# Weighted Graph Implementation for Unit Movement Optimization through Uncolonized Provinces of Colonial Colombia Region in Europa Universalis IV

Rayhan Hanif Maulana Pradana - 13521112[1]
*Program Studi Teknik Informatika*
*Sekolah Teknik Elektro dan Informatika*
*Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia*
*[1]13521112@std.stei.itb.ac.id*

***Abstract*—Europa Universalis IV is a grand strategy game developed by Paradox Development Studio where a player is able to control a nation from the Late Middle Ages through the early modern period (1444 - 1821), conducting trade, administration, diplomacy, colonization, and warfare. Unit or Army movement is a crucial part of the game, as armies can win battles, keep peace, crush rebellions, and explore uncharted lands. Unit movement in uncolonized provinces can be tricky however, as a unit that travels through them risks getting attacked by the natives, which slows down movement and risks losing men. By implementing weighted graph to this problem, we can find the most efficient route to traverse through uncolonized provinces by using Kruskal's Algorithm to find its minimum spanning tree. As a note, this paper limits the study to Colonial Colombia Region.**

***Keywords*—Weighted Graph, Minimum Spanning Tree, Kruskal's Algorithm, Unit Movement, Uncolonized Provinces, Colonial Colombia Region, Europa Universalis IV**

## I. Introduction

Europa Universalis IV is a grand strategy game in the Europa Universalis series developed by Paradox Development Studio and is released on Steam on August 13, 2013. It is a sequel to Europa Universalis III.

In Europa Universalis IV, a player can govern and control a nation which existed in a particular starting date of the game (e.g., playing as the Grand Duchy of Lithuania with the starting date at 11 November 1444). The player will conduct trade, administration, diplomacy, colonization if possible or necessary, and warfare with other nations. The way in which the player controls the nation can vary depending on the nation's government types, region, and whether it's a part of the Holy Roman Empire or the Shogunate or not. Victory condition is not defined in the game, rather the player itself defines the victory condition, whether it is world domination or military superiority, great economy, leading technological advances, roleplaying historical moments, restoring the Roman Empire, unifying into a bigger nation (e.g., Kingdom of Castille, Aragon, and Navarra can unite as Spain) or simply surviving. However, the game provides achievements which the player can get by playing the ironman mode, a game mode in which the player cannot go back in time. The game provides a route which the player can take as a particular nation as well, that is usually advantageous to the nation, but is not a must.


Figure 1. Europa Universalis IV Gameplay
(*Source : https://www.gameskinny.com/nn1ou/europa-universalis-iv-rights-of-man-patch-118-revolutionizes-eu4-gameplay*)

One of the most crucial parts of the game is unit movement. A unit can be divided into two, the Land Unit or troops which can be moved on land, and the Naval Unit or ships which can be moved on waterbodies. Units can be used for battles, keeping revolts and uprisings at bay and crush it if rebellion breaks out, and exploring uncharted lands and seas. Naval units in particular can be used for trade as well. Movement simply means moving units from a particular province to another province. Movement speed can be affected by terrains and monsoons. A unit can move or traverse through uncolonized provinces as well, which are provinces that are not currently owned by any nation. However, any unit that traverses through an uncolonized province may or may not receive an attack from the natives. This paper will provide the most efficient route for traversing uncolonized provinces in Colonial Colombia Region, the route which takes into account the terrains for movement speed and natives attack risks for safer travel by drawing a weighted graph and find its minimum spanning tree using Kruskal's Algorithm.

## II. THEORETICAL BASIS

### *A. Graph*

A graph is a discrete mathematical structure that is used to represent objects and the relations between those objects. A graph is structured by vertices and edges.

$$G = (V, E)$$

G = Graph
V = Non-empty set of vertices {v1, v2, v3, ..., vn}
E = Non-empty set of edges {e1, e2, e3, ..., en}

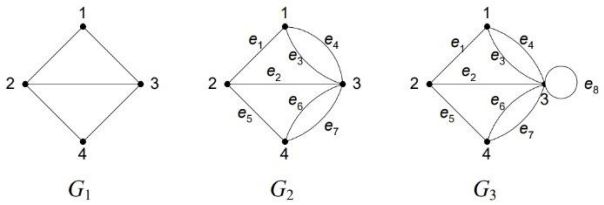


Figure 2. Graph Examples
(*Source :* *https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian1.pdf*)

Based on whether or not its edges have directional orientation, a graph can be divided into two:

1. Undirected Graph
   A graph in which its edges do not have directions.
2. Directed Graph
   A graph with arcs as it edges. Arcs are edges with a direction or arrow pointing to a location.

