p-ISSN: 2502-3691
e-ISSN: 2502-4361

Perbandingan Algoritma Dijkstra dan Algoritma Floyd-Warshall Penentuan Jalur Lintasan Terpendek Stasiun Tegal Menuju Hotel

Gunawan[1*)], Wresti Andriani[2]
[1,2]Program Studi Teknik Informatika STMIK YMI Tegal
[1,2]Jl. Pendidikan No. 1 Kota Tegal, Jawa Tengah, Indonesia
[1]gunawan.gayo@gmail.com, [2]wresty.andriani@gmail.com

Abstrak

Penentuan jalur lintasan terpendek merupakan masalah penting yang sering ditemukan dalam dunia komputer dan teknologi informasi. Algoritma Dijkstra merupakan salah satu algoritma graf yang digunakan untuk mencari jalur terpendek antara dua simpul dalam sebuah graf berbobot positif. Sedangkan algoritma Floyd-Warshall digunakan untuk mencari jalur terpendek antara semua pasang simpul dalam sebuah graf berbobot positif atau negatif. Tujuan penelitian ini adalah untuk mengetahui kelebihan dan kekurangan dari masing-masing algoritma dalam menyelesaikan masalah yang sama. Berdasarkan penelitian ini, maka algoritma Dijkstra memiliki kompleksitas pada source code yang lebih kecil dibandingkan dengan penerapan pada algoritma Floyd-Warshall, berarti bahwa algoritma Dijkstra memiliki efisiensi yang lebih tinggi dalam menentukan rute terpendek berdasarkan inputan titik awal dan titik tujuannya. Data dan graf yang dibuat juga dapat diimplementasikan dengan baik. Dengan demikian, algoritma Dijkstra dan algoritma Floyd-Warshall dapat digunakan untuk menyelesaikan persoalan rute terpendek. Namun, tidak menutup kemungkinan akan hasil yang berbeda pada kasus lainnya. Penerapan algoritma Dijkstra juga lebih mudah dipahami dan diterapkan khususnya dalam menentukan rute terpendek pada suatu objek penelitian. Rute terpendek serta informasi jarak yang dihasilkan pun juga diharapkan dapat membantu masyarakat terutama para turis yang ingin menginap di Kota Tegal dengan lebih efisien, selain itu guna untuk mengoptimalkan penggunaan kebutuhan seperti bahan bakar, estimasi waktu untuk menempuh perjalanan.

Kata kunci: **Algoritma Dijkstra, Algoritma Floyd Warshall, Jalur Lintasan Terpendek**

Abstract

Determining the shortest trajectory is a significant problem that is often found in the world of computers and information technology. The Dijkstra algorithm is one of the graph algorithms used to find the shortest path between two vertices in a positive-weighted graph. The Floyd-Warshall algorithm finds the quickest way between all pairs of vertices in a positively or negatively weighted graph. This study aims to discover the advantages and disadvantages of each algorithm in solving the same problem. Based on this research, the Dijkstra algorithm has a little complexity in source code compared to the application of the Floyd-Warshall algorithm, meaning that the Dijkstra algorithm has higher efficiency in determining the shortest route based on the input of the starting point and destination point. The data and graphs created can also be implemented well. Thus, the Dijkstra algorithm and the Floyd-Warshall algorithm can be used to solve the shortest route problem. However, it does not rule out the possibility of different results in other cases. Applying the Dijkstra algorithm is also easier to understand and use, especially in determining the shortest route to an object of study. The shortest path and the resulting distance information are also expected to help the community, especially tourists who want to stay in Tegal City more efficiently, in addition to optimizing the use of needs such as fuel and estimated time to travel.

Keywords: Dijkstra algorithm, Floyd Warshall algorithm, the shortest path

I. PENDAHULUAN

Penentuan jalur lintasan terpendek merupakan masalah penting yang sering ditemukan dalam dunia komputer dan teknologi informasi. Pada MANET, konstruksi perutean dan transmisi data didasarkan pada panjang jalur terpendek (Liu et al., 2019). Setiap panjang tepi sebagai variabel acak mengikuti distribusi probabilitas yang tidak diketahui dan bertujuan untuk menemukan jalur terpendek (Xia et al., 2019). jalur terpendek menetapkan dan memeriksa metode perkiraan untuk masalah interdiksi jaringan jalur terpendek yang dimodifikasi (Wei et al., 2018). Salah satu metode yang sering digunakan untuk menyelesaikan masalah ini adalah dengan menggunakan algoritma graf. Beberapa algoritma graf yang populer adalah algoritma Dijkstra dan algoritma Floyd-Warshall.

Algoritma Dijkstra merupakan salah satu algoritma graf yang digunakan untuk mencari jalur terpendek antara dua simpul dalam sebuah graf berbobot positif. Menggunakan metode Peta Multi-Skala dan algoritma Dijkstra, jalur optimal Peta Skala Kasar dihitung (Luo et al., 2019). Model jaringan saraf tiruan dan algoritma Dijkstra digunakan untuk prediksi kualitas udara dan pencarian jalur yang paling tidak tercemar di jaringan jalan (El Fazziki et al., 2017). Algoritma Dijkstra berdasarkan time window tertanam ke dalam algoritma genetika untuk mencari rute terpendek, mendeteksi tabrakan untuk beberapa kendaraan secara bersamaan (Lyu et al., 2019). Algoritma Dijkstra tradisional hanya menyimpan satu node perantara sehingga hanya satu rute terpendek yang dapat ditemukan melalui satu pencarian (Zhang et al., 2018). Algoritma Dijkstra dikembangkan guna merepresentasikan

p-ISSN: 2502-3691
e-ISSN: 2502-4361

grafik jaringan memilih rute yang tersedia untuk menghitung perkiraan probabilitas pertukaran data (Saajid et al., 2019). Algoritma Dijkstra sedang diimplementasikan untuk menentukan simpul cut-edge yang optimal (Cheng et al., 2019).

