

# *The Usage of Greedy Algorithm to Find the Best Route in the Tournament of Power Mode in Dragon Ball Legends*

Hafizh Hananta Akbari - 13522132  
Program Studi Teknik Informatika  
Sekolah Teknik Elektro dan Informatika  
Institut Teknologi Bandung, Jalan 10 Ganesha Bandung  
E-mail: 13522132@std.itb.ac.id

**Abstract**—Dragon Ball Legends is a game based on the Dragon Ball franchise that is widely popular in countries such as the United States and France. It has multiple game modes and depending on the game modes, it utilizes different game structures within the game modes. One of such game modes is the Tournament of Power mode. The Tournament of Power mode revolves around navigating different routes and take certain battles based on the routes taken and get certain amounts of points based on the battles taken. Due to how the points are calculated, it is mathematically possible to find calculate the best route that can be taken to maximise the amount of points that can be acquired within the Tournament of Power game mode. This paper researches the usage of the greedy algorithm to find the best route to take in the Tournament of Power mode.

**Keywords**—Dragon Ball Legends; Tournament of Power; The Best Route; Greedy Algorithm

## I. INTRODUCTION

Dragon Ball Legends is a free-to-play mobile game that is based on the Dragon Ball franchise. It was released in 2018 in multiple countries for both Android and IOS. Unlike Dragon Ball Z: Dokkan Battle which has different release dates as well as versions depending on the location, Dragon Ball Legends only has 1 version and is released in multiple countries during the same timeframe. By June 11, 2024, Dragon Ball Legends is still receiving new content with the game still in the middle of celebrating its 6<sup>th</sup> anniversary.

Ever since its release, Dragon Ball Legends have introduced multiple kinds of contents for players to enjoy, whether those contents revolve around the “Player vs Player” system (also known as PvP) or the “Player vs Enemy” system (also known as PvE). The contents of this game generally use the main gameplay mechanic of Dragon Ball Legends which is a 3D card-based fighting game. However, a certain game mode called the Tournament of Power takes a departure from the normal 3D card-based fight gameplay format. Instead, it takes a more board-game based approach with multiple different nodes and multiple different battles to choose based on the nodes taken. The difference in game mechanics allows for a different method of points calculation when compared to other game modes such as Rating Matches and Full Power Battles.

Due to the gameplay mechanic as well as the method used to calculate the points, this allows for the points calculation to be done by theory. This would mean that it is possible to mathematically deduce the total amount of points that can be gained in the Tournament of Power game mode as well as the best route to take to maximise the points gained. Determining the best route to take can be done with many methods, one of which being the greedy algorithm. This paper researches the usage of greedy algorithm on determining the best route to take in the Tournament of Power mode in Dragon Ball Legends.

## II. DRAGON BALL LEGENDS

### A. Dragon Ball Legends

Dragon Ball Legends is a free-to-play mobile game based on the Dragon Ball franchise that is developed by Dimps and published by Bandai Namco Entertainment. This game is compatible for both Android and IOS devices. This game was released in 2018 and has received continuous amounts of support since its release.

Within over six years, the game has introduced multiple different game modes. Such game modes include Story, Ultra Space-Time Rush, Rating Matches, Hyperdimensional Co-Op, and Tournament of Power. Most of these game modes have similar game mechanics with most of the game modes sharing the same skeleton.

The core gameplay mechanic of Dragon Ball Legends consists of an action based gameplay with a 3v3 (3 characters against 3 characters) system in a three-dimensional space with each battle lasting for 180 timer counts. The player brings a party consisting of six characters. Out of the six characters, the player uses up to three characters in a single battle against the opponent who is also using up to three characters. The player swipes and drags the screen around in order to move and dodge the opponent’s attacks and reposition. The gameplay itself is card-based with the main methods of attacks being the cards that are available in the four card slots on the screen. The image below showcases the main gameplay mechanic of Dragon Ball Legends.



Fig 2.1 The gameplay of Dragon Ball Legends

The gameplay loop revolves around bringing up to three characters and defeat the opponent by defeating all of their characters. This can be done by using arts cards to deal damage, tapping the screen with various methods to do a small attack, swiping the screen to dodge oncoming attacks which includes small attacks that can be done by tapping the screen as well as different arts cards, and tapping a different character's character icon to perform a switch at any time during the battle. This gameplay loop is affected by the characters' unique set of abilities which are called unique abilities. For most characters, another ability called main ability can be activated when a player taps the character icon when it is glowing white. Doing the same thing with some characters may have them do a tag-switch which have them switch characters within the same character unit. Some characters also have a unique gauge that is depicted as a circle with a symbol in the middle which will do special things depending on the character.

