Cynthia Joseph (938358)
COMP20003 - Algorithms and Data Structures Assignment 3 – Experimentation – 30/10/20

Introduction:

The purpose of this report is to identify trends in runtime and solution quality in comparison to the initial number of pegs when running a depth first search algorithm to solve a game of peg solitaire. The algorithm will either return the first winning solution or have reached its maximum budget of expanded nodes. Each possible board state defines a vertex, and each legal jump defines an edge in an implicitly defined graph <V,E>. The algorithm scans the board by traversing through the x coordinate, then the y coordinate to look for a peg that can perform a jump (left, right, up, down in that order).

As the number of initial pegs increase, the number of expanded nodes needed for the AI solver to find a solution (one peg remaining) will grow at an exponential rate due to each state generating more edges and nodes. Therefore, it is predicted that the solution quality of the puzzle will be inversely proportional to its starting nodes, meaning more initial nodes will lead to a greater number of pegs left for identical budgets. In addition, the solution quality is predicted to be directly proportional to the budget (maximum number of explored nodes) as the algorithm will have more "chances" to find the winning solution. This report will consider the trends between the number of pegs left as a function of pegs initially in the board before exceeding the given budget, as well as the solution quality of different layouts in comparison to its budget.

Methodology:

1. Set budgets of increments "10,000", "100,000", "1,000,000" and "1,500,000".
2. Test each of the 9 layouts with the given budgets.
3. Record the initial number of pegs, expanded nodes, generated nodes, number of pegs left, expanded nodes per second and total time (in seconds) to complete the peg solver for each layout.
4. Repeat the test 3 times and get the average results for each layout and budget.
5. Create graphs to visualize the trends found in the experiment.

Results from experimentation:

Level 0						
Initial pegs:	Budget:	Expnaded Nodes:	Generated Nodes:	NumPegsLeft:	Expanded/seconds:	Time (seconds):
3	10,000	2	2	1	15	0.131538
3	100,000	2	2	1	14	0.135056
3	1,000,000	2	2	1	15	0.129798
3	1,500,000	2	2	1	12	0.158329
Level 1						
Initial pegs:	Budget:	Expnaded Nodes:	Generated Nodes:	NumPegsLeft:	Expanded/seconds:	Time (seconds):
4	10,000	3	3	1	22	0.132757
4	100,000	3	3	1	22	0.132278
4	1,000,000	3	3	1	23	0.129462
4	1,500,000	3	3	1	21	0.13704
Level 2						
Initial pegs:	Budget:	Expnaded Nodes:	Generated Nodes:	NumPegsLeft:	Expanded/seconds:	Time (seconds):
7	10,000	7	8	1	50	0.139166
7	100,000	7	8	1	52	0.132204
7	1,000,000	7	8	1	50	0.138929
7	1,500,000	7	8	1	54	0.128878
Level 3						
Initial pegs:	Budget:	Expnaded Nodes:	Generated Nodes:	NumPegsLeft:	Expanded/seconds:	Time (seconds):
17	10,000	3541	10282	1	23620	0.149915
17	100,000	3541	10282	1	24212	0.146247
17	1,000,000	3541	10282	1	23802	0.148767
17	1,500,000	3541	10282	1	24521	0.144403
Level 4						
Initial pegs:	Budget:	Expnaded Nodes:	Generated Nodes:	NumPegsLeft:	Expanded/seconds:	Time (seconds):
32	10,000	1065	2418	1	7097	0.150056
32	100,000	1065	2418	1	7793	0.136651
32	1,000,000	1065	2418	1	7796	0.136593
32	1,500,000	1065	2418	1	7769	0.13707

Figure 1 – Data collected for Levels (0-4).

