Aplikasi *Dijkstra Algorithm* untuk Perencanaan Rute Perdagangan Emas dan Strategi Perang *Game Age of Empires II: HD Edition*

Farel Winalda - 13522047[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13522047@mahasiswa.itb.ac.id

***Abstract*—Age of Empires II: HD Edition merupakan game sebuah game yang mengusung tema peperangan antar bangsa. Setiap bangsa dapat membangun bangunan, menciptakan penduduk dan prajurit dengan menukarkan 4 sumber daya utama, yaitu kayu, makanan, emas, dan batu. Setiap pemain dapat bersekutu dengan bangsa apapun. Seiring berkembangnya jaman, sumber daya yang ada tidak lagi banyak, sehingga satu-satunya sumber daya seperti kayu, makanan, dan batu akan menjadi langka, sehingga hanya emas yang tidak dapat habis. Emas dapat digunakan untuk berdagang dan menjadi mata uang antar bangsa baik sekutu maupun lawan, akan tetapi jalur masuk ke daerah perdagangan yang dilalui harus melalui wilayah sekutu. Perdagangan ini harus dilakukan untuk mempertahankan bangsa kita dari kehancuran. Makalah ini membahas perencanaan rute dagang dari suatu bangsa ke bangsa lain dengan memilih rute terpendek dan keuntungan tertinggi dengan menggunakan algoritma Dijkstra yang diimplementasikan pada program dalam bahasa C++.**

***Keywords*—Age of Empires II: HD Edition, Perdagangan, Algoritma Dijkstra, C++**

I. Pendahuluan

Age of Empires II: HD Edition merupakan game remaster dari Age of Empires II: The Age of Kings. Game ini dirilis pada tanggal 9 April 2013 di platform Steam untuk Pengguna Windows (Sumber: [3]). Game ini mengusung tema peperangan antar kebangsaan dan memiliki maksimal 8 pemain dalam satu kali periode permainan. Setiap bangsa dapat membangun bangunan, menciptakan penduduk dan prajurit dengan menukarkan 4 sumber daya utama, yaitu kayu, makanan, emas, dan batu. Pemain harus menaklukkan kekaisaran musuh yang dilalui oleh 4 masa, yaitu "Zaman Kegelapan" dari rentang tahun 500 to 1500 AD, "Zaman Feudal" dari rentang abad 5 sampai 12, "Zaman Kastil" dari abad 12 hingga 14, dan "Zaman Kekaisaran" yaitu saat Renaisans. Setiap pemain dapat bersekutu dengan bangsa apapun dan meminta bantuan berupa prajurit maupun sumber daya, namun bangsa lain belum tentu mau bersekutu. Setiap pemain saling serang menyerang dan mengatur strategi untuk merebut wilayah musuh dan mempertahankan wilayah sendiri yang diklaim oleh pihak musuh. Suatu bangsa akan dinyatakan hancur atau hilang dari peradaban ketika tidak ada lagi desa, penduduk dan bangunan penting (kastil, tempat ibadah), pasukan, serta sumber daya.

Pada game ini, terdapat beberapa peta yang diisi oleh 8 pemain. Perdagangan sendiri hanya dapat dilakukan melalui jalur darat. Sehingga, jika peta yang digenerate berbentuk kepulauan / archipelago, maka tidak dapat melakukan perdagangan antar negara. Tempat perdagangan setiap bangsa berbeda-beda dan memiliki nama / simbol *'market'* di dalam peta, namun setiap tempat perdagangan terdapat di wilayah masing-masing tiap daerah. Sehingga jika ingin berdagang dengan musuh, maka dapat dilakukan dengan melewati daerah sekutu yang tidak bermusuhan dengan wilayah musuh. Sebagai ilustrasi, A, B, dan C. A dan B bersekutu, B dan C bersekutu, namun A dan C saling bermusuhan. Sehingga jika A ingin berdagang dengan C harus melalui B sebagai bangsa netral.

Salah satu strategi untuk memenangkan permainan ini adalah dengan cara menghancurkan bangsa lain dengan menggunakan prajurit yang beragam sesuai dengan bangsa yang didapatkan. Berlaku juga untuk pertahanan untuk mempertahankan wilayah dapat dilakukan dengan membangun benteng, menara pemanah, dan kastil, serta dapat menggunakan prajurit untuk berada di garis pertahanan untuk melindungi bangsa. Untuk memperoleh sumber daya di awal zaman (Zaman Kegelapan dan Zaman Feudal) dapat dengan memanfaatkan penduduk / *'villager'* untuk menebang kayu, berternak hewan, bertani, dan juga menambang emas / batu. Namun, sumber daya yang ada dapat habis sehingga pemain harus secepat mungkin mencapai Zaman Kerajaan sehingga dapat melakukan perdagangan antar bangsa. Sehingga biaya untuk menciptakan prajurit tidak akan habis dengan cara perdagangan antar negara ini dan menjadi kunci untuk memenangkan suatu peperangan antar bangsa.

Gambar 1 Peta Wilayah Bangsa & Jalur Biru ke Ungu
(Sumber: Dokumen Pribadi)

Dalam peta tersebut, ditunjukkan 8 pemain yang berada dalam satu wilayah dan ditampilkan rute dari bangsa biru menuju bangsa ungu. Pemain biru, merah, kuning, dan hijau saling bersekutu, namun bermusuhan dengan ungu, biru muda, jingga, dan abu-abu. Sedangkan pemain hijau dan kuning bersekutu dengan pemain ungu, jingga, abu-abu, dan biru muda.Sehingga, jika pemain biru ingin berdagang dengan pemain ungu, maka harus melewati wilayah hijau atau kuning. Dapat dilihat dari **Gambar 1**, jalur berwarna abu-abu.