Generally, there are four different types of cards. The first and second ones being strike arts and blast arts cards, each of them dealing strike or blast damage depending on the arts card. The third and fourth types are special arts and special move arts cards with special arts cards giving a special effect depending on the character that is being used and special move arts card dealing more damage than both strike and blast arts cards. Some characters have other arts cards such as ultimate arts and awakened arts cards, both of which deals more damage than both strike and blast arts. The two special arts card types can only be acquired by certain characters mostly through the usage of the characters' main ability.

### B. Tournament of Power

Tournament of Power (shortened as T.O.P) is a PvE game mode within Dragon Ball Legends that follows a different kind of gameplay loop. It was first implemented on May 30, 2021 and has since been permanently implemented within Dragon Ball Legends with every season lasting for two weeks and the players divided into different leagues which are C, B, A, S, and Z.

depending on their results in the previous season. It is a game mode that requires the player to create a team of six characters alongside two replacement characters that can be tagged with the six characters that is brought to battle. The team-building aspect of the game mode is as follows.



Fig 2.2 The team-building aspect of Tournament of Power

Tournament of Power follows a board game approach, hence the importance of strategizing. This game mode has the player starting from their initial node and then traversing through 25 rows of nodes, each row consisting of 5 separate nodes with the exception for the first row. The player can only choose the node that is in front of the player's current node and the nodes on both diagonal sides. The following is the board-game aspect of the game mode.



Fig 2.3 The board-game aspect of Tournament of Power

As seen in the image above, different nodes come in different types of spaces as well as different codes for each space. The code for every space follows a numerical order that goes from left to right. Other than the codes, there are four different tiles or spaces that can be used within Tournament of Power. First is the recovery space, symbolized by the green colored heart image. This tile recovers the health of all team members except the ones with zero health. Second is the reform team space, symbolized by the yellow color and the three people shaped silhouettes. This tile averages the cumulative health every team member has, effectively reviving characters with zero health while also allow the player to manage their team setup. Third is the battle space, symbolized with the red colored fist image. This tile is the most common tile and acts as the tile that affects the total points calculation by the end of the season. Fourth is the boss space, symbolized by the rectangular shape and character icon. This tile is similar to the battle space but with harder opponents. The following is the details within a battle space.



Fig 2.4 The battle space in Tournament of Power

As seen in the image, there are three major things to be considered when entering a battle tile. First is the rival team which has characters with certain colors and effects. Second is the layout bonus that can be used by putting characters in the tile where the bonus applies. Third is the difficulty level of the battle tile, symbolized by the amount of fist images are brightened. The amount of fists within a battle tile will determine the difficulty of the battle tile and the point rewards for completing the battle tile with more fists equates to more points given. Other than the three major things to consider, there are also two pieces of information which are the space-code (B-45 being the code in figure 2.4) and the total amount of power level. To achieve great results within Tournament of Power, a player must be able to maximize the amount of points that can be acquired. With the assumption the player is in the Z league, these are the details for the points calculation in Tournament of Power.

$$VP = N - 1 . 20000 + 480000$$

This equation calculates the Victory Points. N is the number of the fight that ranges from 1 to 25.

$$BP = \text{More points for less damage a player takes}$$

This is the explanation for Battle Points. Battle Points are also affected by the difficulty level within a battle space. The exact equation to find the value of Battle Points is currently unknown.

$$FK = 50000$$

This is the value of Full K.O. Bonus. This bonus will only be applicable if all opponents are defeated.

$$BC = (VP + BP + FK) . \frac{\text{Total Boost}}{100}$$

This equation calculates the Boost Character Bonus with Total Boost being the total amount of boost a team of six characters have.

$$CW = (VP + BP + FK + BC) . \frac{\text{Consecutive Wins}}{100}$$

This equation calculates the Consecutive Wins Bonus with Consecutive Wins being the amount of wins a player has in a row.

$$\text{Total} = (VP + BP + FK + BC + CW)$$

This equation calculates the total amount of points by adding all of the other factors together.