Level 5						
Initial pegs:	Budget:	Expnaded Nodes:	Generated Nodes:	NumPegsLeft:	Expanded/seconds:	Time (seconds):
36	10,000	10000	26495	4	60510	0.165261
36	100,000	100000	359818	3	174056	0.574526
36	1,000,000	1000000	4488464	2	199398	5.015073
36	1,500,000	1090275	4898609	1	203715	5.351951
Level 6						
Initial pegs:	Budget:	Expnaded Nodes:	Generated Nodes:	NumPegsLeft:	Expanded/seconds:	Time (seconds):
44	10,000	10000	29368	5	60105	0.166373
44	100,000	100000	374378	4	175538	0.569676
44	1,000,000	1000000	4481233	3	205773	4.859701
44	1,500,000	1500000	7020668	3	201597	7.440579
Level 7						
Initial pegs:	Budget:	Expnaded Nodes:	Generated Nodes:	NumPegsLeft:	Expanded/seconds:	Time (seconds):
38	10,000	10000	32469	4	58222	0.171755
38	100,000	100000	386440	2	167615	0.596605
38	1,000,000	1000000	4790308	2	193002	5.181267
38	1,500,000	1500000	7173504	2	196898	7.618146
Level 8						
Initial pegs:	Budget:	Expnaded Nodes:	Generated Nodes:	NumPegsLeft:	Expanded/seconds:	Time (seconds):
40	10,000	10000	27562	6	58985	0.169534
40	100,000	100000	349921	4	184903	0.540823
40	1,000,000	1000000	4073028	4	208940	4.786055
40	1,500,000	1500000	6361454	4	206023	7.280712

Figure 2 – Data collected for Levels (5-8).

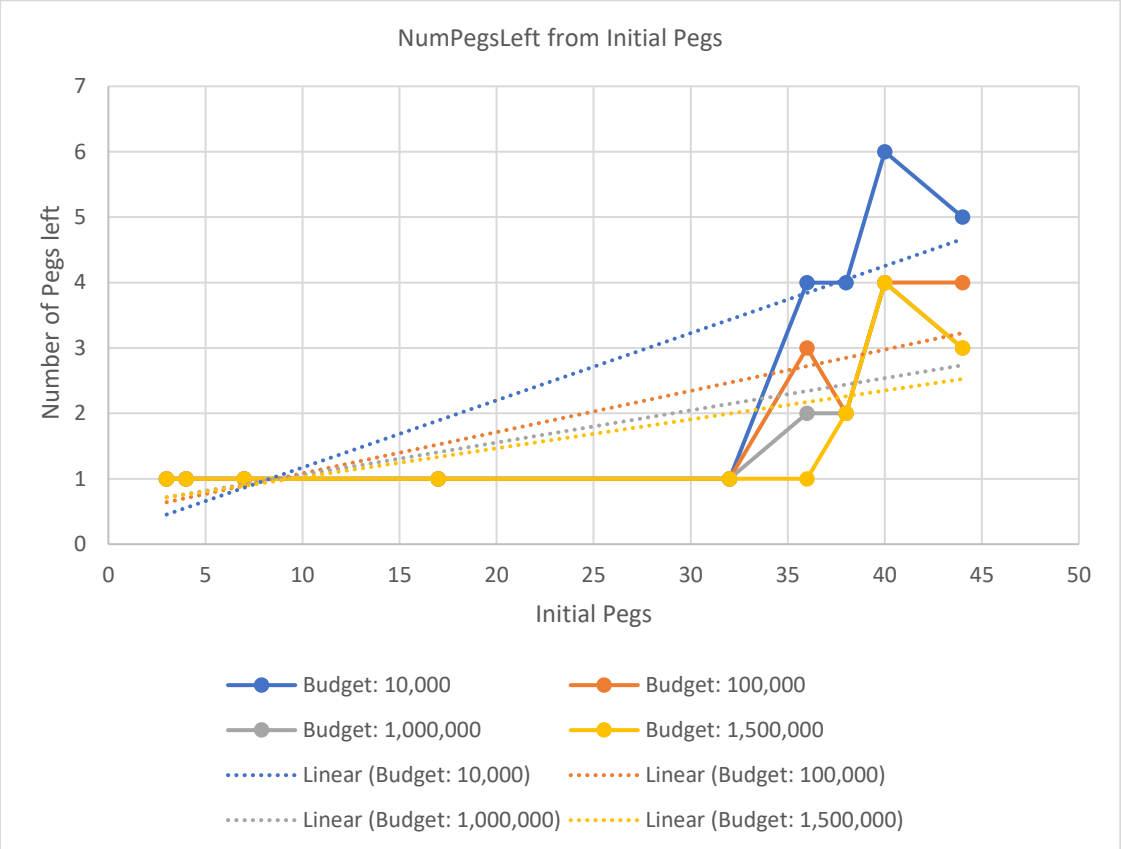

Figure 3a – Number of pegs left as a function of the number of pegs initially in the board (linear trend).