Sedangkan algoritma Floyd-Warshall digunakan untuk mencari jalur terpendek antara semua pasang simpul dalam sebuah graf berbobot positif atau negatif. algoritma Floyd-Warshall, adalah sebuah algoritma untuk mencari jalur terpendek memandang solusi yang diperoleh sebagai sebuah keputusan yang saling terkait (Ratnasari et al., 2013). Algoritma Floyd Warshall dan menerapkan Algoritma Floyd Warshall untuk menentukan lintasan terpendek dalam pengangkutan sampah dari 10 TPS ke TPA yang ada di Kabupaten Kubu Raya (Setiawan et al., 2017). Algoritma Floyd-Warshall sangat efisien dari sudut pandang penyimpanan data karena dapat diimplementasikan dengan hanya pengubahan sebuah matriks jarak (Darnita et al., 2017). Floyd Warshall merupakan salah satu algoritma pencarin yang dapat digunakan dalam menghitung jalur terpendek, dan mampu membandingkan semua kemungkinan lintasan pada graph untuk setiap sisi dari semua simpul yang ada (Ni Ketut, 2014).

Karena keduanya dapat digunakan untuk menyelesaikan masalah yang sama, yaitu mencari jalur terpendek dalam sebuah graf, maka perbandingan antara kedua algoritma ini sangat menarik untuk dilakukan. Dalam penelitian ini, akan dilakukan perbandingan antara algoritma Dijkstra dan algoritma Floyd-Warshall dalam menentukan jalur lintasan terpendek pada beberapa kasus uji tertentu. Tujuan penelitian ini adalah untuk mengetahui kelebihan dan kekurangan dari masing-masing algoritma dalam menyelesaikan masalah yang sama. Diharapkan hasil penelitian ini dapat memberikan kontribusi dalam pengembangan algoritma graf yang lebih efisien dan efektif dalam menyelesaikan masalah penentuan jalur lintasan terpendek. Pada penelitian ini, diperlukan beberapa aplikasi tambahan guna mendukung implementasi graf, diantaranya visio, Google Earth dan Google Maps guna membuat tampilan graf. Data yang diperoleh terkait 8 hotel yang akan digunakan dalam penelitian ini, yaitu Premiere Hotel, Karlita Hotel, KHAS Hotel, Riez Palace Hotel, Alexander Hotel, Prembiz Hotel, Bahari Inn Hotel, Plaza Hotel.

II. METODE PENELITIAN

A. Analisa Kebutuhan

Pada penelitian ini, diperlukan beberapa aplikasi tambahan guna mendukung implementasi graf, diantaranya Microsoft Visio, Google Earth dan Google Maps guna membuat tampilan graf. Kemudian, setelah perhitungan manual pada masing-masing algoritma, implementasi akan diterapkan dengan bantuan bahasa pemrograman C serta source code akan dieksekusi melalui Online C Compiler (https://www.onlinegdb.com/online_c_compiler), untuk source code yang digunakan selama penelitian ditampilkan pada pembahasan dalam jurnal ini.

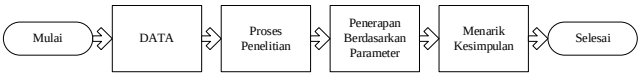
B. Skema Penelitian

Penelitian ini menggunakan metode komparatif, dimana metode ini digunakan untuk perbandingan terhadap satu variabel terhadap satu maupun lebih variabel pembanding. Teknik pengumpulan data dilakukan secara kualitatif, dalam arti data tidak hanya berupa angka atau numerik, melainkan dapat berupa kata-kata dan data pendukung lainnya. Tujuan dari penelitian ini adalah untuk menghasilkan luaran (output) berupa perbandingan dari kedua algoritma yang digunakan dalam penerapan graf dengan titik awal Stasiun Tegal menuju Hotel yang populer di Kota Tegal. Perbandingan ini didasarkan pada dua aspek yang menjadi parameter terhadap hasil penelitian terhadap ketiga algoritma, diantaranya:

1. Perhitungan manual dan program.
2. Kompleksitas.

Berdasarkan tujuan tersebut, penelitian dilakukan dengan beberapa tahapan, diantaranya :

1. Menentukan objek penelitian serta mengumpulkan data dan informasi sebanyak mungkin.
2. Melakukan penelitian sesuai perhitungan yang telah ditentukan.
3. Melakukan penelitian dengan implementasi serta perbandingan berdasarkan parameter yang telah ditentukan.
4. Menarik kesimpulan dari hasil implementasi pada objek penelitian.



Gambar 1. Skema Alur Penelitian

Data yang diperoleh terkait 8 hotel yang akan digunakan dalam penelitian ini, yaitu Premiere Hotel, Karlita Hotel, KHAS Hotel, Riez Palace Hotel, Alexander Hotel, Prembiz Hotel, Bahari Inn Hotel, Plaza Hotel.

C. Algoritma Dijkstra

Algoritma Dijkstra ditemukan oleh Edsger W. Dijkstra yang merupakan salah satu varian bentuk algoritma populer dalam pemecahan persoalan yang terkait dengan masalah optimasi dan bersifat sederhana. Algoritma ini menyelesaikan masalah mencari sebuah lintasan terpendek (sebuah lintasan yang mempunyai panjang minimum) dari vertex a ke vertex z dalam graph berbobot, bobot tersebut adalah bilangan positif jadi tidak dapat dilalui oleh node negatif, namun jika terjadi demikian, maka penyelesaian yang diberikan adalah infinity [5].

1) Pengenalan Algoritma Dijkstra

Dijkstra merupakan algoritma menemukan jalur dengan biaya terendah (yaitu rute terpendek) antara simpul tersebut dengan setiap simpul lainnya. Algoritma ini juga dapat digunakan untuk menemukan jalur terpendek dari simpul asal ke simpul tujuan dengan cara menghentikan algoritma ketika jalur terpendek simpul tujuan telah ditentukan. Salah satu komponen dari algoritma Dijkstra adalah graf dan matriks ketetanggaan. Graf merupakan pasangan himpunan G = (V,E). Secara geometri graf digambarkan sebagai sekumpulan noktah (simpul) didalam bidang dwimatra yang dihubungkan dengan sekumpulan garis (sisi). Dalam matriks ketetanggaan terdapat komponen utama sebagai penyusunnya yaitu lintasan. Dua buah simpul dalam sebuah graf

p-ISSN: 2502-3691
e-ISSN: 2502-4361

dinyatakan bertetangga apabila keduanya terhubung langsung dalam sebuah sisi[6].

