Git Gud Pathfinding: Algorithmic Route Optimization in Dark Souls Speedrunning Using Pathfinding Algorithms

Aramazaya - 13523082
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
Email: 13523082@std.stei.itb.ac.id, aramazaya21@gmail.com

Abstract – **Speedrunning has been a staple of the gaming community for a considerable amount of time now. It is one of the most successful fields of gaming to date due to its high retention of viewers. The introduction of Dark Souls to the speedrunning community has only further enhance this proposition. Dark Souls Speedrunning community is one of the biggest speedrunning community there is, with it's high number of community members working together to create the most efficient route ever seen in the game. Even with its old age of 16 years, its speedrunning community is still as lively as ever. With that in mind, a question trainwrecks through my brain. What if we can use algorithms to create the most mathematically based optimized route to speedrun Dark Souls REMASTERED. This paper aims to create a tool that players can use to create their own personalized speedrunning route using Uniformed Cost Search Algorithm. Unlike standard UCS Algorithm, my approach uses pruning to reduce the amount of unnecessary node expansion. It also analyzes triggers for shortcuts and locked areas to complete the game in the fastest way possible. This approach not only can be used in other program, it may also have uses in the field of robotics with conditional obstacle path planning.**

Index Terms – Dark Souls, Uniformed Cost Search, Pathfinding, Speedrunning

INTRODUCTION

Completing an activity in an accelerated phase is a part of the human nature that people just cannot shake off. Somone who achieved a degree in less time than the normal population is considered to be better than most. People are even willing to watch cars go around a bend multiple hundreds of times just to see which one is faster. This view is also prevalent in gaming. In the gaming industry, the concept of finishing a game quicker than the rest of the player base is a common thing to witness. This concept is fittingly called Speedrunning.

The industry of speedrunning is a highly competitive one. As the industry grows, more and more people realized that watching a player finish a game in less time they took to cook a rice is an oddly satisfying experience. Many people benefitted from this concept. Starting with the runner themselves, to charity foundations supported by events that showcases speedrunning accomplishments.

Most of speedrunners achieve the optimal path to complete their game by using their vast amount of game knowledge and experience in the game itself. This author, however, beg to differ. I believe that using pathfinding algorithms with fine-tuned heuristics can achieve a "mathematically perfect" route in speedrunning, specifically in Dark Souls speedrunning.

As someone with a perfect achievement in Dark Souls REMASTERED, I believe that I am qualified enough to create this paper. The purpose of this paper is to find the most optimal path in a Dark Souls speedrun, using pathfinding algorithms that take notice of variables such as enemies and items. I believe this paper will have a colossal impact on the world as we know it.

THEORETICAL FOUNDATION

A. Speedrunning

FIGURE I
REPRESENTATION OF A SPEEDRUNNER BY YOUTUBE CREATOR "HECK NO"

Speedrunning is an activity in which players of a game is prompted to finish said game in near impossible time.

Many online platforms now support this community by being the official place where a player can submit their "time". There are many categories within speedrunning such as TAS (Tool Assisted Speedrun) where the player can use special tool designated to making button inputs at the most perfect moment in the game.

Since speedrunning on itself does not generate much revenue, many runners opt to creating videos on other platforms such as YouTube or Twitch to stimulate more revenue from this hobby that they love. Moreover, most official platforms to submit time requires proof of video recordings of the run itself to verify the integrity of the run. This becomes more incentive for runners to stream their run in a third party platform.

Due to the competitive nature of the subgenre, many individuals choose to cheat their way through the leaderboard. One of the most notorious cheater is Todd Rogers which has swindled Guiness Book of World Records with his speedrun of Dragster. While using a tool to help you during a speedrun may be considered cheating, using algorithms to optimize the run beforehand is not considered a tool assisted run since the algorithm did not directly interfered in the actual run.