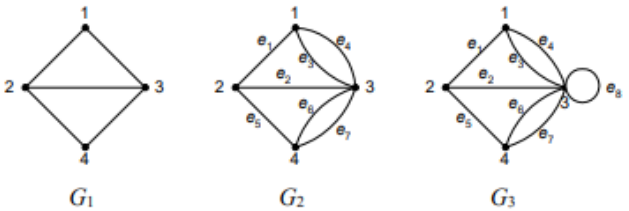
II. Teori Dasar

A. Graf

Graf adalah sebuah struktur data yang terdiri dari simpul (*node / vertices*) dan sisi (*edge*) yang menjadi penghubung antara simpul. Graf digunakan untuk merepresentasikan objek-objek diskrit dan hubungan antara objek-objek tersebut. Graf didefinisikan dalam bentuk *G = (V, E)* di mana G adalah graf, V (*vertices*) himpunan simpul-simpul tidak kosong dan E (*edge*) adalah himpunan sisi yang menghubungkan simpul dalam graf.

Graf dapat dibedakan menjadi dua berdasarkan keberadaan gelang atau sisi ganda, yaitu:

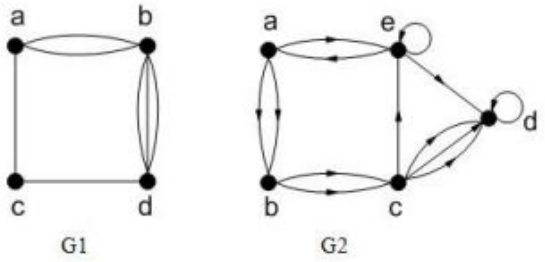
1. Graf Sederhana, yaitu graf yang tidak memiliki sisi ganda atau gelang.
2. Graf tak-sederhana, yaitu graf yang memiliki sisi ganda atau gelang. Dan dapat dibagi menjadi dua, yaitu : Graf ganda yaitu graf yang mengandung sisi ganda dan Graf Semu yaitu graf yang mengandung sisi gelang



Gambar 2.1 *G1* adalah Graf Sederhana, *G2* adalah Graf ganda, *G3* adalah Graf semu (Sumber: [1])

Lalu, berdasarkan orientasinya, graf juga dapat dibedakan menjadi dua, yaitu:

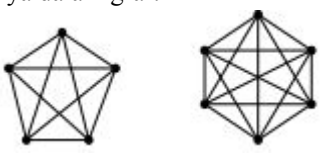
1. Graf tak-berarah, yaitu graf yang tidak mempunyai orientasi arah.
2. Graf berarah, yaitu graf yang setiap sisi yang menghubungkan simpul memiliki arah.



Gambar 2.2 G1 adalah Graf tak-berarah, G2 adalah Graf berarah (Sumber: [1])

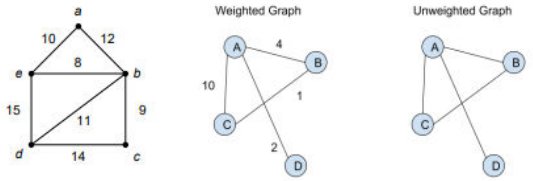
Beberapa terminologi dalam teori graf:

1. Ketetanggaan (*adjacency*)
   Dua buah simpul dalam graf dikatakan bertetangga jika kedua simpul terhubung oleh setidaknya satu sisi.
2. Bersisian (*incidency*)
   Sebuah sisi dalam graf dikatakan bersisian dengan simpul v_i dan v_j jika sisi saling menghubung v_i dan v_j .
3. Derajat (*degree*)
   Derajat suatu simpul adalah jumlah sisi yang bersisian dengan simpul tersebut.
4. Lintasan (*path*)
   Lintasan adalah barisan simpul-simpul dan sisi-sisi yang berbentuk e_0, v_0, e_1, v_1, e_2, v_2, …, e_n, v_n yang menghubungkan simpul awal v_0 ke simpul tujuan v_n, dengan panjang lintasan *n* adalah jumlah sisi dalam lintasan tersebut.
5. Siklus (*cycle*)
   Siklus adalah lintasan yang memiliki awal dan akhir pada simpul yang sama.
6. Keterhubungan (*connected*)
   Dua buah simpul v_i dan v_j dinyatakan terhubung bila terdapat lintas yang menghubungkan v_i dan v_j.
7. Graf Lengkap (*completed graph*)
   Graf lengkap merupakan graf yang memiliki sisi yang menghubungkan setiap simpul dengan semua simpul lainnya dalam graf.



Gambar 2.3 Graf Lengkap

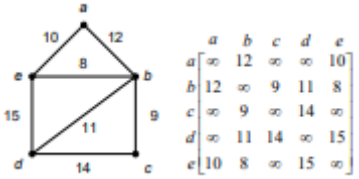
8. Graf berbobot (*weighted graph*)
   Graf berbobot adalah graf yang memiliki nilai atau bobot yang spesifik di setiap sisi yang menghubungkan simpul-simpul.



