

Creating a Hybrid Maze Generation Algorithm

SID:1608183

**MOD002691 Final Project**

**Final Project Report**

**BSc (Hons) Computer Gaming Technology**

**Submitted: May 2019**

# Abstract

Many algorithms exist for generating procedural mazes. Unfortunately, many of them suffer from weakness which affects the mazes they generate. This paper looks randomised Prim’s and recursive backtracking and how the two algorithms can be integrated together to eliminate the weaknesses that each algorithm suffers from. The project uses the recursive backtracking algorithm to generate the path from the maze start to end, and then uses randomised Prim’s to generate branches out. To analyse the effectiveness of this, data was collected on randomised Prim’s, recursive backtracking and the hybrid version of the two, to look at path length, branch count, branch size, maximum branch size and generation time. Results showed that the hybrid generation had the solution path that a standalone recursive backtracking maze generates, with similar branch properties which can be generated using randomised Prim’s.

Contents

[Abstract ii](#_Toc7778974)

[1 Introduction 1](#_Toc7778975)

[1.1 Aims 1](#_Toc7778976)

[1.2 Background 2](#_Toc7778977)

[2 Methodology 7](#_Toc7778978)

[3.1 Randomised Prim’s 10](#_Toc7778979)

[3.2 Prim’s Generation Information 10](#_Toc7778980)

[3.3 Recursive Backtracking 12](#_Toc7778981)

[3.4 Recursive Backtracking Generation Information 15](#_Toc7778982)

[3.5 Hybrid 15](#_Toc7778983)

[3.6 Hybrid Generation Information 16](#_Toc7778984)

[3 Implementation and Results 18](#_Toc7778985)

[4 Discussion 30](#_Toc7778986)

[5 Conclusion 35](#_Toc7778987)

[5.1 Summary 35](#_Toc7778988)

[5.2 Achieved Aims 35](#_Toc7778989)

[5.3 Future Work 36](#_Toc7778990)

[6 References iv](#_Toc7778991)

[Appendix 1 vi](#_Toc7778992)

[Appendix 2 xii](#_Toc7778993)

[Appendix 3 xix](#_Toc7778994)

[Appendix 4 xxvi](#_Toc7778995)

[Appendix 5 xxvii](#_Toc7778996)

[Appendix 6 iv](#_Toc7778997)

[Appendix 7 iv](#_Toc7778998)

[Appendix 8 iv](#_Toc7778999)

[Appendix 9 iv](#_Toc7779000)

# 1 Introduction

## 1.1 Aims

The primary aim of this project is to create a maze generation algorithm, that combines two or more currently existing maze generation algorithm to create a maze that meets the projects maze requirements. They are:

* Long, non-linear path connecting the start and end of the maze.
* Maximum number of branches.
* Fast maze generation time.
* Optional generation into Unity.

The success of this project will be based upon how well the final hybrid maze algorithm meets the projects maze requirements and how much so. A long, non-linear path is key in any maze as it is the main problem of the maze that has to be solved. If the path is short and linear then no difficulty for the maze is provided and this would fail to function as a challenging maze. Furthermore, the number of branches that derive from the solution path is important as for every one of these that is connecting, there is greater chance for the maze participant to go down a branch which will lead to dead end/s. The more dead ends there are, the greater the difficulty of the generated maze. In addition, the generation time will need to be quick so that quick maze creation and play can be achieved. If the maze takes to long as a product then it offers less incentive as a product to be used for maze generation. Whilst a complex maze generator that takes a longer period to generate is fair, the balance between complexity and generation time must be balanced. Finally, as the artefact is a maze generator that has to output the generated maze to a selected software, the maze generator must be able to generate and then create the maze within Unity so that the maze can be used.

The methodology chapter of this study will look at the chosen algorithms that will be implemented into the hybrid algorithm and how they work. It will also look at how to generate results for each of the existing algorithms which is implemented into the hybrid version. These results are regarding, path length, generation time, and branch data. These generated results will be what determines how successful this project is and if the hybrid algorithm is successful or not.

The implementation and results chapter of this study will present the generated data in the forms of graphs and tables to make the data concise and easy to understand. The complete data sets will be in the Appendix, with data averages and key results used in the graphs and tables within the implementation and results chapter.

Finally, the discussion chapter analyses the data to see how each of the individual algorithms performed so that the hybrid algorithm has a base of comparison. Each aspect of the maze will be analysed in detail to see if the hybrid fulfils the requirements to make it a successful artefact of this project.

## 1.2 Background

The dissertation focuses on maze generation algorithms and because of this will feature research and review on existing maze generation algorithms and how they are implemented in software.

Graph

Graph theory uses mathematical based graph structed to model relations between connecting objects and is a triplet comprised of a vertex set *V(G),* an edge set *E(G),* and a mapping function *ε : E → V2* called *relation and incidence.* This vertex set associates a relation pair of vertices {x, y} to each edge *e ∈*  E. Edge *e* and its associated vertices are incident. Furthermore, one graph attribute is *connectivity* which is used to determine between connected and disconnected graphs. A connected graph has all of the vertices connected which means that there is an unbroken chain of edges which translates to a path. However, a disconnected graph with vertices which are not connected and thus is a broken chain of edges (Foltin, 2008). Karlsson visualises graphs in Figure 1.1 (Karlsson, 2018).

|  |
| --- |
| **Figure 1.1 (a) displaying a disconnected graph and (b) displays a connected graph, with red dots as vertices and black lines are connecting edges.** |
|  |

Spanning Tree

A tree graph is a connected graph that contains no cycles and has exactly one path between each pair of vertices. A connected graph with *n* vertices and *n*-1 edges is a tree graph. If any edge in the graph were to be removed, the graphs *connectivity* would be disconnected, while adding an edge would cause an additional relation making a loop/ cycle in the graph (Foltin, 2008). A spanning tree is a tree graph which means that it contains all graph vertices, and can be defined as a “*maximal set of edges that contains no cycle, or as a minimal set of edges that connects all vertices”* (E. Black, 2014 ; Zhang and Yin, 2012).

Weighted Graph

A weighted graph is a tree graph, in which every edge has an associated label. A minimal weighted spanning tree has a weight value equal to or less than every other potential spanning tree of the graph, where as a maximum weighted spanning tree has the greatest or equal to value than all other variations of the spanning tree graph (Foltin, 2008).

The A\* search algorithm uses weighted graphs to determine the lowest and/or highest value path from one vertex on the graph to another.

Greedy

The greedy algorithm uses weight values and picks the best immediate local solution to the problem, but this may not always be the most optimal solution. The algorithm however is strong at finding a solution to a given problem quickly (E. Black, 2014).

Procedural Content Generation

Procedural content generation (PCG) refers to an algorithmic creation of content by using a set of parameters and rules. Togelius, states that PCG, is content that affects everything apart from Non-Player Characters (NPC’s) behaviour and the game engine itself. Furthermore, Togelius states that there are 3 main types of PCG (Togelius, Yannakakis, Stanley, and Browne, 2011). PCG allows for content to be generated automatically and is slowly becoming a common workplace tool for the games industry amongst others (Linden, Lopes and Bidarra, 2013). Many indie games utilise PCG tool for aspects such as level generation and item creation. Most notable games which feature PCG are Minecraft (Minecraft, 2011) and Terraria (Terraria, 2011) which uses PCG for their level/world creation. In addition, the Borderlands franchise (2K, 2009) generates loot-able weapons which can be found throughout the games.

Mazes are good example products of PCG as they can require few limitations to generate a maze, and with further limitations and requirements through maze algorithms and the PCG software itself, more complex mazes can be generated. Simple algorithms such as randomized Prim’s algorithms can generate vast quantities of varying mazes whilst still following the definitions of a maze (Prims, 1957 ; Foltin, 2008). As well as Prims, algorithms such as Recursive Backtracker (*Recback*), and Recursive Division (*RecDiv*) are use more complex algorithms to generate more complex mazes.

Perfect Maze

Kim and Crawfis, state that a perfect maze every node in a spanning tree is connected without cycles, that every node in the maze is accessible and connecting to the same spanning tree, and that the start and end points of the maze can be placed anywhere, while still guaranteeing a unique solution to the maze. Adding to this, this means that every node in the spanning tree/ maze has a unique path to every other node in the spanning tree and thus for any single spanning there is numerous possible perfect mazes (Kim and Crawfis).

Randomised Prim’s

Prim’s algorithm can find a minimum spanning tree for a connected graph that is connected and weighted (Foltin, 2008). However, the randomised versions of Prim’s algorithm don’t use a weighted graph to connected vertices, but instead connects vertices randomly without biased (Karlsson, 2018). Furthermore, Jamis Buck states that randomised Prim’s can be a disconnected spanning tree that starts from a single point, and then expands outwards randomly from that point until no more points are disconnected from the minimal spanning tree (Buck, 2011). In addition, randomised Prim’s algorithm is an example of a wall carving algorithm which creates a new cell and makes unused adjacent cells walls (Jeong and Kim, 2016).

Recursive Backtracking / Depth-First-Search

The recursive backtracker/backtracking algorithm is based upon the Depth-First Search method, which creates a list of connecting vertices until the current vertex has no available connections, at which point, it backtracks its list of connecting vertices until the current one has a available connection, then starts connecting vertices again. This repeats until the graph has all the vertices connected (Foltin, 2008). Bucks, describes recursive backtracking as a path carving algorithm, which treats every cell as a wall, until it is connected to the spanning tree which then makes the cell a path (Bucks, 2010).

Recursive Division

Recursive division is a wall adding algorithm, which starts by dividing a maze either vertically or horizontally, and then adds an opening along the division line which has become a wall in the maze. (Foltin, 2008). This process repeats until every division has been completed and this algorithm creates a minimal spanning tree with the start and end being able to be placed anywhere.

# 2 Methodology

As this study involves generating mazes and aims to find either a more efficient or/and better output maze, the mazes will need to be generated on a platform which can record data and output the results of each maze generation. The game engine Unity (version: 2018.2.0f2) will be used as the generation platform to display generated mazes and to record and show data that will be collected. Within Unity, C# scripts will be created and developed externally in Microsoft Visual Studio 2017, which will virtually create the mazes that will be generated, collect and store data and Unity be used to visually represent the maze if required. To ensure fair data results and collection, testing variables and apparatus will be consistent throughout testing, data collection and processing to eliminate any variables that could influence an affect software processing and performance, which would then affect data results.

Figure 2.1 states the specifications other PC that all testing will be conducted on.

|  |  |
| --- | --- |
| **Figure 2.1 Specifications of PC Used for Testing.** | |
| Processor | Intel Core i7-7700HQ @ 2.80GHz |
| RAM (Random Allocated Memory) | 8192MB |
| Virtual RAM | 9351MB |
| Graphics Card | NVIDIA GeForce GTX 1060 w/ 2987MB Memory |
| Storage | SSD: 120GB  Hybrid HDD/SSD: 1000GB |
| Operating System | Windows 10 Home |

For each of the maze generating algorithm that will be used will meet set requirements to be regarded as a successful generation.

For testing, Prim’s, recursive backtracking and a hybrid version which utilises both algorithms will need to generate mazes which will meet set requirements to be regarded as a successful generation.

All three of the maze generators will need to fulfil the following requirements:

* Perfect maze; Every node in the graph is used, no loops meaning that the tree is a tree graph, a start and end point.
* Two dimensional.
* Equal grid dimensions so that every maze generator a tree graph with equal number of vertices.
* Solvable; the start point must be connected to the exit within the tree graph.
* Determinism; each generator can repeat itself and output identical mazes.

Each of the maze generators will generate square mazes to eliminate possible generation result variation caused by a rectangular maze grid. The start point will be in the bottom most left corner of the maze grid, with the exit point being in the upper most right corner of the maze grid. The start and exit are in these locations so as the longest achievable straight line within a square grid is the hypotenuse from one corner to the far diagonal corner. Each maze is going to have a custom dimension variable, that will generate square mazes and will uses the dimensions 10, 20, 30, 40, and 50. These sizes are set at equal increments, so data analysis of each maze algorithm van be equally compared. Furthermore, each maze generator will need the option to output the generated maze into Unity so that the maze can be visually represented. To avoid the instantiation of the maze within Unity affecting the generation of the mazes, all the maze will be generated first before any instantiation of objects in Unity occurs.

For each maze generator, a 2-dimensional int array will be used to store the values of each cell within the maze. As the default value of every element in an int array is 0 and the algorithms being used are path carving, if an element in the array is 0, it is a wall, however if it is a 1, it is either a path or a node. As the maze generators are connecting node together, much like how a spanning tree connects vertices together, each maze generator will need to create a list of vertices which will all be connected in a successfully generated maze, much like a connected spanning tree. Figure 2.2 below shows the pseudocode for how each grid point is generated.

|  |
| --- |
| **Figure 2.2 Pseudocode for how each map grid point is generated.** |
| *N = Width of Maze in Nodes*  *HashSet of Vector2Int to store nodes*  *for (int i=1 to N)*  *{*  *for (int j=1 to N)*  *{*  *Add Vector2Int (i,j) to HashSet.*  *}*  *}* |

This list of nodes will save checking if an adjacent position within the maze array is valid as the generator can instead check if the position is a node or not. If the list contains the adjacent position, then it is a node, if not it is outside of the maze array field and is not available. Figure 2.3 below shows the pseudocode for the initialisation of the map data array.

|  |
| --- |
| **Figure 2.3 Pseudocode for the initialisation of the map data array.** |
| *Int[,] MapData*  *Int NumberOfNodes*  *Int ArraySize = (NumberOfNodes \* 2)-1*  *// Sets distance so that every node has a space between it and the next adjacent node.*  *ArraySize+=2*  *// Adds 2 to the ArraySize so that the maze has an outside edge of walls.*  *MapData = new int[ArraySize,ArraySize]* |

## 3.1 Randomised Prim’s

The randomised Prim’s algorithm starts with an available node, which then has its neighbours added to list. In this randomised Prim’s maze generation, the starting node is the start of the maze, but could be any node within the grid of nodes, as with Prim’s every node is connected on the same spanning tree. This list is used when the algorithm expands the maze, by picking a random node within this list, and connecting it to an adjacent existing node. This new node then adds its unused neighbours to the list and removes itself as it has now been expanded to. This process repeats until the requirement is met, which in this maze generation example is until there are no available points left within the list. Figure 2.4 below shows the randomised Prim’s pseudocode.

|  |
| --- |
| **Figure 2.4 Pseudocode for Randomised Prim’s algorithm.** |
| *HashSet of Nodes*  *Int Array MapLayout*  *HashSet of UsedNodes*  *HashSet of AvailableNodes*  *Do*  *{*  *CurrentNode is random node from AvailableNodes*  *Set CurrentNode position in MapLayout to empty*  *HashSet of adjacent nodes to CurrentNode*  *AvailableNodes adds adjacent nodes HashSet*  *Foreach (Node in AvailableNodes)*  *{*  *Set position in between Node and Current Node to empty*  *}*  *AvailableNodes removes CurrentNode*  *UsedNodes adds CurrentNode*  *} while (UsedNodes is not equal to HashSet of Nodes)* |

## 3.2 Prim’s Generation Information

Due to this study looking at how to create a better maze generation algorithm, data on how randomised Prim’s performs and the type of maze that it generates needs to be recorded. The first set of data that will be recorded is the length of the solution path that connects the starting point to the end point and how many nodes the solution path takes. For this, a depth-first search (DFS) will be used to search through the maze until it reaches end point. This will then return the path in the form of a C# stack and the number of nodes in the stack will be recorded.

To collect this data efficiently, a new script is created so that a set number of randomised Prim’s mazes are generated virtually and with each of the mazes being solved with a DFS algorithm and then the length of each maze’s solution path being recorded into an array. This is done to automate the process of gathering the data so that the process is only run once but will return a user defined amount of results. Figure 2.5 shows the pseudo code for a loop that generates a maze and then find the solution path before recording the solution path length.

|  |
| --- |
| **Figure 2.5 Pseudocode for a looping maze generation which records solution path.** |
| *Int LoopingAmount*  *Int[] PathLengths*  *For (i=0; i< LoopingAmount; i++)*  *{*  *Generate Maze as shown in* ***Figure 2.4***    *List of ConsumedNodes*  *Stack of CurrentNodes*  *Add starting node to Stack.*  *Do*  *{*  *Node = Top Node in Stack*  *Search for connecting nodes of Node.*  *If one is found add it to Stack.*  *If not remove Node from Stack.*  *Add Node to ConsumedNodes*  *} while Stack does not contain end node.*  *PathLengths[i] = Stack length.*  *}* |

To collect data on the branches that are generated a script is developed that will generate a randomised Prim’s maze, find the solution path between the start point and end point, and will then search for any nodes that connect to this solution path. The nodes that it will find, will be the first nodes in each of the branches of dead ends that connect to the main path. Then for each of these nodes, a DFS algorithm will search each branch to find how many nodes it contains and will then return this value to an array. The purpose of this is to collect data on the number of branches that a randomised Prim’s algorithm makes, but to also find out the average amount of nodes that each branch contains.

## 3.3 Recursive Backtracking

The second algorithm that is used as a maze generator, is recursive backtracking (RecBack) which is also known as recursive backtracker or depth-first search. The maze generator will generate its maze until all available nodes have been connected to the spanning tree regardless of if the end node has been connected to the spanning tree graph or not. The generator will follow the set rules of no loop, start and end point and no inaccessible areas as previously stated and will produce a maze with matching dimensions to the five sets of dimensions. The first step of the recursive backtracking generator is creating a HashSet of coordinates which will represent the position of every node inside a 2-dimesional array. This 2-dimensional array initialisation is shown in Figure 2.2. The purpose of this HashSet is to check if an adjacent node to a selected node is valid and within the size of the 2-dimensional array which will hold the map values. As the maze dimensions are customisable within the Unity inspector menu and in the C# script itself, the array which holds the map data will need to be initialised during runtime and not pre initialised in the C# script. The pseudocode for the initialisation of the map data array is shown in Figure 2.3.