2) Cara Kerja Algoritma Dijkstra

Cara kerja algoritma Dijkstra memakai strategi greedy. Dimana strategi greedy pada algoritma Dijkstra menyatakan bahwa setiap langkah, ambil sisi yang berbobot minimum yang menghubungkan sebuah simpul yang sudah terpilih dengan sebuah simpul lain yang belum terpilih. Lintasan dari simpul asal ke simpul yang baru haruslah merupakan lintasan terpendek diantara semua lintasannya ke simpul-simpul yang belum dipilih. Algoritma ini mencari panjang lintasan terpendek dari vertex a ke vertex z dalam sebuah graf (graph) berbobot positif dan tersambung [7].

D. Algoritma Floyd Warshall

Algoritma Flyod Warshall adalah algoritma untuk menemukan jalur terpendek dalam grafik tertimbang dengan bobot tepi positif atau negatif (tetapi tidak ada siklus negatif). Algoritma ini membandingkan semua jalur yang mungkin melalui grafik antara setiap pasangan simpul. Akurasi algoritma ini selalu menunjukkan nilai 100%. Mekanisme algoritma Flyod Warshall dilakukan dalam beberapa langkah. Langkah-langkah algoritma Flyod Warshall adalah:

1. Mewakili grafik berbobot sebagai matriks. Berat untuk masing-masing adalah:
   $w\,i\,j \;= 0, i\,f\;\; i = j$
   $= w\,(i\,,j\,), i\,f\;\; i \neq j\;\; a\,n\,d\;\; (i\,,j\,) \in E$
   $= \infty\; i\,f\;\; i \neq j\;\; a\,n\,d\;\; (i\,,j\,) \notin E$
   Format output adalah matriks $n \times n$ dengan jarak/waktu tempuh $D = [d\,i\;]$, $d\,i\,j$ adalah a jarak/waktu tempuh dari vertex i ke vertex j.
2. Dikomposisi Flyod Warshall.
   a. dij$^{(k)}$ merupakan panjang dari shortest path dari i ke j, sehinggasemua vertex intermediate yang terdapat pada path (jika ada) terkumpul pada {1,2,....,k}
   b. dij$^{(0)}$ dikumpulkan pada wij, yaitu tidak ada vertex intermediate.
   c. D$^{(k)}$ menjadi matriks n x n [dij(k)] Tentukan dij(n) sebagai jarak dari i ke j kemudian hitung D(n)
   d. Hitung D(k) untuk k = 0,1,...., nPada langkah ini, lakukan pengamatan.
3. Langkah ketiga adalah menentukan struktur shortest path. Dalam hal ini, harus dilakukan dua pengamatan terlebih dahulu sebelum melangkah lebih jauh, yaitu:
   a. Sebuah *shortest path* tidak memuat *vertex* yang sama sebanyak dua kali
   b. Untuk sebuah shortest path dari i ke j dengan beberapa vertex intermediate pada path dipilih dari kumpulan {1, 2, ..., k}, dengan kemungkinan k bukan merupakan *vertex* pada *path vertex* pada path (path terpendek memiliki panjang dij(k-1) + dij(k-1)).
4. Setelah melakukan pengamatan diatas, kemudian dlakukan penentuan shortest path dari I k j yang memuat vertex k:
   a. Shorter path tersebut Memuat sebuah path dari k ke j
   b. 2. Setiap subpath hanya dapat memuat pertex intermediate pada (1,.., k-1.) dan sedapat mungkin memiliki nilai terpendek kemudian beri nama panjangnya dik(k-1 dan dkj(k-1) sehingga path memiliki panjang (dik(k-1) + dkj(k-1).
5. Langkah-langkah adalah melakukan iterasi yang dimulai dari iterasi ke 0 sampai dengan. Perhitungan yang dilakukan adalah:
   a. Menentukan D(0) (iterasi ke 0) = [wij] merupakan matriks bobot.
   b. Menentukan D(k) dengan menggunakan rumus, dij(k) = min {dij(k-1), dik(k-1) + dkj(k-1)}, untuk k = 1, ..., n dimana n adalah jumlah vertex.

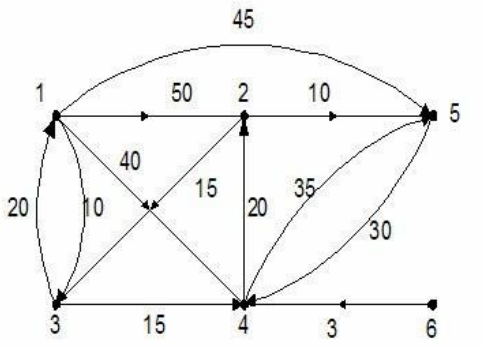
   Hasil akhir dari algoritma *Floyd-Warshall* adalah matriks untuk iterasi ke-n. Dari matriks ke-n ini, dapat dilihat *shortest path* untuk setiap *vertex* padasuatu *graph*.

