

**TODO:**

* Overview of the project

> clear statement that describes the problem area and specific problem being solved/ investigated

* Website based roguelike game, with levels based on a series of procedurally generated mazes
* Aim to give the user an amount of control over the generation, e.g. maze sizes, weighting on the generation algorithm etc.

This is documentation for a website based rogue-lite game with levels based on a series of procedurally generated mazes. It will be built upon a variety of technologies, including the Flask application framework for Python and Object Oriented Programming in Javascript. The application will procedurally generate mazes with customisable features **(such as size, \_\_\_\_)** using a Python backend to be sent to the client.

* Background to project

> background in sufficient detail for a third party to understand the problem being solved/ investigated.

* Why did I choose this project
* What is procedural generation
* brief history e.g what is it used for & pros and cons
* Procedurally generated mazes in games
* What are roguelike games
* history of roguelike and roguelite games (talk about maze ones)
* Why is this a roguelike game - fits with maze structure

*Why did I choose this project?*

I have always been interested in games that offer longevity and replayability without repeating the same activities over and over. This can generally be achieved in two ways, either by continuously updating the game or by using procedural generation. The downside of having to continuously update a game is the large time commitments required by developers over the game’s lifetime, to create more new levels and experiences for the end users. Procedural generation however, especially in the example of roguelike games, offers an efficient way of creating a game which is different every time you play, whilst retaining similar challenges and structure for users to play through.

The idea of a maze based game enticed me as it seemed like an interesting and exciting way to experience a roguelike dungeon crawler (a game where the player must make their way through a series of rooms and levels, known as a ‘dungeon’), as players would have the added challenge of navigating through the maze, on top of any enemies they might have to fight or challenges they might have to complete.

*What is procedural generation?*

Procedural generation is a method of producing data automatically, without the need for it to be manually designed. In games, it is when an aspect of the game is controlled by algorithms which determine what that aspect is, either randomly or based on a set of criteria. Examples of procedural generation include:

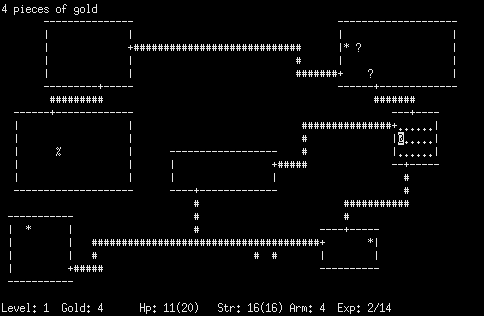
* Changing loot players receive depending on how far through the game they have progressed, e.g. giving more powerful rewards to higher level players
* Generating terrain or a game map , allowing near infinite possibilities of completely different worlds with one piece of code.
* Deciding where, when, how many, and what enemies should be spawned, e.g. based on the area of the map, the in-game time, the difficulty of a level or the level of a player.

Many popular games such as Minecraft, Terraria, Stardew Valley, No Man’s Sky, the Civilisation series, the Borderlands series, Rogue, and Rogue Legacy use procedural generation to enhance the player’s experience, offering slightly unique experiences each time they are played.

*Procedurally Generated Mazes in Games*

This project will mainly focus on using procedural generation to create mazes. This style of game is not one with many well known or renowned games - however, there are still several examples of games that use procedural generation to create mazes. Most of these examples fall into the RougeLike genre.

*What are Roguelike and Rogue-lite Games? (+history)*



Roguelike is a genre of games characterised by a dungeon crawl through procedurally generated levels, with permanent death of the player (i.e. you lose all progress on death). 1980’s *Rogue*, the namesake of the genre, is commonly considered as the original roguelike - although some games matching the criteria were released before it - and is a terminal based ASCII game, where the player would navigate through a series of dungeons, and everything was represented using different ASCII characters.

Over time, many more roguelike games were released, and the genre grew in popularity. However, some of the more modern games were not incorporating all of the features of a typical roguelike, by allowing players to save data between runs or using procedural generation, for example. Because of this, the term ‘rogue-lite’ was proposed to make the distinction between these games and true roguelikes.

