Assignment 2

1 Heuristic (15 pts)

Define a novel heuristic function for this puzzle. Include an explanation of this heuristic in your report along with a justification for why it will guarantee an optimal solution with A* Search. Use this heuristic in your implementation below.

Answer:
In the Globe Puzzle, after each move, the user has the option to make six moves. Thus when we consider this as a search problem, there are 6 different possible configurations it can lead to.
They are:
Clockwise rotation of the equator
Clockwise rotation of the first longitude
Clockwise rotation of the second longitude
Anti-clockwise rotation of the equator
Anti-clockwise rotation of the first longitude
Anti-clockwise rotation of the second longitude

As a novel heuristic, I decided the take a relation of the sum of number of moves each piece needs for reaching the target state from a current configuration along with the number of moves made to reach that configuration.
Now, since the heuristic value shouldn't be over-estimated, I decided to divide the obtained distance metric by the maximum number of moves each piece can make (6) and since each piece works in pairs, I further halved my heuristic value to develop an ideal metric (ie. The north-pole piece(0,0) and the south-pole piece(180,180) are always opposite to each other).
My heuristic value is determined by the sum of Manhattan distance for each piece in a configuration divided by the product of number of possible rotations from current configuration and symmetric property of pieces in the upper and lower hemispheres given by $1/12 \sum MD_{piece} +$ N, where MD is the Manhattan Distance and N is the total number of moves made to reach that configuration.

Reason why the heuristic guarantees an optimal solution with A*.
The heuristic considers two factors when determining optimality - Admissibility and consistency. We can say that the heuristic is admissible as it never overestimates the value and the estimated cost is never more than the cost from the current state to target state. From the test cases provided (Path files), it was determined that the actual cost from the current state to the next state through Manhattan distance h(n) and the estimated state determined by the number of moves made to reach the configuration g(n) is always lesser than or equal to the estimated cost from the current configuration to the goal configuration. We can say that when the sum of Manhattan Distance of the pieces decrease, the configuration is a closer representation of the solved puzzle state. Thus, after each move, we check if the position of

the pieces are closer than the ones in the previous state. We need to incorporate the number of moves made as an additional feature for the heuristic because the same state X can be reached by performing more moves.

2 Implementation (65 pts)

Implement three search algorithms to solve the puzzle:

1. Breadth-First Search
2. A*
3. Recursive Best-First Search

Answer:
The code to develop a search algorithm for solving the Globe Puzzle is written and compiled in Python. The attached zip file contains the code to solve the puzzle using the three algorithms.
The code is implemented in a single package called Search.py. This code will be called as follows:
$Search.py < ALG >< FILE >$
Where ALG is one of: "BFS", "$AStar$", "$RBFS$". And $FILE$ is the puzzle file name.

3 Analysis (20 pts)

Using the code above calculate solution results for each of your algorithms on the Puzzle files. Report the minimum, average, and maximum values for the number of states expanded and the queue size for each algorithm and identify the hardest puzzle for each. Consider the relative performance of the algorithms and state which algorithm is best for this problem. Justify your answer.

Answer:
Below are the solution results obtained for each of the puzzle files:

Puzzle2-0.mb
AStar:
Number of states expanded: 191458
The maximum size of the queue during search: 914863
The final path length: 13
Final Path as a sequence of steps:
Increment Long 0/180 —> Increment Equator —> Increment Long 90/270 —> Increment Long 0/180 —> Increment Long 90/270 —> Increment Long 0/180 —> Increment Long 0/180 —> Increment Long 90/270 —> Decrement Equator —> Decrement Long 90/270 —> Decrement Equator —> Decrement Long 0/180 —> Increment Equator
Running time: 286.9282958507538 secs

Puzzle2-1.mb
AStar:
Number of states expanded: 163739
The maximum size of the queue during search: 797854
The final path length: 14
Final Path as a sequence of steps:
Decrement Long 90/270 —> Increment Equator —> Decrement Long 90/270 —> Increment Equator —> Increment Long 0/180 —> Increment Long 0/180 —> Increment Equator —> Increment Long 0/180 —> Increment Equator —> Increment Long 90/270 —> Decrement Long 0/180 —> Increment Long 90/270 —> Increment Long 0/180 —> Increment Long 0/180
Running time: 248.05612754821777 secs