E. Jalur Lintasan Terdekat

Kata Terpendek pada persoalan sebuah lintasan memiliki pengertian yaitu proses inimalisasi bobot pada sebuah lintasan graph. Beberapa jenis persoalan lintasan terpendek antara lain:

1) Lintasan terpendek antara dua buah simpul.
2) Lintasan terpendek antara semua pasangan simpul.
3) Lintasan terpendek dari simpul tertentu ke semua simpul yang lain.
4) Lintasan terpendek antara dua buah simpul

yang melalui beberapa simpul tertentu [1]. Dalam pemecahan persoalan tentang graf
berbobot
G = (V,E)
dan sebuah simpul a. Penentuan lintas terpendek dari a ke setiap simpul lainnya di G. Asumsi yang kita buat adalah bahwa semua sisi berbobot positif seperti yang ditunjukkan pada Gambar 2:



Gambar 2 . Graph Contoh Persoalan Lintasan Terpendek

F. Evaluasi

Untuk parameter pertama yaitu terkait perbandingan terhadap perhitungan secara manual dan secara program. Pada parameter ini, dilakukan perbandingan terhadap perhitungan manakah yang dinilai paling efektif untuk mencari rute terpendek berdasarkan titik awal dan tujuan tertentu. Untuk perhitungan secara manual dilakukan oleh penulis dengan menganalisa data, graf, dan proses perhitungan total secara manual. Kemudian, untuk perhitungan secara program akan dilakukan dengan bantuan bahasa pemrograman C untuk melihat apakah hasil yang diperoleh sama besar dan perhitungan mana

p-ISSN: 2502-3691
e-ISSN: 2502-4361

yang paling efektif dan efisien untuk digunakan dalam jangka waktu yang lebih cepat dan praktis.

Untuk parameter kedua, yakni mengenai kompleksitas. Pada perbandingan ini, ketiga algoritma yang digunakan akan dilihat seberapa efisien penggunaan algoritma tersebut untuk menentukan rute terpendek pada graf yang tersedia. Kompleksitas disini berbicara mengenai rumit atau tidaknya algoritma yang digunakan, sehingga indikasi yang dapat dijadikan yaitu seberapa efisien rute yang terbentuk oleh suatu algoritma yang dapat ditempuh oleh wisatawan, sehingga para wisatawan dapat melakukan perjalanan menuju destinasi tertentu dengan cepat dan dengan rute yang paling tepat.

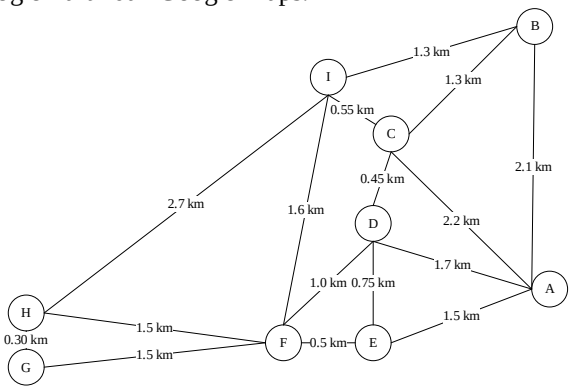
III. HASIL DAN PEMBAHASAN

Adapun data yang akan digunakan untuk perhitungan (sumber googlemap.com):

Tabel 1. Daftar Jarak Stasiun Tegal ke Hotel

Kode	Nama Tempat	Jarak
1.	Stasiun Tegal	0.0 km
2.	Premiere Hotel	2.2 km
3.	Khas Hotel	2.2 km
4.	Riez Palace Hotel	1.7 km
5.	Alexander Hotel	1.5 km
6.	Primebiz Tegal Hotel	2.0 km
7.	Bahari Inn	3.4 km
8.	Plaza Hotel	3.7 km
9.	Karlita Hotel	2.6 km

Untuk graf yang digunakan dalam penelitian ini adalah graf berbobot dimana simpul dalam graf ini merepresentasikan nama tempat dari titik awal (stasiun Tegal dan Hotel-hotel, dan sisi/edges sebagai rute dengan nilai bobot (jarak) dalam satuan kilometer berdasarkan data pada Tabel 1. dan software yang digunakan, yaitu Google Earth dan Google Maps.



Gambar 3. Tampilan Graf

A. Penerapan Algoritma *Dijkstra*

Perhitungan berdasarkan data dan graf yang telah dibuat dilakukan dengan 2 perhitungan, yaitu perhitungan secara manual dan perhitungan melalui bantuan bahasa pemrograman C. Berikut merupakan penerapan algoritma Dijkstra yang dilakukan secara manual. Langkah pertama diawali dengan membuat Matriks Ketetanggaan (*Adjacency Matrix*) berdasarkan simpul, sisi, dan nilai bobot pada graf (dalam kilometer).

Tabel 2. Matriks Ketetanggaan

	1	2	3	4	5	6	7	8	9
	0	2.10	2.20	1.70	1.50	∞	∞	∞	∞
	2.10	0	1.30	∞	∞	∞	∞	∞	1.30
	2.20	1.30	0	0.45	∞	∞	∞	∞	0.55
	1.70	∞	0.45	0	0.75	1.00	∞	∞	∞
	1.50	∞	∞	0.75	0	0.50	∞	∞	∞
	∞	∞	∞	1.00	0.50	0	1.50	1.50	1.60
	∞	1.30	0.55	∞	∞	1.50	0	0.30	∞
	∞	∞	∞	∞	∞	1.50	0.30	0	2.70
	∞	1.30	0.55	∞	∞	1.60	2.70	2.70	0

Setelah matriks ketetanggaan terbentuk, maka selanjutnya dapat dilakukan perhitungan secara manual dengan menerapkan algoritma Dijkstra. Hasil didapatkan melalui pencarian rute terpendek secara manual dari titik awal menuju titik akhir sesuai baris dan kolom tabel, kemudian rute yang paling pendek yang akan dijadikan sebagai nilai (dalam kilometer) dari masing-masing baris dan kolom. Untuk hasil rute terpendek direpresentasikan oleh baris pertama, dikarenakan fokus dari pembahasan ini adalah untuk mencari lintasan yang paling efisien dari titik awal Stasiun Tegal yang diwakilkan oleh simpul nomor 1.

Tabel 3. Hasil Penerapan Algoritma Dijkstra

	1	2	3	4	5	6	7	8	9
	0	2.2	2.2	1.7	1.5	2.0	3.4	3.7	2.6
	2.2	0	1.3	1.7	2.5	2.7	4.1	4.5	1.4
	2.2	1.3	0	0.45	1.2	1.5	2.8	3.2	0.55
	1.7	1.7	0.45	0	0.75	1.0	2.4	2.7	1.0
	1.5	2.5	1.2	0.75	0	0.5	1.9	2.2	1.8
	2.0	2.7	1.5	1.0	0.5	0	1.5	1.8	1.6
	3.4	4.4	3.1	2.6	2.4	2.2	0	0.3	3.0
	3.7	4.1	2.8	2.3	1.9	1.6	0.3	0	2.7
	2.6	1.7	0.55	1.0	1.8	1.6	2.7	3.0	0

Baris 1 merepresentasikan hasil pencarian rute terpendek dari Stasiun Tegal menuju ke semua titik Hotel. Rute yang terbentuk terdiri dari beberapa lintasan dengan adanya pembulatan pada nilai bobot (jarak) nya untuk mempermudah penulisan.