*Why is this a Rogue-lite game?*

I thought that the procedurally generated maze structure of the game would fit well with the classic traits of a roguelike game, notably the procedural generation and dungeon crawling aspects. However, the game will be a rogue-lite as it will not have a permanent death mechanic. I made this decision so players could save and revisit old mazes that they had played**,** and would be able to save progress between runs, as it allows for a more enjoyable and less frustrating experience.

* **Interview an end user**

> what genres of games do they typically find enjoyable

> do they enjoy fixed level based games

> would they enjoy a game based on navigating mazes

* why?

> would they prefer more randomness in level generation than fixed structure? Would this increase the time they spend? Would they prefer a mixture of the two?

This project will be aimed at:

* Users who enjoy the roguelike and rogue-lite genres of games
* Users who find mazes interesting and engaging challenges
* Users who appreciate the ease of access of a web-based game

The following is an interview with a prospective end user, someone who fits into all of the above categories.

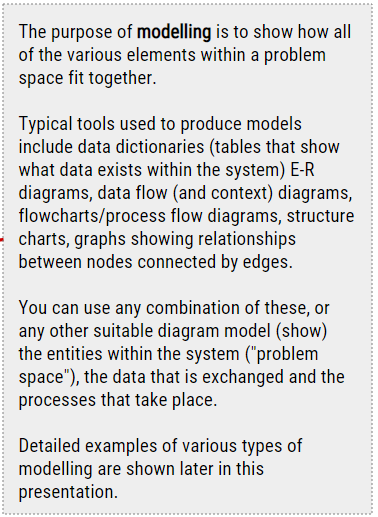
| **1. What genres of games do you enjoy playing?** |
| --- |
| I enjoy a variety of game genres however I prefer roguelikes, puzzle games, and action-adventure games |
| **2. Do you find games with fixed level based structures enjoyable?** |
| Yes I do, I think it can allow developers to create balanced and enjoyable challenges as the game progress |
| **3. Do you think you would enjoy a game based on navigating mazes? If so, why?** |
| I do think I would enjoy a game about navigating mazes. I think that mazes are not only incredibly intellectually stimulating but would also be a great and versatile medium to keep a player engaged |
| **4. Would you prefer more randomness over fixed structure in a game, or a balance of the two, and do you think this would increase the time you spend in the game?** |
| I think some randomness is required in order to keep a game feeling fresh and unique however, too much randomness would make it impossible for a player to be invested in the experience so I think a balance between the two options would be the best way to increase the time I spend in a game |
| *The interviewee desires a balance between randomness and fixed structure* |
| **5. Do you think that a good example of the balance would be keeping game mechanics the same throughout, but randomising the layout of the map or terrain?** |
| I think that is an excellent example of balance. Keeping the core game mechanics as a rock while having randomised elements such as layout and enemy spawns can provide a unique and gripping experience every time you play |
| *Consistent game mechanics, but changing the maze for each level is a good example of this balance. They would also expect a game such as this to have some kind of enemy system* |
| **6. Looking at these maze examples, which do you think best suits a roguelite style game?** |
| The maze that utilises recursive backtracking is the most difficult and would pose the greatest blend of challenge and interesting gameplay while rewarding the player for significant problem solving. The other maps are poor due to the simple pathways, multitude of short dead ends, and lack of real player choice as choosing the wrong path will result in immediate failure and backtracking which would be disheartening. |
| *The interviewee regards the recursive backtracking algorithm as the best in terms of the shape of the maze, and how it will have to be solved* |
| **7. Would you like to have a level of control over how each maze generates? If so, what factors would you like to be able to change or control?** |
| I think that it could significantly improve the game after completing the main game to have access to a type of “free play” that would allow them to create mazes based on their own parameters in order to increase the longevity of the game. |
| *They would like have the option to generate custom mazes once they have completed the rest of the game* |
| **8. Is there anything else you would like to see from this project? Are there any other requirements that you have?** |
| I would like to be able to play the game from multiple devices and have my save data transfer between these devices. Furthermore, it would be very pleasant to be able to play the game on the go or when I am out and about. |
| *The interviewee would like to be able to play the game on any device, and have their progress save between those devices* |

As evidenced by their answers to the first few questions, the interviewee is a good example of a prospective user of the game, as they prefer roguelikes and puzzle games, and enjoy solving mazes and playing games that involve them.

