KNIGHT'S TOUR

STRATEGY DESCRIPTION

AIM

This programming assignment focusses on generating *closed* Knight's tours, keeping time efficiency as a key benchmark. Using Brute-force Depth first search, the program employs varying constraints to generate faster solutions.

STRATEGIES

Strategy 0

The most basic approach, Strategy 0 uses a depth first search whereby the program non-deterministically chooses a move from the available moves at each step. Although useful as a starting point using smaller chessboards, this approach owing to its "random" nature rarely ever generates a solution for an 8*8 board. While traversing the board, if a failure condition is met, the program traces its route back and remembers which moves were previously made from the current square.

Strategy 1

For this strategy, our program randomly picks from the available moves to unvisited squares and squares it has not yet searched as an extension to its current path that have the *lowest fixed degree*. In a similar fashion, it proceeds forward picking from the minimum fixed degree neighbour nodes until it reaches either a solution or a point of failure, in which case it backtracks from this square.

Strategy 2

This strategy initialises the dynamic degrees to each square's static degree. Every time the program moves to a square, it decrements the dynamic degree of all its neighbors that have not been visited yet. When the program reaches a dead end (or a solution!), and backs up out of a square, it increments the dynamic degree of all its unvisited neighbors to reset them. Thus at any given moment the dynamic degree of a square is the number of its neighbors which are legal moves and are not yet part of the current path. With Strategy 2, the program chooses randomly from the available moves with the lowest dynamic degree (Warnsdorf's rule) (and hasn't backed up from along this path).

Strategy 3

In Strategy 3, the program checks if more than one of the *available* neighbours has a dynamic degree of 1 while choosing a move. If so, this is a state of failure since no two neighbours of a square can have a dynamic degree of 1 since this would imply that on making a move, there would still be one pending square with one move left, but no where to move to. Hence, the program backtracks from this square. In cases where this doesn't apply, it uses Strategy 1 to pick the next square.

Strategy 4

When selecting a move, if there is only one available neighbor with a dynamic degree of 1, the program moves there. In addition, in backing up from such a move (after success or failure), the other available moves are not tried, but the system continues to back up to the previous move. This is because no other move besides to the move with the dynamic degree of 1 can ever lead to a successful path. In scenarios where this does not apply. Strategy 1 is used to decide the next move.

Strategy 5

The most optimal strategy, this strategy uses a combination of 2, 3 and 4 such that all three are used at the same time. It uses the dynamic degree to choose its next node while moving forward.

GLOSSARY

Knight's tour : Sequence of moves of a knight on a chessboard such that the knight visits every square only once.

Open solution : If the knight ends on a square that is one knight's move from the beginning square, the tour is closed.

Closed solution : If the knight ends on a square that is *not* one knight's move from the beginning square, the tour is open.

Fixed degree : Number of available moves at the start of the search

Dynamic Degree : Number of available moves from a square given the search so far

Valid move : A knight moves in an L, two squares in one direction and one in an orthogonal direction.

PERFORMANCE

Strategy 0

Mean : Time taken	0.1948681195
Median : Time taken	0.2237242073
SD : Time taken	0.08161734131

Mean : Number of moves	1000000
Median : Number of moves	1000000
SD : Number of moves	0

Mean : Number of moves/second	5625053.98
Median : Number of moves/second	4792095.507
SD : Number of moves/second	2355962.339

Mean : Number of solutions/second	0
Median : Number of solutions/second	0
SD : Number of solutions/second	0

Mean : Number of solutions	0
Median : Number of solutions	0
SD : Number of solutions	0

Strategy 1

Mean : Time taken	0.1982612745
Median : Time taken	0.2279640328
SD : Time taken	0.08401208711

Mean : Number of moves	1000000
Median : Number of moves	1000000
SD : Number of moves	0

Mean : Number of moves/second	5541349.612
Median : Number of moves/second	4711165.461
SD : Number of moves/second	2348115.372

Mean : Number of solutions/second	4287.562887
Median : Number of solutions/second	4947.79044
SD : Number of solutions/second	1867.405519

Mean : Number of solutions	928.5
Median : Number of solutions	1186.75
SD : Number of solutions	730.441305

Strategy 2 :

Mean : Time taken	0.210820148
Median : Time taken	0.2395537435
SD : Time taken	0.0812708809

Mean : Number of moves	1000000
Median : Number of moves	1000000
SD : Number of moves	0

Mean : Number of moves/second	5124125.776
Median : Number of moves/second	4425736.373
SD : Number of moves/second	1975343.532

Mean : Number of solutions/second	7704.502558
Median : Number of solutions/second	10255.83337
SD : Number of solutions/second	7216.253281