B. Dark Souls

FIGURE II
CREATOR OF THE SOULS SERIES INCLUDING DARK SOULS REMASTERED, HIDETAKA MIYAZAKI

In *Shizuoka, Japan*, at September 19, 1974, a legend was born. Hidetaka Miyezaki, the creator of the souls series, grew up in *Shizuoka* prefecture and came to FromSoftware game as a programmer. He was then more wildly known after directing FromSoftware's Armored Core 4 and Armored Core : For Answer[1]. He is the sole reason why many people grow to love the Dark Souls series.

The Dark Souls series is a collection of games starting from Demon's Souls that share a connection lore wise and gameplay wise. Surprisingly, the first game of this multi-hundred-million dollar selling series does not achieve much recognition. The series was then popularized as games that are difficult to beat but very well balanced. With it's infamy, many gamer across the world view the game as a mountain to be conquered. This resulted in an influx of sale of games from the series.

After Demon's Souls popularity surge, FromSoftware produced another game within the same universe as Demon's Souls called Dark Souls. This game will be the main focus of this paper.

Dark Souls is an Action Role-Playing Game developed by FromSoftware and directed by Hidetaka Miyazaki. The game popularity was caused by its high difficulty due to the combination of its battle mechanics and enemy status. The game took to popularity due to its unique world building and story-telling elements. Unlike most game of the genre, Dark Souls did not reward player with a coherent story. Rather, the story of Dark Souls is one to be understood from the world itself.

The popularity of the franchise prompted other game studios to create similar games which are now called "souls-like games" within the community. The newest instalment in the series is called ELDEN RING which estimated to have sold over 20 million copies worldwide[2]. To say that this series is just a genre is an understatement. It is a whole community teeming with many creation from many different people from many different part of the world. One of such creation is the Dark Souls Speedrun.

C. Dark Souls Speedrunning

The Dark Souls speedrunning scene is one with many notable events within its history. With rich community, many players contributed to the discovery of glitches and exploits that made the game capable to be beaten below the 20-minute mark. One such event is the discovery of the ladder glitch. The ladder glitch allows a player to set their character location in the game file to a top of the ladder while the actual character model and camera may move independently. This exploit is the final brick to finish the sub-20 minute run.

While it is possible to finish the game below 20 minute using glitches, most of the glitches used are known to be difficult to input. Not unlike games like Halo: CE, the amount of useful but insanely difficult glitches that requires frame perfect actions are abundant. While the movement of Dark Souls REMASTERED are unlike most games that are more freeing, the community still managed to find many movement glitch and exploits that allowed the game to be finished insanely quickly. However, due to the difficulty of the glitches, this paper will focus on delivering the fastest route during a "glitchless" run rather than an any % run which allows glitches and extreme exploits.

In order to give a comparison for how quick the speedrun of Dark Souls Remastered is, the current any % world record that allows force quits is held by tamatama with a time of 17 minutes and 41 seconds. The category in which no glitch are allowed or glitchless is currently held by m3dic37 with a time of 45 minutes and 23 seconds[3]. The average time a person finishes the game is 29 hours[4]. So to say that this players are godlike in the game are an understatement.

D. Pathfinding Algorithms

Pathfinding, or more appropriately path routing, algorithms are a type of algorithm that focuses on finding a path, usually within a graph or graph-like data, to a finish node from a start node. The two most known path routing algorithms is BFS and DFS. Both focuses on finding one path to the finish node from the start node.

There are two types of path routing algorithms, uninformed search algorithms and informed search algorithms. While algorithms like BFS and DFS are quite effective on finding a path, it is however not so good when each edge of the graph has a cost or weight. Due to the uninformed nature, most uninformed search algorithms does not find the most optimal path to the finish node. This is one of the drawbacks of uninformed search algorithms. On the other hand, while usually taking more resources, the informed search algorithms take notice of the weight or cost of each edge within the search graph. This prompted the algorithm to find the most optimal path from the start node to the finish node.