Summarising the results of the interview, it is clear that the project should include:

* Recursive-backtracking as the main algorithm used to generate mazes
* Game levels that feature randomly generated mazes, but keep core mechanics the same throughout
* ‘Enemies’ to face within each maze
* A ‘free-play’ mode after a user has completed the levels, which would allow them to have more control over how the mazes are generated
* Some kind of easy portability of the game, and a way to save data between devices
* **Survey more end users - ask more people the interview questions?**

> Based on results of interview



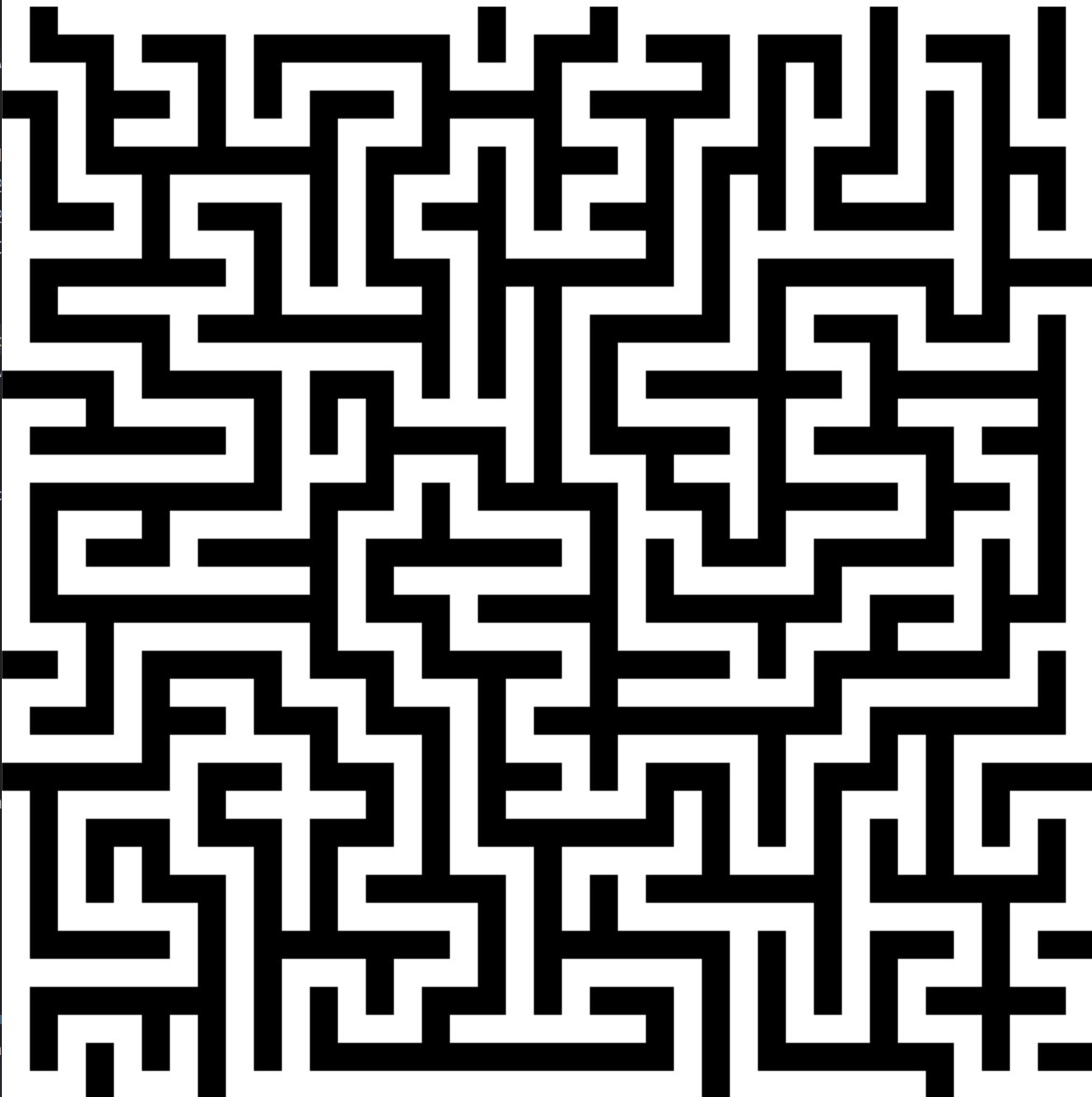
* **Modelling the project**
  + **Maze generation algorithm analysis**