Gambar 2.3 Graf berbobot (kiri dan tengah) dan Graf Tidak berbobot (kanan) (Sumber: [1])

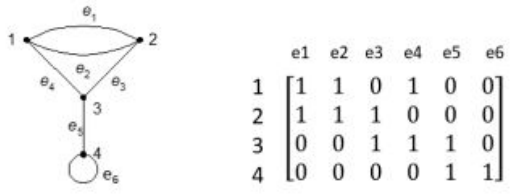
Dalam graf sendiri ada banyak cara untuk merepresentasikan dalam bentuk matriks tergantung dari pemilihan struktur data yang digunakan dalam pemrosesan graf. Beberapa representasi yang umum, yaitu:

1. Matriks Ketetanggaan (*Adjacent Matrix*)
   Dalam representasi ini, sebuah matriks memiliki ukuran sesuai dengan jumlah simpul dan digunakan untuk menyimpan informasi tentang sisi yang ada dalam graf. Setiap elemen matriks menunjukkan apakah ada keterhubungan antara dua simpul yang sesuai dengan indeks baris dan kolom dari elemen matriks tersebut.



Gambar 2.4 Representasi matriks ketetanggan untuk graf berbobot (Sumber: [2])

2. Matriks Bersisian (*Incidence Matrix*)
   Dalam representasi ini, sebuah matriks yang memiliki ukuran baris sesuai jumlah simpul dan ukuran kolom sesuai dengan jumlah sisi dalam graf. Setiap elemen matriks yang bernilai 1 jika simpul bersisian dengan simpul lainnya atau simpul diri sendiri dan 0 jika tidak saling bersisian.



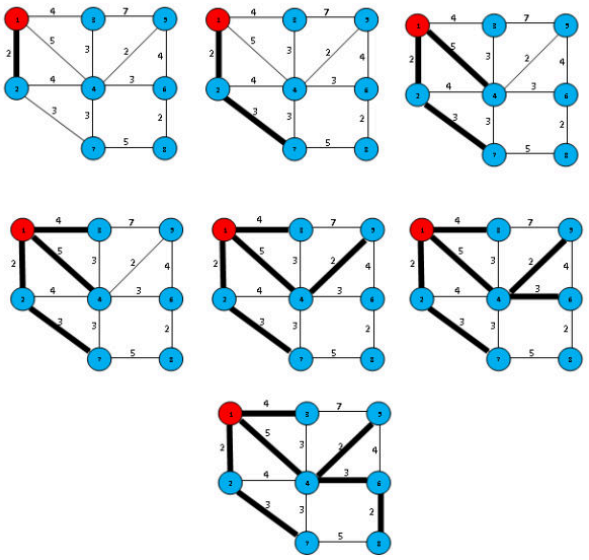
Gambar 2.5 Representasi matriks bersisian (Sumber: [2])

B. Algortima Dijkstra

Algoritma Dijkstra adalah algoritma yang digunakan untuk menemukan permasalahan menemukan jalur terpendek (*shortest path*) dari sebuah simpul asal menuju simpul lainnya dalam sebuah graf yang memiliki bobot. Algoritma ini ditemukan oleh ilmuwan Edsger W. Dijkstra pada tahun 1956 dan kemudian dirilis pada tahun 1959 di jurnal *Numerische Mathematik* yang berjudul "*A Note on Two Problems in Connexion with Graphs*" dan dianggap sebagai *Algoritma Greedy*.

Berikut langkah-langkah dalam menentukan jalur terpendek menggunakan Algoritma Dijkstra (Sumber: [4]):

1. Menentukan simpul awal dan memberikan bobot untuk semua sisi dari simpul-simpul yang terhubung.
2. Membuat sebuah set ***S*** yang berisi simpul-simpul, lalu set dengan nilai 0 untuk simpul awal dan nilai tak hingga untuk simpul lain yang belum terisi.
3. Cari semua simpul yang dapat dilalui dan dimasukkan ke dalam set ***S*** untuk menuju ke simpul tujuan.
4. Dari simpul awal keberangkatan, pertimbangkan simpul tetangga yang belum dilalui dan hitung jarak dari awal titik keberangkatan. Jika jarak lebih kecil dari jarak sebelumnya, simpan data lama dan diubah dengan data yang baru.
5. Setelah selesai mempertimbangkan setiap jarak dengan simpul tetangga, simpul yang telah dilalui harus ditandai. Simpul yang telah ditandai tidak akan di cek kembali, jarak yang disimpan adalah jarak terakhir yang memiliki bobot minimal.
6. Mengulangi langkah nomor 4 dan 5 hingga semua simpul telah dimasukkan ke dalam set ***S***.
7. Setelah semua simpul telah dimasukkan ke dalam set ***S***, didapatkan jalur terpendek dengan mengambil urutan simpul ke simpul lainnya yang memiliki jarak terpendek.



Gambar 2.6 Langkah-langkah menentukan jalan tercepat dari simpul 1 ke simpul lainnya (Sumber : [5])

III. Metodologi

A. Membuat Graf Berbobot dari Peta

Diinisiasi dengan pembuatan game dari sebuah data *Peta* kustomisasi yang telah dibuat sebelumnya, dipilih kebangsaan serta tim dan urutan pemain.