To generate the maze by connecting the nodes together, the process starts with the starting point of the maze, collects all its neighbours, if only one neighbour is found the current node is removed from the Stack of nodes, and if more than one neighbour is found, then it picks one at random to connect too and then adds it to the maze Stack. Every time the current node has no available neighbours, it is removed from the list. When a node is connected to, it is added to a HashSet of nodes which stores all used nodes. This process repeats until the number of nodes in the HashSet of connected nodes is equal to the number of nodes in the maze which means that every node has been connected to at least once.

For a node to be considered a neighbour, it first must not have been previously connected too, meaning that it is not in the list of connected nodes, and that it is a valid node point in the maze. The reason why a random node neighbour is picked from a list of all available neighbouring nodes is so that a random valid neighbour is picked without influence from which direction the neighbour is. If the method which searches for neighbouring nodes stops when a valid neighbour is found, for it to be random direction, a set of directions would need to be shuffled and randomised each time a node searches for a neighbour. The process of picking a random neighbour from a list of available neighbours saves this unnecessary shuffling of directions. Furthermore, the node is removed from the Stack if only a single neighbour is found because, that node cannot be used again as it will be adding its single available neighbour to the Stack, meaning that if the backtracking comes back to the node in mention, it will return zero neighbours. By removing the node with a single neighbour at this point, it prevents the do while loop from selecting the node again, which would only end up removing it after zero neighbours are found. Figure 2.6 is the pseudocode for the recursive backtracking algorithm.

|  |
| --- |
| **Figure 2.6 Pseudocode for Recursive Backtracking.** |
| *Stack of nodes*  *Hashset of nodes that can no longer be connected to*  *Hashset of all grid points.*  *Do*  *{*  *Current node is top grid point in Stack of nodes*  *Get all available neighbours*  *If one neighbour is found*  *{*  *Remove current node from Stack and add the single found neighbour.*  *Set the space in between neighbour and current node to 1*  *So that it is empty which is equivalent to a path.*  *}*  *If two or more available neighbours are found*  *{*  *Add one at random to Stack*  *Set the space in between neighbour and current node to 1*  *So that it is empty which is equivalent to a path.*  *}*  *If no available neighbours are found*  *Remove current node from stack.*  *} while Hashset of nodes which has been connect is not equal to the number of grid spaces in the maze.* |

The reason why a Stack is used in the maze generation is because only the top element in a Stack can be accessed with Stack.Peek() and because a Stack is last in, first out, it returns the most recent element added to the Stack. This has benefits over a C# list, in which the last in element would need to be accessed by using the count number of elements in the list minus one. Furthermore, due to the size of the list changing as nodes are added and removed, every time the most recent node will need to be accessed, the count of elements would need to be calculated.

## 3.4 Recursive Backtracking Generation Information

One set of data that is collected regarding recursive backtracking, is the number of nodes in the solution path that connects the start node to the end node. To collect all the nodes, the DFS method is used to search through the maze until it reaches the end node. The process of doing this, is the exact same as the recursive backtracking maze generation, but instead of seeing if a node has been carved into the maze yet, the space in between the current node and the adjacent node is checked in the map grid array to see if the value is equal to one. If the value is one, it means that a path has been carved between each of the nodes and that the nodes are connecting. If the value is equal to zero, it means that whilst the nodes are adjacent to each other, they do not directly link to each other and thus means they are not neighbours of each other.

## 3.5 Hybrid

For the hybrid maze that is the combination of randomised Prim’s and recursive backtracking, the solution path between the start and end node will be generated first, and branches will be generated from the solution path using randomised Prim’s until all nodes are connecting to the graph tree of the maze. The path which generates using a Stack, will follow algorithm until it reaches the end node. If the algorithm backtracks, it adds the removed nodes into a second Stack which will store unavailable nodes. This prevents the path Stack from adding back nodes which did not provide a path from the starting node to the end node. This path generation fulfils the requirement of the maze start and end node connecting. This method can be compared to a process of elimination with nodes either being valid or not valid for the path.

The branches that derive from the solution path are then generated using a modified version of randomised Prim’s, which will start with the solution path as all available nodes that can be expanded from, instead of a single node. As the branch generation will loop until there is no available nodes to connect to.

## 3.6 Hybrid Generation Information

As the generated solution path using recursive backtracking is generated in an empty node grid, and functionally works the same as the recursive backtracking generator for the path, the path data for this hybrid algorithm will follow the same trends as pattern as the path which the recursive backtracking generator will produce. As this is the case, the recursive backtracking path data generator will also be used for the hybrid path data.

For quicker and more automated data generation and collection, a base classed is used to store maze solution path length, branch count, minimum branch size, and maximum branch size. To store this information, four arrays are used which lengths all match the looping amount of the data collection. If the looping amount is five, then each array length will be five, and five mazes generated with their unique data sets returned to the arrays. Each maze class which will generate a maze and collect data, is given an index number which matches the current loop amount and when data is returned from that maze, its data is put into the index position of each array. This means that each maze has its data in matching positions in each array.

For the branch data using the hybrid algorithm, a maze generator is made which will generate a user defined number of mazes, will collect the solution path length, then will search through every branch connecting to the solution path and will store all the branch lengths into an array. This looping generator for each maze, creates a new maze class for each loop iteration, and within this class will generate the maze, collect and store this data, and then return the desired information to the parent class. For the branch information, every branch length is stored inside a List, and then to get the minimum and maximum branch sizes, the .Min() and .Max() functions are used to get these. Once the path solution length, branch count, minimum branch length and maximum branch length is collected, the information is returned to the parent class. Figure 2.7 shows the pseudocode for this method.

|  |
| --- |
| **Figure 2.7 Pseudocode for the Hybrid algorithm which loops maze generation and returns its data to the parent class.** |
| *Parent Class*  *{*  *Loop Amount*  *Maze Size*  *Array of Path Length*  *Array of Branch Count*  *Array of Minimum Branch Size*  *Array of Maximum Branch Size*  *For loop each maze generation.*  *{*  *Generate A Maze*  *Collect Data*  *Return Data*  *}*  *}*  *Maze Class*  *{*  *Generate Recursive Backtracking Maze*  *Set Path Length*  *Generate Randomised Prim’s Branches*  *Get every branch length*  *Get Min and Max*  *Return Data.*  *}* |

# 3 Implementation and Results

The randomised Prim’s maze generator was looped a total of 50 times to virtually generate and maze and record the time to do so but did not generate game objects into Unity for each maze. The looping function automated the collection of generating mazes and stored each timing inside a C# array which was then serialized in the Unity inspector menu. This method first started as a single run method but was changed into a looping function for efficiency and quicker data recording. The timing results were collected in milliseconds (ms) and were recorded in 5, 10 loop segments for ease of reading the times inside the Unity inspector menu. This was repeated for each of the maze sizes as stated in the methodology and were recorded into tables for each maze size, with an average calculated. Table 3.1 shows the average timing for each stated maze size using randomised Prim’s.

|  |
| --- |
| **Table 3.1 Randomised Prim’s maze generation time.** |
|  |

The method of collecting maze generation times that randomised Prim’s used, was the same for recursive backtracking and the hybrid algorithm and was done for quick data collection. Table 3.2 shows the average timing for recursive backtracking and Table 3.3 shows the average timing for the hybrid algorithm.

|  |
| --- |
| **Table 3.2 Recursive backtracking maze generation time.** |
|  |

|  |
| --- |
| **Table 3.3 Hybrid maze generation time.** |
|  |

Figure 3.1 below is a graph of the data from the 3 maze generation algorithms and their average time to generate each maze size. The graph uses the data from Table 3.1, Table 3.2, and Table 3.2.

|  |
| --- |
| **Figure 3.1 Graph of the Maze Generation Time for Randomised Prim’s, Recursive Backtracking and Hybrid.** |
|  |

APPENDIX 1 shows the whole timing data set for Prim’s, recursive backtracking and the hybrid algorithm.

As stated in the methodology, the solution path size for randomised Prim’s and recursive backtracking was going to be collected to see how much of the maze grid is used up by the solution path. This data set was not recorded for the hybrid algorithm as its solution path shares the same algorithm as recursive backtracking and would result in the same trends and correlations. This value is calculated as a percentage of nodes that are the path itself. Due to time constraints this process was not looped and had to be repeated inside Unity 50 times for each maze dimension, for both randomised Prim’s and recursive backtracking. To calculate the percentage, the length of the path in nodes was returned into the inspector menu inside Unity and recorded inside Excel, which then automated the percentage calculation using the total nodes in the maze, and the path length. This formula is below in Equation 3.1.

|  |
| --- |
|  |
| Equation 3.1.  *P = Percentage (%)*  *SPL = Solution Path Nodes (#)*  *TN = Total Nodes (Maze Width \* Maze Height)* |

Table 3.4 shows the randomised Prim’s solution path average results for each maze dimension.

|  |
| --- |
| **Table 3.4 Randomised Prim’s solution path average data for each maze dimension.** |
|  |

Table 3.5 shows the recursive backtracking solution path average results for each maze dimension.

|  |
| --- |
| **Table 3.5 Recursive backtracking solution path average data for each maze dimension.** |
|  |

APPENDIX 2 shows the complete solution path data for randomised Prim’s.

APPENDIX 3 shows the complete solution path data for recursive backtracking.

Whilst it wasn’t planned, solution path data was collected for the all three algorithms as several nodes used in the branch data collection required this information. From this, Figure 3.2 shows the solution path averages for all three algorithms at all 5 maze dimensions and Table 3.6 is the data used in Figure 3.2

|  |
| --- |
| **Figure 3.2 Maze Generation Solution Path Averages for Each Dimension.** |
|  |

|  |
| --- |
| **Table 3.6 Maze Generation Solution Path Average at Each Dimension.** |
|  |

The solution path data script had further functionality added, which was the ability to visually show multiple solution paths of the desired script overlapping on a map grid. This was done to collect visual representation of the solution path so any patterns and correlations could be analysed. The branches length for these visual representations were not recorded or colour coded per solution path generated as the sole purpose was the overall placement of solution path nodes and not individual solution path placements. This was done for both randomised Prim’s and recursive backtracking but not the hybrid algorithm as it shares the same solution path generation algorithm as recursive backtracking.

APPENDIX 4 shows overlapping solution paths generated by randomised Prim’s, with (a) displaying a single solution path in a 30X30 grid, (b) displaying 5 solution paths in a 10X10 grid, (c) displaying 5 solution paths in a 20X20 grid, (d) displaying 5 solution paths in a 30X30 grid, (e) displaying 5 solution paths in a 40X40 grid, and (e) displaying 5 solution paths in a 50X50 grid.

APPENDIX 5 shows overlapping solution paths generated by recursive backtracking, with (a) displaying a single solution path in a 30X30 grid, (b) displaying 5 solution paths in a 10X10 grid, (c) displaying 5 solution paths in a 20X20 grid, (d) displaying 5 solution paths in a 30X30 grid, (e) displaying 5 solution paths in a 40X40 grid, and (e) displaying 5 solution paths in a 50X50 grid.

The branch data for the randomised Prim’s was generated inside Unity, and then displayed inside the inspector menu. This information did not use any looping and had to be collected individually per test, copied into Excel, and then the Unity script restarted to collect another set of data. Looping was not implemented due to time constraints, but per data set individually did not take much time to generate. The data collected was the solution path length, branch count, minimum branch size, maximum branch size, and branch size range. The average branch nodes were also collected but was calculated within Excel using Equation 3.2.

|  |
| --- |
|  |
| Equation 3.2.  *ABN = Average Branch Nodes*  *TN = Total Nodes*  *PL = Path Length*  *BC = Branch Count* |

APPENDIX 6 shows the average and whole data set results for the randomised Prim’s branch data results for each maze dimension.

The branch data for the recursive backtracking was generated inside Unity, and displayed in the inspector menu, the same way that the randomised Prim’s branch data was represented. Due to the greater complexity of mazes that recursive backtracking generates, looping was not implemented due to time constraints and the risk of if time spent adding generation looping would take longer to implement than the individual testing. However, looping was implemented for the hybrid algorithm to collect the same branch data as collected for the randomised Prim’s and recursive backtracking. The collected data is presented the same way as the randomised Prim’s branch data and uses Equation 3.2.

APPENDIX 7 shows the average and whole data set results for the recursive backtracking branch data results for each maze dimension.

APPENDIX 8 shows the average and whole data set results for the hybrid branch data results for each maze dimension.

Figure 3.3 below shows the average branch count for all three of the maze generation algorithms at each maze dimension.

|  |
| --- |
| **Figure 3.3 The Average Branch Count for each Maze Generation Algorithm at Each Dimension.** |
|  |

Table 3.7 presents the data used in Figure 3.3.

|  |
| --- |
| **Table 3.7 Maze Generation Average Branch Count for Each Algorithm at Each Dimension.** |
|  |

Figure 3.4 presents the average number of nodes for each maze generation algorithm and each maze dimension.

|  |
| --- |
| **Figure 3.4 Maze Generation Average Branch Size for each Maze Generation Algorithm at Each Dimension.** |
|  |

Table 3.8 presents the average branch size of each maze generation algorithm at each maze dimension.

|  |
| --- |
| **Table 3.8 Average Branch Size of Each Maze Generation Algorithm at Each Maze Dimension.** |
|  |

Figure 3.5 presents the average maximum branch size for each maze generation algorithm at each dimension.

|  |
| --- |
| **Figure 3.5 Maze Generation Average Maximum Branch Size for each Maze Generation Algorithm at Each Dimension.** |
|  |

Table 3.9 presents the average maximum branch size for each maze generation algorithm at each maze dimension.

|  |
| --- |
| **Table 3.9 Average Maximum Branch Size for Each Maze Generation Algorithm at Each Maze Dimension.** |
|  |

Figure 3.6 shows a 30X30 maze generated by randomised Prim’s, Figure 3.7 shows a 30X30 maze generated by recursive backtracking and Figure 3.8 generates a 30X30 maze using a hybrid algorithm.

|  |
| --- |
| **Figure 3.6 30X30 Maze Generated by Randomised Prim’s.** |
|  |

|  |
| --- |
| **Figure 3.7 30X30 Maze Generated by Recursive Backtracking.** |
|  |

|  |
| --- |
| **Figure 3.8 30X30 Maze Generated by the Hybrid Algorithm.** |
|  |

# 4 Discussion

Figure 3.1and Table 3.1, 3.2 and 3.3 show evidence that the randomised Prim’s algorithm is the fastest for each maze dimension, followed by recursive backtracking which is minimum dimension is 1.3X slower, and at maximum tested dimension is 8.2X slower. Comparing the randomised Prim’s timing to the hybrid algorithm timing shows that at minimum size, the hybrid is 10.3X slower and at maximum tested size is 39.4X slower. These results are expected because Figure 2.4 demonstrates how randomised Prim’s single generation condition is to expand until all grid nodes are consumed, whilst Figure 2.6 demonstrates that recursive backtracking must generate its path forward and backtrack until all grid nodes are consumed. Figure 2.7 demonstrates how the hybrid algorithm must follow the recursive backtracking algorithm until it has connected the start node to the end node (Figure 2.6) and then must generate the maze branches using randomised Prim’s (Figure 2.4) until all remaining unconnected grid nodes are connected. As the hybrid algorithm uses both randomised Prim’s and recursive backtracking, the average generation time trend in Figure 3.1 is expected.

Comparing Figures 3.4 and 3.5 shows that at the minimum tested dimension size, that the randomised Prim’s is 19.12% of all maze grid nodes, compared to the 49.52% of the recursive backtracking solution path, which is 2.59X greater than the randomised Prim’s. This trend is present throughout the 4 other maze dimensions that were tested, and at maze size of 50X50, the randomised Prim’s path is only 4.05% of all nodes, compared to recursive backtracking’s 29.71% which is 7.34X greater. As the hybrid algorithm utilises the recursive backtracking algorithm for its solution path and one of the stated aims is to create a better non-linear path, that the recursive backtracking algorithm is suitable.

Figure 3.6 demonstrates that at dimensions 40X40 and greater, that the hybrid solution path nodes count with margin of variation is 500 nodes. This result is not expected as the hybrid path generation uses recursive backtracking for its solution path and could suggest that not enough data samples were collected in implementation for an accurate average. Furthermore, the small maze dimensions of 10X10, 20X20 and 30X30 have the greatest average number of nodes for the solution path using the hybrid algorithm which further supports that not enough data samples were collected for an accurate average.

Figure 3.6, Table 3.6 and APPENDIX 4 (a-f) demonstrates that the solution path generated by randomised Prim’s at each tested maze dimension is placed along the hypotenuse of the maze grid as previously mentioned in methodology. This can be visually seen in APPENDIX 4(a) within a margin of variation and Table 3.6 shows that at each solution path average for randomised Prim’s is the maze dimension multiplied by 2 within the margin average of 2.8 nodes at 50X50 maze size. It is important to note that the minimum solution path length for each dimension is Equation 4.1 below.

|  |
| --- |
|  |
| **Equation 4.1**  *ML = Minimum Length.*  *DW = Dimension.* |

The solution path averages generated by recursive backtracking (Table 3.6) are far greater than the minimum length of each dimension size meaning that solution paths at each tested dimension size has variation and can be seen in APPENDIX 5 (a-f). This information further suggests that recursive backtracking generates are non linear path from the start node to end node, where as randomised Prim’s does to due to lack of variation as previously stated. As previously stated, one of the aims of the hybrid algorithm was to create a long, non linear solution path, and the current interpreted results interpret that using recursive backtracking has done this.