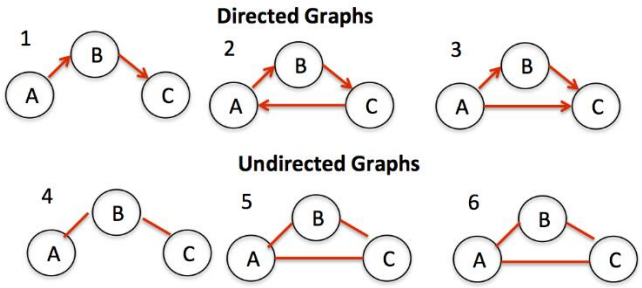


Figure 3. Directed Graph and Undirected Graph Examples
(*Source :* *https://sites.google.com/a/cs.christuniversity.in/discrete-mathematics-lectures/graphs/directed-and-undirected-graph*)

Several graph terminologies that are to be used in this paper include:

1. Cycle or Circuit
   A circuit is a series of adjacent vertices starting and ending at the same vertex.
2. Path
   A path is a series of edges from a vertex to another vertex.
3. Subgraph
   Suppose G = (V, E) is a graph. $G1 = (V1 , E1)$ is the subgraph of G if $V1 \subseteq V$ and $E1 \subseteq E$.

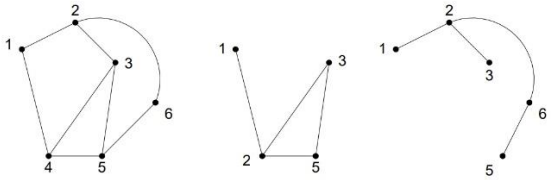


Figure 4. A graph (leftmost) and its subgraphs
(*Source :* *https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian1.pdf*)

4. Spanning Subgraph
   A spanning subgraph is a subgraph that has all the vertices of its original graph.

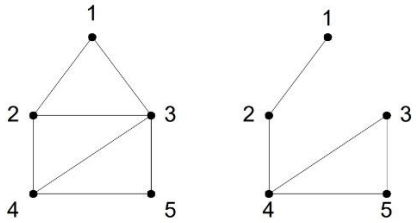


Figure 5. A graph (left) and its spanning subgraph
(*Source :* *https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian1.pdf*)

5. Connected Graph
   A connected graph is a graph with all its vertices can be reached from another vertices.
6. Weighted Graph
   A weighted graph is a graph in which its edges have values.

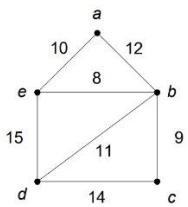


Figure 6. A weighted graph
(*Source :* *https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian1.pdf*)

### *B. Tree*

A tree is an undirected connected graph without a circuit or cycle.
Suppose G = (V, E) is an undirected graph and has n vertices. The following statements are equivalent:

1. G is a tree.
2. Every vertex in G is connected by a single path.
3. G is connected and has n – 1 edge.
4. G doesn't have a circuit.

5. Addition of edge by one will make the graph G has one circuit.

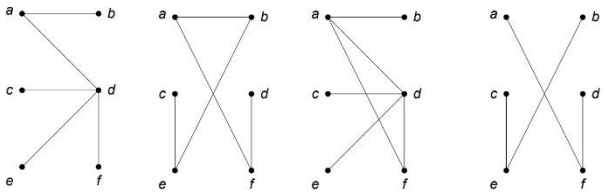


Figure 7. The first two leftmost are trees and the rest are not
(*Source : https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Pohon-2020-Bag1.pdf*)

*C. Spanning Tree and Minimum Spanning Tree*

The spanning tree of a connected graph is its spanning subgraph that is a tree.

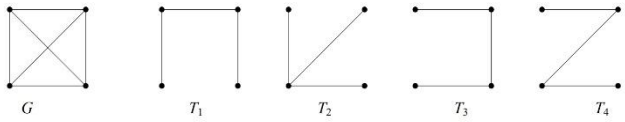


Figure 8. Graph G and its spanning trees
(*Source : https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Pohon-2020-Bag1.pdf*)

A minimum spanning tree is a spanning tree of a weighted graph in which its total edge value is the minimum.