Algorithms like Uniform Cost Search or Greedy Best-First Search are examples of informed search algorithms. Most informed search algorithms uses a cost function or evaluation function to find the most efficient route within the search graph. This cost function usually looks like this:

$$f(n) = g(n) + h(n) \tag{1}$$

where f(n) is the evaluation function, g(n) is the cost to reach node n, and h(n) is the heuristic evaluation of node n. Not all heuristic can be used to evaluate the cost function. The heuristic evaluation used must show a lesser value than the actual cost to reach the finish node from node n or

$$h(n) \leq h^*(n) \tag{2}$$

where $h^*(n)$ is the true cost from node n to the finish node. A heuristic that fulfils this condition is called an admissible heuristic and will give optimal solution to A* with tree-search.

In order to find the most efficient path in Dark Souls Speedrunning, this paper will use the UCS or Uniformed Cost Search Algorithm. The reasoning for this is that UCS will always result in the most optimal path as long as the cost in the search graph edges is always positive or more than 0.

UCS is an algorithm that only uses the current cost to reach node n as its evaluation function. Ergo, its evaluation function will look like this:

$$f(n) = g(n) \tag{3}$$

where g(n) is the cost to reach node n. Due to this evaluation function, UCS will always give the most optimal path to reach a node within a search graph. The algorithm extend its node following the BFS algorithm where it adds all node within the same depth to the current priority queue. The only downside to UCS is that due to the nature of the algorithm's evaluation function, the space and time needed to process the algorithm is higher than algorithm such as A* which has a heuristic evaluation function that reduces the amount of unnecessary node expansion. However, since fast search time is not needed in the problem of Dark Souls speedrunning, UCS is the best search algorithm to find the optimal path.

ANALYSIS

A. Overview

The first step of finding the optimal path is to make the search graph first. Dark Souls REMASTERED has a complex map where the world is an open-world format. This means that players can go anywhere in the world with some limitations. However, the world was built so that the player is directed in more of a linear pathing rather than a true open world game such as Assassins Creed or Farcry series.

In order to finish the game, a player must complete these following steps:

1. First the player must make their way out of the Undead Asylum to the Firelink Shrine;
2. Second, the player must ring two bells that are located in the Undead Parish and also in Blighttown;
3. Then, the player must make their way to Anor Londo to take the Lordvessel which allows the player to warp between some chosen bonfire;
4. The player then needed to collect four lord souls located in the Duke's Archive, New Londo Ruins, Lost Izalith and lastly Tomb of Giants;
5. Lastly, the player must make their way back to Firelink shrine and then go to the Kiln of the First Flame where they will fight the final boss which marks the end of the game.

This steps mark out some important places that a player must visit to complete the game. Notably, those places are Undead Settlement, Blighttown and in extension Lost Izalith, Tomb of Giants, Anor Londo and in extension Duke's Archive, and New Londo Ruins.

There are myriads of other places that the player can visit and are optional such as the Artorias of the Abyss DLC areas. This paper, however, will not take notice of those places due to the fact that they are not required to finish the game. There are also many areas in between the required one with many looping path from area to area.

To achieve a smaller node size, each area will be divided into smaller cells which contains its own sets of enemies and also loots.

B. The Dark Souls REMASTERED Map

Considering the already stated facts, we can now create a simplified graph-like map of the Dark Souls REMASTERED world. To do this, I divided all area into sub-areas I call cells that indicates in which direction will the player be going.

