFS, DFS, Greedy Best, and A-star search algorithms. I utilized several different heuristics, including a manhattan distance, a heuristic to identify stuck positions, a distance from the robot to the nearest box, and a minimum matching distance between boxes and goals. In addition, I implemented a bidirectional search which was considerably faster although it introduced some bugs that caused some algorithms to find less-than-optimal paths when they otherwise would have expected an optimal result. In addition, I used memoization to remember states, not only for not duplicating the path but for optimizing and calculating my heuristic.

Puzzle 1:

BFS, Greedy best, and AStar solved the puzzle in 46 moves, the optimal number of moves. BFS and A-star each took about 0.5 seconds and Greedy best and DFS took less than 0.1 seconds. BFS searched through the most states while greedy best searched through the least.

Puzzle 2:

BFS and A-star solved the puzzle in 54 moves, while Greedy best came close with 55 moves. BFS was slowest with 0.7 seconds followed by A-star at 0.4 seconds. Greedy was just above 0.1 seconds while DFS maintained less than 0.1 seconds. Greedy and DFS searched through the least states at about 2300 while BFS searched nearly 18000.

Puzzle 3:

BFS did not solve this problem in an adequate amount of time, but DFS and A-star took about 48 seconds and Greedy took about 19 seconds. While DFS and A-star took the same total time, A-star took much longer on each state, only searching 320000 states to DFS searching through 1350000 states, a factor of over 4. A-star also found a path in 130 moves compared to DFS generating a path of nearly 700 moves.

Puzzle 4:

All algorithms solved this puzzle, with BFS the slowest at 5 seconds and Greedy the fastest at 0.7 seconds. Greedy looked at the fewest states as well, only searching 7600 compared to BFS searching 110000. However, only BFS and A-star found the optimal path, only 127 moves.

Puzzle 5:

All the algorithms solved this puzzle with BFS the slowest at 10 seconds and Greedy the fastest at 0.25 seconds. In this puzzle, A-star and DFS had nearly the same time at 0.76 and 0.68 seconds respectively. Again though, A-star searched many fewer states, taking almost 3 times as long per state compared to DFS. Again only A-star and BFS completed the puzzle optimally in 72 moves.

Puzzle 6:

I used DFS, Greedy, and A-Star to solve this problem with greedy taking 27 seconds and A-star taking almost 3 minutes. DFS solved in about 2 minutes however taking a path 3621 moves long. Greedy's solution took 264 moves compared to A-star which only took 142. Greedy searched 190000 states while A-star searched 920000 and DFS over 3000000.

Heuristic Analysis:

The first heuristic used was a heuristic assigning an arbitrarily large value to a puzzle where a box was stuck in a corner. This typically created a minor reduction in the completion time and states searched, although this could not hav any of the algorithms get over the hump of solving a previously unsolved puzzle. This heuristic is admissible since a stuck puzzle would never be able to complete.

Next I used a heuristic to calculate the distance between the robot and the nearest box. This simply return the minimum number of steps it would take for the robot to reach the nearest box. This prioritized moving boxes over moving just the player. This typically affected the puzzles with larger empty spaces the most. The heuristic is admissible since in order to complete the puzzle, the robot must move boxes and the last move must be moving a box.

As required, I implemented the Manhattan heuristic to calculate the distance between each box and the nearest goal. This prioritized moving boxes to the nearby goals. This heuristic is clearly admissible since moving each box to a goal solves the puzzle and this heuristic attempts to return the smallest distance for each box. This heuristic typically best improved the algorithm search speed compared to the previously mentioned heuristics although it would marginally increase the time spent on each state. I used memoization to save time computing the distance since any box at the same coordinate should have the same manhattan value.

Finally, I used a minimum matching heuristic, similar to the Manhattan heuristic, where each box is matched to a unique goal such that the total distances are minimized. This computes a slightly better value than manhattan since boxes cannot overlap and similarly is admissible since it searches for the least moves to match all the boxes to goals. This typically slightly increased the time spent on each state but also reduced the number of states searched compared to Manhattan.

Algorithm Analysis:

BFS typically had the slowest time to completion and searched the most states. However it typically also spent the least amount on each state. While it should be able to complete and find the optimal solution for each of the puzzles, it would also take an unreasonable amount of time and space for the most complex puzzles.

DFS often had one of the shortest search times but also had some of the longest paths. It would spend some of the shortest time on each state and had the smallest fringe. My implementation included memoization and removed repeated states, so it theoretically should be complete, but it is clearly not optimal. DFS often had problems in puzzles where it might get stuck and it could not backtrack for a significant amount of time.

Greedy Best was able to solve every single puzzle in the shortest amount of time although it often found a non-optimal path. The solution is complete since it also utilized memoization for states. It typically searched the fewest states out of any of the algorithms.

A-star took a comparable time to DFS while taking longer on each state and returning an optimal path. It was able to solve each of the puzzles although its completion time was heavily influenced by how well the heuristics informed it. In complex puzzles, it would often beat DFS in time, while taking 4-5 times longer on each state.