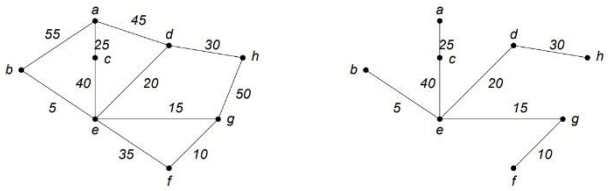


Figure 9. A weighted graph and its minimum spanning tree
(*Source : https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Pohon-2020-Bag1.pdf*)

*D. Kruskal's Algorithm*

A way to find the minimum spanning tree of a weighted graph is by using Kruskal's Algorithm:
Suppose T is the minimum spanning tree of graph G.

(0) Order G's vertices in an ascending order.
(1) T is empty
(2) Pick an edge (u, v) with the minimum value and doesn't form a circuit in T. Add the edge to T.
(3) Repeat step (2) n – 1 times.

*E. Provinces in Europa Universalis IV*

A province in Europa Universalis IV has attributes. Its attribute may affect a nation's economy, power, etc. and it may affect unit movement speed. Provinces in Europa Universalis IV is divided into owned/colonized provinces and uncolonized provinces:

1. Owned/Colonized Provinces
   Owned provinces affects its nation directly. Units traversing in their own provinces may travel safely.


Figure 10. A Castilian owned province, Segovia and its attributes can be viewed in bottom left corner
(*Source : Private document*)

2. Uncolonized Provinces
   A unit may traverse through an uncolonized province. It has only a handful of basic attributes, such as the terrains and additional native attributes.


Figure 11. Uncolonized province of Llanos
(*Source : Private document*)

3. Uncolonized Provinces Native Attributes
   Native attributes of an uncolonized province determines the number of natives that will attack your unit, the population attribute, the likelihood of the attack will happen, the aggressiveness attribute, and how hard that the natives will fight, that is, how much tactical advantage will the natives have against your unit, the ferocity attribute.


Figure 12. Native Population attribute
(*Source : Private document*)


Figure 13. Aggressiveness attribute
(*Source : Private document*)


Figure 14. Ferocity attribute
(*Source : Private document*)

*F. Unit Movement in Europa Universalis IV*

1. Unit Movement
   A unit may traverse through provinces. Their speed are directly affected by the province's terrain. The size of the unit may affect its movement speed as well.


Figure 12. Castilian Regiments moving from Segovia to Albacete
(*Source : Private document*)

2. Movement Cost
   The movement cost affects a unit movement negatively. The movement cost of a province depends on its terrain and monsoons if it is currently occurring. This paper assumes that monsoons don't occur in the provinces that will be studied. Movement cost cannot be found in the ingame UI, so we must look for it in the terrain file of the map folder (local file for Europa Universalis IV).

```
farmlands = {
        color = { 179 255 64 }

        type = plains
        sound_type = plains

        movement_cost = 1.10
        local_development_cost = -0.05
        supply_limit = 10
        allowed_num_of_buildings = 1
        nation_designer_cost_multiplier = 1.05
```

Figure 15. farmlands terrain attributes. The movement cost of farmlands can be seen.
(*Source : Private document*)

*G. Colonial Colombia Region in Europa Universalis IV*

Regions in Europa Universalis IV are the subset of super-regions. Colonial Regions are simply regions that has plentiful of uncolonized provinces and provide great benefits for the colonizers. Colonial Colombia Region is a subset of South America super-region.

Figure 16. South America super-region, political map mode
(*Source : Private document*)


Figure 17. South America super-region, colonial region map mode. Colonial Colombia is the red one
(*Source : Private document*)

Figure 18. South America super-region viewed as a 2d map. Colonial Colombia is the lime colored one (*Source : https://eu4.paradoxwikis.com/Central_and_South_America_regions*)

Colonial Colombia is arguably one of the most important colonial regions in the game. Aside from its colony benefits, Colonial Colombia Region is one of the colonial regions in South America nearest to Europe, particularly to Spain and Portugal, the biggest colonizers. It also connects to other valuable colonial regions, such as the Colonial Mexico Region, Colonial Caribbean Region, Colonial Brazil Region, Colonial Peru Region, and Colonial La Plata Region.