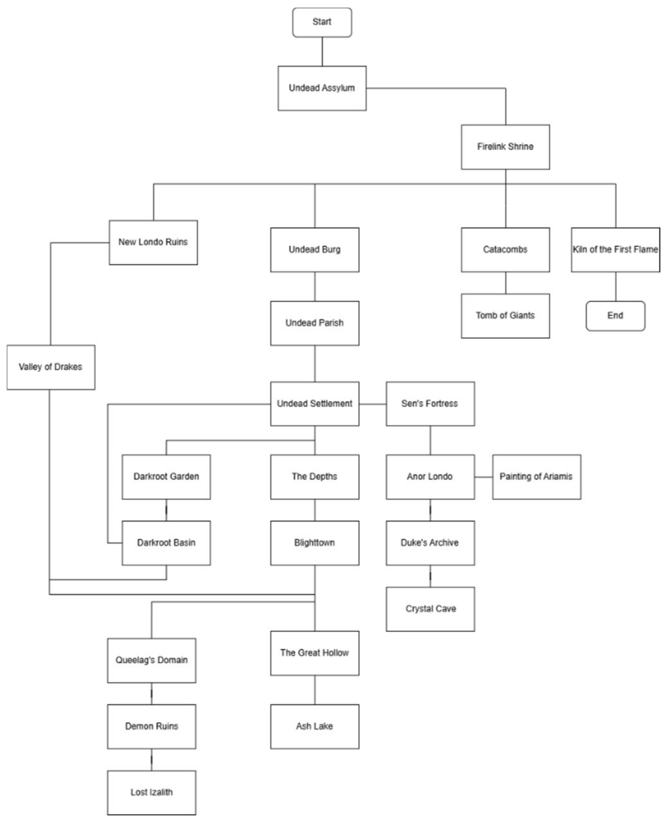
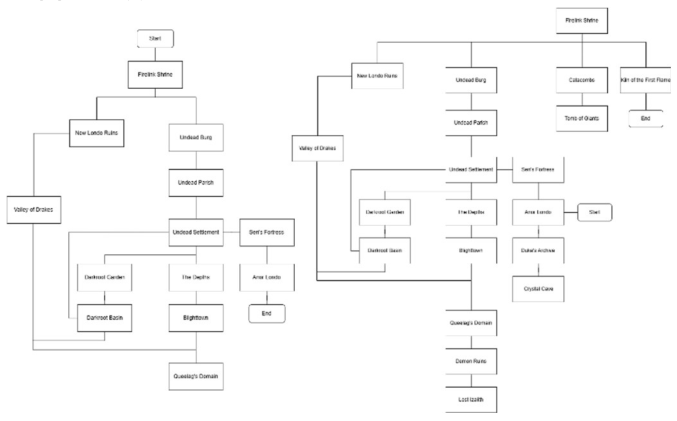


FIGURE III
THE ENTIRE DARK SOULS REMASTERED MAP

As we can see from the map, there are many areas that leads to dead-end. Another consideration to add is that all edges in the graph follows a two-way approach with some of the edges requiring certain items to traverse. After trimming the unneeded area such as Ash Lake or Painting of Ariamis, and dividing the map into two which follows the story progression of the game, we now have the graph as seen in FIGURE IV.



(a) (b)

FIGURE IV
THE ENTIRE DARK SOULS REMASTERED MAP DIVIDED INTO FIRST ACT (a) AND SECOND ACT (b)

In FIGURE IV (a) the player needs to reach both Undead Parish and Queelag's Domain to ring both bells, after which they need to reach Anor Londo to take the Lordvessel. To simulate this, I will first run the algorithm to find the shortest path to Undead Parish per cell. Afterwards, I will then run the algorithm again after modifying the graph to add shortcuts that are unlocked such as the one from Undead Parish to Firelink Shrine. This second algorithm run will find the shortest path from Top of undead parish straight to Quelaag's Domain. Lastly, the algorithm will run one more time to get to Anor Londo through Sen's Fortress.

The second act will be the more complex one within this research. After a player has the Lordvessel in their inventory, they can now warp between notable bonfires. This complicates things because then there are many more edges in the search graph connecting each cells that contain said notable bonfires while also considering which bonfires have been lit up. To do this, I will make the program take notes on which bonfires on which cells has already been lit up and make the algorithm react accordingly. I will divide the search into four parts for all the lord souls. The last run from whichever boss is last into the Kiln of the First Flame is not needed since the fastest path to it is to just warp from the boss bonfire into the Lordvessel bonfire.

It is important to note that some area that are locked by items are mostly due to players not having keys. For example the path from New Londo Ruins to Blighttown through Valley of Drakes is locked behind a door which requires the Master Key to open. For this reason, the class best suited for the Speedrun is the Thief considering they get the Master Key by default unlike other class which requires the player to choose Master Key as their Gift. Moreover, to finish the Undead Asylum quickly, the player will need to pick the Black Firebomb Gift to kill the boss quickly without the need to traverse further into the Asylum.