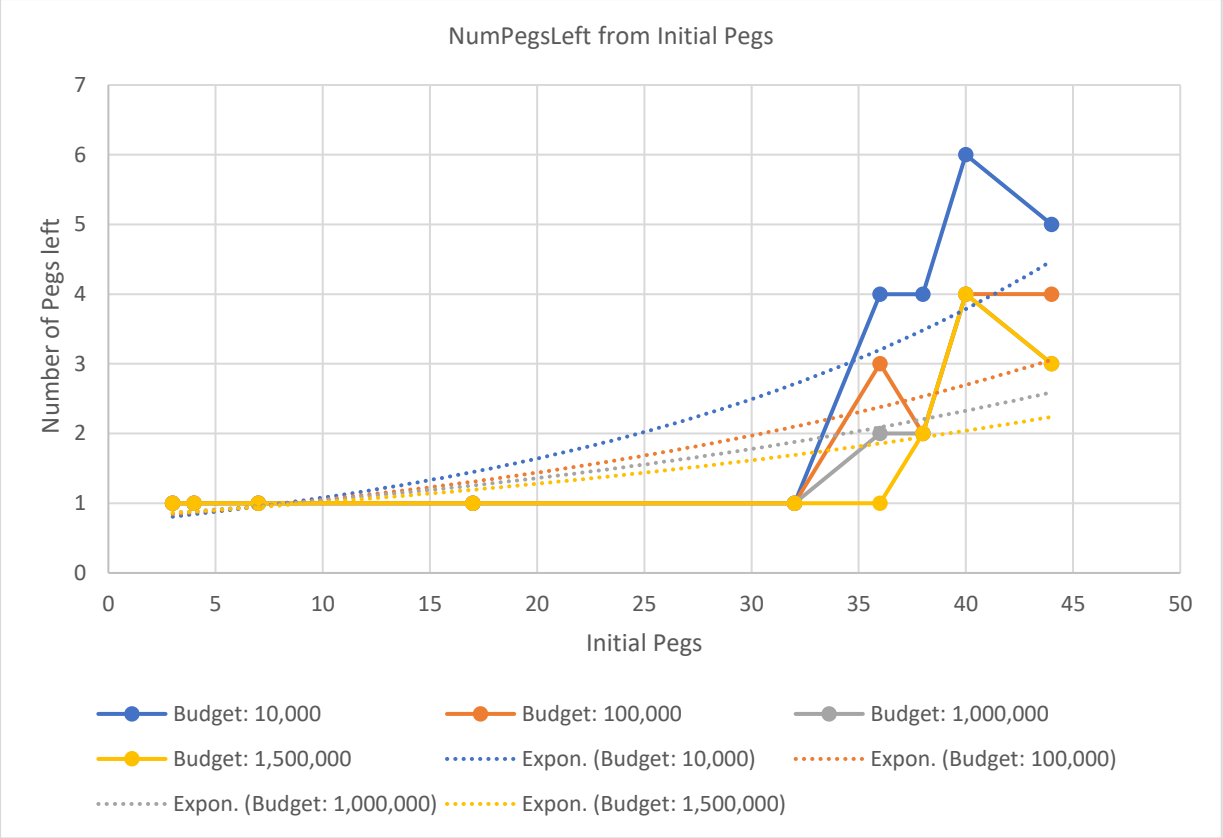

Figure 3b – Number of pegs left as a function of the number of pegs initially in the board (exponential trend).