## III. IMPLEMENTATION

### *A. Undirected Graph of Colonial Colombia Region*

The graph that will be implemented represents all of the possible routes in traversing uncolonized provinces in Colonial Colombia Region. The vertices of the graph represents the uncolonized provinces and the edges of the graph represents possible movement of a unit trough adjacent provinces, that is if an uncolonized province u borders uncolonized province v, then it will be connected by the (u, v) edge.

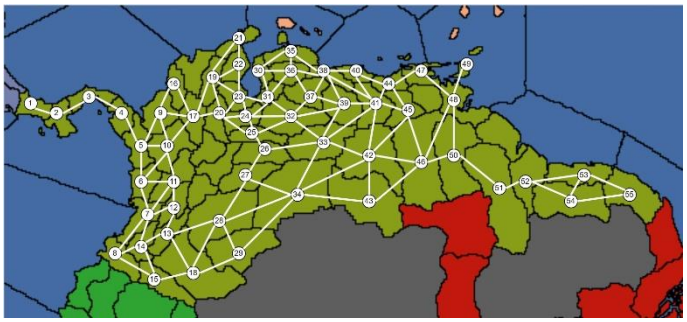

Figure 19. The graph representation of Colonial Colombia Region routes (*Source : Private document*)

### *B. Data of the Graph's Vertices*

The weight or value of the edges depends on the vertices or provinces attributes that are represented by integers. The total value of each vertex is the sum of the movement cost, the native population, the native's aggressiveness, and the native's ferocity. The movement cost, as can be seen in figure 15. is not an integer. For simplicity's sake and the nature of the Kruskal's Algorithm, which require integer values, this paper represents movement cost of 1.0 to 1.1 as one, 1.2 to 1.3 as two, and 1.4 to 1.5 as 3.

The native populations range from 500 to 2500. Again, for simplicity's sake and the fact that native populations do not make too big of a difference inside a range of every 500, this paper represents the native population by dividing the actual value of the native population by 500.