C. Algorithm Building

The factors that are used to determine the cost of the evaluation function in the algorithm is only the time taken to traverse a path and also the time taken to defeat any essential enemies in the path. For example, from node Taurus Demon Entrance to the Undead Burg Bridge Entrance, some people may have some difficulty in fighting the Taurus Demon, ergo, increasing the time it takes to traverse Taurus Demon Entrance →Undead Burg Bridge Entrance edge. Due to this issue, it is more preferable to create a program that can take user simulation of segments and then creating an optimized path for each person. It is also possible for the algorithm to takes into account the preferred weapon of choice of the player by using trigger mechanism. The program uses this mechanism to handle shortcuts that are only open from a certain time on a certain location.

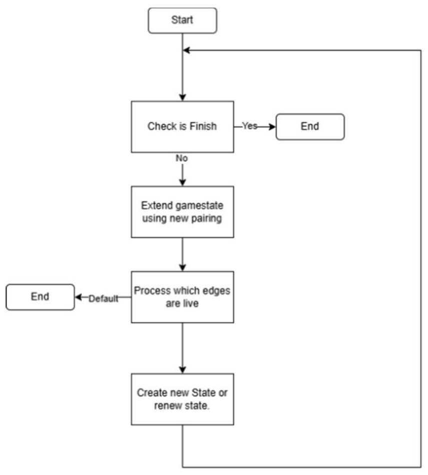

Start
Check is Finish
Yes
End
No
Extend gamestate using new pairing
End
Default
Process which edges are live
Create new State or renew state.

FIGURE V
ALGORITHM PROCESS FLOWCHART

The algorithm will follow the flow chart in FIGURE V. Firstly it will check whether or not this node is the finish node. Afterwards, it will renew the gamestate pairing if it is already present. It will then extends the priority queue using a function that searches for triggered connection and normally available connection. It then added the resulting nodes of this connection query to the priority queue to be looked over net. Next the algorithm will pop the head of the priority queue and renew the game state if available. It then repeats itself until the finish node is found or there are no more nodes in the priority queue. The formula used to count the cost function is

$$f(n) = g(n) = T_{base} \quad (4)$$

where T_{base} is the base traversal time which includes boss fights.

D. Code

The code uses python for simplicity. It views the visited_nodes, location combination as a whole state. This is due to the fact that there are some connection that requires the player to be in a certain area first before they are able to go to the another area. For example, the shortcut from Firelink Shrine straight to the Undead Parish. To use the shortcut, players must first reach Undead Parish through Undead Burg and then turn on the elevator. Until then the only way to go to Undead Parish is through Undead Burg. So now, when the algorithm reaches a certain trigger, the state of that branch is saved as a different state rather than every branch in the queue viewing the shortcut as already opened.

```
@dataclass(frozen=True)
class GameState:
    location: str
    visited_nodes: frozenset
```

It then initialize the prioqueue and the best_cost variable which is used to track the shortest path to reach a certain game state. Afterwards, it went into the main loop. In here, the first thing the algorithm do is pop the head of the priority queue. It then run a check of whether the current location is the goal location.

```
while priority_queue:
    current_cost, _, current_state, path = heappop(priority_queue)
    if current_state.location == goal:
        return path, current_cost
```

if the current location is not the goal location, it then checks whether or not the current state is already in the best_cost variable. If it is, it then see whether or not the cost is better than the one's in the best_cost variable.
this is one of the major differences between this algorithm and normal UCS algorithm. This is done to improve the performance of the search.

```
if current_state in best_costs and current_cost >
best_costs[current_state]:
        continue
```