The results from the experiment showed that there was an upwards trend when comparing the number of remaining pegs from the number of initial pegs when running the algorithm at certain budgets (Figure 3a-b). When given 36 initial pegs and above, the number of remaining pegs begin to spike. The data shows that the AI solver was able to solve the puzzle (1 peg remaining) for levels 1-4 for all tested budgets (10,000 – 1,500,000) (Figure 1). Starting from level 5 (36 pegs), budgets of 10,000 – 1,000,000 were not able to solve the puzzle until a budget of 1,500,000 was given. Levels 6-8 were not able to solve the puzzle for all given budgets (Figure 2). Thus, an exponential trend can be seen, as an increase in initial pegs exponentially increases the remaining number of pegs for all given budgets.

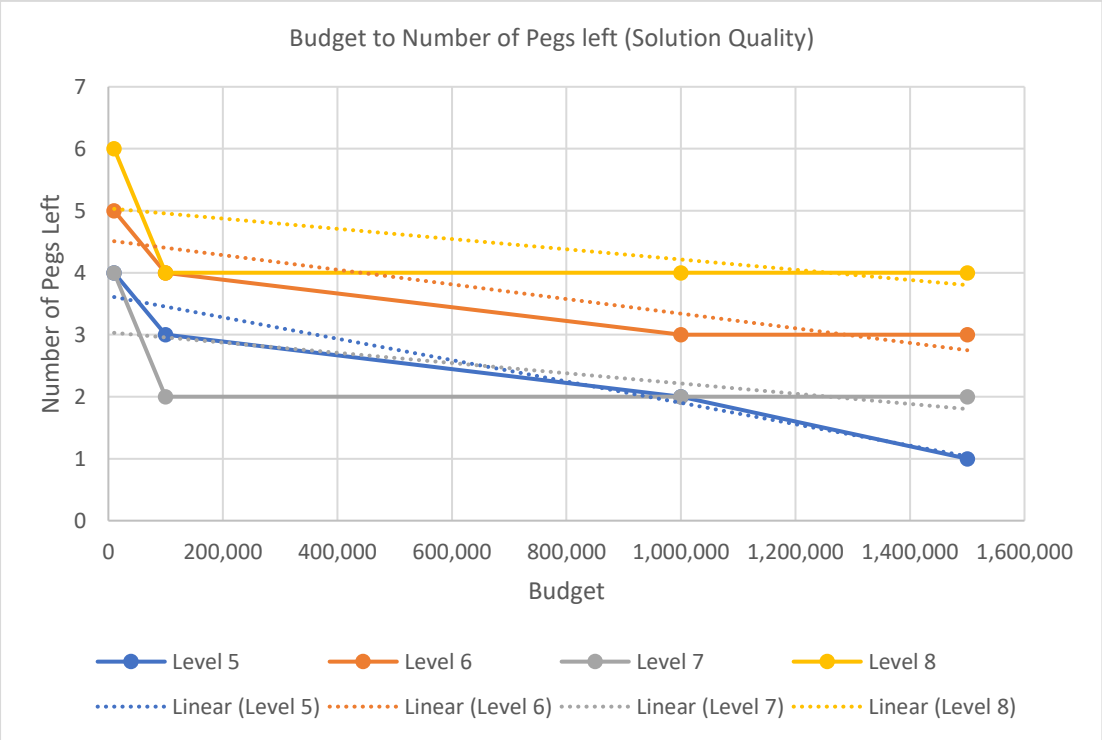

Figure 4a – Budgets as a function of number of pegs left to determine the solution quality in the last 4 layouts (linear trend).