For randomised Prim’s, Figure 3.3 and Table 3.7 show that at each tested maze dimension that randomised Prim’s has the fewest average branch count except for dimension 10X10. With the information in Figure 3.4 and Table 3.8, randomised Prim’s is seen to have the greatest average branch size for dimensions 20X20, 30X30, and 50X50 which with the average branch count information previously stated, that randomised Prim’s for 3 maze dimensions has the fewest branches, but the greatest average branch size when compared to recursive backtracking and the hybrid algorithm. This information is expected as randomised Prim’s has been shown to have the smallest solution path, which returns the fewest possible branch points, but the greatest number of nodes available for these branches to use. Figure 3.5 and Table 3.9 show that randomised Prim’s maximum branch size average is the second greatest out of the three tested algorithms at all five maze dimensions and at its furthest, is fifteen average nodes away from recursive backtracking which has greatest maximum branch size average for all five maze dimensions.

With recursive backtracking, Figure 3.3 and Table 3.7 show that for recursive backtracking with each tested maze dimension it has the second greatest average branch count except for the 10X10 maze dimension, in which recursive backtracking has the fewest average branch count. The information stated in Figure 3.4 and Table 3.8 inform that recursive backtracking has the second greatest average branch size for all dimensions with exception to 10X10 and 40X40 dimensions, in which recursive backtracking is the greatest. This means that whilst recursive backtracking has the second greatest branch count in four of the tested dimensions, each of these only three of these average branch sizes is less than that of the randomised Prim’s. This result could be linked to how in Figure 3.2 recursive backtracking has an average solution path average which is at least 2X greater in every maze dimension than that of randomised Prim’s. APPENDIX 5(a) could further support this as it shows multiple path nodes with all neighbouring grid nodes being a part of the solution path. However, without further testing this cannot be proven. Figure 3.5 and Table 3.9 prove that for every maze dimension, recursive backtracking has the greatest average maximum branch length and whilst this is most likely caused by an algorithm (Figure 2.6) expanding and backtracking until no further nodes are possible, it cannot be proven with the current information. This does show that with the previously stated information that recursive backtracking generates can generate the largest range of branch sizes, but with the average on three of the maze dimensions being less than randomised Prim’s.

For the hybrid algorithm, Figure 3.3 and Table 3.7 show that for the hybrid algorithm all five of tested maze dimensions using the hybrid algorithm generated the greatest average amount of branches out of the three algorithms. Furthermore, for the first four tested maze dimensions, the number of averaged generated branches is over 2X what the second greatest average number of branches is. This means that the greater average solution path that the hybrid algorithm generates has more nodes, which can then generate randomised Prim’s branches which derive from the solution path. In addition, Figure 3.4 and Table 3.8 show that out of the three algorithms, the hybrid generator has the lowest average branch size for all five tested maze dimensions. However, this is expected as the hybrid algorithm generates the greatest number of branches meaning that all nodes that are not the solution path are divided more greatly over more branches, which results in smaller branches. This trend is the opposite for randomised Prim’s which has the least number of branches but for three of the five maze dimensions had the greatest average branch size. Following this, Figure 3.5 and Table 3.9 show that the hybrid average maximum branch size was at best 8.92 nodes away from randomised Prim’s which was the second-best algorithm for average maximum branch size. From the hybrid algorithm branch information, it shows that it generated the greatest number of branches with the lowest average size which with the solution path data means that the hybrid algorithm generates a long solution path maze, with many branches deriving from it.

Overall, the data collected implies that a hybrid maze generator that uses a recursive backtracking solution path and randomised Prim’s branches generates a maze, which meets the aims of this study. The solution path length is long and doesn’t follow any trends such as the randomised Prim’s solution path as shown in APPENDIX 4 but instead uses a longer and less linear solution path (APPENDIX 5) which the recursive backtracking algorithm generates. The use of randomised Prim’s branches generates a greater number of smaller branches of that generated by recursive backtracking.

# 5 Conclusion

## 5.1 Summary

In conclusion, the hybrid maze generation algorithm, which first uses recursive backtracking for a solution path generation and then randomised Prim’s to generate the branches deriving from the solution path was successful. Firstly, the solution path of the hybrid is long and non-linear path which at the upper most tested maze dimensions had a length around 500 nodes, against the minimum number of required nodes of 99. With this, solution path covers an average of 29% of the total maze grid which ultimately creates a challenging path for the user to follow. The solution paths length means at 50X50 size, an average of 310 branches giving optimal branch counts, whilst maintaining a complex solution path.

Further evidence to support the successfulness of the artefact and the hybrid generation, is that the it only used high branch count of randomised Prim’s and not the solution path that the algorithm generates. In addition, the hybrid utilises the long paths that are generated by recursive backtracking whilst avoiding the lower branch count that it generates compared to randomised Prim’s. Furthermore the hybrid algorithm is able to generate a 50X50 maze which contains 2500 nodes, quickly in an average of 898.98 milliseconds and whilst this compared to both randomised Prim’s and recursive backtracker is significantly slower, the results can be considered better by how the hybrid algorithm meets the success criteria stated in the introduction.

## 5.2 Achieved Aims

The four main aims of this project were to create a maze generation algorithm, that could generate a long non-linear solution path, maximum branch count, fast generation time and optional ability for the maze to be generated into Unity. The hybrid maze generation has met the success criteria, as it generates a long path, many branches deriving from the solution path, a quick generation time that has an average generation time of under a second and can instantiate the maze into Unity.

## 5.3 Future Work

For further work and analysis of creating hybrid maze generating algorithms in the future, different algorithms could be combined instead of randomised Prim’s and recursive backtracking. Furthermore, for the hybrid algorithm, only 50 data sets for each maze dimension was completed and for greater accuracy in the future, an increased number of data sets would yield a more accurate average. In addition, the branch data collection for randomised Prim’s and recursive backtracking was not made so that the process of collecting the data could be automated and this is something that could be improved if similar projects were to happen.

# 6 References

707 Games, 2011. *Terraria.* [PC Game] 707 Games.

Black, P.E., 2005. *Greedy Algorithm.* [online] Available at: https://www.nist.gov/dads/HTML/greedyalgo.html [Accessed: 19/04/2019]

Buck, J., 2011. *Maze Generation: Prim’s Algorithm.* [online] https://weblog.jamisbuck.org/2011/1/10/maze-generation-prim-s-algorithm.html [Accessed: 17/04/2019]

Foltin, M., 2008. *Automated Maze Generation and Human Interaction*. [online] Available at: https://www.semanticscholar.org/paper/Automated-Maze-Generation-and-Human-Interaction-Foltin/e882d6927e29fa1afb2a0cb7e9f2a01fa540ec31 [Accessed: 19/04/2019]

Ganley, J. L., 2014. *Minimum Spanning Tree.* [online] Available at: https://xlinux.nist.gov/dads/HTML/minimumSpanningTree.html [Accessed: 19/04/2019]

Gearbox Software, 2009. *Borderlands.* [Game] 2K Australia.

Kim, P.H., & Crawfis, R., 2015. *The Quest for the Perfect Perfect-maze.* [online] 2015 Computer Games: AI, Animation, Mobile, Multimedia, Educational and Serious Games (CGAMES), pp. 65-72.

Mojang, 2011. *Minecraft.* [PC Game] Mojang.

van der Linden, R., Lopes. R., & Bidarra. R., 2014 *Procedural Generation of Dungeons.* [online] IEEE Transactions on Computational Intelligence and AI in Games, 6(1), pp. 78-89. Doi 10.1109/TCIAIG.2013.2290371

Togelius, J. Yannakakis, G.N., Stanley, K.O., & Browne, C., 2011. *Search-Based Procedural Content Generation: A Taxonomy and Survey.* [online] IEEE Transactions on Computational Intelligence and AI in Games, 3(3), pp. 172-186. Doi 10.1109/TCIAIG.2011.2148116

Zhang, T., & Yin, Y., 2012. *The minimum spanning tree problem with non-terminal set.* [online] Information Processing Letters, 112 (17-18), pp. 688-690. Doi 10.1016/j.ipl.2012.06.012

# Appendix 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Size (Nodes)** | **Node Count** | **Randomised Prim's (ms)** | **Recursive Backtracking (ms)** | **Hybrid (ms)** |
| 10 | 100 | 0.2 | 0.26 | 2.06 |
| 20 | 400 | 2.02 | 7.42 | 27.38 |
| 30 | 900 | 5.62 | 30.68 | 118.48 |
| 40 | 1600 | 12.52 | 80.96 | 364.32 |
| 50 | 2500 | 22.82 | 187.66 | 898.98 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Size** | **Nodes** |  |
|  | 10 | 100 |  |
|  | **Randomised Prim's** | **Recursive Backtracking** | **Hybrid** |
|  | 2 | 2 | 6 |
|  | 0 | 0 | 1 |
|  | 0 | 0 | 1 |
|  | 0 | 0 | 2 |
|  | 0 | 1 | 1 |
|  | 0 | 0 | 1 |
|  | 0 | 0 | 1 |
|  | 0 | 0 | 1 |
|  | 0 | 0 | 2 |
|  | 0 | 0 | 2 |
|  | 2 | 2 | 6 |
|  | 0 | 0 | 2 |
|  | 0 | 0 | 1 |
|  | 0 | 0 | 2 |
|  | 0 | 0 | 2 |
|  | 0 | 0 | 2 |
|  | 0 | 0 | 1 |
|  | 0 | 0 | 2 |
|  | 0 | 1 | 2 |
|  | 0 | 0 | 2 |
|  | 2 | 2 | 5 |
|  | 0 | 0 | 2 |
|  | 0 | 0 | 2 |
|  | 0 | 0 | 2 |
|  | 0 | 0 | 2 |
|  | 0 | 0 | 1 |
|  | 0 | 0 | 2 |
|  | 0 | 0 | 2 |
|  | 0 | 0 | 2 |
|  | 0 | 0 | 2 |
|  | 2 | 2 | 5 |
|  | 0 | 0 | 1 |
|  | 0 | 0 | 1 |
|  | 0 | 0 | 2 |
|  | 0 | 1 | 2 |
|  | 0 | 0 | 2 |
|  | 0 | 0 | 1 |
|  | 0 | 0 | 2 |
|  | 0 | 0 | 2 |
|  | 0 | 0 | 2 |
|  | 2 | 2 | 6 |
|  | 0 | 0 | 1 |
|  | 0 | 0 | 1 |
|  | 0 | 0 | 2 |
|  | 0 | 0 | 2 |
|  | 0 | 0 | 2 |
|  | 0 | 0 | 2 |
|  | 0 | 0 | 1 |
|  | 0 | 0 | 2 |
|  | 0 | 0 | 2 |
| **Average:** | 0.2 | 0.26 | 2.06 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Size** | **Nodes** |  |
|  | 20 | 400 |  |
|  | **Randomised Prim's** | **Recursive Backtracking** | **Hybrid** |
|  | 4 | 9 | 24 |
|  | 2 | 7 | 33 |
|  | 2 | 7 | 22 |
|  | 2 | 7 | 30 |
|  | 2 | 7 | 16 |
|  | 2 | 8 | 26 |
|  | 2 | 6 | 32 |
|  | 2 | 8 | 28 |
|  | 2 | 8 | 24 |
|  | 3 | 7 | 35 |
|  | 2 | 10 | 23 |
|  | 2 | 8 | 26 |
|  | 2 | 10 | 37 |
|  | 2 | 8 | 22 |
|  | 2 | 8 | 28 |
|  | 2 | 8 | 20 |
|  | 2 | 8 | 24 |
|  | 2 | 7 | 26 |
|  | 1 | 6 | 29 |
|  | 2 | 8 | 23 |
|  | 4 | 9 | 21 |
|  | 2 | 7 | 28 |
|  | 2 | 7 | 23 |
|  | 2 | 7 | 24 |
|  | 2 | 8 | 22 |
|  | 2 | 7 | 33 |
|  | 2 | 6 | 33 |
|  | 2 | 7 | 25 |
|  | 1 | 6 | 21 |
|  | 2 | 8 | 34 |
|  | 4 | 9 | 24 |
|  | 2 | 6 | 36 |
|  | 1 | 7 | 24 |
|  | 1 | 7 | 27 |
|  | 1 | 7 | 30 |
|  | 2 | 9 | 26 |
|  | 2 | 6 | 27 |
|  | 2 | 8 | 23 |
|  | 2 | 8 | 34 |
|  | 2 | 7 | 29 |
|  | 3 | 8 | 33 |
|  | 1 | 6 | 23 |
|  | 2 | 7 | 20 |
|  | 2 | 6 | 34 |
|  | 2 | 7 | 35 |
|  | 1 | 8 | 23 |
|  | 2 | 7 | 30 |
|  | 2 | 8 | 30 |
|  | 2 | 7 | 35 |
|  | 2 | 6 | 34 |
| **Average:** | 2.02 | 7.42 | 27.38 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Size** | **Nodes** |  |
|  | 30 | 900 |  |
|  | **Randomised Prim's** | **Recursive Backtracking** | **Hybrid** |
|  | 7 | 30 | 110 |
|  | 5 | 31 | 113 |
|  | 6 | 31 | 159 |
|  | 5 | 34 | 139 |
|  | 6 | 31 | 104 |
|  | 5 | 31 | 89 |
|  | 6 | 31 | 84 |
|  | 5 | 37 | 100 |
|  | 5 | 30 | 84 |
|  | 6 | 32 | 152 |
|  | 8 | 31 | 131 |
|  | 5 | 29 | 125 |
|  | 6 | 30 | 161 |
|  | 6 | 31 | 123 |
|  | 8 | 33 | 137 |
|  | 7 | 29 | 165 |
|  | 5 | 29 | 82 |
|  | 5 | 31 | 120 |
|  | 6 | 32 | 88 |
|  | 5 | 28 | 116 |
|  | 7 | 33 | 151 |
|  | 5 | 33 | 91 |
|  | 5 | 29 | 152 |
|  | 6 | 30 | 91 |
|  | 5 | 30 | 94 |
|  | 5 | 30 | 90 |
|  | 6 | 32 | 135 |
|  | 5 | 42 | 111 |
|  | 5 | 31 | 110 |
|  | 6 | 29 | 148 |
|  | 8 | 30 | 94 |
|  | 5 | 31 | 155 |
|  | 6 | 29 | 132 |
|  | 5 | 30 | 151 |
|  | 5 | 24 | 150 |
|  | 6 | 29 | 140 |
|  | 5 | 30 | 90 |
|  | 6 | 34 | 93 |
|  | 6 | 28 | 127 |
|  | 5 | 32 | 94 |
|  | 7 | 29 | 136 |
|  | 5 | 27 | 119 |
|  | 5 | 27 | 107 |
|  | 5 | 27 | 86 |
|  | 5 | 32 | 127 |
|  | 5 | 28 | 104 |
|  | 5 | 30 | 123 |
|  | 5 | 34 | 110 |
|  | 5 | 33 | 135 |
|  | 5 | 30 | 96 |
| **Average:** | 5.62 | 30.68 | 118.48 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Size** | **Nodes** |  |
|  | 40 | 1600 |  |
|  | **Randomised Prim's** | **Recursive Backtracking** | **Hybrid** |
|  | 14 | 87 | 295 |
|  | 12 | 62 | 415 |
|  | 12 | 80 | 272 |
|  | 14 | 96 | 284 |
|  | 12 | 77 | 263 |
|  | 12 | 86 | 350 |
|  | 12 | 69 | 267 |
|  | 13 | 81 | 353 |
|  | 12 | 78 | 317 |
|  | 12 | 91 | 346 |
|  | 14 | 90 | 332 |
|  | 12 | 86 | 297 |
|  | 12 | 79 | 455 |
|  | 12 | 84 | 439 |
|  | 13 | 94 | 473 |
|  | 12 | 76 | 373 |
|  | 12 | 88 | 434 |
|  | 12 | 81 | 449 |
|  | 12 | 81 | 276 |
|  | 11 | 79 | 352 |
|  | 15 | 79 | 328 |
|  | 12 | 75 | 280 |
|  | 12 | 75 | 402 |
|  | 12 | 88 | 460 |
|  | 12 | 85 | 370 |
|  | 12 | 81 | 282 |
|  | 12 | 84 | 294 |
|  | 13 | 87 | 368 |
|  | 12 | 75 | 260 |
|  | 12 | 88 | 463 |
|  | 14 | 81 | 287 |
|  | 12 | 91 | 462 |
|  | 14 | 71 | 322 |
|  | 13 | 88 | 417 |
|  | 12 | 85 | 407 |
|  | 13 | 74 | 311 |
|  | 13 | 72 | 334 |
|  | 12 | 88 | 526 |
|  | 11 | 72 | 401 |
|  | 12 | 93 | 416 |
|  | 14 | 83 | 291 |
|  | 12 | 78 | 439 |
|  | 13 | 76 | 407 |
|  | 13 | 81 | 358 |
|  | 15 | 75 | 282 |
|  | 12 | 81 | 278 |
|  | 12 | 79 | 406 |
|  | 12 | 85 | 366 |
|  | 12 | 55 | 444 |
|  | 14 | 78 | 513 |
| **Average:** | 12.52 | 80.96 | 364.32 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Size** | **Nodes** |  |
|  | 50 | 2500 |  |
|  | **Randomised Prim's** | **Recursive Backtracking** | **Hybrid** |
|  | 24 | 188 | 750 |
|  | 23 | 179 | 622 |
|  | 22 | 192 | 1093 |
|  | 22 | 186 | 888 |
|  | 22 | 179 | 1163 |
|  | 22 | 158 | 772 |
|  | 22 | 191 | 1089 |
|  | 22 | 217 | 1043 |
|  | 23 | 203 | 597 |
|  | 22 | 178 | 1143 |
|  | 25 | 174 | 913 |
|  | 23 | 180 | 1051 |
|  | 24 | 202 | 858 |
|  | 23 | 170 | 1052 |
|  | 24 | 210 | 561 |
|  | 22 | 165 | 901 |
|  | 22 | 188 | 742 |
|  | 22 | 163 | 1084 |
|  | 24 | 192 | 867 |
|  | 23 | 205 | 1064 |
|  | 26 | 162 | 758 |
|  | 23 | 206 | 898 |
|  | 23 | 206 | 953 |
|  | 22 | 169 | 935 |
|  | 23 | 203 | 1052 |
|  | 23 | 189 | 784 |
|  | 23 | 175 | 745 |
|  | 23 | 175 | 986 |
|  | 22 | 201 | 1209 |
|  | 22 | 177 | 727 |
|  | 24 | 194 | 891 |
|  | 24 | 148 | 640 |
|  | 22 | 213 | 823 |
|  | 22 | 202 | 1011 |
|  | 24 | 183 | 1095 |
|  | 22 | 215 | 787 |
|  | 22 | 188 | 798 |
|  | 22 | 214 | 736 |
|  | 24 | 190 | 1051 |
|  | 25 | 195 | 853 |
|  | 23 | 177 | 872 |
|  | 23 | 193 | 689 |
|  | 23 | 170 | 1235 |
|  | 22 | 207 | 689 |
|  | 21 | 186 | 946 |
|  | 22 | 157 | 1056 |
|  | 23 | 208 | 1026 |
|  | 22 | 165 | 792 |
|  | 21 | 205 | 593 |
|  | 24 | 190 | 1066 |
| **Average:** | 22.82 | 187.66 | 898.98 |