Next, the algorithm starts expanding its priority queue using neighbor nodes and triggers. Triggers are used to denote changes in pathing due to certain events in certain locations. The openning of the elevator that carries player from Firelink Shrine to Undead Parish is one of them. To achieve this, it first takes all connections that are available and then it creates a new state, and lastly it does the usual UCS stuff.

```
for connection in connections:
    neighbor_loc = connection['neighbor']
    edge_cost = connection['edge_cost']
visited nodes
    new_visited = current_state.visited_nodes | {neighbor_loc}
    new_state = GameState(neighbor_loc, new_visited)
    total_cost = current_cost + edge_cost
    if new_state not in best_costs or total_cost < best_costs[new_state]:
        best_costs[new_state] = total_cost
        new_path = path + [neighbor_loc]
        heappush(priority_queue, (total_cost, counter, new_state, new_path))
```

```
def get_available_connections(self, location: str, visited_nodes: frozenset):
    connections = []
    connections.extend(node.connections)
    for conn in node.triggered_connections:
        trigger_id = conn['trigger_id']
        trigger_active = any(
            trigger_id in self.nodes[visited].triggers
            for visited in visited_nodes
        )
        if trigger_active:
            connections.append(conn)
    return connections
```

E. Result

The result of the program on the act 1 of the Dark Souls REMASTERED Game are as follows:

1. Firelink Shrine (00:00)
2. Undead Merchant Male (Residence Key) (01:05)
3. Undead Burg Bonfire (01:22)
4. Taurus Demon Entrance (02:06)
5. Undead Burg Bridge Entrance (02:41)
6. Sunlight Altar Bonfire (03:09)
7. Undead Parish Bell (05:13)
8. Firelink Shrine (05:16)
9. Valley of Drakes via Master Key Door in New Londo (05:50)
10. Quelaag's Bell (11:13)
11. Firelink Shrine (11:16)
12. Andre of Astora (12:21)

13. Sen’s Fortress Entrance (12:35)
14. Sen’s Fortress Top (14:11)
15. Anor Londo Warp Point After Iron Golem (15:48)
16. Anor Londo Ariamis Painting Building 2F (16:45)
17. Rotating Bridge (17:27)
18. Hell (Two Silver Knight archers) (17:47)
19. Ornstein and Smough Bonfire (18:23)
20. Gwynevere Bonfire (Lordvessel) (21:21)

Obviously, this result is skewed to my playstyle and some historic data of successful speedrunning attempts. It is, however, recommended for someone that wants to use the program to fill in their own capable time for each segments by using simulations. This is merely a representation of some successful speedrun combined with my own attempt on the segments. Moreover, the result does not takes into account exploits such as movement exploit to capitalize off of undead settlement mage. Another thing to note is that the program does not have a feature that takes into account items and its usefulness.

Conclusion

In this paper, I have managed to create a simple pathfinding algorithm to complete Dark Souls REMASTERED in the fastest way possible. However, the algorithm’s heuristic still doesn’t take into account the effects of using certain weapon or certain Items has into the calculation. To use the algorithms, runners will need to simulate segments that they need and count the time to use in the algorithm. While using historical data is pretty nice, in the end I think its better for each player to input their own simulation data rather than just using historical data of previous runners record. The result of my own simulation data combined with some historical data taken from speedrunner soldi can be seen above on the *Result* section.

References

[1] "Hidetaka Miyazaki," Dark Souls Wiki, Fandom, Accessed: Jun. 23, 2025. [Online]. Available: https://darksouls.fandom.com/wiki/Hidetaka_Miyazaki.

[2] "ELDEN RING stats, graphs, and player estimates," PlayTracker Insight, Accessed: Jun. 23, 2025. [Online]. Available: https://playtracker.net/insight/game/71621.

[3] "Dark Souls Remastered," Speedrun.com, Accessed: Jun. 23, 2025. [Online]. Available: https://www.speedrun.com/darksoulsremastered?h=Any_Force_Quit-no-dropmod&x=xd173pzd-kn0kmk08.1dk2wz4l.

[4] "Game Details," HowLongToBeat, Accessed: Jun. 23, 2025. [Online]. Available: https://howlongtobeat.com/game/56677.

[5] Maulidevi, N. U., & Munir, R. (2025). Penentuan Rute (Route/Path Planning) - Bagian 1: BFS, DFS, UCS, Greedy Best First Search. Bahan Kuliah IF2211 Strategi Algoritma, Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika ITB.