Gambar 3.1 Inisiasi Permainan dengan saya sebagai pemain 1 (biru) dan semua bangsa saling bersekutu
(Sumber: Dokumen Pribadi)

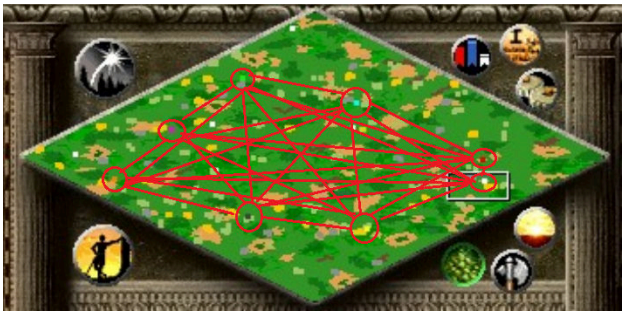
Dari sebuah *Peta* dibuat simpul-simpul yang akan mewakili simpul yaitu *'market'* untuk tujuan perdagangan. Pada *Peta* game ini sendiri terdapat tekstur layaknya peta umumnya seperti daerah pegunungan / dataran tinggi, hutan, perairan, dan juga daerah tandus.

Gambar 3.2 *'market'* pemain 1 (saya sendiri) dengan warna biru dan pedagang yang ditunjuk oleh panah hitam
(Sumber: Dokumen Pribadi)

Gambar 3.3 *'market'* pemain lainnya yang memiliki bentuk bangunan sesuai dengan kebangsaan pemain
(Sumber: Dokumen Pribadi)

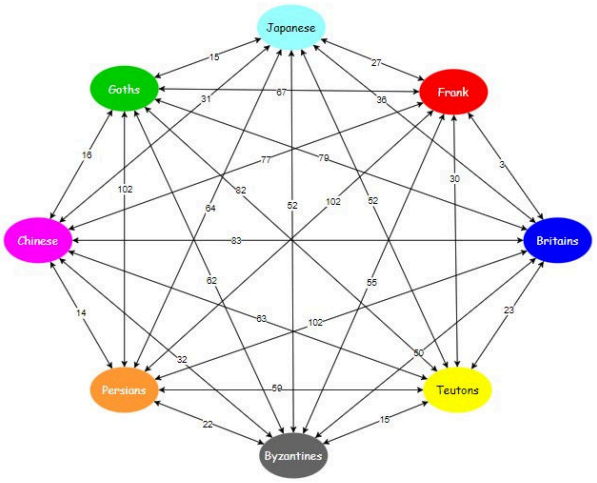
Gambar 3.4 Peta yang menunjukkan semua simpul
(Sumber: Dokumen Pribadi)

Dapat dilihat terdapat 8 daerah *'market'* yang telah ditandai dengan simpul merah, simpul 1 yang memiliki kotak adalah *'market'* pemain 1 (biru) berkebangsaan *Britains*, simpul 2 adalah *'market'* pemain 2 (merah) berkebangsaan *Frank*, simpul 3 adalah *'market'* pemain 3 (hijau) berkebangsaan *Goths*, simpul 4 adalah *'market'* pemain 4 (kuning) berkebangsaan *Teutons*, simpul 5 adalah *'market'* pemain 5 (biru muda) berkebangsaan *Japanese*, simpul 6 adalah *'market'* pemain 6 (ungu) berkebangsaan *Chinese*, simpul 7 adalah *'market'* pemain 7 (abu-abu) berkebangsaan *Byzantines*, simpul 8 adalah *'market'* pemain 8 (jingga) berkebangsaan *Persians*.



Gambar 3.5 Peta yang menunjukkan keterhubungan antar simpul (Sumber: Dokumen Pribadi)

Setelah membuat simpul, dibuat sisi yang menghubungkan antar *'market'*. Terdapat batasan dalam membuat sisi, di mana garis sisi tidak lurus yang disebabkan adanya tekstur wilayah penghalang seperti pegunungan, hutan, dan lain-lain. Sehingga dengan men-simplifikasi sisi menjadi garis lurus untuk menunjukkan keterhubungan antar simpul.



Gambar 3.6 Ilustrasi dan pemberian bobot pada Peta (Sumber: Dokumen Pribadi)

Memberikan bobot pada setiap sisi dapat dilihat pada **Gambar 3.6** simpul-simpul yang telah disimplifikasi menjadi sebuah diagram berarah dan memiliki bobot di setiap sisinya.

B. Implementasi Data ke dalam File Penyimpanan

Dari data-data yang didapatkan dari pembuatan permainan, dimasukkan ke dalam sebuah file penyimpanan, dalam kasus ini saya membuat data dalam format *Javascript Object Notation / JSON*.

```
{
    "Nations": [
        { "country": "Britains", "name":"Farel W" },
        { "country": "Frank", "name":"Philip II Augustus" },
        { "country": "Goths" , "name":"King Euric the Visigoth"},
        { "country": "Teutons" , "name":"Frederick Barbarossa"},
        { "country": "Japanese" , "name":"Imagawa Yoshimoto"},
        { "country": "Chinese" , "name":"Yue Fei"},
        { "country": "Byzantines" , "name":"General Manuel Comnenus"},
        { "country": "Persians" , "name":"Emp. Hormizd"}
    ],

    "distances": [
        [0, 3, 79, 23, 36, 83, 50, 102],
        [3, 0, 67, 30, 27, 77, 55, 102],
        [79, 67, 0, 82, 15, 16, 62, 49],
        [23, 30, 82, 0, 52, 63, 15, 59],
        [36, 27, 15, 52, 0, 31, 52, 64],
        [83, 77, 16, 63, 31, 0, 32, 14],
        [50, 55, 62, 15, 52, 32, 0, 22],
        [102, 102, 49, 59, 64, 14, 22, 0]
    ],

    "Ally":[
        [0, 1, 1, 1, 1, 1, 1, 1],
        [1, 0, 1, 1, 1, 1, 1, 1],
        [1, 1, 0, 1, 1, 1, 1, 1],
        [1, 1, 1, 0, 1, 1, 1, 1],
        [1, 1, 1, 1, 0, 1, 1, 1],
        [1, 1, 1, 1, 1, 0, 1, 1],
        [1, 1, 1, 1, 1, 1, 0, 1],
        [1, 1, 1, 1, 1, 1, 1, 0]
    ]
}
```