# Appendix 2

|  |  |  |  |
| --- | --- | --- | --- |
| **Randomised Prim's Solution Path Data** | | |  |
| **Size** | **Node Count** | **Average Nodes** | **Average Percentage** |
| 10 | 100 | 19.12 | 19.12 |
| 20 | 400 | 39.68 | 9.92 |
| 30 | 900 | 61.36 | 6.817777778 |
| 40 | 1600 | 81.8 | 5.1125 |
| 50 | 2500 | 101.28 | 4.0512 |

|  |  |  |
| --- | --- | --- |
|  | **Width:** | 10 |
|  | **Nodes:** | 100 |
|  |  |  |
|  | **Nodes Used** | **Percentage of Map Covered** |
|  | 19 | 19 |
|  | 19 | 19 |
|  | 19 | 19 |
|  | 19 | 19 |
|  | 19 | 19 |
|  | 19 | 19 |
|  | 19 | 19 |
|  | 19 | 19 |
|  | 19 | 19 |
|  | 19 | 19 |
|  | 19 | 19 |
|  | 19 | 19 |
|  | 19 | 19 |
|  | 19 | 19 |
|  | 19 | 19 |
|  | 19 | 19 |
|  | 19 | 19 |
|  | 19 | 19 |
|  | 19 | 19 |
|  | 19 | 19 |
|  | 19 | 19 |
|  | 19 | 19 |
|  | 19 | 19 |
|  | 19 | 19 |
|  | 19 | 19 |
|  | 19 | 19 |
|  | 19 | 19 |
|  | 19 | 19 |
|  | 19 | 19 |
|  | 19 | 19 |
|  | 19 | 19 |
|  | 19 | 19 |
|  | 19 | 19 |
|  | 19 | 19 |
|  | 19 | 19 |
|  | 23 | 23 |
|  | 19 | 19 |
|  | 19 | 19 |
|  | 19 | 19 |
|  | 19 | 19 |
|  | 19 | 19 |
|  | 21 | 21 |
|  | 19 | 19 |
|  | 19 | 19 |
|  | 19 | 19 |
|  | 19 | 19 |
|  | 19 | 19 |
|  | 19 | 19 |
|  | 19 | 19 |
|  | 19 | 19 |
| **Average:** | 19.12 | 19.12 |

|  |  |  |
| --- | --- | --- |
|  | **Width:** | 20 |
|  | **Nodes:** | 400 |
|  |  |  |
|  | **Nodes Used** | **Percentage of Map Covered** |
|  | 39 | 9.75 |
|  | 39 | 9.75 |
|  | 39 | 9.75 |
|  | 39 | 9.75 |
|  | 39 | 9.75 |
|  | 39 | 9.75 |
|  | 39 | 9.75 |
|  | 39 | 9.75 |
|  | 41 | 10.25 |
|  | 39 | 9.75 |
|  | 41 | 10.25 |
|  | 41 | 10.25 |
|  | 43 | 10.75 |
|  | 39 | 9.75 |
|  | 39 | 9.75 |
|  | 39 | 9.75 |
|  | 39 | 9.75 |
|  | 39 | 9.75 |
|  | 41 | 10.25 |
|  | 39 | 9.75 |
|  | 41 | 10.25 |
|  | 41 | 10.25 |
|  | 39 | 9.75 |
|  | 39 | 9.75 |
|  | 39 | 9.75 |
|  | 41 | 10.25 |
|  | 41 | 10.25 |
|  | 39 | 9.75 |
|  | 39 | 9.75 |
|  | 39 | 9.75 |
|  | 39 | 9.75 |
|  | 39 | 9.75 |
|  | 39 | 9.75 |
|  | 39 | 9.75 |
|  | 39 | 9.75 |
|  | 39 | 9.75 |
|  | 41 | 10.25 |
|  | 39 | 9.75 |
|  | 39 | 9.75 |
|  | 39 | 9.75 |
|  | 39 | 9.75 |
|  | 41 | 10.25 |
|  | 41 | 10.25 |
|  | 39 | 9.75 |
|  | 41 | 10.25 |
|  | 41 | 10.25 |
|  | 41 | 10.25 |
|  | 39 | 9.75 |
|  | 41 | 10.25 |
|  | 39 | 9.75 |
| **Average:** | 39.68 | 9.92 |

|  |  |  |
| --- | --- | --- |
|  | **Width:** | 30 |
|  | **Nodes:** | 900 |
|  |  |  |
|  | **Nodes Used** | **Percentage of Map Covered** |
|  | 61 | 6.777777778 |
|  | 63 | 7 |
|  | 59 | 6.555555556 |
|  | 61 | 6.777777778 |
|  | 59 | 6.555555556 |
|  | 63 | 7 |
|  | 63 | 7 |
|  | 69 | 7.666666667 |
|  | 59 | 6.555555556 |
|  | 65 | 7.222222222 |
|  | 63 | 7 |
|  | 61 | 6.777777778 |
|  | 63 | 7 |
|  | 59 | 6.555555556 |
|  | 63 | 7 |
|  | 61 | 6.777777778 |
|  | 61 | 6.777777778 |
|  | 69 | 7.666666667 |
|  | 63 | 7 |
|  | 63 | 7 |
|  | 59 | 6.555555556 |
|  | 59 | 6.555555556 |
|  | 61 | 6.777777778 |
|  | 63 | 7 |
|  | 61 | 6.777777778 |
|  | 61 | 6.777777778 |
|  | 63 | 7 |
|  | 59 | 6.555555556 |
|  | 59 | 6.555555556 |
|  | 61 | 6.777777778 |
|  | 59 | 6.555555556 |
|  | 59 | 6.555555556 |
|  | 59 | 6.555555556 |
|  | 61 | 6.777777778 |
|  | 63 | 7 |
|  | 59 | 6.555555556 |
|  | 63 | 7 |
|  | 61 | 6.777777778 |
|  | 67 | 7.444444444 |
|  | 59 | 6.555555556 |
|  | 63 | 7 |
|  | 61 | 6.777777778 |
|  | 59 | 6.555555556 |
|  | 57 | 6.333333333 |
|  | 61 | 6.777777778 |
|  | 59 | 6.555555556 |
|  | 61 | 6.777777778 |
|  | 61 | 6.777777778 |
|  | 59 | 6.555555556 |
|  | 63 | 7 |
| **Average:** | 61.36 | 6.817777778 |

|  |  |  |
| --- | --- | --- |
|  | **Width:** | 40 |
|  | **Nodes:** | 1600 |
|  |  |  |
|  | **Nodes Used** | **Percentage of Map Covered** |
|  | 81 | 5.0625 |
|  | 81 | 5.0625 |
|  | 79 | 4.9375 |
|  | 85 | 5.3125 |
|  | 79 | 4.9375 |
|  | 81 | 5.0625 |
|  | 83 | 5.1875 |
|  | 79 | 4.9375 |
|  | 83 | 5.1875 |
|  | 81 | 5.0625 |
|  | 79 | 4.9375 |
|  | 83 | 5.1875 |
|  | 87 | 5.4375 |
|  | 85 | 5.3125 |
|  | 79 | 4.9375 |
|  | 79 | 4.9375 |
|  | 81 | 5.0625 |
|  | 81 | 5.0625 |
|  | 81 | 5.0625 |
|  | 79 | 4.9375 |
|  | 81 | 5.0625 |
|  | 91 | 5.6875 |
|  | 83 | 5.1875 |
|  | 81 | 5.0625 |
|  | 79 | 4.9375 |
|  | 79 | 4.9375 |
|  | 81 | 5.0625 |
|  | 81 | 5.0625 |
|  | 79 | 4.9375 |
|  | 83 | 5.1875 |
|  | 79 | 4.9375 |
|  | 81 | 5.0625 |
|  | 81 | 5.0625 |
|  | 79 | 4.9375 |
|  | 81 | 5.0625 |
|  | 79 | 4.9375 |
|  | 81 | 5.0625 |
|  | 93 | 5.8125 |
|  | 81 | 5.0625 |
|  | 93 | 5.8125 |
|  | 85 | 5.3125 |
|  | 83 | 5.1875 |
|  | 81 | 5.0625 |
|  | 79 | 4.9375 |
|  | 83 | 5.1875 |
|  | 81 | 5.0625 |
|  | 87 | 5.4375 |
|  | 79 | 4.9375 |
|  | 79 | 4.9375 |
|  | 81 | 5.0625 |
| **Average:** | 81.8 | 5.1125 |

|  |  |  |
| --- | --- | --- |
|  | **Width:** | 50 |
|  | **Nodes:** | 2500 |
|  |  |  |
|  | **Nodes Used** | **Percentage of Map Covered** |
|  | 101 | 4.04 |
|  | 101 | 4.04 |
|  | 99 | 3.96 |
|  | 101 | 4.04 |
|  | 105 | 4.2 |
|  | 105 | 4.2 |
|  | 111 | 4.44 |
|  | 105 | 4.2 |
|  | 13 | 0.52 |
|  | 99 | 3.96 |
|  | 107 | 4.28 |
|  | 103 | 4.12 |
|  | 101 | 4.04 |
|  | 107 | 4.28 |
|  | 105 | 4.2 |
|  | 103 | 4.12 |
|  | 101 | 4.04 |
|  | 105 | 4.2 |
|  | 99 | 3.96 |
|  | 99 | 3.96 |
|  | 109 | 4.36 |
|  | 103 | 4.12 |
|  | 105 | 4.2 |
|  | 105 | 4.2 |
|  | 101 | 4.04 |
|  | 99 | 3.96 |
|  | 101 | 4.04 |
|  | 101 | 4.04 |
|  | 101 | 4.04 |
|  | 109 | 4.36 |
|  | 99 | 3.96 |
|  | 101 | 4.04 |
|  | 99 | 3.96 |
|  | 107 | 4.28 |
|  | 103 | 4.12 |
|  | 101 | 4.04 |
|  | 103 | 4.12 |
|  | 105 | 4.2 |
|  | 103 | 4.12 |
|  | 105 | 4.2 |
|  | 103 | 4.12 |
|  | 103 | 4.12 |
|  | 101 | 4.04 |
|  | 103 | 4.12 |
|  | 103 | 4.12 |
|  | 109 | 4.36 |
|  | 101 | 4.04 |
|  | 103 | 4.12 |
|  | 105 | 4.2 |
|  | 103 | 4.12 |
| **Average:** | 101.28 | 4.0512 |

# Appendix 3

|  |  |  |
| --- | --- | --- |
|  | **Width:** | 10 |
|  | **Nodes:** | 100 |
|  | **Nodes Used** | **Percentage of Map Covered** |
|  | 27 | 27 |
|  | 31 | 31 |
|  | 73 | 73 |
|  | 67 | 67 |
|  | 51 | 51 |
|  | 35 | 35 |
|  | 39 | 39 |
|  | 55 | 55 |
|  | 35 | 35 |
|  | 65 | 65 |
|  | 59 | 59 |
|  | 63 | 63 |
|  | 55 | 55 |
|  | 33 | 33 |
|  | 25 | 25 |
|  | 71 | 71 |
|  | 31 | 31 |
|  | 47 | 47 |
|  | 75 | 75 |
|  | 55 | 55 |
|  | 67 | 67 |
|  | 49 | 49 |
|  | 67 | 67 |
|  | 69 | 69 |
|  | 37 | 37 |
|  | 51 | 51 |
|  | 67 | 67 |
|  | 27 | 27 |
|  | 23 | 23 |
|  | 27 | 27 |
|  | 51 | 51 |
|  | 53 | 53 |
|  | 23 | 23 |
|  | 49 | 49 |
|  | 47 | 47 |
|  | 55 | 55 |
|  | 61 | 61 |
|  | 65 | 65 |
|  | 35 | 35 |
|  | 33 | 33 |
|  | 67 | 67 |
|  | 53 | 53 |
|  | 55 | 55 |
|  | 49 | 49 |
|  | 45 | 45 |
|  | 33 | 33 |
|  | 67 | 67 |
|  | 65 | 65 |
|  | 45 | 45 |
|  | 49 | 49 |
| **Average:** | 49.52 | 49.52 |

|  |  |  |
| --- | --- | --- |
|  | **Width:** | 20 |
|  | **Nodes:** | 400 |
|  | **Nodes Used** | **Percentage of Map Covered** |
|  | 187 | 46.75 |
|  | 211 | 52.75 |
|  | 201 | 50.25 |
|  | 131 | 32.75 |
|  | 125 | 31.25 |
|  | 189 | 47.25 |
|  | 197 | 49.25 |
|  | 263 | 65.75 |
|  | 83 | 20.75 |
|  | 113 | 28.25 |
|  | 171 | 42.75 |
|  | 215 | 53.75 |
|  | 125 | 31.25 |
|  | 223 | 55.75 |
|  | 165 | 41.25 |
|  | 165 | 41.25 |
|  | 229 | 57.25 |
|  | 171 | 42.75 |
|  | 79 | 19.75 |
|  | 205 | 51.25 |
|  | 123 | 30.75 |
|  | 93 | 23.25 |
|  | 149 | 37.25 |
|  | 127 | 31.75 |
|  | 133 | 33.25 |
|  | 67 | 16.75 |
|  | 179 | 44.75 |
|  | 155 | 38.75 |
|  | 107 | 26.75 |
|  | 135 | 33.75 |
|  | 119 | 29.75 |
|  | 149 | 37.25 |
|  | 119 | 29.75 |
|  | 107 | 26.75 |
|  | 115 | 28.75 |
|  | 127 | 31.75 |
|  | 169 | 42.25 |
|  | 273 | 68.25 |
|  | 79 | 19.75 |
|  | 177 | 44.25 |
|  | 187 | 46.75 |
|  | 93 | 23.25 |
|  | 185 | 46.25 |
|  | 181 | 45.25 |
|  | 219 | 54.75 |
|  | 95 | 23.75 |
|  | 127 | 31.75 |
|  | 237 | 59.25 |
|  | 111 | 27.75 |
|  | 119 | 29.75 |
| **Average:** | 154.08 | 38.52 |

|  |  |  |
| --- | --- | --- |
|  | **Width:** | 30 |
|  | **Nodes:** | 900 |
|  | **Nodes Used** | **Percentage of Map Covered** |
|  | 249 | 27.66666667 |
|  | 299 | 33.22222222 |
|  | 305 | 33.88888889 |
|  | 269 | 29.88888889 |
|  | 247 | 27.44444444 |
|  | 447 | 49.66666667 |
|  | 209 | 23.22222222 |
|  | 243 | 27 |
|  | 373 | 41.44444444 |
|  | 311 | 34.55555556 |
|  | 303 | 33.66666667 |
|  | 339 | 37.66666667 |
|  | 239 | 26.55555556 |
|  | 231 | 25.66666667 |
|  | 153 | 17 |
|  | 157 | 17.44444444 |
|  | 311 | 34.55555556 |
|  | 297 | 33 |
|  | 359 | 39.88888889 |
|  | 361 | 40.11111111 |
|  | 315 | 35 |
|  | 495 | 55 |
|  | 153 | 17 |
|  | 431 | 47.88888889 |
|  | 165 | 18.33333333 |
|  | 373 | 41.44444444 |
|  | 313 | 34.77777778 |
|  | 273 | 30.33333333 |
|  | 477 | 53 |
|  | 405 | 45 |
|  | 383 | 42.55555556 |
|  | 307 | 34.11111111 |
|  | 229 | 25.44444444 |
|  | 155 | 17.22222222 |
|  | 475 | 52.77777778 |
|  | 159 | 17.66666667 |
|  | 457 | 50.77777778 |
|  | 335 | 37.22222222 |
|  | 343 | 38.11111111 |
|  | 349 | 38.77777778 |
|  | 319 | 35.44444444 |
|  | 225 | 25 |
|  | 409 | 45.44444444 |
|  | 305 | 33.88888889 |
|  | 489 | 54.33333333 |
|  | 221 | 24.55555556 |
|  | 363 | 40.33333333 |
|  | 323 | 35.88888889 |
|  | 247 | 27.44444444 |
|  | 245 | 27.22222222 |
| **Average:** | 308.8 | 34.31111111 |