$1 \rightarrow 1 = 0$ km
$1 \rightarrow 2 = 2.2$ km
$1 \rightarrow 3 = 2.2$ km
$1 \rightarrow 4 = 1.7$ km
$1 \rightarrow 5 = 1.5$ km
$1 \rightarrow 6 = 1 - 5 - 6 = 2.0$ km
$1 \rightarrow 7 = 1 - 5 - 6 = 3.4$ km
$1 \rightarrow 8 = 1 - 5 - 6 - 7 = 3.7$ km
$1 \rightarrow 9 = 2.6$ km

Berdasarkan hasil pada Tabel 3. diperoleh bahwa rute terpendek (dalam kilometer) dari Stasiun Tegal

p-ISSN: 2502-3691
e-ISSN: 2502-4361

menuju beberapa hotel di Kota Tegal, diantaranya dengan tujuan Bahari Inn Hotel dapat ditempuh dengan jarak 3.4 km melalui rute dari Stasiun Tegal melewati Alexander Hotel dan Primebiz Tegal Hotel. Kemudian, dengan tujuan Karlita Hotel memiliki rute terpendek sejauh 2.6 km dari Stasiun Tegal. Untuk tujuan dari Stasiun Tegal menuju Plaza Hotel dapat ditempuh sejauh 3.7 km, Khas Hotel sejauh 2.2 km, Riez Palace Hotel 1.7 km.

Kemudian, untuk perhitungan selanjutnya yaitu menggunakan bantuan program. Disini bahasa pemrograman yang digunakan adalah bahasa C, yang dimana proses eksekusi program dilakukan dengan Online C Compiler. Berikut merupakan kode program untuk implementasi algoritma Dijkstra terhadap data dan graf yang sudah tersedia.

main.c

```c
#include<stdio.h>
#include<stdbool.h>
#define inf 9999
int n;
int table[100][100];
int dijkstra (int start, int stop){
    int dist[n];
    bool visit[n];
    int i,j,v;
    for (i=1; i<=n; i++){
        dist[i] = inf;
    }
    for (i=1; i<=n; i++){
        visit[i] = false;
    }
    dist[start] = 0;
    while(true){
        int u = -1;
        int minDist = inf;
        for (i=1; i<=n; i++){
            if((visit[i] == false) && (dist[i] < minDist)){
                u = i;
                minDist = dist[i];
            }
        }
        if((u == -1) || (dist[u] == inf)){
            break;
        }
        visit[u] = true;
        for(v = 1; v <= n; v++){
```

Gambar 4. Program Algoritma Dijkstra

```c
            if(table[u][v] != 0){
                if(dist[v] > dist[u]+table[u][v]){
                    dist[v] = dist[u]+table[u][v];
                }
            }
            }
        }
        return dist[stop];
}
int main(){
    int start, stop;
    int i,j;
    printf("Masukkan banyak node : ");
    scanf("%d", &n);
    printf("Masukkan jarak node dalam matriks : \n");
    for(i=1; i<=n; i++){
        for(j=1; j<=n; j++){
            scanf("%d", &table[i][j]);
        }
    }
    int answer = 0, x;
    while(answer == 0){
        printf("\n");
        printf("Masukkan titik awal : ");
        scanf("%d", &start);
        printf("Masukkan titik tujuan : ");
        scanf("%d", &stop);
        printf("Bobot minimal dari titik %d ke %d: %d km \n"
        , start, stop, dijkstra(start,stop));
    }
}
```

Gambar 5. Program Algoritma Dijkstra

Melalui penerapan bahasa C tersebut, pertama akan diminta inputan mengenai seberapa banyak jumlah node yang akan digunakan, node disini berperan dalam menentukan seberapa banyak node yang akan membentuk ukuran matriks. Node yang telah diinputkan akan dilanjutkan dengan proses menginput jarak setiap node dalam bentuk matriks, bisa ditulis jarak menyerupai bentuk matriks dengan bantuan spasi pada keyboard hingga muncul inputan berikutnya. Kemudian, akan diminta inputan mengenai titik awal dan titik tujuan sehingga dapat diperoleh maksimal nilai bobot pada rute tersebut. Untuk penentuan hasil digunakan node sebanyak 9, kemudian untuk jarak node dalam bentuk matriks digunakan tampilan yang menyerupai seperti pada Tabel 3. Kemudian dilakukan perhitungan oleh program berdasarkan titik awal dan titik tujuan yang diinputkan.

```
Masukkan banyak node : 11
Masukkan jarak node dalam matriks :
0 4 0 10 0 4 0 0 10 1 5
4 0 0 7 0 0 0 0 0 3 0
0 0 0 0 4 0 0 9 14 0 0
10 7 0 0 0 0 13 0 13 0 0
0 0 4 0 0 0 0 6 0 0 7
4 0 0 0 0 0 0 0 0 3 1
0 0 0 13 0 0 0 0 2 0 0
0 0 9 0 6 0 0 0 6 0 0
10 0 14 13 0 0 2 6 0 0 13
1 3 0 0 0 3 0 0 0 0 0
5 0 0 0 7 1 0 0 13 0 0

Masukkan titik awal : 1
Masukkan titik tujuan : 1
Bobot minimal dari titik 1 ke 1: 0 km

Masukkan titik awal : 1
Masukkan titik tujuan : 2
Bobot minimal dari titik 1 ke 2: 4 km

Masukkan titik awal : 1
Masukkan titik tujuan : 3
Bobot minimal dari titik 1 ke 3: 16 km

Masukkan titik awal : 1
Masukkan titik tujuan : 4
Bobot minimal dari titik 1 ke 4: 10 km

Masukkan titik awal : 1
Masukkan titik tujuan : 5
Bobot minimal dari titik 1 ke 5: 12 km

Masukkan titik awal :
```

Gambar 6. Hasil Program Dijkstra

Berdasarkan program tersebut, dilakukan input untuk titik awal dari Stasiun Tegal menuju ke beberapa hotel di Kota Tegal, hasil menyatakan bahwa diperoleh nilai bobot minimal yang sama seperti hasil pada perhitungan yang dilakukan secara manual.