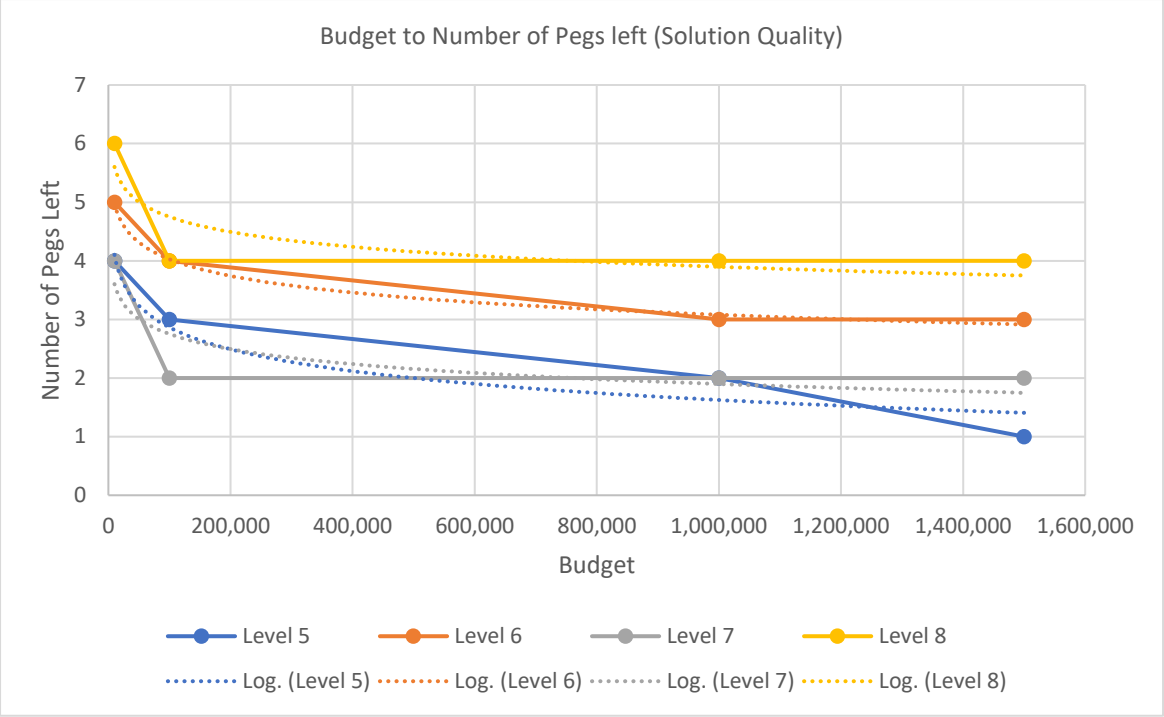

Figure 4b – Budgets as a function of number of pegs left to determine the solution quality in the last 4 layouts (logarithmic trend).

For this experiment, solution quality will be inversely proportional to the number of pegs left after running the algorithm, with a lesser number of pegs left indicating a better solution. The results show that the algorithm will continue to find better solutions as the budget increases, creating a downwards trend until one peg remains or the budget is exceeded (Figure 4a-b). The steepest decline of "number of pegs left" is encountered during earlier budgets, where every level was able to improve its solution by at least one peg from a budget of 10,000 – 100,000. Afterwards, increasing the budget from 100,000-1,000,000 did not improve the solution quality for levels 7-8 but improved the solution quality by one peg for levels 5 and 6 (Figure 2). At a budget of 1,500,000, the algorithm was able to solve level 5. Therefore, a logarithmic trend is observed as fewer declines are seen as budgets increase for each layout.