**> how do maze gen algorithms work**

> overview of different algorithms (with efficiency) -> evolution of generation algs

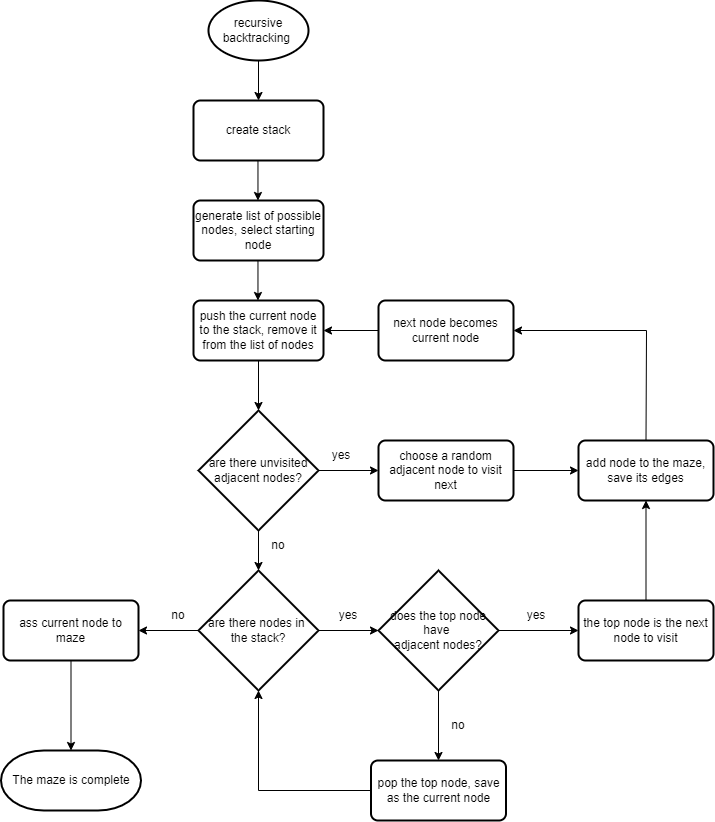
Img sauce = <https://en.wikipedia.org/wiki/Maze_generation_algorithm>