Mean : Number of solutions	1917.5
Median : Number of solutions	2676.75
SD : Number of solutions	2147.483294

Strategy 3 :

Mean : Time taken	0.242597829
Median : Time taken	0.273436269
SD : Time taken	0.08722428018

Mean : Number of moves	1000000
Median : Number of moves	1000000
SD : Number of moves	0

Mean : Number of moves/second	4406889.867
Median : Number of moves/second	3846696.854
SD : Number of moves/second	1584465.113

Mean : Number of solutions/second	15478.10815
Median : Number of solutions/second	11502.10098
SD : Number of solutions/second	11245.84655

Mean : Number of solutions	3264.5
Median : Number of solutions	2777.25
SD : Number of solutions	1378.151117

Strategy 4 :

Mean : Time taken	0.26280229
Median : Time taken	0.2936251035
SD : Time taken	0.08718008176

Mean : Number of moves	1000000
Median : Number of moves	1000000
SD : Number of moves	0

Mean : Number of moves/second	4026704.245
Median : Number of moves/second	3554431.518
SD : Number of moves/second	1335788.989

Mean : Number of solutions/second	161651.1251
Median : Number of solutions/second	127789.9572
SD : Number of solutions/second	95773.84593

Mean : Number of solutions	38307.5
Median : Number of solutions	34391.25
SD : Number of solutions	11076.82773

Strategy 5 :

Mean : Time taken	0.308897438
Median : Time taken	0.350037806
SD : Time taken	0.1163625328

Mean : Number of moves	1000000
Median : Number of moves	1000000
SD : Number of moves	0

Mean : Number of moves/second	3484558.631
Median : Number of moves/second	3020469.238
SD : Number of moves/second	1312643.026

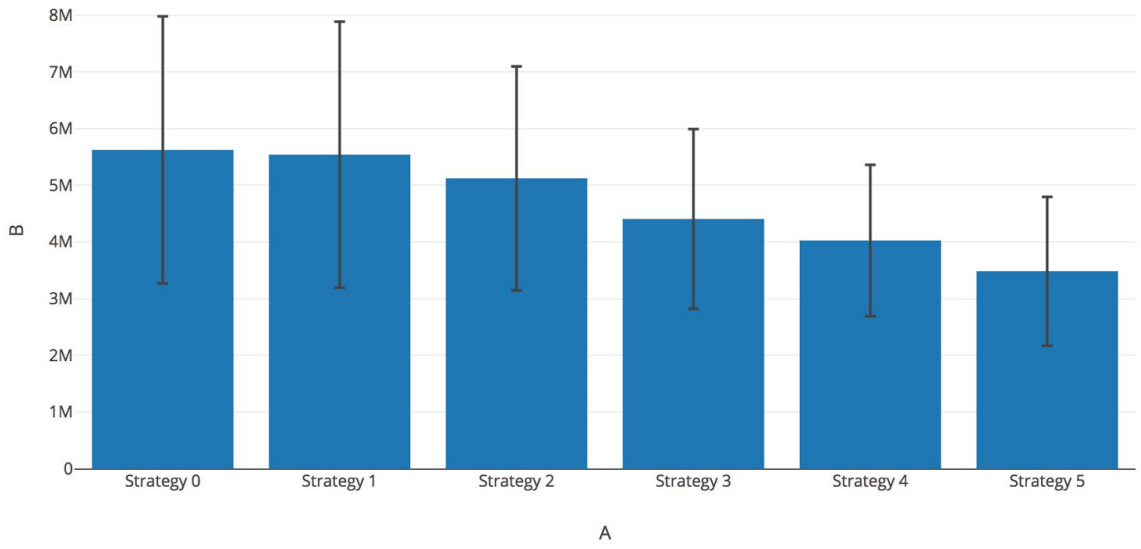
Mean : Number of solutions/second	152953.077
Median : Number of solutions/second	140563.703
SD : Number of solutions/second	35042.44136

Mean : Number of solutions	45208
Median : Number of solutions	47673.5
SD : Number of solutions	6973.487076