### III. GREEDY ALGORITHM

Greedy algorithm is a type of algorithm that is widely popular in its implementation. It is a type of algorithm that evaluates optimization problems. The greedy algorithm is characterized by how it solves those problems, that of which is how the greedy algorithm searches for the locally optimum solution in all of its iterations. The idea behind the greedy algorithm is by choosing the best and most optimum option within an iteration (also known as the locally optimum solution), it will give the most optimum result in the entire problem (also known as the globally optimum solution).

The following are the different implementations of the greedy algorithm.

#### A. Activity Selection Problem

Activity Selection Problem is an optimization problem that covers an issue concerning multiple activities, each of them having their own start and end times. The goal of this problem is to create the maximum amount of activities that do not overlap with each other. This is done with the assumption that a person

can only do one activity at a time.

There are multiple methods that can be employed to solve this problem. A method that is widely used to solve this problem is using the greedy algorithm. The way the greedy algorithm works in this context is by first, sorting out the activities based on their end times. Then, the first activity with the earliest end time can be selected. Next, an iteration through all the other activities can be done if their start times are greater or equal to the end time of the previously selected activity. This process can be done until the iterations have gone through all the activities.

### *B. Integer Knapsack Problem*

The Integer Knapsack Problem is an optimization problem that covers an issue which concerns a number of items with their own weight and value as well as a knapsack which has a limit on the amount of weight it can hold. The goal of this problem is to maximize the amount of value that can be put within the knapsack. This refers to the total amount of value held by the amount of items that can be put within the knapsack without violating the weight limit that the knapsack has.

Typically, an Integer Knapsack Problem can be solved with three types of greedy methods which are Greedy by Profit, Greedy by Weight, and Greedy by Density. First is Greedy by Profit which is an implementation of the greedy algorithm by picking the items that have the biggest value. Next is Greedy by Weight which is an implementation of the greedy algorithm by picking the items based on the lowest amount of weight. Finally, there is the Greedy by Density which is an implementation of the greedy algorithm by picking the items based on the highest amount of density in which the amount of density can be calculated by dividing the value of an item with its weight. All three methods can be used in different cases although all three methods are not guaranteed to be able to reach the optimal solution.

### *C. Fractional Knapsack Problem*

The Fractional Knapsack Problem is an optimization problem that is very similar to the Integer Knapsack Problem. It features the same core principles as the Integer Knapsack Problem. It also covers the same issues as the Integer Knapsack Problem. The difference lies in how the Fractional Knapsack Problem solves the issues that are being tackled. This is shown by how the Fractional Knapsack Problem allows the division of items in its calculations. This means that the knapsack will always be full even if some items are divided to fit within the knapsack.

The Fractional Knapsack Problem can be solved in an efficient manner by using the greedy algorithm. The way to implement the greedy algorithm is by first calculating the density of the values that every item has by dividing each of the values with each of the weights. Next, all of the items can be sorted by density from highest to lowest. Next, selecting the items can be done by initializing the value and total weight of the knapsack and iterate through all the items. For every iteration, the density of every item is checked to determine if said item can fit within the knapsack. If the item is unable to fit within the knapsack, a fraction of said item is taken and put into the knapsack. This is done to maximize the amount of weight that can be put within the knapsack hence maximizing the amount of values within the knapsack.

### *D. Huffman Coding*

Huffman Coding is a method of data compression that is done to reduce a data's size without losing any piece of information. This method is used in multiple ways for different types of file compressions formats such as ZIP. It is also used for data transmission as well as to handle the compression of different types of media. This method works by two principles, the first of which being that the character that appears most often will be assigned to codes that are shorter by length. The second principle is how the character that appears less often will be assigned to codes that are longer by length. This allows for the minimization of the code size.

The way Huffman Coding works is by utilizing the greedy algorithm. This works by first reading through every character within a single data to calculate the amount of times each character appears in the data. Every character that builds the data is initialized as a tree and every node is assigned by the amount of times said character appears. Next, the greedy algorithm can be applied by combining two trees that have the least amount of frequency for their roots in every iteration. The root of the combined tree is the total amount of frequency from the two trees that make up the combined tree. The iterations stop once there is only one tree left. Then, a label of 0's and 1's can be added to the tree so that the full route from the root to the leaf would represent the binary string for each character. This will result in the binary string for each and every character.