- recursive backtracking

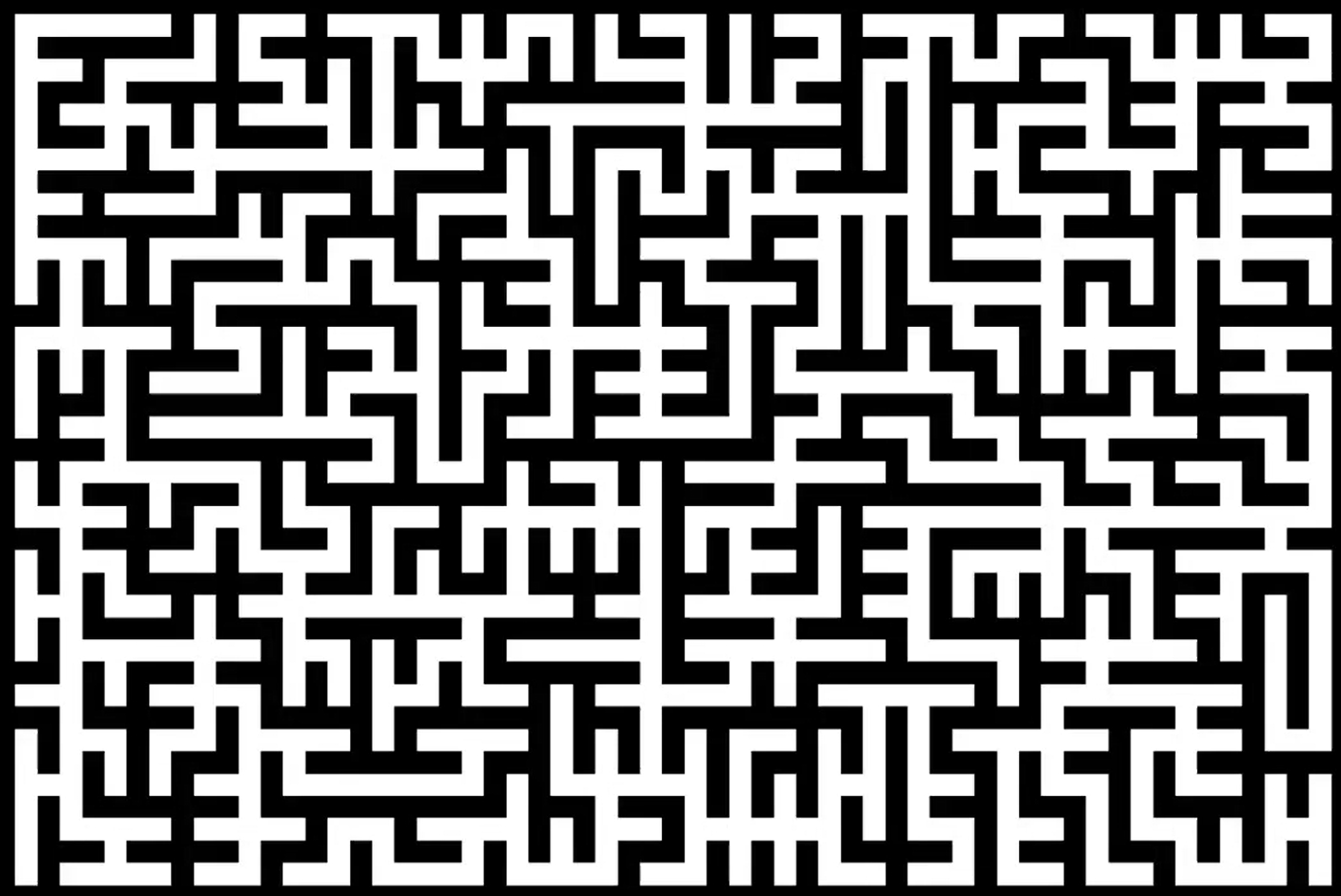


Recursive backtracking is an algorithm that is used for both generating and solving mazes. The algorithm does a random ‘walk’ until it reaches a dead end, then will check each previously visited node until it can continue again. This algorithm produces mazes that look like typical mazes - having long winding corridors and not too many dead ends.

The time complexity of the recursive backtracking algorithm is O(2n) where n is the number of nodes in the maze. This is because each node will be visit once as the algorithm randomly walks around the maze, then again when it backtracks through.



