Aplikasi *Travelling Salesman Problem* dalam Rute Pengangkutan Sampah di Kecamatan Cakung, Jakarta Timur

Randy Verdian - 13522067[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13522067@itb.ac.id

***Abstract*—Metode *Travelling Salesman Problem (TSP)* digunakan dalam pengelolaan rute pengangkutan sampah di Kecamatan Cakung, Jakarta Timur. TSP adalah permasalahan optimasi kombinatorial yang bertujuan untuk menemukan rute terpendek yang melibatkan kunjungan ke semua titik tetapi setiap titik hanya dilalui sekali dan memiliki titik mulai dan akhir yang sama. Salah satu penyelesaian TSP dilakukan dengan menggunakan algoritma dynamic programming, yang merupakan metode yang efisien untuk menyelesaikan masalah yang memiliki struktur suboptimal dan submasalah yang tumpang tindih. Dengan menggunakan fungsi rekursif, tabel dinamis, dan matriks parent, algoritma ini dapat menemukan rute optimal pengangkutan sampah. Hasil penelitian ini diharapkan dapat memberikan solusi yang efisien dan berkelanjutan dalam pengelolaan sampah di wilayah tersebut.**

***Keywords*—pengangkutan sampah, *Travelling Salesman Problem, dynamic programming,* rute optimal.**

I. PENDAHULUAN

Pengelolaan sampah merupakan salah satu aspek dalam menjaga kebersihan dan kelestarian lingkungan. Dalam skala kota atau kecamatan, pengelolaan sampah menjadi semakin kompleks, terutama dalam hal pengangkutan sampah. Salah satu solusi yang dapat diusulkan adalah menggunakan *Travelling Salesman Problem (TSP)* untuk mengoptimalkan rute pengangkutan sampah di wilayah tertentu. TSP adalah suatu permasalahan optimasi kombinatorial yang bertujuan untuk menemukan rute terpendek yang melibatkan kunjungan ke semua titik tetapi setiap titik hanya dilalui sekali dan memiliki titik mulai dan akhir yang sama. TSP merupakan salah satu permasalahan NP-hard, yaitu permasalahan yang tidak dapat diselesaikan secara optimal dalam waktu polinomial dengan menggunakan algoritma heuristik atau eksploratori.

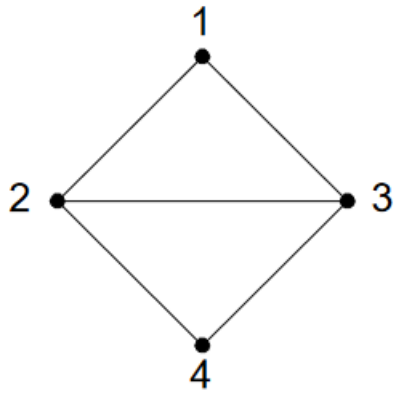
Kecamatan Cakung, Jakarta Timur, sebagai salah satu wilayah perkotaan yang padat penduduk dan wilayah industri, memerlukan pendekatan yang efektif dalam pengelolaan rute pengangkutan sampah. Keberhasilan dalam menyelesaikan permasalahan ini tidak hanya meminimalkan jarak yang ditempuh oleh armada pengangkut sampah tetapi juga dapat berkontribusi pada pengurangan dampak negatif terhadap lingkungan, seperti emisi gas buang dan kepadatan lalu lintas.

Dalam makalah ini, akan dibahas penerapan konsep dan metode *Travelling Salesman Problem* untuk mengoptimalkan rute pengangkutan sampah di Kecamatan Cakung, Jakarta Timur. Penelitian ini diharapkan dapat memberikan solusi yang efisien dan berkelanjutan dalam pengelolaan sampah di wilayah tersebut.

II. TEORI DASAR

A. *Graf*

Graf adalah struktur diskrit berupa kumpulan simpul yang terhubung melalui himpunan sisi. Graf G dinyatakan sebagai pasangan himpunan (V, E) dimana V adalah himpunan tidak kosong dari simpul-simpul (*vertices*) $\{v_1, v_2, v_3, \ldots, v_n\}$ dan E adalah himpunan sisi (*edges*) yang menghubungkan sepasang simpul $\{e_1, e_2, \ldots, e_n\}$. Gambar 2.1 menunjukkan graf $G = (V, E)$ dengan himpunan titik $V = \{1, 2, 3, 4\}$ dan E = {(1, 2), (1, 3), (2, 3), (2, 4), (3, 4)}.