### *E. Minimum Spanning Tree*

A minimum spanning tree is a subset of a fully connected graph that connects each and every one of the vertices. This subset is characterized by the lack of cycles and how all the edges that are used to connect the vertices totals in a minimum amount that is necessary to connect all the vertices. This means that the total amount of weight the edges have results in the minimum amount of weight overall. A minimum spanning tree can be used in designs, more specifically the designs that work like a network. This includes a network of cables, roads, and roads.

To create a minimum spanning tree, there are two widely known methods. These two methods are the Prim algorithm and Kruskal algorithm. The usage of the greedy algorithm lies within these two algorithms. With the Prim algorithm, the way it works is by first taking an edge from a graph that has the least amount of weight and putting it in a tree. Next, the greedy algorithm can be used by picking an edge that has the least amount of weight that is adjacent to the vertex that is included in the tree that does not form a cycle in the tree. The edge that has been picked can be added to the tree. This can be repeated until the tree is complete, hence making the entire minimum spanning tree.

Meanwhile, the Kruskal algorithm has a slightly different approach. The way the Kruskal algorithm works is by first sorting all the edges in a graph based on the weight from the smallest to biggest. Next, the greedy algorithm is implemented by iterating through all the edges and adding the smallest edge to the tree if it does not form a cycle. This is done until a minimum spanning tree has been made with all of the vertices being present within the tree.

## IV. CALCULATIONS

### A. Method of Research

To start with the calculations, it is important to understand the subject matter as well as the problem that has been presented. First of all, this is a research paper that covers the route traversal within the Tournament of Power mode in Dragon Ball Legends. The solution that is proposed is the usage of the greedy algorithm to determine the best route to take. The results of this research come in the form of an analysis that is done by taking the routes that exist within the 81<sup>st</sup> season of Tournament of Power. The amount of factors that can affect this research means that certain limitations should be implemented in order to keep this research within boundaries that are still within the topic of this research. The following are the limitations used in this research.

1. This research will not account for the other factors that may affect the points calculation within Tournament of Power. These factors include the amount of consecutive wins a player has, certain characters that provide a boost in points, as well as the points that are given based on the amount of damage a player takes. This is due to the fact that this research would not fulfill its original intentions if every factor that affects the points calculation were included within this research.
2. This research will not account for different characters that can be used within a team. This is due to the fact that this research only covers the routes within Tournament of Power that are optimal from a calculations standpoint. To include the usage of different characters within a team would be to stray away from the main topic of this research.
3. This research will use the assumption that every character will not die in battle spaces. This is due to how characters have limited amounts of health and the amount of damage taken can affect this research by taking it in a different direction.

Finding the best route to take in Tournament of Power can be done with the usage of the greedy algorithm. The greedy algorithm can be implemented with the constraints of the Tournament of Power mode to determine the route that can give a player the most amount of points.

### B. Algorithm Implementation

Tournament of Power is a game mode in Dragon Ball Legends with its own method of points calculation. One of the factors behind the calculations is the difficulty level within a battle space. Battle spaces may have different difficulty levels and are distributed randomly within the board. Other spaces, although rarer than battle spaces, are also distributed randomly. This allows for multiple different routes to choose from.

To find the best theoretical route without bringing in other factors such as a player's six character team, the characters that the rival use in every battle space, and any other factors, the usage of a greedy algorithm can be done. This is due to the fact that the principle behind the greedy algorithm is to choose the best option locally in hopes that it will lead to an optimal global

solution. The following is the board for the B league in the 81<sup>st</sup> season of Tournament of Power.

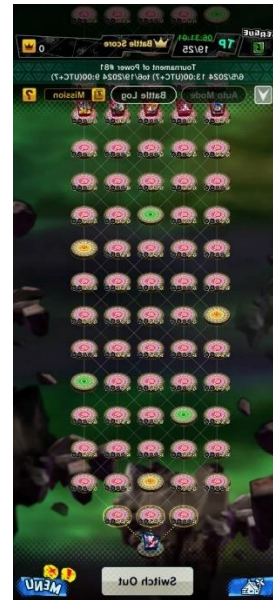


Fig 3.1 The First 13 Rows in Tournament of Power



Fig 3.2 The Last 12 Rows in Tournament of Power

By applying the greedy algorithm to the board within Tournament of Power, it is possible to find the best route. The following is the method to apply the greedy algorithm into the Tournament of Power game mode.