| Vertex | Province Name | Movement Cost | Native Population | Aggressiveness | Ferocity | Total Value |
|---|---|---|---|---|---|---|
| 1 | Guaymi | 3 | 1 | 2 | 1 | 7 |
| 2 | Veraguas | 3 | 1 | 2 | 1 | 7 |
| 3 | Panama | 3 | 4 | 5 | 2 | 14 |
| 4 | Darien | 3 | 4 | 5 | 2 | 11 |
| 5 | Kuna | 3 | 3 | 3 | 1 | 10 |
| 6 | Choco | 3 | 2 | 2 | 1 | 8 |
| 7 | Cauca | 3 | 3 | 3 | 1 | 10 |
| 8 | Pastos | 1 | 3 | 3 | 1 | 8 |
| 9 | Sinu | 3 | 3 | 3 | 1 | 10 |
| 10 | Antioquia | 3 | 2 | 2 | 1 | 8 |
| 11 | Mariquita | 3 | 2 | 1 | 1 | 7 |
| 12 | Pijaos | 3 | 2 | 2 | 1 | 8 |
| 13 | Neiva | 3 | 3 | 3 | 1 | 10 |
| 14 | Popayan | 3 | 3 | 3 | 1 | 10 |
| 15 | Putumayos | 3 | 3 | 3 | 1 | 10 |
| 16 | Cartagena | 3 | 3 | 2 | 1 | 9 |
| 17 | Magangue | 3 | 3 | 3 | 1 | 10 |
| 18 | Caqueta | 3 | 2 | 1 | 1 | 7 |
| 19 | Chimila | 3 | 3 | 3 | 1 | 10 |
| 20 | Motilones | 3 | 2 | 2 | 1 | 8 |
| 21 | Guajira | 1 | 3 | 6 | 1 | 11 |
| 22 | Maracaibo | 1 | 3 | 3 | 1 | 8 |
| 23 | Yukpa | 1 | 3 | 3 | 1 | 8 |
| 24 | Merida | 3 | 3 | 3 | 1 | 10 |
| 25 | Tuneo | 2 | 2 | 2 | 1 | 7 |
| 26 | Pore | 1 | 3 | 5 | 1 | 10 |
| 27 | Achaguas | 1 | 2 | 2 | 1 | 6 |
| 28 | Meta | 1 | 3 | 3 | 1 | 8 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 29 | Guaviire | 3 | 3 | 3 | 1 | 10 |
| 30 | Altagracia | 1 | 3 | 3 | 1 | 8 |
| 31 | Wayuu | 3 | 3 | 3 | 1 | 10 |
| 32 | Barinas | 2 | 3 | 5 | 1 | 11 |
| 33 | Apure | 1 | 3 | 3 | 1 | 8 |
| 34 | Puerto Carreno | 1 | 3 | 3 | 1 | 8 |
| 35 | Coro | 1 | 3 | 5 | 1 | 10 |
| 36 | Variquicimeto | 3 | 3 | 3 | 1 | 10 |
| 37 | Guanaguanare | 2 | 3 | 3 | 1 | 9 |
| 38 | Tacarigua | 3 | 3 | 3 | 1 | 10 |
| 39 | Llanos | 1 | 3 | 5 | 1 | 10 |
| 40 | Caracas | 3 | 5 | 5 | 1 | 14 |
| 41 | Zaraza | 1 | 3 | 6 | 2 | 12 |
| 42 | Atabapo | 3 | 3 | 3 | 1 | 10 |
| 43 | Guaju | 3 | 3 | 3 | 1 | 10 |
| 44 | Anzoategui | 1 | 3 | 3 | 1 | 8 |
| 45 | Guanipa | 1 | 3 | 3 | 1 | 8 |
| 46 | Angostura | 3 | 3 | 6 | 2 | 14 |
| 47 | Cumana | 1 | 3 | 2 | 1 | 7 |
| 48 | Orinoco Delta | 2 | 3 | 2 | 1 | 8 |
| 49 | Trinidad | 1 | 5 | 9 | 2 | 17 |
| 50 | Guasipati | 3 | 3 | 2 | 1 | 9 |
| 51 | Pirara | 3 | 3 | 6 | 2 | 14 |
| 52 | Demerara | 3 | 2 | 3 | 1 | 9 |
| 53 | Paramaribo | 3 | 2 | 3 | 1 | 9 |
| 54 | Surinam e | 3 | 2 | 3 | 1 | 9 |
| 55 | Cayenne | 3 | 3 | 3 | 2 | 11 |

Table 1. The data of each vertex in Colonial Colombia Region Graph
(*Source : Private Document*)

*C. Weighted Graph Data*

The weight of the edges is the sum of total value of each vertex adjacent to the edges.

| **Edges** | **Weight** |
|---|---|
| 1,2 | 14 |
| 2,3 | 21 |
| 3,4 | 28 |
| 4,5 | 24 |
| 5,6 | 18 |
| 5,9 | 20 |
| 5,10 | 18 |
| 6,7 | 18 |
| 6,10 | 16 |
| 6,11 | 15 |
| 7,8 | 18 |
| 7,11 | 17 |
| 7,12 | 18 |
| 7,14 | 20 |
| 8,14 | 18 |
| 8,15 | 18 |
| 9,10 | 18 |
| 9,16 | 19 |
| 9,17 | 20 |
| 10,11 | 15 |
| 10,17 | 18 |
| 11,12 | 15 |
| 12,13 | 18 |
| 13,14 | 20 |
| 13,18 | 17 |
| 13,28 | 18 |
| 14,15 | 20 |
| 15,18 | 17 |
| 16,17 | 19 |
| 17,19 | 20 |
| 17,20 | 18 |
| 18,28 | 15 |
| 18,29 | 17 |
| 19,20 | 18 |
| 19,21 | 21 |
| 19,22 | 18 |
| 19,23 | 18 |
| 20,23 | 16 |
| 20,24 | 18 |
| 20,25 | 15 |
| 21,22 | 19 |
| 22,23 | 16 |
| 23,24 | 18 |
| 23,31 | 18 |
| 24,25 | 17 |
| 24,31 | 20 |
| 24,32 | 21 |
| 25,32 | 18 |
| 25,26 | 17 |
| 26,27 | 16 |
| 26,33 | 18 |
| 27,28 | 14 |
| 27,34 | 14 |
| 28,29 | 18 |
| 28,34 | 16 |
| 29,34 | 18 |
| 30,31 | 18 |
| 30,35 | 18 |
| 30,36 | 18 |
| 31,32 | 21 |
| 31,36 | 20 |
| 32,33 | 19 |
| 32,37 | 20 |
| 32,39 | 21 |
| 33,34 | 16 |
| 33,39 | 18 |
| 33,41 | 20 |