* kruskals

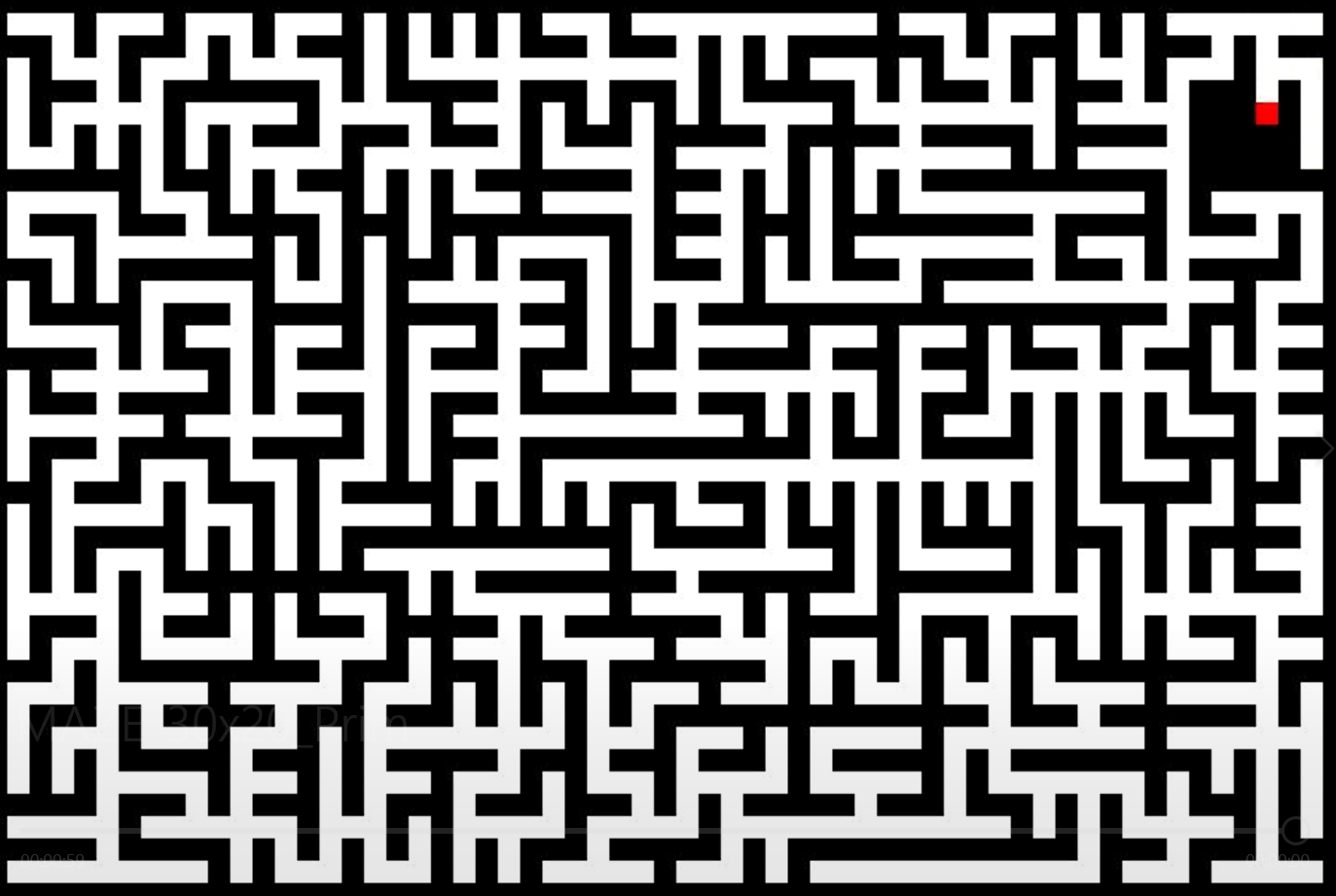


Kruskal’s algorithm is one that was designed to produce a minimum spanning tree from a list of weighted nodes. By giving each node a random weight, it can be used to produce a maze. Each node starts as its own ‘group’ (known as a disjoint set) then two adjacent nodes are chosen to be merged into one group, bringing any other nodes in their group with them. The maze is complete when all the nodes are in one group. Unfortunately, Kruskal’s algorithm produces mazes with lots of very short dead ends, in an almost repetitive pattern, which, as confirmed by the results of the interview, is not as well suited to a maze as the recursive backtracking algorithm.

The time complexity of Kruskal’s is O(nlogn), where n is the area, due to the implementation of a disjoint tree.