Gambar 3.6 Data yang telah dimasukkan ke dalam file penyimpanan (Sumber: Dokumen Pribadi)

Dari data-data tersebut, *"Nations"* menunjukkan bangsa-bangsa yang ada dalam permainan. Di dalam variabel *"Nations"* sendiri terdapat variabel *"country"* yang menunjukkan negara dan variabel *"name"* yang menunjukkan nama bangsa tersebut. Lalu, graf sendiri direpresentasikan dalam bentuk matriks ketetanggaan (*adjacency matrix*) dalam variabel *"distances"*. Di dalam variabel *"ally"* sendiri ditunjukkan relasi antar pemain, angka 1 berarti sekutu dan 0 berarti musuh yang juga ditampilkan dalam bentuk matriks ketetanggaan (*adjacency matrix*).

C. Implementasi Dijkstra Algorithm dalam Bahasa C++

Berdasarkan dasar teori pada bab sebelumnya, dibuat implementasi Algoritma Dijkstra untuk penyelesaian jarak *'market'* terdekat dengan menggunakan bahasa pemrograman C++. Implementasi kode ini menggunakan library <bits/stdc++.h> untuk sebagian besar implementasi dan <json/json.h> untuk pembacaan data dari file *.JSON* yang telah dimasukkan sebelumnya.

Pertama-tama diinisialisasi struktur data Graf dan data-data yang ada ke dalam sebuah struktur data *Graph*, yang terdiri dari *list* berukuran 8 bangsa *'Nation'*, matriks ketetanggaan berbobot (*adjacency matrix*) berukuran *8 x 8* yang menunjukkan jarak antar bangsa, matriks ketetanggan sekutu (*adjacency matrix*) berukuran *8 x 8* yang menunjukkan hubungan kedua bangsa yang bersesuaian dengan indeks baris dan kolom, dan index yaitu jumlah data masukan.

```
#include <json/json.h>
#include <bits/stdc++.h>

using namespace std;

#define MAX 8
#define MARK 9999

struct Nation {
    string country;
    string name;
};

struct Graph {
    struct Nation nations[MAX];
    int distances[MAX][MAX];
    int allies[MAX][MAX];
    int index;
};

void addNation(Graph* graph, string& country, string& name);

bool readNationDataFromJson(const string& filename, Graph& nationGraph);

int findNationIndex(Graph* graph, const string& name);

void dijkstra(Graph* graph, const string& start, const string& end);

void printGraph(Graph* graph);

#endif
```

Gambar 3.7 Implementasi struktur data dan primitif-primitif dalam file *header C++* (Sumber: Dokumen Pribadi)

Lalu, dari primitif-primitif tersebut dibuat program berdasarkan fungsi masing-masing. Terdapat beberapa primitif seperti *'void addNation'* untuk menambahkan bangsa ke dalam struktur data, *'bool readNationDataFromJson'* untuk membaca file data *JSON* dan memasukkan ke dalam struktur data, *'int findNationIndex'* untuk mencari indeks keberadaan suatu bangsa dalam matriks maupun *list*, *'void printGraph'* untuk menampilkan Graf.

Dalam implementasi permasalahan utama dari penelitian ini, yaitu pada fungsi *'void dijkstra'* adalah implementasi *Algoritma Dijkstra*. Dengan pertama-tama mencari indeks tujuan awal dan akhir bangsa dalam *list*. Lalu, inisialisasi variabel penyimpanan untuk menyimpan jarak terpendek, keterkunjungan, dan juga bangsa sebelum. Lalu, dengan menggunakan *loop* untuk mencari semua jarak yang berbeda dan membandingkan dengan jarak sebelumnya.

```cpp
void dijkstra(Graph* graph, const string& start, const string& end) {
    int startIdx = findNationIndex(graph, start); // Mencari index nation awal
    int endIdx = findNationIndex(graph, end); // Mencari index nation tujuan

    if (startIdx == -1 || endIdx == -1) { // Bangsa tidak ditemukan
        cout << "Invalid nation names for Dijkstra's algorithm." << endl;
        return;
    }

    int dist[MAX]; // Array untuk menyimpan jarak terpendek dari bangsa awal
    bool visited[MAX]; // Array penanda kunjungan ke setiap bangsa
    int previous[MAX]; // Array untuk menyimpan predecessor atau bangsa sebelumnya dalam jalur terpendek

    for (int i = 0; i < MAX; ++i) { // Inisialisasi array jarak, visited, dan previous
        dist[i] = MARK;
        visited[i] = false;
        previous[i] = -1;
    }
    dist[startIdx] = 0;

    for (int count = 0; count < MAX; ++count) { // Loop untuk menemukan jalur terpendek
        int u = findMinDistance(dist, visited); // Mencari bangsa dengan jarak terpendek yang belum dikunjungi
        visited[u] = true; // Menandai bangsa yang sudah dikunjungi

        for (int v = 0; v < MAX; ++v) { // Loop untuk mencari jarak lainnya terpendek ke bangsa lain
            if (!visited[v] && graph->distances[u][v] != 0 && dist[u] != MARK) {
                // Memeriksa sekutu atau bukan dan membandingkan dengan jarak sebelumnya
                if (graph->allies[u][v] == 1 && dist[u] + graph->distances[u][v] < dist[v]) {
                    dist[v] = dist[u] + graph->distances[u][v];
                    previous[v] = u;
                }
            }
        }
    }
```