Puzzle2-2.mb
BFS:
Number of states expanded: 474732
The maximum size of the queue during search: 2262216
The final path length: 9
Final Path as a sequence of steps:
Decrement Long 0/180 —> Increment Long 90/270 —> Increment Equator —> Decrement Long 90/270 —> Increment Equator —> Increment Equator —> Increment Equator —> Decrement Long 90/270 —> Decrement Long 90/270
Running time: 11351.840588331223 secs

AStar:
Number of states expanded: 250
The maximum size of the queue during search: 1247
The final path length: 9
Final Path as a sequence of steps:
Decrement Long 0/180 —> Increment Long 90/270 —> Increment Equator —> Decrement Long 90/270 —> Increment Equator —> Increment Equator —> Increment Equator —> Decrement Long 90/270 —> Decrement Long 90/270
Running time: 0.35938262939453125 secs

Puzzle2-3.mb
BFS:
Number of states expanded: 83199
The maximum size of the queue during search: 398985
The final path length: 8
Final Path as a sequence of steps:
Increment Long 0/180 —> Decrement Equator —> Increment Long 90/270 —> Increment Long 90/270 —> Decrement Equator —> Decrement Long 90/270 —> Increment Long 0/180 —> Increment Equator
Running time: 342.9807107448578 secs

AStar
Number of states expanded: 711
The maximum size of the queue during search: 3494
The final path length: 8
Final Path as a sequence of steps:
Increment Long 0/180 —> Decrement Equator —> Increment Long 90/270 —> Increment Long 90/270 —> Decrement Equator —> Decrement Long 90/270 —> Increment Long 0/180 —> Increment Equator
Running time: 1.015784502029419 secs

Puzzle2-4.mb
AStar:
Number of states expanded: 17594
The maximum size of the queue during search: 84560
The final path length: 11
Final Path as a sequence of steps:
Decrement Long 0/180 —> Decrement Long 0/180 —> Increment Equator —> Decrement Long 0/180 —> Increment Long 90/270 —> Increment Long 0/180 —> Increment Equator —> Decrement Long 90/270 —> Increment Equator —> Increment Long 90/270 —> Increment Long 0/180
Running time: 24.798535585403442 secs

Puzzle2-5.mb
AStar:
Number of states expanded: 17299
The maximum size of the queue during search: 84686
The final path length: 12
Final Path as a sequence of steps:
Increment Long 0/180 —> Increment Long 0/180 —> Increment Long 0/180 —> Increment Equator —> Increment Long 0/180 —> Increment Long 0/180 —> Decrement Equator —> Decrement Equator —> Increment Long 90/270 —> Decrement Equator —> Decrement Long 90/270 —> Decrement Long 90/270
Running time: 24.798969268798828 secs

Puzzle2-8.mb
BFS:
Number of states expanded: 9763
The maximum size of the queue during search: 47324
The final path length: 7
Final Path as a sequence of steps:
Increment Long 0/180 —> Increment Long 90/270 —> Increment Long 0/180 —> Decrement Long 90/270 —> Decrement Long 0/180 —> Decrement Long 0/180 —> Decrement Equator
Running time: 6.3339760303497314 secs

AStar:
Number of states expanded: 312
The maximum size of the queue during search: 1516
The final path length: 7
Final Path as a sequence of steps:
Increment Long 0/180 —> Increment Long 90/270 —> Increment Long 0/180 —> Decrement Long 90/270 —> Decrement Long 0/180 —> Decrement Long 0/180 —> Decrement Equator
Running time: 0.4218583106994629 secs

Puzzle2-9.mb
AStar:
Number of states expanded: 738
The maximum size of the queue during search: 3676
The final path length: 10
Final Path as a sequence of steps:
Increment Long 0/180 —> Increment Long 0/180 —> Increment Long 90/270 —> Decrement Equator —> Decrement Equator —> Increment Long 0/180 —> Increment Equator —> Increment Long 90/270 —> Increment Long 90/270 —> Increment Long 0/180
Running time: 1.1095314025878906 secs