1. Start from the starting node.
2. Iterate for every row within the board.
3. For every iteration, choose the node with the most amount of fist symbols highlighted
4. Repeat until the greedy algorithm has reached the 25<sup>th</sup> iteration.

With the complete method of research specified, the following is the complete list of iterations with the greedy algorithm within Tournament of Power.

1. 1<sup>st</sup> iteration.  
In the first iteration, choose B-2 as it is the first battle space with the highest amount of difficulty level or fist symbols within the row. The route is currently as follows:
  - B-2
2. 2<sup>nd</sup> iteration.  
In the second iteration, choose B-5 as it is an accessible battle space with the highest amount of difficulty level or fist symbols within the row. The route is currently as follows:
  - B-2, B-5
3. 3<sup>rd</sup> iteration.  
In the third iteration, choose B-10 as it is an accessible battle space with the highest amount of difficulty level or fist symbols within the row. The route is currently as follows:
  - B-2, B-5, B-10
4. 4<sup>th</sup> iteration.  
In the fourth iteration, choose B-14 as it is an accessible battle space with the highest amount of difficulty level or fist symbols within the row. The route is currently as follows:
  - B-2, B-5, B-10, B-14
5. 5<sup>th</sup> iteration.  
In the fifth iteration, choose B-19 as it is an accessible battle space with the highest amount of difficulty level or fist symbols within the row. The route is currently as follows:
  - B-2, B-5, B-10, B-14, B-19
6. 6<sup>th</sup> iteration.  
In the sixth iteration, choose B-24 as it is an accessible battle space with highest amount of difficulty level or fist symbols within the row. The route is currently as follows:
  - B-2, B-5, B-10, B-14, B-19, B-24
7. 7<sup>th</sup> iteration.  
In the seventh iteration, choose B-30 as it is the only battle space accessible. The route is currently as follows:
  - B-2, B-5, B-10, B-14, B-19, B-24, B-30
8. 8<sup>th</sup> iteration.  
In the eight iteration, choose B-34 as it is an accessible battle space with the highest amount of difficulty level or fist symbols within the row. The route is currently as follows:
  - B-2, B-5, B-10, B-14, B-19, B-24, B-30, B-34
9. 9<sup>th</sup> iteration.  
In the ninth iteration, choose B-40 as it is an accessible battle space with the highest amount of difficulty level or fist symbols within the row. The route is currently as follows:
  - B-2, B-5, B-10, B-14, B-19, B-24, B-30, B-34, B-40
10. 10<sup>th</sup> iteration.  
In the tenth iteration, choose B-45 as it is an accessible battle space with the highest amount of difficulty level or fist symbols within the row. The route is currently as follows:
  - B-2, B-5, B-10, B-14, B-19, B-24, B-30, B-34, B-40, B-45
11. 11<sup>th</sup> iteration.  
In the eleventh iteration, choose B-50 as it is an accessible battle space with the highest amount of difficulty level or fist symbols within the row. The route is currently as follows:
  - B-2, B-5, B-10, B-14, B-19, B-24, B-30, B-34, B-40, B-45, B-50
12. 12<sup>th</sup> iteration.  
In the twelfth iteration, choose B-54 as it is an accessible battle space with the highest amount of difficulty level or fist symbols within the row. The route is currently as follows:
  - B-2, B-5, B-10, B-14, B-19, B-24, B-30, B-34, B-40, B-45, B-50, B-54
13. 13<sup>th</sup> iteration.  
In the thirteenth iteration, choose B-59 as it is an accessible battle space with the highest amount of difficulty level or fist symbols within the row. The route is currently as follows:
  - B-2, B-5, B-10, B-14, B-19, B-24, B-30, B-34, B-40, B-45, B-50, B-54, B-59
14. 14<sup>th</sup> iteration.  
In the fourteenth iteration, choose B-65 as it is an accessible battle space with the highest amount of difficulty level or fist symbols within the row. The route is currently as follows:
  - B-2, B-5, B-10, B-14, B-19, B-24, B-30, B-34, B-40, B-45, B-50, B-54, B-59, B-65
15. 15<sup>th</sup> iteration.  
In the fifteenth iteration, choose B-69 as it is an accessible battle space with the highest amount of difficulty level or fist symbols within the row. The route is currently as follows:
  - B-2, B-5, B-10, B-14, B-19, B-24, B-30, B-34, B-40, B-45, B-50, B-54, B-59, B-65, B-69
16. 16<sup>th</sup> iteration.  
In the sixteenth iteration, choose B-75 as it is an accessible battle space with the highest amount of difficulty level or fist symbols within the row. The route is currently as follows:
  - B-2, B-5, B-10, B-14, B-19, B-24, B-30, B-34, B-40, B-45, B-50, B-54, B-59, B-65, B-69, B-75
17. 17<sup>th</sup> iteration.  
In the seventeenth iteration, choose B-80 as it is an accessible battle space with the highest amount of difficulty level or fist symbols within the row. The route is currently as follows:
  - B-2, B-5, B-10, B-14, B-19, B-24, B-30, B-34, B-40, B-45, B-50, B-54, B-59, B-65, B-69, B-75, B-80