| | |
|---|---|
| 33,42 | 18 |
| 34,42 | 18 |
| 34,43 | 18 |
| 35,36 | 20 |
| 35,38 | 20 |
| 36,37 | 19 |
| 36,38 | 20 |
| 37,38 | 19 |
| 37,39 | 19 |
| 38,39 | 20 |
| 38,40 | 24 |
| 38,41 | 22 |
| 39,41 | 22 |
| 40,41 | 26 |
| 40,44 | 22 |
| 41,42 | 22 |
| 41,44 | 20 |
| 41,45 | 20 |
| 42,43 | 20 |
| 42,45 | 18 |
| 42,46 | 24 |
| 43,46 | 24 |
| 44,45 | 16 |
| 44,47 | 15 |
| 45,46 | 22 |
| 46,48 | 22 |
| 46,50 | 23 |
| 47,48 | 15 |
| 48,49 | 25 |
| 48,50 | 17 |
| 50,51 | 23 |
| 51,52 | 23 |
| 52,53 | 18 |
| 52,54 | 18 |
| 53,54 | 18 |
| 53,55 | 20 |
| 54,55 | 20 |

Table 2. The edges' weight in Colonial Colombia Region Graph

*D. Kruskal's Algorithm Program*

The program used is not programmed by the writer of this paper. The implementation of Kruskal's Algorithm will be coded in C++.

```cpp
// Kruskal's algorithm in C++

#include <algorithm>
#include <iostream>
#include <vector>
using namespace std;

#define edge pair<int, int>

class Graph {
   private:
  vector<pair<int, edge> > G;  // graph
  vector<pair<int, edge> > T;  // mst
  int *parent;
  int V;  // number of vertices/nodes in graph
   public:
  Graph(int V);
  void AddWeightedEdge(int u, int v, int w);
  int find_set(int i);
  void union_set(int u, int v);
  void kruskal();
  void print();
};
```

Figure 20. Kruskal's Algorithm code in C++ (1)
(*Source : https://www.programiz.com/dsa/kruskal-algorithm*)

```cpp
Graph::Graph(int V) {
  parent = new int[V];

  //i 0 1 2 3 4 5
  //parent[i] 0 1 2 3 4 5
  for (int i = 0; i < V; i++)
    parent[i] = i;

  G.clear();
  T.clear();
}
void Graph::AddWeightedEdge(int u, int v, int w) {
  G.push_back(make_pair(w, edge(u, v)));
}
int Graph::find_set(int i) {
  // If i is the parent of itself
  if (i == parent[i])
    return i;
  else
    // Else if i is not the parent of itself
    // Then i is not the representative of his set,
    // so we recursively call Find on its parent
    return find_set(parent[i]);
}
```

Figure 21. Kruskal's Algorithm code in C++ (2)
(*Source : https://www.programiz.com/dsa/kruskal-algorithm*)

```
void Graph::union_set(int u, int v) {
  parent[u] = parent[v];
}
void Graph::kruskal() {
  int i, uRep, vRep;
  sort(G.begin(), G.end());  // increasing weight
  for (i = 0; i < G.size(); i++) {
    uRep = find_set(G[i].second.first);
    vRep = find_set(G[i].second.second);
    if (uRep != vRep) {
      T.push_back(G[i]);  // add to tree
      union_set(uRep, vRep);
    }
  }
}
void Graph::print() {
  cout << "Edge :"
     << " Weight" << endl;
  for (int i = 0; i < T.size(); i++) {
    cout << T[i].second.first << " - " << T[i].second.second << " : "
       << T[i].first;
    cout << endl;
  }
}
```

Figure 22. Kruskal's Algorithm code in C++ (3)
(*Source : https://www.programiz.com/dsa/kruskal-algorithm*)