Puzzle2-11.mb
BFS:
Number of states expanded: 13250
The maximum size of the queue during search: 64097
The final path length: 7
Final Path as a sequence of steps:
Increment Long 90/270 —> Decrement Long 0/180 —> Increment Long 90/270 —> Decrement Equator —> Decrement Long 90/270 —> Decrement Equator —> Decrement Equator
Running time: 9.544635772705078 secs

AStar:
Number of states expanded: 87
The maximum size of the queue during search: 432
The final path length: 7
Final Path as a sequence of steps:
Increment Long 90/270 —> Decrement Long 0/180 —> Increment Long 90/270 —> Decrement Equator —> Decrement Long 90/270 —> Decrement Equator —> Decrement Equator
Running time: 0.14084219932556152 secs

Puzzle2-12.mb
BFS:
Number of states expanded: 14510
The maximum size of the queue during search: 70112
The final path length: 7
Final Path as a sequence of steps:
Decrement Long 90/270 —> Decrement Equator —> Increment Long 0/180 —> Increment Equator —> Increment Equator —> Decrement Long 90/270 —> Decrement Long 0/180
Running time: 12.397239685058594 secs

AStar:
Number of states expanded: 260
The maximum size of the queue during search: 1290
The final path length: 7
Final Path as a sequence of steps:
Decrement Long 90/270 —> Decrement Equator —> Increment Long 0/180 —> Increment Equator —> Increment Equator —> Decrement Long 90/270 —> Decrement Long 0/180
Running time: 0.3906090259552002 secs

Puzzle2-14.mb
AStar:
Number of states expanded: 20240
The maximum size of the queue during search: 100128
The final path length: 12
Final Path as a sequence of steps:
Decrement Long 90/270 —> Decrement Long 0/180 —> Decrement Long 0/180 —> Decrement Equator —> Decrement Long 90/270 —> Decrement Long 90/270 —> Increment Long 0/180 —> Increment Long 0/180 —> Decrement Equator —> Increment Long 0/180 —> Increment Long 0/180 —> Decrement Long 90/270
Running time: 29.23625683784485 secs

Puzzle2-15.mb
AStar:
Number of states expanded: 201080
The maximum size of the queue during search: 941438
The final path length: 12
Final Path as a sequence of steps:
Decrement Equator —> Increment Long 0/180 —> Increment Equator —> Decrement Long 0/180 —> Decrement Long 90/270 —> Decrement Long 0/180 —> Decrement Equator —> Increment Long 0/180 —> Decrement Long 90/270 —> Increment Equator —> Increment Long 90/270 —> Decrement Equator
Running time: 270.7199490070343 secs

Puzzle2-16.mb
BFS:
Number of states expanded: 63339
The maximum size of the queue during search: 303960
The final path length: 8
Final Path as a sequence of steps:
Increment Long 90/270 —> Decrement Long 0/180 —> Increment Equator —> Decrement Long 0/180 —> Decrement Long 0/180 —> Increment Equator —> Decrement Long 90/270 —> Increment Long 0/180
Running time: 201.66765928268433 secs

AStar:
Number of states expanded: 834
The maximum size of the queue during search: 4125
The final path length: 8
Final Path as a sequence of steps:
Increment Long 90/270 —> Decrement Long 0/180 —> Increment Equator —> Decrement Long 0/180 —> Decrement Long 0/180 —> Increment Equator —> Decrement Long 90/270 —> Increment Long 0/180
Running time: 1.203416109085083 secs

Puzzle2-18.mb
AStar:
Number of states expanded: 24836
The maximum size of the queue during search: 120460
The final path length: 12
Final Path as a sequence of steps:
Decrement Long 0/180 —> Increment Long 90/270 —> Increment Long 90/270 —> Decrement Equator —> Increment Long 0/180 —> Decrement Long 90/270 —> Decrement Long 90/270 —> Increment Equator —> Increment Equator —> Increment Equator —> Increment Long 90/270 —> Increment Equator
Running time: 34.98663806915283 secs

Puzzle2-19.mb
AStar:
Number of states expanded: 297601
The maximum size of the queue during search: 1401698
The final path length: 13
Final Path as a sequence of steps:
Decrement Long 90/270 —> Increment Equator —> Increment Equator —> Increment Long 90/270 —> Increment Long 0/180 —> Increment Long 0/180 —> Decrement Equator —> Increment Long 90/270 —> Increment Equator —> Increment Long 0/180 —> Decrement Long 90/270 —> Decrement Long 0/180 —> Decrement Equator
Running time: 409.0256793498993 secs

Runtime issues:
Since the algorithm is run on a personal computer, larger depths couldn't be achieved due to memory restrictions. The puzzles numbered 6,7,10 and 13 were of larger depths and the path for them could not be determined. While running the A* algorithm, the above mentioned puzzles either couldnt reach the desired goal state in 3hrs or the memory exceeded.
When the algorithms ran for 3hrs, I was able explore 2407000 nodes before the memory limit was reached.