B. Penerapan Algoritma Floyd-Warshall

Perhitungan selanjutnya adalah dengan menerapkan algoritma Floyd-Warshall, dengan algoritma tersebut akan dilakukan proses secara manual dan secara program menggunakan bahasa C. Perhitungan dengan algoritma Floyd-Warshall ini, terdiri dari 3 variabel diantaranya k, i, dan j. Dimana nantinya untuk perhitungan akan diselesaikan dengan menggunakan rumus:

$$[i\ ,j\ ]> X\ [i\ ,k\ ]+X\ [k\ ,j\ ]$$

Keterangan:

$[i\ ,j\ ]$ = baris dan kolom

p-ISSN: 2502-3691
e-ISSN: 2502-4361

k = matriks ke-n

Berdasarkan pada graf di Gambar 2. Langkah awal untuk menerapkan algoritma Floyd-Warshall ini adalah dengan membuat matriks hubung graf (K = 0) terlebih dahulu. Untuk variabel i, j, dan k diketahui:

I = {1,2,3,4,5,6,7,8,9,10,11}
J = {1,2,3,4,5,6,7,8,9,10,11}
K = {0,1,2,3,4,5,6,7,8,9,10,11}

Berikut merupakan matriks hubung graf (K = 0) yang telah dibuat. Matriks berikut dibuat berdasarkan derajat yang mewakili setiap simpul pada graf (dalam kilometer).

Tabel 4. Matriks Hubung Graf

	1	2	3	4	5	6	7	8	9
	0	2.2	2.2	1.7	1.5	2.0	3.4	3.7	2.6
	2.2	0	1.3	1.7	2.5	2.7	4.1	4.5	1.4
	2.2	1.3	0	0.45	1.2	1.5	2.8	3.2	0.55
	1.7	1.7	0.45	0	0.75	1.0	2.4	2.7	1.0
	1.5	2.5	1.2	0.75	0	0.5	1.9	2.2	1.8
	2.0	2.7	1.5	1.0	0.5	0	1.5	1.8	1.6
	3.4	4.4	3.1	2.6	2.4	2.2	0	0.3	3.0
	3.7	4.1	2.8	2.3	1.9	1.6	0.3	0	2.7
	2.6	1.7	0.55	1.0	1.8	1.6	2.7	3.0	0

Setelah diperoleh matriks hubung graf seperti pada Tabel 4. Langkah berikut adalah dengan menentukan masing-masing matriks, dengan ketentuan K = (1, 2, 3, …, n) dimana nilai dari K ini mewakilkan matriks ke-berapa yang dibuat serta baris dan kolom berapa yang diharuskan memiliki letak yang tetap pada matriks, nilai pada variabel K juga harus sama dengan jumlah simpul yang terdapat pada graf. Apabila nilai K = 1, maka baris dan kolom 1 haruslah berada pada angka yang sama, kemudian dilakukan perhitungan pada baris dan kolom lainnya dengan menggunakan Persamaan 1.

Tabel 5. Hasil Algoritma Floyd-Warshal

	10	11	12	13	14	15	16	17	18
	0	2.2	2.2	1.7	1.5	2.0	3.4	3.7	2.6
	2.2	0	1.3	1.7	2.5	2.7	4.1	4.5	1.4
	2.2	1.3	0	0.45	1.2	1.5	2.8	3.2	0.55
	1.7	1.7	0.45	0	0.75	1.0	2.4	2.7	1.0
	1.5	2.5	1.2	0.75	0	0.5	1.9	2.2	1.8
	2.0	2.7	1.5	1.0	0.5	0	1.5	1.8	1.6
	3.4	4.4	3.1	2.6	2.4	2.2	0	0.3	3.0
	3.7	4.1	2.8	2.3	1.9	1.6	0.3	0	2.7
	2.6	1.7	0.55	1.0	1.8	1.6	2.7	3.0	0

Dari Tabel 7. dapat diperoleh hasil rute terpendek dari titik awal Stasiun Tegal yang direpresentasikan oleh baris pertama menuju ke semua titik lainnya yang merupakan Hotel di Kota Tegal. Berikut merupakan hasil untuk rute dengan bobot paling minimum.

1 → 1 = 0 km
1 → 2 = 4 km
1 → 3 = 16 km
1 → 4 = 10 km
1 → 5 = 12 km
1 → 6 = 4 km
1 → 7 = 12 km
1 → 8 = 16 km
1 → 9 = 10 km

Berdasarkan Tabel 5. hasil yang diperoleh juga memiliki nilai bobot yang sama pada masing-masing rute dengan perhitungan pada algoritma sebelumnya, untuk jarak dari Stasiun Tegal menuju Alexander Hotel memiliki rute terpendek sejauh 1.5 km, untuk tujuan ke Plaza Hotel sejauh 3.7 km.