18. 18<sup>th</sup> iteration.

In the eighteenth iteration, choose B-84 as it is an accessible battle space with the highest amount of difficulty level or fist symbols within the row. The route is currently as follows:

- B-2, B-5, B-10, B-14, B-19, B-24, B-30, B-34, B-40, B-45, B-50, B-54, B-59, B-65, B-69, B-75, B-80, B-84

19. 19<sup>th</sup> iteration.

In the nineteenth iteration, choose B-89 as it is an accessible battle space with the highest amount of difficulty level or fist symbols within the row. The route is currently as follows:

- B-2, B-5, B-10, B-14, B-19, B-24, B-30, B-34, B-40, B-45, B-50, B-54, B-59, B-65, B-69, B-75, B-80, B-84, B-89

20. 20<sup>th</sup> iteration.

In the twentieth iteration, choose B-94 as it is the only accessible battle space. The route is currently as follows:

- B-2, B-5, B-10, B-14, B-19, B-24, B-30, B-34, B-40, B-45, B-50, B-54, B-59, B-65, B-69, B-75, B-80, B-84, B-89, B-94

21. 21<sup>st</sup> iteration.

In the twenty-first iteration, choose B-100 as it is an accessible battle space with the highest amount of difficulty level or fist symbols within the row. The route is currently as follows:

- B-2, B-5, B-10, B-14, B-19, B-24, B-30, B-34, B-40, B-45, B-50, B-54, B-59, B-65, B-69, B-75, B-80, B-84, B-89, B-94, B-100

22. 22<sup>nd</sup> iteration.

In the twenty-second iteration, choose B-105 as it is an accessible battle space with the highest amount of difficulty level or fist symbols within the row. The route is currently as follows:

- B-2, B-5, B-10, B-14, B-19, B-24, B-30, B-34, B-40, B-45, B-50, B-54, B-59, B-65, B-69, B-75, B-80, B-84, B-89, B-94, B-100, B-105

23. 23<sup>rd</sup> iteration.

In the twenty-third iteration, choose B-110 as it is an accessible battle space with the highest amount of difficulty level or fist symbols within the row. The route is currently as follows:

- B-2, B-5, B-10, B-14, B-19, B-24, B-30, B-34, B-40, B-45, B-50, B-54, B-59, B-65, B-69, B-75, B-80, B-84, B-89, B-94, B-100, B-105, B-110

24. 24<sup>th</sup> iteration.

In the twenty-fourth iteration, choose B-115 as it is an accessible battle space with the highest amount of difficulty level or fist symbols within the row. The route is currently as follows:

- B-2, B-5, B-10, B-14, B-19, B-24, B-30, B-34, B-40, B-45, B-50, B-54, B-59, B-65, B-69, B-75, B-80, B-84, B-89, B-94,

B-100, B-105, B-110, B-115

25. 25<sup>th</sup> iteration.

In the twenty-fifth iteration, choose B-119 as it is an accessible battle space with the highest amount of difficulty level or fist symbols within the row. The route is currently as follows:

- B-2, B-5, B-10, B-14, B-19, B-24, B-30, B-34, B-40, B-45, B-50, B-54, B-59, B-65, B-69, B-75, B-80, B-84, B-89, B-94, B-100, B-105, B-110, B-115, B-119

After all these iterations have been done, the following is the complete route that is returned by the greedy algorithm.