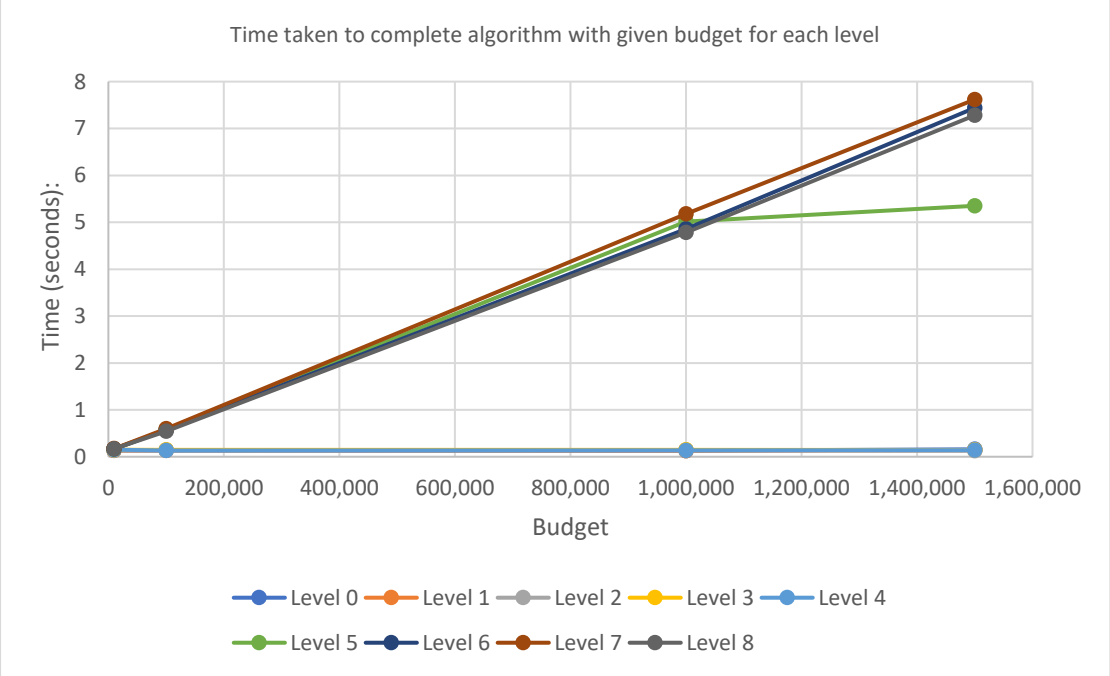

Figure 5 – Time taken to complete algorithm with given budget for each level.

The data shows that the runtime of the algorithm will continue to grow linearly and upwards if no solution is found and the number of explored nodes has exceeded the budget (Figure 5). For levels 0-4, the algorithm completed within 0.15 seconds (Figure 1), as it was able to find the solution without having to exceed the budget.

Data and Comparisons to Theory:

It was predicted that the number of remaining pegs will increase as the algorithm attempts to solve a greater number of pegs. This theory stems from the idea that more pegs will result in an exponential growth in branches to find a better solution to replace its previous one. However, the data shows that this will not always be the case, as a layout of 44 starting pegs was left with fewer pegs than that of a layout with 40 pegs, indicating that a better solution was found for the former (Figure 2).

Lower budgets displayed a steeper upwards trend (larger gradients) when comparing the number of remaining pegs as a function of initial pegs. The steepness of the linear line continues to decrease as the given budget for each layout increases (Figure 3a). This indicates that there is a growing difference between the maximum budget needed to solve more difficult puzzles and coincides with the theory that an exponential growth is present when attempting to solve the puzzles with more pegs (Figure 3b).

The greatest reduction in remaining pegs could be found during the first few budget increments, implicating that better solutions are quickly found and updated within the earlier stages of the search (Figure 4a). After the search exceeded a budget of 1,000,000, it was difficult for the algorithm to better the quality of its solution despite being given an additional 500,000 budget increase. This shows that larger puzzles will exhaust the budget quickly when attempting to update its current best solution.

Conclusion and Recommendations:

Overall, the results from the experiment shows that the AI solver is unable to solve the puzzle after introducing 36 pegs on the board or greater when given a budget of 10,000. As opposed to previous prediction, the initial number of pegs in the board does not always allow for better solution quality when given the same budget. Running the AI on layout 6 of 44 initial pegs resulted in a solution of "3 pegs left" after given a budget of 1,500,000 whereas running the AI on layout 8 of 40 initial pegs with the same budget resulted in a solution of "4 pegs left", showing that increasing the initial number of pegs on the board will not always lead to a decrease in solution quality. Future experiments must test different boards to find a more reliable trend between initial pegs and solution quality. It may also consider the size of the board with differing amounts of empty spaces when computing this trend.

The greatest improvements in solution quality occurred during the earlier tests with smaller budgets as opposed to ones exceeding larger budgets, despite being of smaller increments. This may be due to the exponential complexity of increasing the number of initial pegs, as it is essentially dependent on the number of moves (edges) that branch out from each board state. Therefore, more nodes would have to be explored to find a better solution than the previous.

Additionally, future experiments may involve running the algorithm at greater budgets to determine the minimum budget required to solve each layout. As more initial pegs are given, the number of generated nodes will increase at a very fast rate and thus the time needed to complete the search will grow.