Hasil tersebut juga dilaksanakan pada pemrograman. Dengan perhitungan selanjutnya yaitu secara program, disini digunakan bahasa C untuk mengeksekusi code untuk melihat apakah hasil yang menjadi luaran sudah tepat dan sama seperti yang dikerjakan dengan cara manual. Untuk compiler, menggunakan Online C Compiler dan diperoleh hasil yang menyatakan bobot minimum dari inputan titik awal dan titik tujuan.

main.c

```c
#include<stdio.h>
#include<conio.h>
#define inf 9999
int table[100][100], pred[100][100];
int main()
{
    int i,j,k,n,bobot;
    printf("Masukkan banyak node : ");
    scanf("%d",&n);
    for(i=1; i<=n; i++)
    {
        for(j=1; j<=n; j++){
            pred[i][j] = i;
            table[i][j] = 0;
        }
    }
    for(i=1; i<=n; i++)
    {
        for(j=1; j<=n; j++)
        {
            if(i==j)
            {
                table[i][j] = 0;
            }
            else {
                printf("Masukkan jarak node dalam matriks :  \n");
                for(i=1; i<=n; i++){
                    for(j=1; j<=n; j++){
                        scanf("%d", &table[i][j]);
                    }
                }
            }
        }
    }
    for(i=1; i<=n; i++)
    {
        for(j=1; j<=n; j++)
        {
```

Gambar 7. Program Algoritma Floyd-Warshall

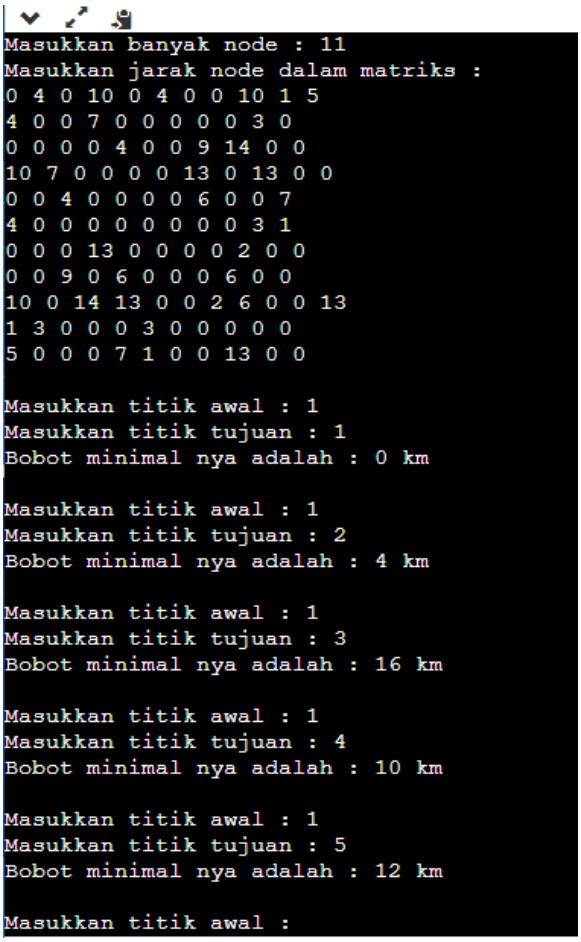
```c
            if(i!=j && table[i][j] == 0)
            {
                table[i][j] = inf;
            }
        }
    }
    for(k=1; k<=n;k++)
    {
        for(i=1; i<=n; i++)
        {
            for(j=1; j<=n; j++)
            {
                if(table[i][k] + table[k][j] < table[i][j])
                {
                    table[i][j] = table[i][k] + table[k][j];
                    pred[i][j] = pred[k][j];
                }
            }
        }
    }
int answer = 0, x;
while(answer == 0){
    int start = 0, stop = 0;
    printf("\n");
    printf("Masukkan titik awal : ");
    scanf("%d", &start);
    printf("Masukkan titik tujuan : ");
    scanf("%d", &stop);
    printf("Bobot minimal nya adalah : %d km \n", table[start][stop]);
    }
}
```

Gambar 8. Program Algoritma Floyd-Warshall

Code tersebut dieksekusi, sehingga luaran akan menampilkan beberapa inputan terlebih dahulu seperti halnya banyak node, jarak node dalam matriks guna

p-ISSN: 2502-3691
e-ISSN: 2502-4361

mendata keseluruhan komponen sehingga dapat terbentuk menjadi sebuah matriks dan dapat diketahui bobot minimal pada titik awal dan tujuan dengan bantuan program. Dilakukan beberapa percobaan terkait inputan titik untuk melihat apakah bobot minimum yang dihasilkan sama dengan bobot pada pengerjaan dengan cara manual.



Gambar 9. Hasil Program Floyd-Warshall

Untuk hasil yang diperoleh memiliki nilai bobot (jarak) minimal yang sama seperti hasil pada perhitungan secara manual pada titik awal dan titik tujuan tertentu.

C. Hasil Perbandingan Algoritma

Berdasarkan dua algoritma yang telah dikerjakan dengan cara manual dan program, yaitu algoritma Dijkstra dan algoritma Floyd-Warshall, diperoleh hasil yang mengacu pada parameter yang digunakan sebagai indikator untuk mengukur algoritma mana yang paling tepat dan efisien dalam menentukan rute terpendek dari Stasiun Tegal menuju ke berbagai hotel di Kota Tegal. Untuk parameter pertama, adalah terkait perhitungan secara manual dan program, kedua algoritma dikerjakan dengan kedua perhitungan tersebut dengan baik, dan diperoleh hasil rute terpendek dan paling dekat yang sama baik dengan cara manual maupun dengan program yaitu pada perjalanan dengan titik awal Stasiun Tegal menuju Jalan A. Yani dengan jarak yang harus ditempuh sejauh 0.8 km. Untuk efektivitas pengerjaan, algoritma Floyd-Warshall terutama dengan cara manual memerlukan waktu pengerjaan yang lebih lama serta memerlukan lebih banyak penggunaan tabel serta perhitungan jika dibandingkan dengan pengerjaan manual pada algoritma Dijkstra.

Selain itu, diperlukan ketelitian yang tinggi untuk memperoleh penyelesaian dengan Algoritma Floyd-Warshall. Untuk outputyang diperoleh, kedua algoritma memiliki hasil yang sama efektif dan tepat. Sedangkan untuk kompleksitas program, algoritma Floyd-Warshall disini juga dinilai lebih kompleks jika dibandingkan dari segi baris source code pada penerapan algoritma Dijkstra seperti halnya pada Gambar 3. dan Gambar 4.

