Reuben Cook
RLCOOK
1270283

Assignment 3: Deliverable 2 – Experimentation

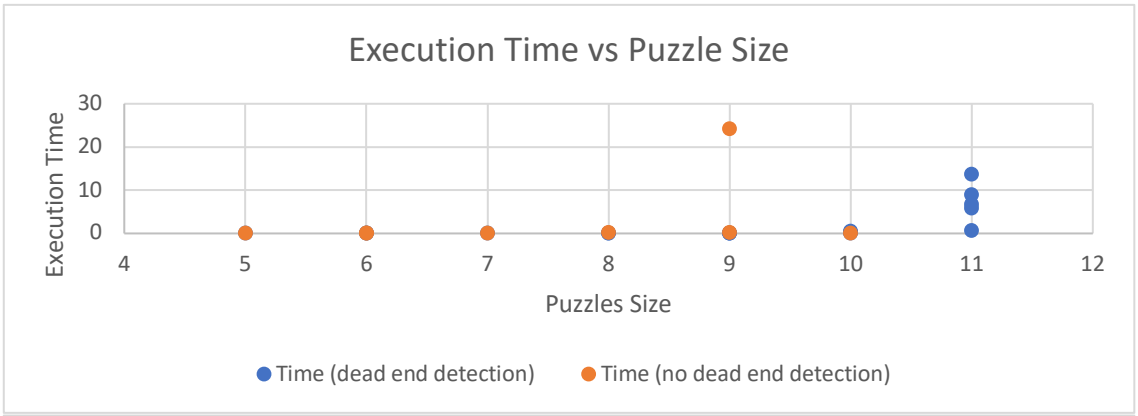
Introduction

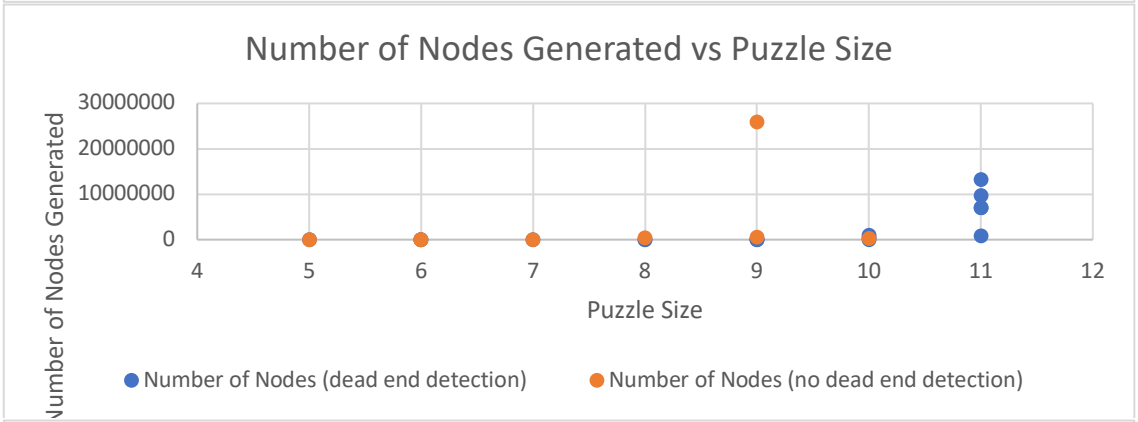
In this assignment Dijkstra's algorithm was used to solve the puzzle flow free. The complexity of Dijkstra's algorithm is usually O(V^2). In this implementation a vertex is a state in the game as such the total number of vertices could expand factorially with the game size. The performance of the program is therefore heavily reliant on the algorithm choosing the best paths and stopping when a path cannot lead to a solution using color selection and dead end detection.
The purpose of this document is to analyse the performance of my solution and the effect of dead end detection on the performance.