|  |  |  |
| --- | --- | --- |
|  | **Width:** | 40 |
|  | **Nodes:** | 1600 |
|  | **Nodes Used** | **Percentage of Map Covered** |
|  | 305 | 19.0625 |
|  | 209 | 13.0625 |
|  | 353 | 22.0625 |
|  | 491 | 30.6875 |
|  | 645 | 40.3125 |
|  | 767 | 47.9375 |
|  | 449 | 28.0625 |
|  | 429 | 26.8125 |
|  | 607 | 37.9375 |
|  | 625 | 39.0625 |
|  | 719 | 44.9375 |
|  | 707 | 44.1875 |
|  | 253 | 15.8125 |
|  | 757 | 47.3125 |
|  | 345 | 21.5625 |
|  | 273 | 17.0625 |
|  | 315 | 19.6875 |
|  | 667 | 41.6875 |
|  | 233 | 14.5625 |
|  | 365 | 22.8125 |
|  | 377 | 23.5625 |
|  | 489 | 30.5625 |
|  | 427 | 26.6875 |
|  | 501 | 31.3125 |
|  | 919 | 57.4375 |
|  | 527 | 32.9375 |
|  | 243 | 15.1875 |
|  | 427 | 26.6875 |
|  | 383 | 23.9375 |
|  | 665 | 41.5625 |
|  | 517 | 32.3125 |
|  | 413 | 25.8125 |
|  | 341 | 21.3125 |
|  | 557 | 34.8125 |
|  | 535 | 33.4375 |
|  | 233 | 14.5625 |
|  | 527 | 32.9375 |
|  | 509 | 31.8125 |
|  | 553 | 34.5625 |
|  | 337 | 21.0625 |
|  | 711 | 44.4375 |
|  | 731 | 45.6875 |
|  | 215 | 13.4375 |
|  | 373 | 23.3125 |
|  | 405 | 25.3125 |
|  | 387 | 24.1875 |
|  | 383 | 23.9375 |
|  | 451 | 28.1875 |
|  | 399 | 24.9375 |
|  | 321 | 20.0625 |
| **Average:** | 467.4 | 29.2125 |

|  |  |  |
| --- | --- | --- |
|  | **Width:** | 50 |
|  | **Nodes:** | 2500 |
|  | **Nodes Used** | **Percentage of Map Covered** |
|  | 1043 | 41.72 |
|  | 823 | 32.92 |
|  | 1067 | 42.68 |
|  | 289 | 11.56 |
|  | 1097 | 43.88 |
|  | 539 | 21.56 |
|  | 719 | 28.76 |
|  | 763 | 30.52 |
|  | 663 | 26.52 |
|  | 711 | 28.44 |
|  | 587 | 23.48 |
|  | 891 | 35.64 |
|  | 675 | 27 |
|  | 771 | 30.84 |
|  | 373 | 14.92 |
|  | 1081 | 43.24 |
|  | 765 | 30.6 |
|  | 609 | 24.36 |
|  | 527 | 21.08 |
|  | 973 | 38.92 |
|  | 751 | 30.04 |
|  | 699 | 27.96 |
|  | 643 | 25.72 |
|  | 731 | 29.24 |
|  | 841 | 33.64 |
|  | 1331 | 53.24 |
|  | 759 | 30.36 |
|  | 649 | 25.96 |
|  | 407 | 16.28 |
|  | 443 | 17.72 |
|  | 533 | 21.32 |
|  | 549 | 21.96 |
|  | 1301 | 52.04 |
|  | 723 | 28.92 |
|  | 725 | 29 |
|  | 967 | 38.68 |
|  | 867 | 34.68 |
|  | 485 | 19.4 |
|  | 999 | 39.96 |
|  | 349 | 13.96 |
|  | 407 | 16.28 |
|  | 1055 | 42.2 |
|  | 1035 | 41.4 |
|  | 881 | 35.24 |
|  | 467 | 18.68 |
|  | 1191 | 47.64 |
|  | 551 | 22.04 |
|  | 619 | 24.76 |
|  | 361 | 14.44 |
|  | 855 | 34.2 |
| **Average:** | 742.8 | 29.712 |

# Appendix 4

|  |  |
| --- | --- |
| **A** | **B** |
|  |  |
| **C** | **D** |
|  |  |
| **E** | **F** |
|  |  |

# Appendix 5

|  |  |
| --- | --- |
| **A** | **B** |
|  |  |
| **C** | **D** |
|  |  |
| **E** | **F** |
|  |  |

# Appendix 6

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Randomised Prim's Solution Path and Branch Data Average (2 dp)** | | | |  |  |  |  |
| **Size (Width X Height)** | **Node Count** | **Solution Path** | **Branch Count** | **Min Branch** | **Max Branch** | **Branch Range** | **Nodes per Branch** |
| 10 | 100 | 19.24 | 16.66 | 1 | 29.58 | 28.58 | 5.04 |
| 20 | 400 | 40.08 | 38.02 | 1 | 143.41 | 139.54 | 9.64 |
| 30 | 900 | 59.52 | 59.38 | 1 | 325.50 | 324.50 | 14.42 |
| 40 | 1600 | 82.12 | 84.08 | 1 | 551.86 | 550.86 | 9.99 |
| 50 | 2500 | 102.80 | 109.98 | 1 | 852.12 | 851.12 | 21.91 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Size:** | 10 |  |  |  |  |
|  | **Nodes:** | 100 |  |  |  |  |
|  | **Path Length** | **Branch Count** | **Minimum Branch** | **Maximum Branch** | **Branch Range** | **Average Branch Nodes** |
|  | 19 | 17 | 1 | 30 | 29 | 4.764705882 |
|  | 19 | 15 | 1 | 26 | 25 | 5.4 |
|  | 19 | 19 | 1 | 26 | 25 | 4.263157895 |
|  | 19 | 16 | 1 | 17 | 16 | 5.0625 |
|  | 19 | 16 | 1 | 23 | 22 | 5.0625 |
|  | 19 | 17 | 1 | 32 | 31 | 4.764705882 |
|  | 19 | 16 | 1 | 31 | 30 | 5.0625 |
|  | 19 | 8 | 1 | 62 | 61 | 10.125 |
|  | 19 | 19 | 1 | 15 | 14 | 4.263157895 |
|  | 19 | 20 | 1 | 31 | 30 | 4.05 |
|  | 21 | 22 | 1 | 39 | 38 | 3.590909091 |
|  | 19 | 21 | 1 | 22 | 21 | 3.857142857 |
|  | 19 | 15 | 1 | 22 | 21 | 5.4 |
|  | 19 | 17 | 1 | 24 | 23 | 4.764705882 |
|  | 19 | 18 | 1 | 21 | 20 | 4.5 |
|  | 19 | 15 | 1 | 36 | 35 | 5.4 |
|  | 19 | 21 | 1 | 27 | 26 | 3.857142857 |
|  | 19 | 15 | 1 | 31 | 30 | 5.4 |
|  | 19 | 13 | 1 | 33 | 32 | 6.230769231 |
|  | 19 | 18 | 1 | 17 | 16 | 4.5 |
|  | 19 | 19 | 1 | 24 | 23 | 4.263157895 |
|  | 19 | 15 | 1 | 46 | 45 | 5.4 |
|  | 19 | 16 | 1 | 23 | 22 | 5.0625 |
|  | 19 | 11 | 1 | 56 | 55 | 7.363636364 |
|  | 19 | 15 | 1 | 36 | 35 | 5.4 |
|  | 19 | 16 | 1 | 32 | 31 | 5.0625 |
|  | 19 | 17 | 1 | 21 | 20 | 4.764705882 |
|  | 19 | 15 | 1 | 31 | 30 | 5.4 |
|  | 19 | 16 | 1 | 26 | 25 | 5.0625 |
|  | 19 | 15 | 1 | 21 | 20 | 5.4 |
|  | 19 | 17 | 1 | 30 | 29 | 4.764705882 |
|  | 19 | 19 | 1 | 22 | 21 | 4.263157895 |
|  | 19 | 19 | 1 | 30 | 29 | 4.263157895 |
|  | 19 | 15 | 1 | 20 | 19 | 5.4 |
|  | 19 | 13 | 1 | 37 | 36 | 6.230769231 |
|  | 19 | 19 | 1 | 23 | 22 | 4.263157895 |
|  | 19 | 11 | 1 | 45 | 44 | 7.363636364 |
|  | 19 | 20 | 1 | 16 | 15 | 4.05 |
|  | 19 | 15 | 1 | 22 | 21 | 5.4 |
|  | 19 | 15 | 1 | 43 | 42 | 5.4 |
|  | 19 | 13 | 1 | 26 | 25 | 6.230769231 |
|  | 23 | 24 | 1 | 35 | 34 | 3.208333333 |
|  | 19 | 18 | 1 | 35 | 34 | 4.5 |
|  | 21 | 21 | 1 | 30 | 29 | 3.761904762 |
|  | 21 | 21 | 1 | 15 | 14 | 3.761904762 |
|  | 21 | 15 | 1 | 40 | 39 | 5.266666667 |
|  | 19 | 19 | 1 | 23 | 22 | 4.263157895 |
|  | 19 | 15 | 1 | 59 | 58 | 5.4 |
|  | 19 | 18 | 1 | 18 | 17 | 4.5 |
|  | 19 | 13 | 1 | 29 | 28 | 6.230769231 |
| **Average:** | 19.24 | 16.66 | 1 | 29.58 | 28.58 | 5.040399773 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Size:** | 20 |  |  |  |  |
|  | **Nodes:** | 400 |  |  |  |  |
|  | **Path Length** | **Branch Count** | **Minimum Branch** | **Maximum Branch** | **Branch Range** | **Average Branch Nodes** |
|  | 41 | 36 | 1 | 202 | 201 | 9.972222222 |
|  | 39 | 44 | 1 | 120 | 119 | 8.204545455 |
|  | 39 | 48 | 1 | 135 | 134 | 7.520833333 |
|  | 39 | 37 | 1 | 210 | 209 | 9.756756757 |
|  | 43 | 50 | 1 | 170 | 169 | 7.14 |
|  | 39 | 39 | 1 | 195 | 194 | 9.256410256 |
|  | 39 | 36 | 1 | 152 | 151 | 10.02777778 |
|  | 41 | 39 | 1 | 132 | 131 | 9.205128205 |
|  | 43 | 37 | 1 | 115 | 114 | 9.648648649 |
|  | 41 | 36 | 1 | 140 | 139 | 9.972222222 |
|  | 41 | 33 | 1 | 263 | 262 | 10.87878788 |
|  | 43 | 41 | 1 | 146 | 145 | 8.707317073 |
|  | 39 | 37 | 1 | 78 | 77 | 9.756756757 |
|  | 39 | 38 | 1 | 123 | 122 | 9.5 |
|  | 39 | 39 | 1 | 185 | 184 | 9.256410256 |
|  | 39 | 32 | 1 | 116 | 115 | 11.28125 |
|  | 39 | 30 | 1 | 178 | 177 | 12.03333333 |
|  | 41 | 40 | 1 | 208 | 207 | 8.975 |
|  | 41 | 39 | 1 | 121 | 120 | 9.205128205 |
|  | 39 | 40 | 1 | 139 | 138 | 9.025 |
|  | 39 | 35 | 1 | 128 | 127 | 10.31428571 |
|  | 43 | 47 | 1 | 58 | 57 | 7.595744681 |
|  | 39 | 38 | 1 | 196 | 195 | 9.5 |
|  | 39 | 34 | 1 | 137 | 136 | 10.61764706 |
|  | 39 | 37 | 1 | 96 | 95 | 9.756756757 |
|  | 39 | 44 | 1 | 199 | 198 | 8.204545455 |
|  | 43 | 36 | 1 | 164 | 163 | 9.916666667 |
|  | 39 | 39 | 1 | 91 | 90 | 9.256410256 |
|  | 39 | 34 | 1 | 138 | 137 | 10.61764706 |
|  | 39 | 40 | 1 | 99 | 98 | 9.025 |
|  | 39 | 37 | 1 | 192 | 191 | 9.756756757 |
|  | 39 | 43 | 1 | 125 | 124 | 8.395348837 |
|  | 39 | 37 | 1 | 88 | 87 | 9.756756757 |
|  | 39 | 39 | 1 | 156 | 155 | 9.256410256 |
|  | 39 | 41 | 1 | 71 | 70 | 8.804878049 |
|  | 43 | 42 | 1 | 90 | 89 | 8.5 |
|  | 39 | 38 | 1 | 166 | 165 | 9.5 |
|  | 41 | 27 | 1 | 268 | 267 | 13.2962963 |
|  | 39 | 29 | 1 | 185 | 184 | 12.44827586 |
|  | 43 | 44 | 1 | 102 | 101 | 8.113636364 |
|  | 39 | 32 | 1 | 152 | 151 | 11.28125 |
|  | 39 | 43 | 1 | 109 | 108 | 8.395348837 |
|  | 39 | 39 | 1 | 90 | 89 | 9.256410256 |
|  | 39 | 33 | 1 | 178 | 177 | 10.93939394 |
|  | 39 | 30 | 1 | 169 | 168 | 12.03333333 |
|  | 43 | 43 | 1 | 108 | 107 | 8.302325581 |
|  | 47 | 45 | 1 | 87 | 86 | 7.844444444 |
|  | 39 | 40 | 1 | 75 | 74 | 9.025 |
|  | 39 | 28 | 1 | 182 | 181 | 12.89285714 |