Tabel 6. Perbandingan Algoritma Berdasarkan Parameter Kompleksitas

Algoritma	**Parameter Pengujian**
	Kompleksitas (Bahasa C)
Dijkstra	60 baris
Floyd-Warshall	69 baris

Berdasarkan tabel tersebut, maka algoritma Dijkstra memiliki kompleksitas pada source code yang lebih kecil dibandingkan dengan penerapan pada algoritma Floyd-Warshall, berarti bahwa algoritma Dijkstra memiliki efisiensi yang lebih tinggi dalam menentukan rute terpendek berdasarkan inputan titik awal dan titik tujuannya.

IV. KESIMPULAN

Berdasarkan perhitungan yang telah dilaksanakan sebelumnya, diperoleh hasil bahwa perbandingan kedua algoritma, yakni algoritma Dijkstradan algoritma Floyd-Warshall dalam penentuan rute terpendek dari Stasiun Tegal menuju berbagai Hotel di Kota Tegal dilaksanakan dengan perhitungan manual dan menggunakan bantuan program. Hasil yang diperoleh menunjukkan bahwa dari kedua cara pengerjaan tersebut menghasilkan luaran yang tepat, sekaligus dengan penggunaan dua algoritma yang dapat memberikan output dengan nilai bobot sama. Data dan graf yang dibuat juga dapat diimplementasikan dengan baik. Dengan demikian, algoritma Dijkstra dan algoritma Floyd-Warshall dapat digunakan untuk menyelesaikan persoalan rute terpendek. Namun, tidak menutup kemungkinan akan hasil yang berbeda pada kasus lainnya.

Mengacu pada parameter yang telah dibuat, diperoleh kesimpulan bahwa algoritma Dijkstra merupakan algoritma yang lebih efisien dan lebih praktis serta dapat memberikan output dengan rute terpendek dan nilai bobot yang tepat pula. Penerapan algoritma Dijkstra juga lebih mudah dipahami dan diterapkan khususnya dalam menentukan rute terpendek pada suatu objek penelitian. Rute terpendek serta informasi jarak yang dihasilkan pun juga diharapkan dapat membantu masyarakat terutama para turis yang ingin menginap di Kota Tegal dengan lebih efisien, selain itu guna untuk mengoptimalkan penggunaan kebutuhan seperti bahan bakar, estimasi waktu untuk menempuh perjalanan.

p-ISSN: 2502-3691
e-ISSN: 2502-4361

Daftar Pustaka

Cheng, K. P., Mohan, R. E., Nhan, N. H. K., & Le, A. V. (2019). Graph Theory-Based Approach to Accomplish Complete Coverage Path Planning Tasks for Reconfigurable Robots. *IEEE Access*, *7*, 94642–94657. https://doi.org/10.1109/ACCESS.2019.2928467

Darnita, Y., Toyib, R., & Rinaldi. (2017). *Dan Lokasi Perusahaan Travel / Rental Mobil Di Kota*. *IV*(September), 144–156.

El Fazziki, A., Benslimane, D., Sadiq, A., Ouarzazi, J., & Sadgal, M. (2017). An Agent Based Traffic Regulation System for the Roadside Air Quality Control. *IEEE Access*, *5*, 13192–13201. https://doi.org/10.1109/ACCESS.2017.2725984

Liu, S., Zhang, D. G., Liu, X. H., Zhang, T., Gao, J. X., Gong, C. Le, & Cui, Y. Y. (2019). Dynamic Analysis for the Average Shortest Path Length of Mobile Ad Hoc Networks under Random Failure Scenarios. *IEEE Access*, *7*, 21343–21358. https://doi.org/10.1109/ACCESS.2019.2896699

Luo, M., Hou, X., & Yang, S. X. (2019). A multi-scale map method based on bioinspired neural network algorithm for robot path planning. *IEEE Access*, *7*, 142682–142691. https://doi.org/10.1109/ACCESS.2019.2943009

Lyu, X., Song, Y., He, C., Lei, Q., & Guo, W. (2019). Approach to Integrated Scheduling Problems Considering Optimal Number of Automated Guided Vehicles and Conflict-Free Routing in Flexible Manufacturing Systems. *IEEE Access*, *7*, 74909–74924. https://doi.org/10.1109/ACCESS.2019.2919109

Ni Ketut, D. A. J. (2014). Penggunaan Algoritma Floyd Warshall Dalam Masalah Jalur Terpendek Pada Penentuan Tata Letak Parkir. *Seminar Nasional Informatika*, *1*, 75–81.

Ratnasari, A., Ardiani, F., & A, F. N. (2013). Penentuan Jarak Terpendek dan Jarak Terpendek Alternatif Menggunakan Algoritma Dijkstra Serta Estimasi Waktu Tempuh. *Semantik 2013*, *3*(1), 29–34.

Saajid, H., Di, W., Wang, X., Memon, S., Bux, N. K., & Aljeroudi, Y. (2019). Reliability and Connectivity Analysis of Vehicular Ad Hoc Networks under Various Protocols Using a Simple Heuristic Approach. *IEEE Access*, *7*, 132374–132383. https://doi.org/10.1109/ACCESS.2019.2940872

Setiawan, V., Kiftiah, M., & Partiwi, W. B. (2017). *LINTASAN TERPENDEK PENGANGKUTAN SAMPAH (Studi Kasus : Pengangkutan Sampah di Kabupaten Kubu Raya)*. *06*(3), 221–230.

Wei, X., Zhu, C., Xiao, K., Yin, Q., & Zha, Y. (2018). Shortest Path Network Interdiction with Goal Threshold. *IEEE Access*, *6*, 29332–29343. https://doi.org/10.1109/ACCESS.2018.2838570

Xia, W., Di, C., Guo, H., & Li, S. (2019). Reinforcement Learning Based Stochastic Shortest Path Finding in Wireless Sensor Networks. *IEEE Access*, *7*, 157807–157817. https://doi.org/10.1109/ACCESS.2019.2950055

Zhang, Z., Guo, Q., Chen, J., & Yuan, P. (2018). Collision-Free Route Planning for Multiple AGVs in an Automated Warehouse Based on Collision Classification. *IEEE Access*, *6*, 26022–26035. https://doi.org/10.1109/ACCESS.2018.2819199