**If you allow nodes to merge under nodes between them, you can achieve a weave maze**

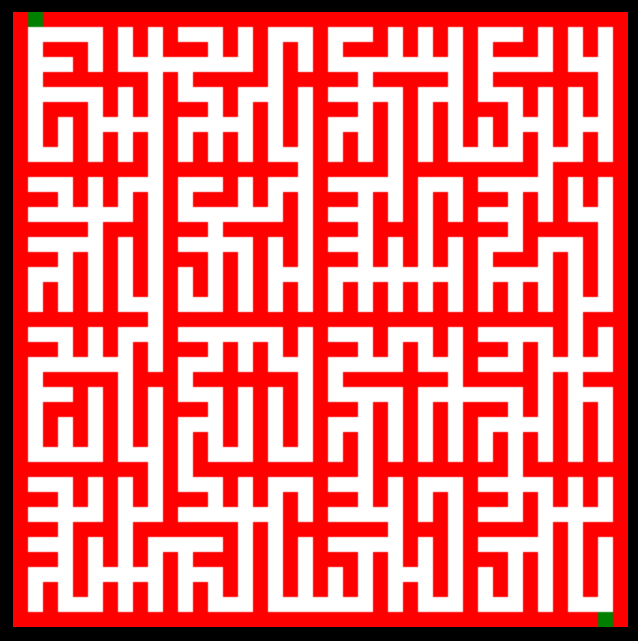
* Primm’s



Primm's algorithm, like Kruskal's, is also for producing a minimum weighted spanning tree. From the starting node,each adjacent node is added to a list,then a random node is chosen to be visited, until there are no nodes in the list. Unsurprisingly, as Primm’s is a similar algorithm to Kruskal’s it produces fairly similar mazes. However, the mazes are worse than those generated by Kruskals as they end up with several ‘branches’ off from the starting node, that if the player takes, they will have to return to the start to choose a different branch - almost as if they were exploring a tree, not a maze.

The time complexity of Primm’s is O(n), where nis the area, as each node is always visited once.

* Recursive backtracking



Recursive division is a very different algorithm to the others. It divides the area in two at a random point, then adds a path at one position in the wall. It will repeat this process for each of the sub areas created until there are none left. The main problem with recursive division is that the mazes are very predictable and simple to solve. As you can see, this maze has clear divisions between 16 areas, and the mazes becomes ‘which major area do I visit next’.

The time complexity of the recursive division algorithm is O(nlogn).

> analysis of shapes and patterns of mazes

> ease of adding weighting and ‘personalisation’ to recursive backtracking

When using a recursive backtracking algorithm, it is relatively easy to adapt the program to allow large variety in how mazes generate. For example, by choosing a node randomly from the stack (a list in this instance) the maze will generate differently. There is also the possibility of adding weights to the adjacent nodes chosen to visit, causing a tendency for more horizontal or vertical corridors.

Due to the nature of the other algorithms described, none of them allow this level of control over the generation of the maze. However, Kruskal’s algorithm can allow ‘weaver’ mazes to be created, if node groups are allowed to merge under - or over - other nodes.

> demonstration of different implementations?

**> why I’m using recursive backtracking**

In summary, the recursive backtracking algorithm is the best for this project. This is because it is the best maze for the intended game scenario. **Data from survey?**

Ideally, the efficiency of the algorithm would be better - however, in the context that it is being used, the mazes won’t be extremely large, so can still be generated in very short times.

* **Existing solutions**

<https://retrospiritgames.blogspot.com/2014/07/instant-dungeon-roguelike-maze-game-for.html>

<https://store.steampowered.com/app/326720/Instant_Dungeon/>

Instant Dungeon! is a roguelike maze game, and is similar to the idea of this project

From its page on Retro Spirit Games:

\_\_\_

*Instant Dungeon* is a cute little maze game available on both PC and the PlayStation Vita (via the PlayStation mobile section on the store) which features roguelike elements to keep things interesting. Created by lone developer Scott Mattot, *Instant Dungeon* keeps things incredibly simple by tasking you with little more than moving around a succession of randomly generated, top down mazes, avoiding the skeletons, vampires, zombies and other monstrosities, nabbing any treasures along the way and finding the key to open the exit door. As you progress deeper into the dungeons the light get more scarce, meaning danger could lurk around every corner.

\_\_\_

Instant Dungeon has several game modes, however, to unlock most of them, you have to play the standard “Adventure Mix” mode, which is a series of fixed levels, without randomly generated mazes, many times. There is a mode available with random mazes, but they do not increase in difficulty without leaving the game and selecting a harder difficulty. This would be a problem for prospective users of this project as having to complete the same repetitive levels over and over, with no random generation is not particularly enjoyable, and goes against what the interviewee suggested.

On top of this, the portability of Instant Dungeon! is not particularly feasible, as data on the steam version does not transfer between devices, something that is also required by users of this project.

However, there are many strengths to Instant Dungeon!, aside from the mazes, the structure of each level and the consistent game mechanics are places that this project can definitely take valuable inspiration from. Essentially, each level requires finding a key, which then unlocks the way to the next level. This adds some extra challenge to navigating the levels.

* **Critical path analysis**
* **Overall objectives**