|  | 39 | 36 | 1 |  | -1 | 10.02777778 |
| **Average:** | 40.08 | 38.02 | 1 | 143.4081633 | 139.54 | 9.63909465 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Size:** | 30 |  |  |  |  |
|  | **Nodes:** | 900 |  |  |  |  |
|  | **Path Length** | **Branch Count** | **Minimum Branch** | **Maximum Branch** | **Branch Range** | **Average Branch Nodes** |
|  | 61 | 68 | 1 | 321 | 320 | 12.33823529 |
|  | 59 | 66 | 1 | 308 | 307 | 12.74242424 |
|  | 59 | 50 | 1 | 185 | 184 | 16.82 |
|  | 61 | 74 | 1 | 300 | 299 | 11.33783784 |
|  | 61 | 46 | 1 | 376 | 375 | 18.23913043 |
|  | 59 | 57 | 1 | 369 | 368 | 14.75438596 |
|  | 59 | 51 | 1 | 329 | 328 | 16.49019608 |
|  | 61 | 64 | 1 | 253 | 252 | 13.109375 |
|  | 67 | 65 | 1 | 357 | 356 | 12.81538462 |
|  | 37 | 57 | 1 | 605 | 604 | 15.14035088 |
|  | 59 | 62 | 1 | 204 | 203 | 13.56451613 |
|  | 63 | 55 | 1 | 255 | 254 | 15.21818182 |
|  | 35 | 37 | 1 | 282 | 281 | 23.37837838 |
|  | 59 | 53 | 1 | 397 | 396 | 15.86792453 |
|  | 59 | 60 | 1 | 284 | 283 | 14.01666667 |
|  | 59 | 66 | 1 | 270 | 269 | 12.74242424 |
|  | 63 | 73 | 1 | 383 | 382 | 11.46575342 |
|  | 59 | 60 | 1 | 327 | 326 | 14.01666667 |
|  | 59 | 55 | 1 | 257 | 256 | 15.29090909 |
|  | 59 | 71 | 1 | 258 | 257 | 11.84507042 |
|  | 59 | 66 | 1 | 318 | 317 | 12.74242424 |
|  | 61 | 56 | 1 | 325 | 324 | 14.98214286 |
|  | 63 | 61 | 1 | 250 | 249 | 13.72131148 |
|  | 61 | 67 | 1 | 205 | 204 | 12.52238806 |
|  | 61 | 51 | 1 | 312 | 311 | 16.45098039 |
|  | 59 | 60 | 1 | 272 | 271 | 14.01666667 |
|  | 63 | 57 | 1 | 170 | 169 | 14.68421053 |
|  | 61 | 50 | 1 | 196 | 195 | 16.78 |
|  | 61 | 55 | 1 | 431 | 430 | 15.25454545 |
|  | 59 | 58 | 1 | 1559 | 1558 | 14.5 |
|  | 59 | 57 | 1 | 186 | 185 | 14.75438596 |
|  | 61 | 61 | 1 | 219 | 218 | 13.75409836 |
|  | 63 | 47 | 1 | 357 | 356 | 17.80851064 |
|  | 59 | 51 | 1 | 226 | 225 | 16.49019608 |
|  | 61 | 63 | 1 | 229 | 228 | 13.31746032 |
|  | 61 | 66 | 1 | 353 | 352 | 12.71212121 |
|  | 61 | 62 | 1 | 252 | 251 | 13.53225806 |
|  | 61 | 70 | 1 | 199 | 198 | 11.98571429 |
|  | 59 | 59 | 1 | 362 | 361 | 14.25423729 |
|  | 59 | 49 | 1 | 327 | 326 | 17.16326531 |
|  | 63 | 70 | 1 | 239 | 238 | 11.95714286 |
|  | 59 | 59 | 1 | 265 | 264 | 14.25423729 |
|  | 61 | 49 | 1 | 191 | 190 | 17.12244898 |
|  | 63 | 65 | 1 | 236 | 235 | 12.87692308 |
|  | 59 | 63 | 1 | 492 | 491 | 13.34920635 |
|  | 59 | 59 | 1 | 306 | 305 | 14.25423729 |
|  | 61 | 64 | 1 | 458 | 457 | 13.109375 |
|  | 59 | 67 | 1 | 433 | 432 | 12.55223881 |
|  | 59 | 61 | 1 | 288 | 287 | 13.78688525 |
|  | 63 | 56 | 1 | 299 | 298 | 14.94642857 |
| **Average:** | 59.52 | 59.38 | 1 | 325.5 | 324.5 | 14.41659705 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Size:** | 40 |  |  |  |  |
|  | **Nodes:** | 1600 |  |  |  |  |
|  | **Path Length** | **Branch Count** | **Minimum Branch** | **Maximum Branch** | **Branch Range** | **Average Branch Nodes** |
|  | 81 | 83 | 1 | 734 | 733 | 18.30120482 |
|  | 85 | 75 | 1 | 452 | 451 | 10.86666667 |
|  | 79 | 84 | 1 | 407 | 406 | 9.773809524 |
|  | 81 | 91 | 1 | 599 | 598 | 9 |
|  | 79 | 84 | 1 | 320 | 319 | 9.773809524 |
|  | 87 | 97 | 1 | 595 | 594 | 8.381443299 |
|  | 83 | 78 | 1 | 319 | 318 | 10.47435897 |
|  | 81 | 82 | 1 | 817 | 816 | 9.987804878 |
|  | 91 | 96 | 1 | 323 | 322 | 8.427083333 |
|  | 83 | 84 | 1 | 587 | 586 | 9.726190476 |
|  | 83 | 93 | 1 | 737 | 736 | 8.784946237 |
|  | 85 | 57 | 1 | 802 | 801 | 14.29824561 |
|  | 83 | 81 | 1 | 500 | 499 | 10.08641975 |
|  | 85 | 94 | 1 | 645 | 644 | 8.670212766 |
|  | 83 | 78 | 1 | 492 | 491 | 10.47435897 |
|  | 79 | 83 | 1 | 577 | 576 | 9.891566265 |
|  | 81 | 104 | 1 | 627 | 626 | 7.875 |
|  | 79 | 82 | 1 | 668 | 667 | 10.01219512 |
|  | 79 | 84 | 1 | 388 | 387 | 9.773809524 |
|  | 83 | 80 | 1 | 534 | 533 | 10.2125 |
|  | 81 | 80 | 1 | 445 | 444 | 10.2375 |
|  | 85 | 77 | 1 | 1019 | 1018 | 10.58441558 |
|  | 85 | 92 | 1 | 934 | 933 | 8.858695652 |
|  | 79 | 77 | 1 | 378 | 377 | 10.66233766 |
|  | 81 | 82 | 1 | 568 | 567 | 9.987804878 |
|  | 79 | 76 | 1 | 686 | 685 | 10.80263158 |
|  | 81 | 81 | 1 | 689 | 688 | 10.11111111 |
|  | 81 | 88 | 1 | 418 | 417 | 9.306818182 |
|  | 85 | 88 | 1 | 443 | 442 | 9.261363636 |
|  | 79 | 77 | 1 | 340 | 339 | 10.66233766 |
|  | 81 | 91 | 1 | 389 | 388 | 9 |
|  | 83 | 89 | 1 | 447 | 446 | 9.179775281 |
|  | 89 | 93 | 1 | 558 | 557 | 8.720430108 |
|  | 79 | 91 | 1 | 302 | 301 | 9.021978022 |
|  | 81 | 75 | 1 | 950 | 949 | 10.92 |
|  | 85 | 79 | 1 | 605 | 604 | 10.3164557 |
|  | 79 | 79 | 1 | 504 | 503 | 10.39240506 |
|  | 81 | 75 | 1 | 378 | 377 | 10.92 |
|  | 87 | 101 | 1 | 397 | 396 | 8.04950495 |
|  | 83 | 84 | 1 | 399 | 398 | 9.726190476 |
|  | 85 | 93 | 1 | 641 | 640 | 8.76344086 |
|  | 79 | 86 | 1 | 621 | 620 | 9.546511628 |
|  | 81 | 82 | 1 | 365 | 364 | 9.987804878 |
|  | 81 | 88 | 1 | 697 | 696 | 9.306818182 |
|  | 81 | 83 | 1 | 342 | 341 | 9.86746988 |
|  | 83 | 81 | 1 | 797 | 796 | 10.08641975 |
|  | 81 | 80 | 1 | 587 | 586 | 10.2375 |
|  | 79 | 78 | 1 | 448 | 447 | 10.52564103 |
|  | 83 | 89 | 1 | 717 | 716 | 9.179775281 |
|  | 79 | 79 | 1 | 406 | 405 | 10.39240506 |
| **Average:** | 82.12 | 84.08 | 1 | 551.86 | 550.86 | 9.988143357 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Size:** | 50 |  |  |  |  |
|  | **Nodes:** | 2500 |  |  |  |  |
|  | **Path Length** | **Branch Count** | **Minimum Branch** | **Maximum Branch** | **Branch Range** | **Average Branch Nodes** |
|  | 103 | 114 | 1 | 917 | 916 | 21.02631579 |
|  | 99 | 103 | 1 | 1099 | 1098 | 23.31067961 |
|  | 107 | 108 | 1 | 953 | 952 | 22.15740741 |
|  | 103 | 113 | 1 | 759 | 758 | 21.21238938 |
|  | 109 | 129 | 1 | 1164 | 1163 | 18.53488372 |
|  | 101 | 99 | 1 | 758 | 757 | 24.23232323 |
|  | 101 | 106 | 1 | 858 | 857 | 22.63207547 |
|  | 103 | 103 | 1 | 649 | 648 | 23.27184466 |
|  | 109 | 112 | 1 | 835 | 834 | 21.34821429 |
|  | 105 | 116 | 1 | 510 | 509 | 20.64655172 |
|  | 105 | 109 | 1 | 842 | 841 | 21.97247706 |
|  | 105 | 119 | 1 | 463 | 462 | 20.12605042 |
|  | 99 | 113 | 1 | 657 | 656 | 21.24778761 |
|  | 101 | 119 | 1 | 848 | 847 | 20.15966387 |
|  | 99 | 104 | 1 | 1018 | 1017 | 23.08653846 |
|  | 101 | 113 | 1 | 663 | 662 | 21.2300885 |
|  | 99 | 103 | 1 | 806 | 805 | 23.31067961 |
|  | 103 | 106 | 1 | 482 | 481 | 22.61320755 |
|  | 99 | 113 | 1 | 843 | 842 | 21.24778761 |
|  | 101 | 114 | 1 | 1016 | 1015 | 21.04385965 |
|  | 103 | 107 | 1 | 612 | 611 | 22.40186916 |
|  | 101 | 117 | 1 | 1064 | 1063 | 20.5042735 |
|  | 99 | 107 | 1 | 1071 | 1070 | 22.43925234 |
|  | 101 | 109 | 1 | 754 | 753 | 22.00917431 |
|  | 107 | 114 | 1 | 1397 | 1396 | 20.99122807 |
|  | 101 | 95 | 1 | 1039 | 1038 | 25.25263158 |
|  | 103 | 109 | 1 | 825 | 824 | 21.99082569 |
|  | 101 | 110 | 1 | 718 | 717 | 21.80909091 |
|  | 103 | 119 | 1 | 735 | 734 | 20.14285714 |
|  | 99 | 100 | 1 | 1093 | 1092 | 24.01 |
|  | 107 | 108 | 1 | 481 | 480 | 22.15740741 |
|  | 107 | 116 | 1 | 1337 | 1336 | 20.62931034 |
|  | 105 | 102 | 1 | 905 | 904 | 23.48039216 |
|  | 101 | 106 | 1 | 729 | 728 | 22.63207547 |
|  | 105 | 103 | 1 | 983 | 982 | 23.25242718 |
|  | 101 | 120 | 1 | 592 | 591 | 19.99166667 |
|  | 105 | 98 | 1 | 1069 | 1068 | 24.43877551 |
|  | 101 | 112 | 1 | 925 | 924 | 21.41964286 |
|  | 99 | 107 | 1 | 843 | 842 | 22.43925234 |
|  | 101 | 114 | 1 | 1004 | 1003 | 21.04385965 |
|  | 107 | 119 | 1 | 786 | 785 | 20.1092437 |
|  | 103 | 104 | 1 | 967 | 966 | 23.04807692 |
|  | 99 | 88 | 1 | 962 | 961 | 27.28409091 |
|  | 103 | 115 | 1 | 511 | 510 | 20.84347826 |
|  | 101 | 116 | 1 | 819 | 818 | 20.68103448 |
|  | 103 | 122 | 1 | 997 | 996 | 19.64754098 |
|  | 103 | 111 | 1 | 1291 | 1290 | 21.59459459 |
|  | 101 | 105 | 1 | 772 | 771 | 22.84761905 |
|  | 109 | 126 | 1 | 533 | 532 | 18.97619048 |
|  | 109 | 104 | 1 | 652 | 651 | 22.99038462 |
| **Average:** | 102.8 | 109.98 | 1 | 852.12 | 851.12 | 21.90938184 |