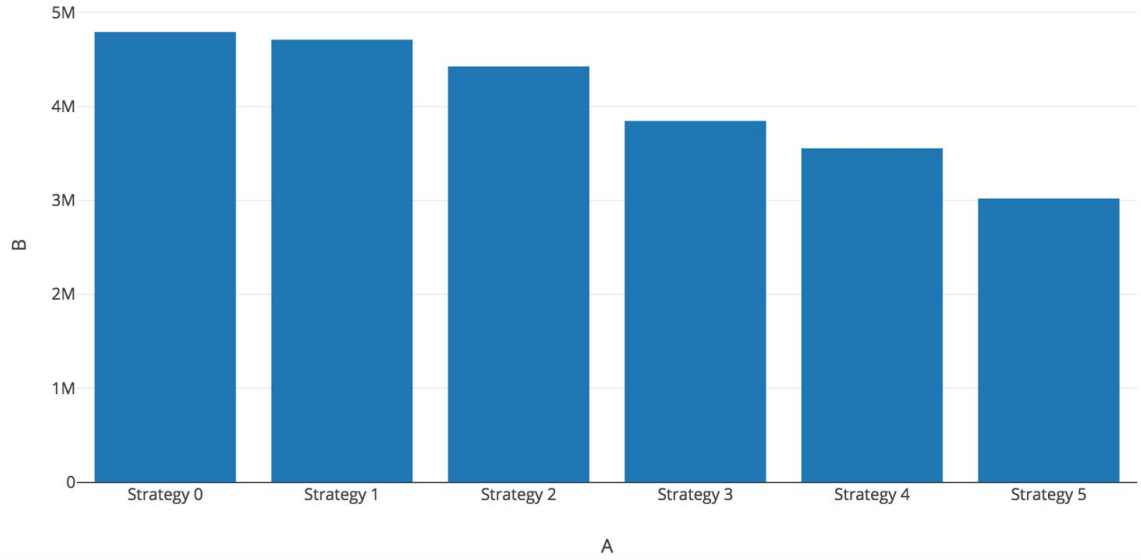
ANALYSIS

Moves per second :