[B-2, B-5, B-10, B-14, B-19, B-24, B-30, B-34, B-40, B-45, B-50, B-54, B-59, B-65, B-69, B-75, B-80, B-84, B-89, B-94, B-100, B-105, B-110, B-115, B-119]

## V. RESULTS

The results gained from this research is that the greedy algorithm is capable of producing a solution although in this case, it is not an optimal solution. This is due to how the battle spaces have different difficulty levels that are scattered throughout the board on a randomized manner. This randomization allows for cases where it could be more beneficial to take a worse node to be able to take better consecutive nodes as opposed to taking the best node only to be left with worse consecutive nodes. This means only using the greedy algorithm would not result in an optimal global solution as taking the locally optimal solutions with each iteration may result in the algorithm missing out on potentially more amounts of points in total.

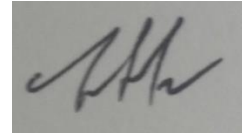
To rectify this, a different algorithm would have to be used in favor of the greedy algorithm, that being the backtracking algorithm. As the issue with using the greedy algorithm is how it may miss the higher difficulty level fights that are behind a lower difficulty level fight, using the backtracking algorithm would solve this issue as the backtracking algorithm would traverse through all the possible routes without exception before it returns the route that contains the most value in total, that being the route with the most amounts of higher difficulty battle spaces. The way this is done is as follows.

1. Start from the starting node.
2. For each node that is traversed, pick the next node from all the available choices. Make sure to not revisit nodes in the same order to avoid infinite cycles.
3. Repeat the second step until it has reached the final node.
4. Calculate the total amount of difficulty levels within the route that has been traversed and compare it to the other routes that have been traversed.
5. If the result is higher than the other routes that have been traversed, update the optimal path. If not, go back to the previous node and try a different node.
6. Repeat all the steps above until every route has been traversed and considered.

## VI. CONCLUSION

Bandung, 12 Juni 2024

In conclusion, although the greedy algorithm is usable, it might not provide the globally optimal solution at all times when used to determine the best route in the Tournament of Power game mode in Dragon Ball Legends. Rather than using the greedy algorithm, a better alternative would be to use the backtracking algorithm in order to find the route that will give the most amount of points based on the difficulty levels of the battle spaces alone. These results might have shown the applicability of the greedy algorithm and to an extent, the backtracking algorithm, these results cannot stand on its own. This is due to the game mechanics within Tournament of Power which is more complicated than the main issue that this research is tackling. However, these results can be used in tandem with other data such as the six characters team that is used, the league in which the player is in, as well as the rival character teams in different battle spaces in creating the best route to traverse through that is both safe and rewarding.



Hafizh Hananta Akbari - 13522132

## VII. ACKNOWLEDGMENT

First of all, the author would like to thank God for his blessings throughout the workings of this paper so that the author is capable of finishing this paper. The author would also like to thank Mr. Monterico Adrian, S.T., M.T. and Mr. Ir. Rila Mandala, M.Eng., Ph.D. as the lecturer of ITB Algorithm Strategies IF2211 for the materials that have been given throughout this entire semester. The author would also like to thank Mr Rinaldi Munir for the materials that have been shared on his website. Finally, the author would also like to thank his family and friends for their support throughout the process of writing this paper.

## REFERENCES

- [1] R. Munir, *Algoritma Greedy (Bagian 1)*, 2021. Retrieved 19:00, June 11, 2024, from [https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-\(2021\)-Bag1.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag1.pdf)
- [2] R. Munir, *Algoritma Greedy (Bagian 2)*, 2021. Retrieved 19:00, June 11, 2024, from [https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-\(2021\)-Bag2.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag2.pdf)
- [3] R. Munir, *Algoritma Greedy (Bagian 3)*, 2022. Retrieved 19:00, June 11, 2024, from [https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Greedy-\(2022\)-Bag3.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Greedy-(2022)-Bag3.pdf)
- [4] R. Munir, *Algoritma Runut-Balik (Backtracking) (Bagian 1)*, Bandung, 2021. Retrieved 21:00, June 12, 2024, from <https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-backtracking-2021-Bagian1.pdf>
- [5] Scipy.sparse.csrgraph.minimum\_spanning\_tree. *Docs.scipy.org*. Retrieved 22:09, June 12, 2024, from [https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.csrgraph.minimum\\_spanning\\_tree.html](https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.csrgraph.minimum_spanning_tree.html)

## 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.