*E. Minimum Spanning Tree of Colonial Colombia Region*

Below is the output of the program. The program receives input from weighted graph in table 2.

```
Edge : Weight
1 - 2 : 14
27 - 28 : 14
27 - 34 : 14
6 - 11 : 15
10 - 11 : 15
11 - 12 : 15
18 - 28 : 15
20 - 25 : 15
44 - 47 : 15
47 - 48 : 15
20 - 23 : 16
22 - 23 : 16
26 - 27 : 16
33 - 34 : 16
44 - 45 : 16
7 - 11 : 17
13 - 18 : 17
```

Figure 23. Minimum Spanning Tree (1)
(*Source : Private Document*)

```
15 - 18 : 17
18 - 29 : 17
24 - 25 : 17
25 - 26 : 17
48 - 50 : 17
5 - 6 : 18
7 - 8 : 18
8 - 14 : 18
8 - 15 : 18
9 - 10 : 18
10 - 17 : 18
19 - 20 : 18
23 - 31 : 18
25 - 32 : 18
30 - 31 : 18
30 - 35 : 18
30 - 36 : 18
33 - 39 : 18
33 - 42 : 18
```

Figure 24. Minimum Spanning Tree (2)
(*Source : Private Document*)

```
34 - 43 : 18
42 - 45 : 18
52 - 53 : 18
52 - 54 : 18
9 - 16 : 19
21 - 22 : 19
36 - 37 : 19
37 - 38 : 19
33 - 41 : 20
53 - 55 : 20
2 - 3 : 21
40 - 44 : 22
45 - 46 : 22
50 - 51 : 23
51 - 52 : 23
4 - 5 : 24
48 - 49 : 25
3 - 4 : 28
```

Figure 25. Minimum Spanning Tree (3)
(*Source : Private Document*)

From the output of the program, we can redraw the graph by of the Colonial Colombia Region to get its Minimum Spanning Tree as follows:

Figure 26. The Minimum Spanning Tree of Colonial Colombia Region
(*Source : Private Document*)

## IV. Conclusion

The implementation of weighted graph and minimum spanning tree with Kruskal's Algorithm in strategy games can be powerful, particularly in Europa Universalis IV. By knowing the minimum spanning tree of a region, we could move our units more efficiently, giving us the edge against our opponents either in wars or colonization race. Additionally, because Europa Universalis IV tries to simulate the map as accurate as possible, it shows that the use of weighted graph and minimum spanning tree with Kruskal's IV tries can be easily translated into real world problems that involve regional maps and can be as powerful.

## V. Acknowledgement

I would like to begin by acknowledging Allah SWT., who has provided me with guidance, the will to learn, knowledge, and determination to go through the discrete math class in this third semester and complete this paper.

I express my deep appreciation to Ms. Fariska Zakhralativa Ruzkanda, S.T. M.T. who provided her students with the necessary knowledge in discrete math, pique her students' interest in discrete math, and who has trained her students enough to increase our problem-solving skills, particularly in discrete math related problems which is essential in the completion of this paper.

And last but not least, I would like to express my gratitude to the mathematicians and computer scientists before us who has advanced the field of discrete math to where it is now. Discrete math has provided us with tools that makes numerous problems possible to solve, especially in the computational fields.

## References

[1] Kenneth H. Rosen, Discrete Mathematics and Application to Computer Science, 8th Edition, Mc Graw-Hill, Inc, 2018

[2] *(https://www.gameskinny.com/nn1ou/europa-universalis-iv-rights-of-man-patch-118-revolutionizes-eu4-gameplay)* Last accessed 12 December 2022, 7.13 GMT+7

[3] *(https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian1.pdf)* Last accessed 12 December 2022, 6.21 GMT+7

[4] *(https://sites.google.com/a/cs.christuniversity.in/discrete-mathematics-lectures/graphs/directed-and-undirected-graph)* Last accessed 12 December 2022, 5.31 GMT+7

[5] *(https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Pohon-2020-Bag1.pdf)* Last accessed 12 December 2022, 6.45 GMT+7

[6] *(https://eu4.paradoxwikis.com/Central_and_South_America_regions) )* Last accessed 12 December 2022, 8.03 GMT+7

[7] *(https://www.programiz.com/dsa/kruskal-algorithm) )* Last accessed 12 December 2022, 9.22 GMT+7

## 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, 3 Desember 2020

Rayhan Hanif Maulana Pradana dan 13521112