Mean and Standard Deviation

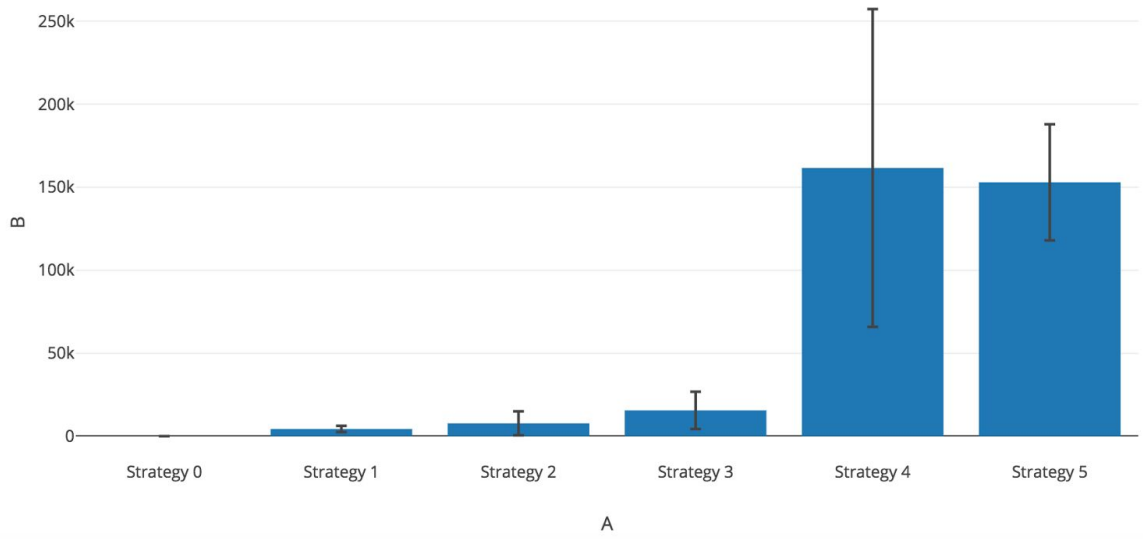


Median

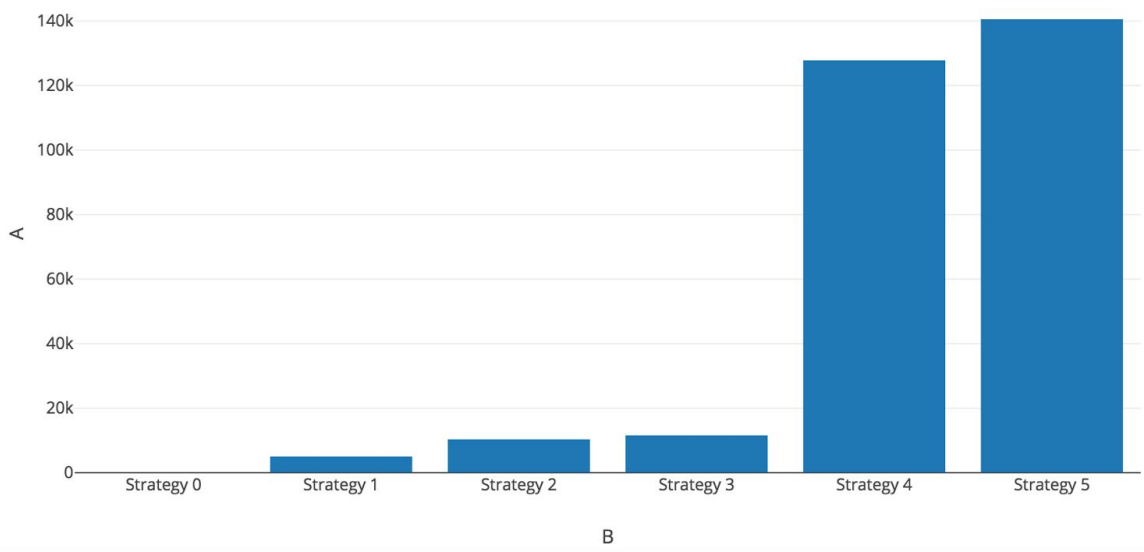


Solutions per second :

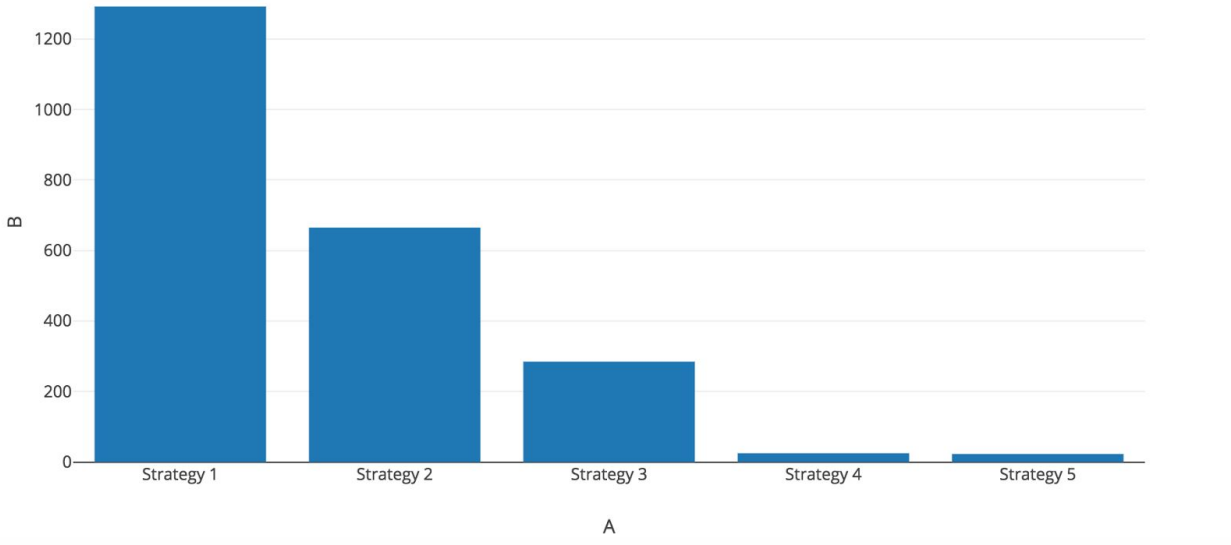
Mean and Standard Deviation



Median



Moves per solution



TAKEAWAYS :

In this assignment, we observe how initially the program chooses the next move randomly thereby generating less optimal solutions. Exploring a node with a larger degree at the start may render a failure solution earlier as we reduce the number of possible moves for all surrounding neighbours. As we progress through the strategies, we enforce stricter constraints upon the method of traversal. For example, Strategy 1 ensures that instead of randomly, a *minimum* valued node is chosen as the next path. To build upon this, Strategy 2 uses a more adaptive technique that learns every time a move is made by setting the states of the affected neighbours (either incrementing or decrementing their respective degrees), yielding more optimal solutions in comparison. With strategy 3, we refine the traversal to account for failures because of multiple nodes with dynamic degree 1. Here, by backing up from this failure node, we limit the erroneous routes that can be taken. While Strategy 3 may traverse longer routes to find that failure node having multiple neighbours with dynamic degree one, Strategy 4 eliminates this use case earlier in the traversal. Strategy 5 combines 2, 3, and 4 to give more optimal solutions because not only does it base its traversal on the dynamic degree but also begins to eliminate failure paths earlier on in the traversal.