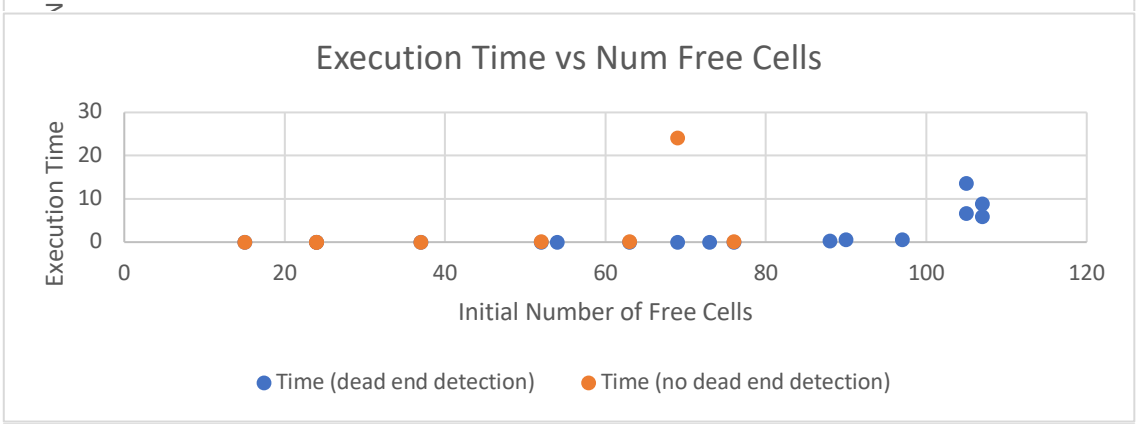
Results

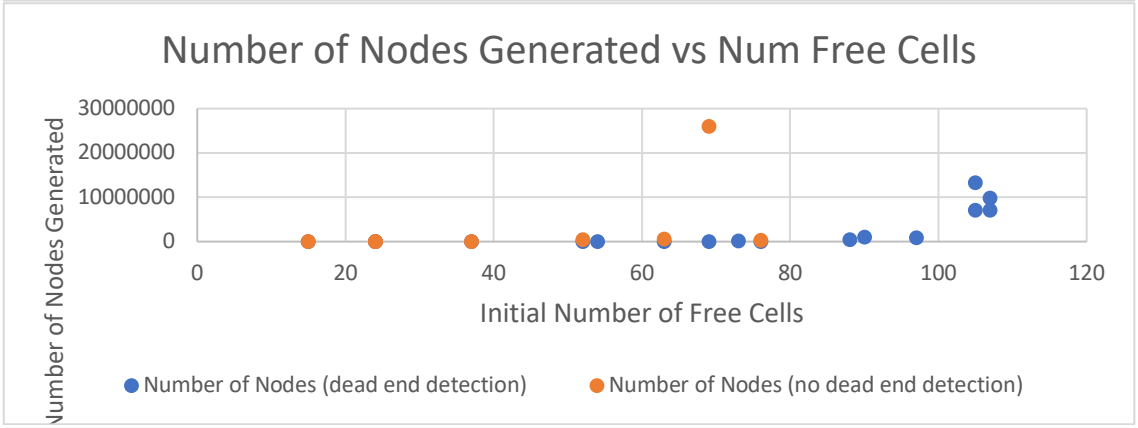
Puzzle Name	Size	Number of Free Cells	Time (dead end detection)	Time (no dead end detection)	Number of Nodes (dead end detection)	Number of Nodes (no dead end detection)
puzzles/deadlock_6x6_01.txt	6	24	0	0	74	202
puzzles/regular_5x5_01.txt	5	15	0	0	16	18
puzzles/regular_6x6_01.txt	6	24	0	0	128	283
puzzles/regular_7x7_01.txt	7	37	0	0.001	213	3,317
puzzles/regular_8x8_01.txt	8	52	0.001	0.127	2,219	409,726
puzzles/extreme_8x8_01.txt	8	54	0.003		5,932	
puzzles/regular_9x9_01.txt	9	63	0.004	0.17	7,031	549,827
puzzles/jumbo_10x10_01.txt	10	76	0.005	0.088	9,016	317,532
puzzles/extreme_9x9_30.txt	9	69	0.016	24.15	34,888	26,027,479
puzzles/extreme_9x9_01.txt	9	73	0.061		123,054	
puzzles/extreme_10x10_30.txt	10	88	0.24		414,633	
puzzles/extreme_10x10_01.txt	10	90	0.545		954,594	
puzzles/jumbo_11x11_01.txt	11	97	0.606		872,016	
puzzles/extreme_11x11_20.txt	11	107	5.855		7,076,023	
puzzles/extreme_11x11_15.txt	11	105	6.65		7,074,370	
puzzles/extreme_11x11_07.txt	11	107	8.926		9,770,583	
puzzles/extreme_11x11_30.txt	11	105	13.681		13,238,659	

Reuben Cook
RLCOOK
1270283









Reuben Cook
RLCOOK
1270283

Discussion

The results are inconclusive regarding the time complexity of my solution, it could be exponential when plotted against both game size and initial number of free cells, game size is much steeper since it is very roughly equivalent to the square root of initial number of free cells.
However there are only a few datapoints due to the large space requirement of the program, and steep complexity. As such the shapes of the graphs are not clear.

The impact of dead end detection seems to be dramatic, allowing the algorithm to solve problems much larger, however the impact on the actual complexity is unclear since there is no visible trend. It is likely reducing the growth rate since there will be much fewer paths needed to be traversed when it detects dead ends but it could just be shifting the graph to the right, only allowing it to solve slightly harder problems but having little difference in performance with huge problems above size 12.

The results do show significant variability between puzzles of similar sizes. This is best seen in the result without dead end detection for 9x9 being 24 seconds while for the same algorithm a 10x10 took less than a second. This shows that some puzzles are much easier for the algorithm to solve than others. Looking at that 10x10 puzzle, it seems that each color has a short path which makes sense since Dijkstra's algorithm will always prioritise the shortest path.