[6] Maulidevi, N. U., & Munir, R. (2025). Penentuan Rute (Route/Path Planning) - Bagian 2: Algoritma A*. Bahan Kuliah IF2211 Strategi Algoritma, Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika ITB.

Pernyataan

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 24 Juni 2025

[illegible]

Aramazaya - 13523082

APPENDIX

Full Code:
https://drive.google.com/file/d/1juZXRJeOyFDI_KQQ8OgpKwbzjQUICv0N/view?usp=sharing

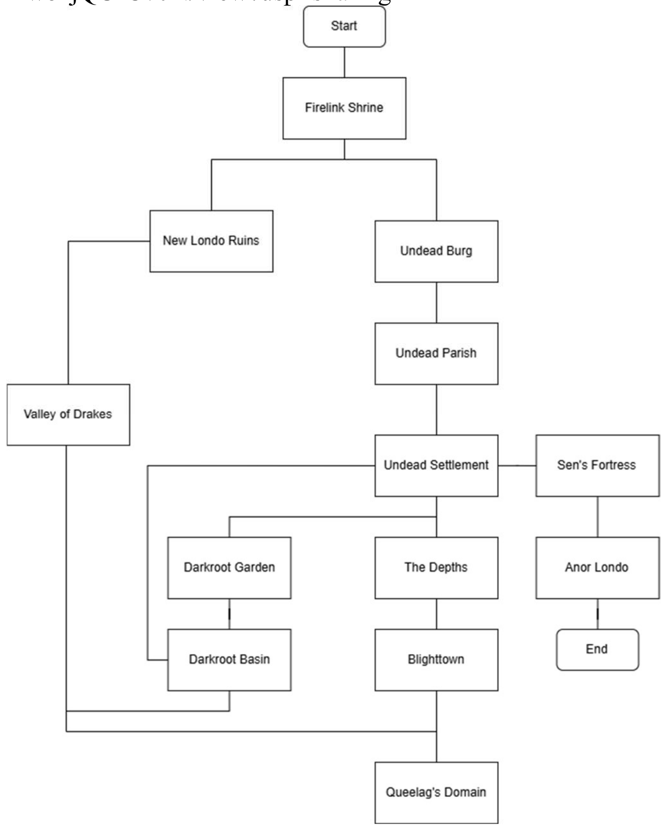


FIGURE IV (a)

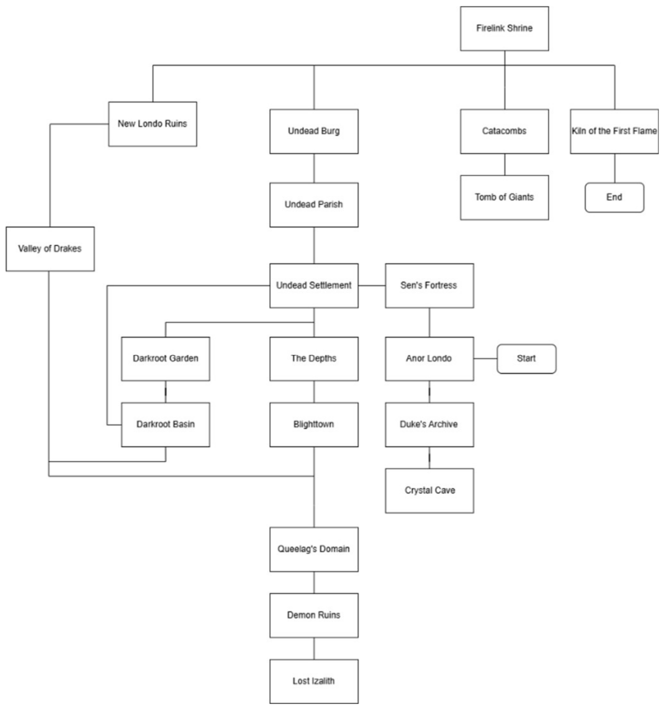


FIGURE IV (b)