# Appendix 7

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Recursive Backtracking Solution Path and Branch Data Average (2dp)** | |  |  |  |  |  |  |
| **Size (Width X Height)** | **Node Count** | **Solution Path** | **Branch Count** | **Min Branch** | **Max Branch** | **Branch Range** | **Nodes per Branch** |
| 10 | 100 | 41.24 | 12.5 | 1.02 | 35.86 | 34.84 | 6.83 |
| 20 | 400 | 140.28 | 39.36 | 1 | 154.62 | 153.62 | 8.05 |
| 30 | 900 | 285.64 | 74.12 | 1 | 340.16 | 339.16 | 9.96 |
| 40 | 1600 | 442.2 | 110.54 | 1 | 636.68 | 635.68 | 13.26 |
| 50 | 2500 | 689.16 | 168.76 | 1 | 865.06 | 864.06 | 13.49 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Size:** | 10 |  |  |  |  |
|  | **Nodes:** | 100 |  |  |  |  |
|  | **Path Length** | **Branch Count** | **Minimum Branch** | **Maximum Branch** | **Branch Range** | **Average Branch Nodes** |
|  | 33 | 14 | 1 | 39 | 38 | 4.785714286 |
|  | 37 | 12 | 1 | 39 | 38 | 5.25 |
|  | 47 | 12 | 1 | 28 | 27 | 4.416666667 |
|  | 33 | 11 | 1 | 53 | 52 | 6.090909091 |
|  | 43 | 11 | 1 | 44 | 43 | 5.181818182 |
|  | 39 | 14 | 1 | 46 | 45 | 4.357142857 |
|  | 59 | 17 | 1 | 9 | 8 | 2.411764706 |
|  | 23 | 5 | 1 | 67 | 66 | 15.4 |
|  | 33 | 12 | 1 | 48 | 47 | 5.583333333 |
|  | 29 | 7 | 1 | 60 | 59 | 10.14285714 |
|  | 39 | 4 | 1 | 34 | 33 | 15.25 |
|  | 37 | 11 | 1 | 49 | 48 | 5.727272727 |
|  | 63 | 15 | 1 | 9 | 8 | 2.466666667 |
|  | 57 | 12 | 1 | 25 | 24 | 3.583333333 |
|  | 25 | 8 | 1 | 49 | 48 | 9.375 |
|  | 53 | 15 | 1 | 21 | 20 | 3.133333333 |
|  | 21 | 2 | 1 | 49 | 48 | 39.5 |
|  | 57 | 20 | 1 | 14 | 13 | 2.15 |
|  | 37 | 13 | 1 | 40 | 39 | 4.846153846 |
|  | 25 | 22 | 1 | 53 | 52 | 3.409090909 |
|  | 49 | 10 | 1 | 34 | 33 | 5.1 |
|  | 61 | 22 | 1 | 7 | 6 | 1.772727273 |
|  | 67 | 26 | 1 | 4 | 3 | 1.269230769 |
|  | 59 | 26 | 1 | 7 | 6 | 1.576923077 |
|  | 51 | 19 | 1 | 21 | 20 | 2.578947368 |
|  | 45 | 14 | 1 | 39 | 38 | 3.928571429 |
|  | 25 | 12 | 1 | 61 | 60 | 6.25 |
|  | 49 | 15 | 1 | 26 | 25 | 3.4 |
|  | 57 | 17 | 1 | 15 | 14 | 2.529411765 |
|  | 29 | 8 | 1 | 48 | 47 | 8.875 |
|  | 33 | 8 | 1 | 54 | 53 | 8.375 |
|  | 27 | 7 | 1 | 49 | 48 | 10.42857143 |
|  | 21 | 3 | 2 | 62 | 60 | 26.33333333 |
|  | 29 | 11 | 1 | 56 | 55 | 6.454545455 |
|  | 39 | 10 | 1 | 28 | 27 | 6.1 |
|  | 29 | 9 | 1 | 47 | 46 | 7.888888889 |
|  | 51 | 15 | 1 | 19 | 18 | 3.266666667 |
|  | 35 | 5 | 1 | 41 | 40 | 13 |
|  | 27 | 5 | 1 | 58 | 57 | 14.6 |
|  | 45 | 16 | 1 | 31 | 30 | 3.4375 |
|  | 31 | 9 | 1 | 47 | 46 | 7.666666667 |
|  | 31 | 8 | 1 | 40 | 39 | 8.625 |
|  | 65 | 10 | 1 | 10 | 9 | 3.5 |
|  | 51 | 17 | 1 | 16 | 15 | 2.882352941 |
|  | 47 | 13 | 1 | 28 | 27 | 4.076923077 |
|  | 31 | 8 | 1 | 53 | 52 | 8.625 |
|  | 35 | 17 | 1 | 46 | 45 | 3.823529412 |
|  | 55 | 25 | 1 | 8 | 7 | 1.8 |
|  | 41 | 8 | 1 | 42 | 41 | 7.375 |
|  | 57 | 15 | 1 | 20 | 19 | 2.866666667 |
| AVERAGE: | 41.24 | 12.5 | 1.02 | 35.86 | 34.84 | 6.829350266 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Size:** | 20 |  |  |  |  |
|  | **Nodes:** | 400 |  |  |  |  |
|  | **Path Length** | **Branch Count** | **Minimum Branch** | **Maximum Branch** | **Branch Range** | **Average Branch Nodes** |
|  | 145 | 58 | 1 | 171 | 170 | 4.396551724 |
|  | 145 | 31 | 1 | 163 | 162 | 8.225806452 |
|  | 85 | 19 | 1 | 273 | 272 | 16.57894737 |
|  | 133 | 38 | 1 | 126 | 125 | 7.026315789 |
|  | 67 | 23 | 1 | 308 | 307 | 14.47826087 |
|  | 189 | 56 | 1 | 72 | 71 | 3.767857143 |
|  | 119 | 39 | 1 | 173 | 172 | 7.205128205 |
|  | 145 | 35 | 1 | 93 | 92 | 7.285714286 |
|  | 219 | 59 | 1 | 43 | 42 | 3.06779661 |
|  | 151 | 34 | 1 | 160 | 159 | 7.323529412 |
|  | 115 | 40 | 1 | 203 | 202 | 7.125 |
|  | 137 | 30 | 1 | 208 | 207 | 8.766666667 |
|  | 199 | 55 | 1 | 65 | 64 | 3.654545455 |
|  | 149 | 45 | 1 | 131 | 130 | 5.577777778 |
|  | 179 | 62 | 1 | 90 | 89 | 3.564516129 |
|  | 175 | 55 | 1 | 68 | 67 | 4.090909091 |
|  | 141 | 43 | 1 | 95 | 94 | 6.023255814 |
|  | 233 | 72 | 1 | 24 | 23 | 2.319444444 |
|  | 89 | 35 | 1 | 228 | 227 | 8.885714286 |
|  | 169 | 50 | 1 | 90 | 89 | 4.62 |
|  | 91 | 16 | 1 | 213 | 212 | 19.3125 |
|  | 135 | 41 | 1 | 183 | 182 | 6.463414634 |
|  | 149 | 29 | 1 | 135 | 134 | 8.655172414 |
|  | 101 | 26 | 1 | 244 | 243 | 11.5 |
|  | 151 | 43 | 1 | 154 | 153 | 5.790697674 |
|  | 149 | 35 | 1 | 146 | 145 | 7.171428571 |
|  | 125 | 45 | 1 | 201 | 200 | 6.111111111 |
|  | 87 | 25 | 1 | 253 | 252 | 12.52 |
|  | 205 | 44 | 1 | 69 | 68 | 4.431818182 |
|  | 165 | 53 | 1 | 90 | 89 | 4.433962264 |
|  | 133 | 47 | 1 | 119 | 118 | 5.680851064 |
|  | 223 | 81 | 1 | 34 | 33 | 2.185185185 |
|  | 155 | 34 | 1 | 179 | 178 | 7.205882353 |
|  | 69 | 26 | 1 | 292 | 291 | 12.73076923 |
|  | 189 | 58 | 1 | 64 | 63 | 3.637931034 |
|  | 121 | 27 | 1 | 226 | 225 | 10.33333333 |
|  | 159 | 39 | 1 | 111 | 110 | 6.179487179 |
|  | 165 | 45 | 1 | 147 | 146 | 5.222222222 |
|  | 95 | 17 | 1 | 238 | 237 | 17.94117647 |
|  | 141 | 22 | 1 | 117 | 116 | 11.77272727 |
|  | 87 | 16 | 1 | 233 | 232 | 19.5625 |
|  | 165 | 33 | 1 | 107 | 106 | 7.121212121 |
|  | 149 | 57 | 1 | 112 | 111 | 4.403508772 |
|  | 125 | 42 | 1 | 193 | 192 | 6.547619048 |
|  | 141 | 36 | 1 | 221 | 220 | 7.194444444 |
|  | 95 | 16 | 1 | 240 | 239 | 19.0625 |
|  | 115 | 30 | 1 | 147 | 146 | 9.5 |
|  | 125 | 41 | 1 | 141 | 140 | 6.707317073 |
|  | 91 | 25 | 1 | 219 | 218 | 12.36 |
|  | 129 | 40 | 1 | 119 | 118 | 6.775 |
| AVERAGE: | 140.28 | 39.36 | 1 | 154.62 | 153.62 | 8.049950184 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Size:** | 30 |  |  |  |  |
|  | **Nodes:** | 900 |  |  |  |  |
|  | **Path Length** | **Branch Count** | **Minimum Branch** | **Maximum Branch** | **Branch Range** | **Average Branch Nodes** |
|  | 385 | 104 | 1 | 202 | 201 | 4.951923077 |
|  | 157 | 48 | 1 | 450 | 449 | 15.47916667 |
|  | 257 | 67 | 1 | 495 | 494 | 9.597014925 |
|  | 367 | 84 | 1 | 240 | 239 | 6.345238095 |
|  | 541 | 88 | 1 | 313 | 312 | 4.079545455 |
|  | 263 | 79 | 1 | 432 | 431 | 8.063291139 |
|  | 441 | 152 | 1 | 94 | 93 | 3.019736842 |
|  | 369 | 109 | 1 | 184 | 183 | 4.871559633 |
|  | 249 | 77 | 1 | 438 | 437 | 8.454545455 |
|  | 295 | 92 | 1 | 397 | 396 | 6.576086957 |
|  | 251 | 54 | 1 | 437 | 436 | 12.01851852 |
|  | 285 | 63 | 1 | 387 | 386 | 9.761904762 |
|  | 261 | 54 | 1 | 355 | 354 | 11.83333333 |
|  | 213 | 84 | 1 | 516 | 515 | 8.178571429 |
|  | 311 | 94 | 1 | 309 | 308 | 6.265957447 |
|  | 147 | 26 | 1 | 664 | 663 | 28.96153846 |
|  | 253 | 63 | 1 | 422 | 421 | 10.26984127 |
|  | 309 | 77 | 1 | 347 | 346 | 7.675324675 |
|  | 415 | 111 | 1 | 110 | 109 | 4.369369369 |
|  | 311 | 83 | 1 | 253 | 252 | 7.096385542 |
|  | 149 | 29 | 1 | 442 | 441 | 25.89655172 |
|  | 239 | 60 | 1 | 283 | 282 | 11.01666667 |
|  | 155 | 38 | 1 | 625 | 624 | 19.60526316 |
|  | 337 | 96 | 1 | 171 | 170 | 5.864583333 |
|  | 409 | 113 | 1 | 112 | 111 | 4.345132743 |
|  | 373 | 65 | 1 | 240 | 239 | 8.107692308 |
|  | 295 | 62 | 1 | 293 | 292 | 9.758064516 |
|  | 339 | 58 | 1 | 216 | 215 | 9.672413793 |
|  | 187 | 44 | 1 | 565 | 564 | 16.20454545 |
|  | 239 | 75 | 1 | 449 | 448 | 8.813333333 |
|  | 349 | 95 | 1 | 226 | 225 | 5.8 |
|  | 155 | 57 | 1 | 499 | 498 | 13.07017544 |
|  | 219 | 49 | 1 | 513 | 512 | 13.89795918 |
|  | 289 | 100 | 1 | 401 | 400 | 6.11 |
|  | 301 | 71 | 1 | 285 | 284 | 8.436619718 |
|  | 265 | 62 | 1 | 355 | 354 | 10.24193548 |
|  | 359 | 94 | 1 | 230 | 229 | 5.755319149 |
|  | 393 | 102 | 1 | 146 | 145 | 4.970588235 |
|  | 267 | 57 | 1 | 336 | 335 | 11.10526316 |
|  | 129 | 31 | 1 | 419 | 418 | 24.87096774 |
|  | 199 | 54 | 1 | 522 | 521 | 12.98148148 |
|  | 301 | 72 | 1 | 212 | 211 | 8.319444444 |
|  | 327 | 82 | 1 | 185 | 184 | 6.987804878 |
|  | 153 | 36 | 1 | 357 | 356 | 20.75 |
|  | 373 | 101 | 1 | 166 | 165 | 5.217821782 |
|  | 339 | 80 | 1 | 256 | 255 | 7.0125 |
|  | 187 | 51 | 1 | 547 | 546 | 13.98039216 |
|  | 253 | 73 | 1 | 325 | 324 | 8.863013699 |
|  | 369 | 104 | 1 | 99 | 98 | 5.105769231 |
|  | 253 | 86 | 1 | 488 | 487 | 7.523255814 |
| AVERAGE: | 285.64 | 74.12 | 1 | 340.16 | 339.16 | 9.963068234 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Size:** | 40 |  |  |  |  |
|  | **Nodes:** | 1600 |  |  |  |  |
|  | **Path Length** | **Branch Count** | **Minimum Branch** | **Maximum Branch** | **Branch Range** | **Average Branch Nodes** |
|  | 441 | 79 | 1 | 805 | 804 | 14.67088608 |
|  | 595 | 183 | 1 | 195 | 194 | 5.491803279 |
|  | 547 | 134 | 1 | 390 | 389 | 7.858208955 |
|  | 511 | 114 | 1 | 474 | 473 | 9.552631579 |
|  | 251 | 77 | 1 | 1162 | 1161 | 17.51948052 |
|  | 637 | 163 | 1 | 297 | 296 | 5.90797546 |
|  | 489 | 117 | 1 | 504 | 503 | 9.495726496 |
|  | 561 | 136 | 1 | 496 | 495 | 7.639705882 |
|  | 441 | 142 | 1 | 832 | 831 | 8.161971831 |
|  | 317 | 66 | 1 | 736 | 735 | 19.43939394 |
|  | 559 | 124 | 1 | 401 | 400 | 8.39516129 |
|  | 773 | 189 | 1 | 110 | 109 | 4.375661376 |
|  | 565 | 146 | 1 | 429 | 428 | 7.089041096 |
|  | 319 | 92 | 1 | 1099 | 1098 | 13.92391304 |
|  | 347 | 89 | 1 | 791 | 790 | 14.07865169 |
|  | 301 | 63 | 1 | 844 | 843 | 20.61904762 |
|  | 407 | 121 | 1 | 807 | 806 | 9.859504132 |
|  | 551 | 138 | 1 | 434 | 433 | 7.601449275 |
|  | 575 | 149 | 1 | 252 | 251 | 6.879194631 |
|  | 575 | 132 | 1 | 297 | 296 | 7.765151515 |
|  | 407 | 98 | 1 | 505 | 504 | 12.17346939 |
|  | 631 | 165 | 1 | 399 | 398 | 5.872727273 |
|  | 473 | 130 | 1 | 518 | 517 | 8.669230769 |
|  | 271 | 55 | 1 | 620 | 619 | 24.16363636 |
|  | 359 | 91 | 1 | 839 | 838 | 13.63736264 |
|  | 395 | 98 | 1 | 840 | 839 | 12.29591837 |
|  | 611 | 180 | 1 | 201 | 200 | 5.494444444 |
|  | 387 | 77 | 1 | 837 | 836 | 15.75324675 |
|  | 513 | 106 | 1 | 480 | 479 | 10.25471698 |
|  | 247 | 73 | 1 | 1166 | 1165 | 18.53424658 |
|  | 663 | 183 | 1 | 303 | 302 | 5.120218579 |
|  | 309 | 69 | 1 | 850 | 849 | 18.71014493 |
|  | 611 | 152 | 1 | 362 | 361 | 6.506578947 |
|  | 351 | 98 | 1 | 743 | 742 | 12.74489796 |
|  | 409 | 100 | 1 | 686 | 685 | 11.91 |
|  | 239 | 53 | 1 | 1037 | 1036 | 25.67924528 |
|  | 381 | 95 | 1 | 611 | 610 | 12.83157895 |
|  | 299 | 67 | 1 | 999 | 998 | 19.41791045 |
|  | 617 | 153 | 1 | 307 | 306 | 6.424836601 |
|  | 499 | 148 | 1 | 622 | 621 | 7.439189189 |
|  | 255 | 64 | 1 | 849 | 848 | 21.015625 |
|  | 209 | 28 | 1 | 756 | 755 | 49.67857143 |
|  | 641 | 164 | 1 | 371 | 370 | 5.847560976 |
|  | 391 | 90 | 1 | 611 | 610 | 13.43333333 |
|  | 163 | 44 | 1 | 1317 | 1316 | 32.65909091 |
|  | 419 | 111 | 1 | 498 | 497 | 10.63963964 |
|  | 387 | 86 | 1 | 607 | 606 | 14.10465116 |
|  | 405 | 99 | 1 | 936 | 935 | 12.07070707 |
|  | 665 | 157 | 1 | 401 | 400 | 5.955414013 |
|  | 141 | 39 | 1 | 1208 | 1207 | 37.41025641 |
| AVERAGE: | 442.2 | 110.54 | 1 | 636.68 | 635.68 | 13.2554602 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Size:** | 50 |  |  |  |  |
|  | **Nodes:** | 2500 |  |  |  |  |
|  | **Path Length** | **Branch Count** | **Minimum Branch** | **Maximum Branch** | **Branch Range** | **Average Branch Nodes** |
|  | 589 | 138 | 1 | 683 | 682 | 13.84782609 |
|  | 759 | 179 | 1 | 850 | 849 | 9.726256983 |
|  | 743 | 188 | 1 | 695 | 694 | 9.345744681 |
|  | 691 | 189 | 1 | 714 | 713 | 9.571428571 |
|  | 725 | 149 | 1 | 1043 | 1042 | 11.91275168 |
|  | 397 | 118 | 1 | 1581 | 1580 | 17.8220339 |
|  | 879 | 252 | 1 | 703 | 702 | 6.432539683 |
|  | 823 | 178 | 1 | 433 | 432 | 9.421348315 |
|  | 741 | 179 | 1 | 738 | 737 | 9.826815642 |
|  | 433 | 117 | 1 | 988 | 987 | 17.66666667 |
|  | 799 | 218 | 1 | 879 | 878 | 7.802752294 |
|  | 1049 | 250 | 1 | 186 | 185 | 5.804 |
|  | 383 | 72 | 1 | 1041 | 1040 | 29.40277778 |
|  | 695 | 166 | 1 | 448 | 447 | 10.87349398 |
|  | 391 | 80 | 1 | 1218 | 1217 | 26.3625 |
|  | 259 | 42 | 1 | 1170 | 1169 | 53.35714286 |
|  | 533 | 188 | 1 | 1371 | 1370 | 10.46276596 |
|  | 849 | 199 | 1 | 691 | 690 | 8.296482412 |
|  | 657 | 153 | 1 | 708 | 707 | 12.04575163 |
|  | 473 | 97 | 1 | 942 | 941 | 20.89690722 |
|  | 835 | 242 | 1 | 617 | 616 | 6.880165289 |
|  | 433 | 110 | 1 | 1646 | 1645 | 18.79090909 |
|  | 671 | 163 | 1 | 1104 | 1103 | 11.2208589 |
|  | 929 | 224 | 1 | 317 | 316 | 7.013392857 |
|  | 721 | 144 | 1 | 929 | 928 | 12.35416667 |
|  | 641 | 156 | 1 | 1120 | 1119 | 11.91666667 |
|  | 983 | 271 | 1 | 336 | 335 | 5.597785978 |
|  | 849 | 200 | 1 | 777 | 776 | 8.255 |
|  | 357 | 77 | 1 | 1031 | 1030 | 27.83116883 |
|  | 1039 | 273 | 1 | 195 | 194 | 5.351648352 |
|  | 995 | 247 | 1 | 425 | 424 | 6.093117409 |
|  | 921 | 216 | 1 | 196 | 195 | 7.310185185 |
|  | 739 | 187 | 1 | 1000 | 999 | 9.417112299 |
|  | 313 | 63 | 1 | 1511 | 1510 | 34.71428571 |
|  | 533 | 106 | 1 | 1473 | 1472 | 18.55660377 |
|  | 737 | 172 | 1 | 686 | 685 | 10.25 |
|  | 417 | 87 | 1 | 1505 | 1504 | 23.94252874 |
|  | 593 | 133 | 1 | 1127 | 1126 | 14.33834586 |
|  | 803 | 182 | 1 | 867 | 866 | 9.324175824 |
|  | 539 | 145 | 1 | 1035 | 1034 | 13.52413793 |
|  | 905 | 250 | 1 | 630 | 629 | 6.38 |
|  | 633 | 144 | 1 | 1161 | 1160 | 12.96527778 |
|  | 881 | 231 | 1 | 369 | 368 | 7.008658009 |
|  | 515 | 116 | 1 | 1356 | 1355 | 17.11206897 |
|  | 1001 | 297 | 1 | 324 | 323 | 5.047138047 |
|  | 403 | 96 | 1 | 1372 | 1371 | 21.84375 |
|  | 863 | 210 | 1 | 537 | 536 | 7.795238095 |
|  | 901 | 224 | 1 | 529 | 528 | 7.138392857 |
|  | 879 | 212 | 1 | 851 | 850 | 7.646226415 |
|  | 561 | 108 | 1 | 1145 | 1144 | 17.9537037 |
| AVERAGE: | 689.16 | 168.76 | 1 | 865.06 | 864.06 | 13.48901391 |