Gambar 3.8 Implementasi *Algoritma Dijkstra* ke dalam Bahasa *C++* (Sumber: Dokumen Pribadi)

Lalu, langkah terakhir adalah dengan membuat program utama yang akan dijalankan dengan cara memanggil fungsi-fungsi yang telah dibuat sebelumnya. Dalam hal ini, direpresentasikan dalam *'int main()'*.

```cpp
#include "src/graph.hpp"

const int MAXLENGTH = 256;

int main() {
    Graph market; // Inisialisasi Graf
    market.index = 0; // Inisialisasi data masukan

    if (!readNationDataFromJson("test/data.json", market)) { // Proses pembacaan data JSON
        cerr << "Failed to read nation data from JSON." << endl;
        return 1;
    }

    printGraph(&market); // Menampilkan daftar bangsa

    string sourceName, destinationName;

    cout << endl << setw(25) << left << "Enter country" << " : "; // Input bangsa awal
    getline(cin, sourceName);

    cout<< setw(25) << left << "Enter destination country" << " : "; // Input bangsa tujuan
    getline(cin, destinationName);

    char sourceNameChar[MAXLENGTH];
    char destinationNameChar[MAXLENGTH];

    strncpy(sourceNameChar, sourceName.c_str(), MAXLENGTH); // Metode mengubah char menjadi string
    strncpy(destinationNameChar, destinationName.c_str(), MAXLENGTH); // Metode mengubah char menjadi string

    dijkstra(&market, sourceNameChar, destinationNameChar); // Memanggil fungsi dijkstra dari file graph.cpp

    return 0;
}
```

Gambar 3.9 Implementasi program utama ke dalam Bahasa *C++* (Sumber: Dokumen Pribadi)

Dalam program yang telah dibuat, pertama-tama akan ditampilkan daftar negara-negara serta nama-nama bangsa yang tersedia sesuai dengan data masukan yang telah dibuat. Lalu, pengguna memasukkan input berupa kata untuk menentukan bangsa awal. Lalu, pengguna juga memasukkan input serupa untuk menentukan bangsa tujuan untuk perdagangan. Lalu, data-data masukan dari pengguna akan diproses di dalam fungsi *'dijkstra'* dan akan dihasilkan keluaran berupa jalur terpendek dari bangsa awal ke negara tujuan hingga kembali lagi ke bangsa awal. Lalu, dihasilkan juga total jarak dan hasil emas yang didapatkan dari perdagangan dengan bangsa tujuan.

IV. Analisis dan Pembahasan

A. Uji coba kasus dan Hasil Eksekusi Program

Berdasarkan data pada **Gambar 3.6**, hasil eksekusi program penyelesaian dengan *Algoritma Dijkstra* dengan bangsa awal adalah bangsa saya sendiri yaitu *Britains* dan bangsa tujuan adalah bangsa *Chinese*, sebagai berikut:

```
farel9@LAPTOP-BMMBK3BU:/mnt/d/Coding/Matematika-Diskrit$ ./main

Nations List:
Country: Britains    Name: Farel W
Country: Frank       Name: Philip II Augustus
Country: Goths       Name: King Euric the Visigoth
Country: Teutons     Name: Frederick Barbarossa
Country: Japanese    Name: Imagawa Yoshimoto
Country: Chinese     Name: Yue Fei
Country: Byzantines  Name: General Manuel Comnenus
Country: Persians    Name: Emp. Hormizd

Enter country             : Britains
Enter destination country : Chinese

Shortest path from Farel W to Yue Fei :

Britains --> Frank --> Japanese --> Chinese --> Japanese --> Frank --> Britains

Total Distance  : 122
Total Reward    : 83
```

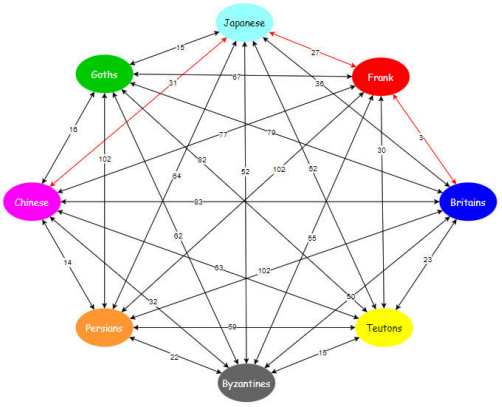
Gambar 4.1 Hasil jalur terpendek dari *Britains* menuju ke *Chinese* (Sumber: Dokumen Pribadi)

Terdapat beberapa cara untuk mencapai *Chinese* baik dengan melewati bangsa lain maupun langsung menuju bangsa tujuan, berikut beberapa jalan untuk menuju *Chinese*:

1. *Britains* → *Chinese*, total jarak 83.
2. *Britains* → *Frank* → *Chinese* , total jarak 80.
3. *Britains* → *Japanese* → *Chinese*, total jarak 67.
4. *Britains* → *Goths* → *Chinese*, total jarak 95.
5. *Britains* → *Teutons* → *Chinese*, total jarak 86.
6. *Britains* → *Byzantines* → *Chinese*, total jarak 82.
7. *Britains* → *Persians* → *Chinese*, total jarak 116.
8. *Britains* → *Frank* → *Japanese* → *Chinese*, total jarak 61.
9. *Britains* → *Frank* → *Japanese* → *Goths* → *Chinese*, total jarak 61.
10. *Britains* → *Frank* → *Goths* → *Chinese*, total jarak 86.
11. *Britains* → *Teutons* → *Byzantines* → *Chinese*, total jarak 116.
12. *Britains* → *Teutons* → *Byzantines* → *Persians* → *Chinese*, total jarak 74.

Dari 12 jalan yang telah dibuat dari total banyak cara, terdapat cara untuk menuju *Chinese* sebanyak C_0^6 / 1 cara tanpa melalui bangsa lain yaitu *Britains* → *Chinese*, C_1^6 / 6 cara dengan melewati tepat 1 bangsa, C_2^6 / 15 cara dengan melewati 2 bangsa, C_3^6 / 20 cara dengan melewati 3 bangsa, C_4^6 / 15 cara dengan melewati 4 bangsa, C_5^6 / 6 cara dengan melewati 5 bangsa, C_6^6 / 1 cara dengan melewati semua negara, dengan semua kombinasi merupakan jalan untuk meraih bangsa-bangsa yang bersangkutan. Sehingga didapatkan banyak kombinasi jalur sebanyak 64 cara. Lalu, didapatkan jarak terpendek untuk mencapai *Chinese* adalah dengan

melalui rute *Britains* → *Frank* → *Japanese* → *Chinese* atau *Britains* → *Frank* → *Japanese* → *Goths* → *Chinese* dengan jarak 61, sehingga jarak keseluruhan yang ditempuh sejumlah 122. Dalam keluaran program ditampilkan *Britains* → *Frank* → *Japanese* → *Chinese*, yang berarti kebenaran program terbukti benar.



Gambar 4.2 Hasil Jalur terpendek (warna merah)
(Sumber: Dokumen Pribadi)

B. Beberapa Studi Kasus

1. Studi Kasus Pertama

Kasus di mana pemain tidak memiliki sekutu dan semua pemain adalah musuh, yang berarti hasil keluaran tidak bisa dicapai sama sekali karena tidak memiliki satupun sekutu.

```
farel9@LAPTOP-BMMBK3BU:/mnt/d/Coding/Matematika-Diskrit$ ./main

Nations List:
Country: Britains    Name: Farel W
Country: Frank       Name: Philip II Augustus
Country: Goths       Name: King Euric the Visigoth
Country: Teutons     Name: Frederick Barbarossa
Country: Japanese    Name: Imagawa Yoshimoto
Country: Chinese     Name: Yue Fei
Country: Byzantines  Name: General Manuel Comnenus
Country: Persians    Name: Emp. Hormizd

Enter country             : Britains
Enter destination country : Persians

Shortest path from Farel W to Emp. Hormizd :

No path found.
```

Gambar 4.3 Hasil Eksekusi program studi kasus pertama
(Sumber: Dokumen Pribadi)

2. Studi Kasus Kedua

Kasus di mana pemain 1 hanya bermusuhan dengan *Persians* (pemain 8) dan *Teutons* (pemain 4) saja dan bersekutu dengan pemain lainnya. Sehingga tidak dapat secara langsung melewati *Teutons,* melainkan harus melalui *Frank* terlebih dahulu.

```
farel9@LAPTOP-BMMBK3BU:/mnt/d/Coding/Matematika-Diskrit$ ./main

Nations List:
Country: Britains    Name: Farel W
Country: Frank       Name: Philip II Augustus
Country: Goths       Name: King Euric the Visigoth
Country: Teutons     Name: Frederick Barbarossa
Country: Japanese    Name: Imagawa Yoshimoto
Country: Chinese     Name: Yue Fei
Country: Byzantines  Name: General Manuel Comnenus
Country: Persians    Name: Emp. Hormizd

Enter country             : Britains
Enter destination country : Persians

Shortest path from Farel W to Emp. Hormizd :

Britains --> Frank --> Teutons --> Byzantines --> Persians --> Byzantines --> Teutons --> Frank --> Britains

Total Distance  : 140
Total Reward    : 102
```

Gambar 4.3 Hasil Eksekusi program studi kasus kedua
(Sumber: Dokumen Pribadi)