Performance of BFS, AStar and RBFS:
From the above results, it can be seen that without a novel heuristic, it is difficult to determine the path from a scrambled state when the number of moves to unscramble is high. My BFS code is seen to run only when the number of moves to unscramble is less than 10. However, once we define a heuristic to tell the algorithm which path is irrelevant, the solution is computed in a much shorter time.

Cases where each algorithm is efficient:
The breadth first search can be used in the cases where the number of moves to unscramble is vary low. For instance, a search tree with a height of 3 can be computed in an instant and no novel heuristic has to be calculated for obtaining a result in a faster time. Eg. PathN-3 time details provided below.

The recursive best first search is ideal in the cases where the heuristic is spot on and the number of recursions needed to determine the path is less. It works the best when the the heuristic value of the child node is always lesser than that of the parent. In such a case, the algorithm never needs to back-track.

The final technique used to solve the Globe Puzzle is the AStar algorithm. It can be seen from the results that for problems where there is no definite heuristic, any sub-optimal heuristic function may give desirable results in comparison to the other techniques. It may be noted that for a puzzle with 10 moves, the breadth first search technique took an average of 11351 seconds to run and AStar was able to determine the path in 0.35 seconds. This difference is massive and thus conveys the significance of a novel heuristic.

From the test cases provided, the comparison of time between a non-heuristic based search and a heuristic based search is given below and the impact can be more clearly understood.

PathN-1.mb :
Non-heuristic - 0.0 seconds, Heuristic - 0.0 seconds
PathN-2.mb :
Non-heuristic - 0.0 seconds, Heuristic - 0.0 seconds
PathN-3.mb :
Non-heuristic - 0.0 seconds, Heuristic - 0.0156 seconds
PathN-4.mb :
Non-heuristic - 0.0624 seconds, Heuristic - 0.0312 seconds
PathN-5.mb :
Non-heuristic - 0.1406 seconds, Heuristic - 0.0312 seconds
PathN-6.mb :
Non-heuristic - 1.8750 seconds, Heuristic - 0.0937 seconds
PathN-7.mb :
Non-heuristic - 26.0940 seconds, Heuristic - 0.1718 seconds
PathN-8.mb :
Non-heuristic - 207.7097 seconds, Heuristic - 1.9688 seconds
PathN-9.mb :
Non-heuristic - 11288.85 seconds, Heuristic - 2.7501 seconds
PathN-10.mb :
Non-heuristic - Memory exceeded, Heuristic - 1.5626 seconds

Hardest case for each puzzle: For the breadth first search technique, the hardest case would be when the solution is the last node (Right most node) of the developed tree of height N. Thus, the solution node would be the $6^{(N_{th})}$ node. Furthermore, for puzzles with a large number of moves, it is time consuming to build solutions level by level. Of the puzzles solvable with BFS, Puzzle2-2 was the hardest as generating 9 levels was time consuming.
For the recursive best first search, the hardest case would be when the the solution node is present in the sibling node but the recursive function assumes gets better f-scores with child nodes. This is however a hypothetical case. From the list of puzzles, Puzzle2-6 was unsolvable even after running it for 2hrs.
The AStar algorithm started to get memory bound issues when the number of moves to solve reaches 16. It can be seen from the results that Puzzles 6, 10 and 13 couldn't run completely before the memory ran out.
Best algorithm for the problem Since the recursive best first search had recursive depth issues for a few cases, I could not test the puzzles with a larger number of moves. Thus, from the results obtained for various paths and puzzles, it is clear that a heuristic technique is essential for determining the solution at a faster time. The A* search algorithm is found to work on most cases and also determines the right path in a few seconds, we conclude that an ideal algorithm to solve the globe puzzle is the A* algorithms.