# Appendix 8

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Size (Width X Height)** | **Node Count** | **Solution Path** | **Branch Count** | **Min Branch** | **Max Branch** | **Branch Range** | **Nodes per Branch** |
| 10 | 100 | 49.24 | 25.14 | 1 | 20.66 | 19.66 | 2.10 |
| 20 | 400 | 161.96 | 93.02 | 1 | 95.02 | 94.02 | 2.80 |
| 30 | 900 | 301.16 | 180.96 | 1 | 265.48 | 264.48 | 3.77 |
| 40 | 1600 | 523.76 | 321.12 | 1 | 414.26 | 413.26 | 3.68 |
| 50 | 2500 | 509.96 | 310.82 | 1 | 466.04 | 465.04 | 7.22 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Size:** | 10 |  |  |  |  |
|  | **Nodes:** | 100 |  |  |  |  |
|  |  |  |  |  |  |  |
|  | **Path Length** | **Branch Count** | **Minimum Branch** | **Maximum Branch** | **Branch Range** | **Average Branch Nodes** |
|  | 41 | 20 | 1 | 21 | 20 | 2.95 |
|  | 31 | 18 | 1 | 39 | 38 | 3.833333333 |
|  | 45 | 26 | 1 | 28 | 27 | 2.115384615 |
|  | 33 | 28 | 1 | 22 | 21 | 2.392857143 |
|  | 35 | 28 | 1 | 28 | 27 | 2.321428571 |
|  | 39 | 28 | 1 | 19 | 18 | 2.178571429 |
|  | 33 | 25 | 1 | 43 | 42 | 2.68 |
|  | 37 | 21 | 1 | 43 | 42 | 3 |
|  | 67 | 26 | 1 | 8 | 7 | 1.269230769 |
|  | 51 | 28 | 1 | 19 | 18 | 1.75 |
|  | 39 | 20 | 1 | 25 | 24 | 3.05 |
|  | 25 | 20 | 1 | 52 | 51 | 3.75 |
|  | 45 | 25 | 1 | 22 | 21 | 2.2 |
|  | 23 | 19 | 1 | 54 | 53 | 4.052631579 |
|  | 55 | 28 | 1 | 16 | 15 | 1.607142857 |
|  | 57 | 26 | 1 | 11 | 10 | 1.653846154 |
|  | 73 | 24 | 1 | 2 | 1 | 1.125 |
|  | 37 | 21 | 1 | 43 | 42 | 3 |
|  | 55 | 30 | 1 | 16 | 15 | 1.5 |
|  | 63 | 30 | 1 | 5 | 4 | 1.233333333 |
|  | 77 | 13 | 1 | 10 | 9 | 1.769230769 |
|  | 39 | 23 | 1 | 20 | 19 | 2.652173913 |
|  | 81 | 17 | 1 | 3 | 2 | 1.117647059 |
|  | 39 | 24 | 1 | 36 | 35 | 2.541666667 |
|  | 65 | 30 | 1 | 4 | 3 | 1.166666667 |
|  | 55 | 21 | 1 | 23 | 22 | 2.142857143 |
|  | 39 | 30 | 1 | 31 | 30 | 2.033333333 |
|  | 35 | 26 | 1 | 39 | 38 | 2.5 |
|  | 49 | 31 | 1 | 19 | 18 | 1.64516129 |
|  | 41 | 35 | 1 | 24 | 23 | 1.685714286 |
|  | 37 | 16 | 1 | 16 | 15 | 3.9375 |
|  | 43 | 22 | 1 | 33 | 32 | 2.590909091 |
|  | 55 | 30 | 1 | 9 | 8 | 1.5 |
|  | 57 | 21 | 1 | 21 | 20 | 2.047619048 |
|  | 49 | 36 | 1 | 9 | 8 | 1.416666667 |
|  | 67 | 23 | 1 | 10 | 9 | 1.434782609 |
|  | 75 | 21 | 1 | 5 | 4 | 1.19047619 |
|  | 45 | 27 | 1 | 28 | 27 | 2.037037037 |
|  | 59 | 30 | 1 | 12 | 11 | 1.366666667 |
|  | 53 | 27 | 1 | 16 | 15 | 1.740740741 |
|  | 45 | 21 | 1 | 14 | 13 | 2.619047619 |
|  | 43 | 32 | 1 | 12 | 11 | 1.78125 |
|  | 53 | 18 | 1 | 29 | 28 | 2.611111111 |
|  | 63 | 25 | 1 | 7 | 6 | 1.48 |
|  | 51 | 27 | 1 | 12 | 11 | 1.814814815 |
|  | 67 | 26 | 1 | 7 | 6 | 1.269230769 |
|  | 41 | 28 | 1 | 17 | 16 | 2.107142857 |
|  | 55 | 28 | 1 | 14 | 13 | 1.607142857 |
|  | 49 | 29 | 1 | 23 | 22 | 1.75862069 |
|  | 51 | 29 | 1 | 14 | 13 | 1.689655172 |
| AVERAGE | 49.24 | 25.14 | 1 | 20.66 | 19.66 | 2.098352497 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Size:** | 20 |  |  |  |  |
|  | **Nodes:** | 400 |  |  |  |  |
|  |  |  |  |  |  |  |
|  | **Path Length** | **Branch Count** | **Minimum Branch** | **Maximum Branch** | **Branch Range** | **Average Branch Nodes** |
|  | 131 | 60 | 1 | 66 | 65 | 4.483333333 |
|  | 155 | 106 | 1 | 81 | 80 | 2.311320755 |
|  | 225 | 107 | 1 | 12 | 11 | 1.635514019 |
|  | 201 | 126 | 1 | 27 | 26 | 1.579365079 |
|  | 123 | 87 | 1 | 168 | 167 | 3.183908046 |
|  | 213 | 112 | 1 | 29 | 28 | 1.669642857 |
|  | 169 | 84 | 1 | 144 | 143 | 2.75 |
|  | 183 | 85 | 1 | 86 | 85 | 2.552941176 |
|  | 157 | 91 | 1 | 121 | 120 | 2.67032967 |
|  | 143 | 102 | 1 | 79 | 78 | 2.519607843 |
|  | 165 | 68 | 1 | 35 | 34 | 3.455882353 |
|  | 251 | 109 | 1 | 8 | 7 | 1.366972477 |
|  | 125 | 82 | 1 | 179 | 178 | 3.353658537 |
|  | 179 | 104 | 1 | 47 | 46 | 2.125 |
|  | 153 | 107 | 1 | 120 | 119 | 2.308411215 |
|  | 131 | 85 | 1 | 151 | 150 | 3.164705882 |
|  | 165 | 96 | 1 | 109 | 108 | 2.447916667 |
|  | 155 | 95 | 1 | 121 | 120 | 2.578947368 |
|  | 177 | 121 | 1 | 39 | 38 | 1.842975207 |
|  | 119 | 77 | 1 | 147 | 146 | 3.649350649 |
|  | 85 | 46 | 1 | 87 | 86 | 6.847826087 |
|  | 215 | 92 | 1 | 73 | 72 | 2.010869565 |
|  | 195 | 137 | 1 | 29 | 28 | 1.496350365 |
|  | 141 | 106 | 1 | 78 | 77 | 2.443396226 |
|  | 103 | 65 | 1 | 218 | 217 | 4.569230769 |
|  | 195 | 127 | 1 | 45 | 44 | 1.614173228 |
|  | 169 | 95 | 1 | 102 | 101 | 2.431578947 |
|  | 187 | 92 | 1 | 103 | 102 | 2.315217391 |
|  | 237 | 87 | 1 | 70 | 69 | 1.873563218 |
|  | 123 | 74 | 1 | 161 | 160 | 3.743243243 |
|  | 205 | 86 | 1 | 16 | 15 | 2.26744186 |
|  | 103 | 55 | 1 | 201 | 200 | 5.4 |
|  | 95 | 73 | 1 | 185 | 184 | 4.178082192 |
|  | 109 | 86 | 1 | 120 | 119 | 3.38372093 |
|  | 135 | 87 | 1 | 129 | 128 | 3.045977011 |
|  | 119 | 89 | 1 | 185 | 184 | 3.157303371 |
|  | 235 | 109 | 1 | 28 | 27 | 1.513761468 |
|  | 129 | 86 | 1 | 116 | 115 | 3.151162791 |
|  | 115 | 75 | 1 | 193 | 192 | 3.8 |
|  | 115 | 80 | 1 | 165 | 164 | 3.5625 |
|  | 175 | 72 | 1 | 29 | 28 | 3.125 |
|  | 109 | 58 | 1 | 113 | 112 | 5.017241379 |
|  | 203 | 117 | 1 | 22 | 21 | 1.683760684 |
|  | 193 | 128 | 1 | 36 | 35 | 1.6171875 |
|  | 303 | 87 | 1 | 5 | 4 | 1.114942529 |
|  | 183 | 127 | 1 | 44 | 43 | 1.708661417 |
|  | 205 | 122 | 1 | 50 | 49 | 1.598360656 |
|  | 181 | 140 | 1 | 37 | 36 | 1.564285714 |
|  | 135 | 88 | 1 | 94 | 93 | 3.011363636 |
|  | 81 | 61 | 1 | 248 | 247 | 5.229508197 |
| AVERAGE | 161.96 | 93.02 | 1 | 95.02 | 94.02 | 2.80250987 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Size:** | 30 |  |  |  |  |
|  | **Nodes:** | 900 |  |  |  |  |
|  |  |  |  |  |  |  |
|  | **Path Length** | **Branch Count** | **Minimum Branch** | **Maximum Branch** | **Branch Range** | **Average Branch Nodes** |
|  | 345 | 160 | 1 | 104 | 103 | 3.46875 |
|  | 255 | 165 | 1 | 334 | 333 | 3.909090909 |
|  | 195 | 155 | 1 | 506 | 505 | 4.548387097 |
|  | 337 | 210 | 1 | 303 | 302 | 2.680952381 |
|  | 325 | 163 | 1 | 365 | 364 | 3.527607362 |
|  | 337 | 216 | 1 | 232 | 231 | 2.606481481 |
|  | 113 | 100 | 1 | 419 | 418 | 7.87 |
|  | 319 | 185 | 1 | 335 | 334 | 3.140540541 |
|  | 145 | 124 | 1 | 566 | 565 | 6.088709677 |
|  | 261 | 215 | 1 | 253 | 252 | 2.972093023 |
|  | 269 | 142 | 1 | 96 | 95 | 4.443661972 |
|  | 211 | 135 | 1 | 321 | 320 | 5.103703704 |
|  | 375 | 224 | 1 | 204 | 203 | 2.34375 |
|  | 255 | 196 | 1 | 269 | 268 | 3.290816327 |
|  | 365 | 222 | 1 | 186 | 185 | 2.40990991 |
|  | 343 | 250 | 1 | 161 | 160 | 2.228 |
|  | 189 | 115 | 1 | 478 | 477 | 6.182608696 |
|  | 347 | 185 | 1 | 296 | 295 | 2.989189189 |
|  | 429 | 199 | 1 | 252 | 251 | 2.366834171 |
|  | 397 | 224 | 1 | 154 | 153 | 2.245535714 |
|  | 139 | 77 | 1 | 171 | 170 | 9.883116883 |
|  | 219 | 154 | 1 | 267 | 266 | 4.422077922 |
|  | 357 | 269 | 1 | 86 | 85 | 2.018587361 |
|  | 459 | 245 | 1 | 129 | 128 | 1.8 |
|  | 335 | 216 | 1 | 255 | 254 | 2.615740741 |
|  | 317 | 176 | 1 | 272 | 271 | 3.3125 |
|  | 377 | 189 | 1 | 189 | 188 | 2.767195767 |
|  | 311 | 156 | 1 | 405 | 404 | 3.775641026 |
|  | 189 | 140 | 1 | 309 | 308 | 5.078571429 |
|  | 415 | 272 | 1 | 80 | 79 | 1.783088235 |
|  | 203 | 101 | 1 | 106 | 105 | 6.900990099 |
|  | 195 | 126 | 1 | 484 | 483 | 5.595238095 |
|  | 467 | 247 | 1 | 133 | 132 | 1.753036437 |
|  | 251 | 170 | 1 | 394 | 393 | 3.817647059 |
|  | 307 | 188 | 1 | 200 | 199 | 3.154255319 |
|  | 133 | 107 | 1 | 607 | 606 | 7.168224299 |
|  | 481 | 259 | 1 | 53 | 52 | 1.617760618 |
|  | 125 | 108 | 1 | 407 | 406 | 7.175925926 |
|  | 307 | 216 | 1 | 210 | 209 | 2.74537037 |
|  | 199 | 175 | 1 | 424 | 423 | 4.005714286 |
|  | 437 | 187 | 1 | 22 | 21 | 2.475935829 |
|  | 399 | 208 | 1 | 101 | 100 | 2.408653846 |
|  | 393 | 228 | 1 | 96 | 95 | 2.223684211 |
|  | 263 | 177 | 1 | 206 | 205 | 3.598870056 |
|  | 177 | 156 | 1 | 422 | 421 | 4.634615385 |
|  | 189 | 107 | 1 | 590 | 589 | 6.644859813 |
|  | 423 | 139 | 1 | 157 | 156 | 3.431654676 |
|  | 405 | 255 | 1 | 127 | 126 | 1.941176471 |
|  | 515 | 248 | 1 | 91 | 90 | 1.552419355 |
|  | 259 | 167 | 1 | 447 | 446 | 3.838323353 |
| AVERAGE | 301.16 | 180.96 | 1 | 265.48 | 264.48 | 3.77114994 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Size:** | 40 |  |  |  |  |
|  | **Nodes:** | 1600 |  |  |  |  |
|  |  |  |  |  |  |  |
|  | **Path Length** | **Branch Count** | **Minimum Branch** | **Maximum Branch** | **Branch Range** | **Average Branch Nodes** |
|  | 587 | 271 | 1 | 63 | 62 | 3.73800738 |
|  | 365 | 221 | 1 | 461 | 460 | 5.588235294 |
|  | 425 | 292 | 1 | 535 | 534 | 4.023972603 |
|  | 753 | 442 | 1 | 106 | 105 | 1.916289593 |
|  | 287 | 244 | 1 | 673 | 672 | 5.381147541 |
|  | 525 | 322 | 1 | 459 | 458 | 3.338509317 |
|  | 541 | 329 | 1 | 316 | 315 | 3.218844985 |
|  | 353 | 294 | 1 | 720 | 719 | 4.241496599 |
|  | 617 | 339 | 1 | 245 | 244 | 2.899705015 |
|  | 729 | 396 | 1 | 330 | 329 | 2.199494949 |
|  | 531 | 292 | 1 | 37 | 36 | 3.660958904 |
|  | 487 | 292 | 1 | 510 | 509 | 3.811643836 |
|  | 505 | 310 | 1 | 535 | 534 | 3.532258065 |
|  | 531 | 417 | 1 | 259 | 258 | 2.563549161 |
|  | 553 | 356 | 1 | 374 | 373 | 2.941011236 |
|  | 407 | 309 | 1 | 432 | 431 | 3.860841424 |
|  | 349 | 266 | 1 | 601 | 600 | 4.703007519 |
|  | 355 | 245 | 1 | 797 | 796 | 5.081632653 |
|  | 787 | 437 | 1 | 127 | 126 | 1.860411899 |
|  | 323 | 218 | 1 | 950 | 949 | 5.857798165 |
|  | 365 | 193 | 1 | 195 | 194 | 6.398963731 |
|  | 449 | 263 | 1 | 443 | 442 | 4.376425856 |
|  | 625 | 393 | 1 | 146 | 145 | 2.480916031 |
|  | 277 | 230 | 1 | 929 | 928 | 5.752173913 |
|  | 485 | 413 | 1 | 144 | 143 | 2.699757869 |
|  | 803 | 442 | 1 | 157 | 156 | 1.803167421 |
|  | 385 | 274 | 1 | 686 | 685 | 4.434306569 |
|  | 501 | 297 | 1 | 672 | 671 | 3.7003367 |
|  | 949 | 494 | 1 | 30 | 29 | 1.317813765 |
|  | 513 | 360 | 1 | 503 | 502 | 3.019444444 |
|  | 627 | 285 | 1 | 98 | 97 | 3.414035088 |
|  | 347 | 227 | 1 | 917 | 916 | 5.519823789 |
|  | 799 | 443 | 1 | 151 | 150 | 1.808126411 |
|  | 349 | 244 | 1 | 619 | 618 | 5.12704918 |
|  | 349 | 225 | 1 | 988 | 987 | 5.56 |
|  | 645 | 356 | 1 | 386 | 385 | 2.68258427 |
|  | 759 | 444 | 1 | 156 | 155 | 1.894144144 |
|  | 455 | 302 | 1 | 699 | 698 | 3.791390728 |
|  | 453 | 271 | 1 | 527 | 526 | 4.232472325 |
|  | 655 | 399 | 1 | 302 | 301 | 2.368421053 |
|  | 365 | 175 | 1 | 160 | 159 | 7.057142857 |
|  | 663 | 366 | 1 | 361 | 360 | 2.56010929 |
|  | 585 | 383 | 1 | 357 | 356 | 2.650130548 |
|  | 473 | 297 | 1 | 343 | 342 | 3.794612795 |
|  | 569 | 338 | 1 | 448 | 447 | 3.050295858 |
|  | 739 | 431 | 1 | 151 | 150 | 1.997679814 |
|  | 593 | 344 | 1 | 386 | 385 | 2.927325581 |
|  | 567 | 321 | 1 | 280 | 279 | 3.218068536 |
|  | 571 | 365 | 1 | 244 | 243 | 2.819178082 |
|  | 263 | 189 | 1 | 705 | 704 | 7.074074074 |
| AVERAGE | 523.76 | 321.12 | 1 | 414.26 | 413.26 | 3.678975737 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Size:** | 50 |  |  |  |  |
|  | **Nodes:** | 2500 |  |  |  |  |
|  |  |  |  |  |  |  |
|  | **Path Length** | **Branch Count** | **Minimum Branch** | **Maximum Branch** | **Branch Range** | **Average Branch Nodes** |
|  | 411 | 199 | 1 | 141 | 140 | 10.49748744 |
|  | 705 | 329 | 1 | 285 | 284 | 5.455927052 |
|  | 425 | 301 | 1 | 669 | 668 | 6.893687708 |
|  | 453 | 258 | 1 | 780 | 779 | 7.934108527 |
|  | 279 | 223 | 1 | 816 | 815 | 9.959641256 |
|  | 623 | 352 | 1 | 506 | 505 | 5.332386364 |
|  | 709 | 431 | 1 | 209 | 208 | 4.155452436 |
|  | 411 | 319 | 1 | 463 | 462 | 6.548589342 |
|  | 811 | 471 | 1 | 79 | 78 | 3.585987261 |
|  | 377 | 243 | 1 | 957 | 956 | 8.736625514 |
|  | 823 | 319 | 1 | 27 | 26 | 5.257053292 |
|  | 721 | 365 | 1 | 185 | 184 | 4.873972603 |
|  | 719 | 347 | 1 | 195 | 194 | 5.132564841 |
|  | 399 | 273 | 1 | 789 | 788 | 7.695970696 |
|  | 213 | 184 | 1 | 656 | 655 | 12.42934783 |
|  | 369 | 256 | 1 | 780 | 779 | 8.32421875 |
|  | 563 | 362 | 1 | 355 | 354 | 5.350828729 |
|  | 563 | 343 | 1 | 314 | 313 | 5.647230321 |
|  | 323 | 256 | 1 | 745 | 744 | 8.50390625 |
|  | 845 | 472 | 1 | 101 | 100 | 3.506355932 |
|  | 325 | 161 | 1 | 105 | 104 | 13.50931677 |
|  | 379 | 204 | 1 | 759 | 758 | 10.39705882 |
|  | 397 | 257 | 1 | 558 | 557 | 8.182879377 |
|  | 319 | 205 | 1 | 977 | 976 | 10.63902439 |
|  | 325 | 221 | 1 | 694 | 693 | 9.841628959 |
|  | 559 | 411 | 1 | 234 | 233 | 4.722627737 |
|  | 701 | 440 | 1 | 199 | 198 | 4.088636364 |
|  | 637 | 383 | 1 | 265 | 264 | 4.864229765 |
|  | 777 | 414 | 1 | 270 | 269 | 4.161835749 |
|  | 367 | 257 | 1 | 842 | 841 | 8.299610895 |
|  | 835 | 333 | 1 | 20 | 19 | 5 |
|  | 421 | 224 | 1 | 543 | 542 | 9.28125 |
|  | 523 | 295 | 1 | 560 | 559 | 6.701694915 |
|  | 463 | 305 | 1 | 611 | 610 | 6.678688525 |
|  | 729 | 363 | 1 | 286 | 285 | 4.878787879 |
|  | 489 | 204 | 1 | 712 | 711 | 9.857843137 |
|  | 379 | 244 | 1 | 905 | 904 | 8.692622951 |
|  | 635 | 357 | 1 | 287 | 286 | 5.224089636 |
|  | 317 | 213 | 1 | 856 | 855 | 10.24882629 |
|  | 509 | 358 | 1 | 389 | 388 | 5.561452514 |
|  | 249 | 134 | 1 | 304 | 303 | 16.79850746 |
|  | 587 | 411 | 1 | 182 | 181 | 4.654501217 |
|  | 533 | 306 | 1 | 333 | 332 | 6.428104575 |
|  | 529 | 658 | 1 | 295 | 294 | 2.995440729 |
|  | 467 | 326 | 1 | 362 | 361 | 6.236196319 |
|  | 587 | 345 | 1 | 370 | 369 | 5.544927536 |
|  | 385 | 342 | 1 | 565 | 564 | 6.184210526 |
|  | 185 | 160 | 1 | 721 | 720 | 14.46875 |
|  | 461 | 302 | 1 | 764 | 763 | 6.751655629 |
|  | 687 | 405 | 1 | 282 | 281 | 4.47654321 |
| AVERAGE | 509.96 | 310.82 | 1 | 466.04 | 465.04 | 7.22384568 |

# Appendix 9

All relevant documentation and files are on both attached discs.