3. Studi Kasus Ketiga

Kasus di mana pemain 1 hanya bersekutu dengan *Frank* (pemain 2) saja dan bermusuhan dengan pemain lainnya. Sedangkan *Frank* bersekutu dengan semua bangsa. Pemain ingin melakukan perdagangan dengan *Chinese* dan tidak akan ada perubahan karena dengan melalui *Frank* kita dapat melakukan perdagangan ke *Chinese* maupun bangsa lain.

```
farel9@LAPTOP-BMMBK3BU:/mnt/d/Coding/Matematika-Diskrit$ ./main

Nations List:
Country: Britains    Name: Farel W
Country: Frank       Name: Philip II Augustus
Country: Goths       Name: King Euric the Visigoth
Country: Teutons     Name: Frederick Barbarossa
Country: Japanese    Name: Imagawa Yoshimoto
Country: Chinese     Name: Yue Fei
Country: Byzantines  Name: General Manuel Comnenus
Country: Persians    Name: Emp. Hormizd

Enter country             : Britains
Enter destination country : Chinese

Shortest path from Farel W to Yue Fei :

Britains --> Frank --> Japanese --> Chinese --> Japanese --> Frank --> Britains

Total Distance  : 122
Total Reward    : 83
```

Gambar 4.3 Hasil Eksekusi program studi kasus ketiga
(Sumber: Dokumen Pribadi)

4. Studi Kasus Keempat

Kasus di mana pemain 1 hanya bersekutu dengan *Frank* (pemain 2) saja dan bermusuhan dengan pemain lainnya. Lalu, *Frank* bersekutu dengan bangsa kita saja dan bermusuhan dengan negara lainnya. Pemain ingin melakukan perdagangan dengan *Chinese* maka tidak ada jalan sama sekali untuk mencapai *Chinese* karena tidak bisa melalui negara sekutu yang bersekutu dengan *Chinese*.

```
farel9@LAPTOP-BMMBK3BU:/mnt/d/Coding/Matematika-Diskrit$ ./main

Nations List:
Country: Britains    Name: Farel W
Country: Frank       Name: Philip II Augustus
Country: Goths       Name: King Euric the Visigoth
Country: Teutons     Name: Frederick Barbarossa
Country: Japanese    Name: Imagawa Yoshimoto
Country: Chinese     Name: Yue Fei
Country: Byzantines  Name: General Manuel Comnenus
Country: Persians    Name: Emp. Hormizd

Enter country             : Britains
Enter destination country : Chinese

Shortest path from Farel W to Yue Fei :

No path found.
```

Gambar 4.3 Hasil Eksekusi program studi kasus keempat
(Sumber: Dokumen Pribadi)

V. Kesimpulan

Perencanaan rute untuk melakukan perdagangan antar bangsa pada permainan *Age of Empires II: HD Edition* dengan melalui jalur wilayah sekutu dapat dicari dengan menggunakan program yang mengimplementasikan *Algoritma Dijkstra.* Solusi ini menghasilkan hasil yang efisien sehingga perdagangan dapat dijalankan dengan lebih cepat dan menghemat waktu dan mendapatkan sumber daya 'emas' sebanyak-banyaknya dari perdagangan tersebut. Seperti yang telah dijelaskan pada Bab 4 hasil rute tercepat dari kasus khusus (semua bangsa bersekutu), maka jalan tercepat untuk mencapai *Chinese* dari *Britains* adalah dengan melalui *Britains* → *Frank* → *Japanese* → *Chinese* → *Japanese* → *Frank* → *Britains,* dengan total jarak total 122 dan emas sebanyak 83. Berlaku juga untuk beberapa studi kasus lainnya yang dapat dilihat pada Bab 4.

VI. Lampiran

Program lengkap dan dokumentasi dari penyelesaian permasalahan dari makalah ini dapat diakses pada link berikut: https://github.com/FarelW/IF2120_13522047_Farel-Winalda

VII. Ucapan Terima Kasih

Penulis menyampaikan ucapan terima kasih kepada:

1. Tuhan Yang Maha Esa atas berkat-Nya sehingga penulis dapat menyelesaikan makalah ini dengan baik,
2. Ibu Dr. Nur Ulfa Maulidevi S.T., M.Sc selaku pembimbing kelas IF2120 Matematika Diskrit Kelas 01,
3. Bapak Dr. Ir. Rinaldi Munir, M.T. dan Ibu Fariska Zakhralativa Ruskanda S.T.,M.T, selaku pembimbing kelas IF2120 Matematika Diskrit kelas paralel.

Penulis juga menyampaikan terima kasih banyak kepada semua orang yang belum disebutkan yang telah memberikan dukungan kepada saya. Tanpa bantuan dari Anda, makalah ini tidak dapat terselesaikan dengan baik.

Akhir kata, penulis mengharapkan makalah ini dapat membawa berkah dan manfaat kepada pihak yang membacanya.

Referensi

[1] Munir, Rinaldi (2023). *Graf : Bagian 1*. https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/19-Graf-Bagian1-2023.pdf (diakses tanggal 7 Desember 2023)

[2] Munir, Rinaldi (2023). *Graf : Bagian 2*. https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/19-Graf-Bagian1-2023.pdf (diakses tanggal 7 Desember 2023)

[3] Sauravt2812 (2022). *Age of Empires II: HD Edition.* https://ageofempires.fandom.com/wiki/Age_of_Empires_II:_HD_Edition (diakses tanggal 8 Desember 2023)

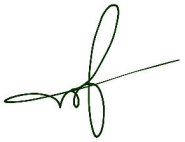
[4] Girsang, Abba S (2017). *Algoritma Dijkstra* https://mti.binus.ac.id/2017/11/28/algoritma-dijkstra/ (diakses tanggal 8 Desember 2023)

[5] Al-Ibadi, Mohammed (2012). *A New Hardware Architecture for Parallel Shortest Path Searching Processor Based-on FPGA Technology.* Hal 2574-2575.

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, 11 Desember 2023



Farel Winalda 13522047