Gambar 2.1. Graf
Sumber: Munir, R. (2023). Graf Bagian 1.

Dua sisi atau lebih yang menghubungkan pasangan simpul yang sama disebut sisi ganda (*multi edges*) dan sisi yang berawal dan berakhir pada simpul yang sama disebut gelang atau kalang (*loop*).

Berdasarkan ada atau tidaknya gelang atau sisi ganda pada suatu graf, graf digolongkan menjadi:

1. Graf sederhana (*simple graph*)
   Graf yang tidak mengandung gelang maupun sisi ganda.
2. Graf tidak sederhana (*unsimple-graph*)
   Graf yang mengandung sisi ganda ataupun gelang.

Berdasarkan orientasi arah pada sisi, graf dibedakan menjadi:

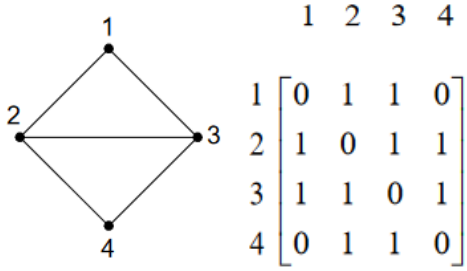
1. Graf tak-berarah (*undirected graph*)
   Graf yang sisinya tidak mempunyai orientasi arah.
2. Graf berarah (*directed graph*)
   Graf yang setiap sisinya diberikan orientasi arah.

Graf memiliki terminologi, diantaranya yaitu:

1. Ketetanggaan (*Adjacent*)
   Dua simpul dikatakan bertetangga jika keduanya dihubungkan oleh sebuah sisi.
2. Bersisian
   Jika sisi e dihubungakn oleh simpul v_i dan v_j, maka sisi e dikatakan bersisian dengan simpul v_i dan v_j.
3. Simpul Terpencil (*Isolated Vertex*)
   Simpul yang tidak mempunyai sisi yang bersisian dengannya.
4. Graf Kosong (*null graph*)
   Graf yang himpunan sisinya adalah himpunan kosong yang dinyatakan dengan (N_n).
5. Derajat (*Degree*)
   Derajat dari suatu simpul adalah jumlah sisi yang bersisian dengannya.
6. Lintasan (*Path*)
   Lintasan yang memiliki panjang n dari simpul awal v_0 ke simpul tujuan v_n adalah barisan berselang-seling simpul-simpul dan sisi-sisi yang berbentuk $v_0, e_1, v_1, e_2, v_2, \dots, v_{n-1}, e_n, v_n$ sehingga $e_1 = (v_0, v_1), e_2 = (v_1, v_2), \dots, e_n = (v_{n-1}, v_n)$ adalah sisi-sisi dari graf G.
7. Siklus (*Cycle*) dan Sirkuit (*Circuit*)
   Lintasan yang berawal dan berakhir pada simpul yang sama.
8. Keterhubungan (*Connected*)
   Simpul v_1 dan v_2 dikatakan terhubung jika terdapat lintasan dari v_1 ke v_2 dalam graf G.
9. Upagraf (*Subgraph*)
   Misalkan terdapat graf $G = (V, E)$. Graf $G_1 = (V_1, E_1)$ adalah upagraf dari G jika V_1 dan E_1 adalah subset dari V dan E berturut-turut.
10. Upagraf Merentang (*Spanning Subgraph*)
    Misalkan terdapat graf $G = (V, E)$. Upagraf $G_1 = (V_1, E_1)$ dikatakan upagraf rentang jika $V_1 = V$ dimana G_1 mengandung semua simpul dari G.
11. Cut-Set
    Himpunan sisi yang jika dibuang dari G menyebabkan G tidak terhubung.
12. Graf Berbobot (*Weighted Graph*)
    Graf yang setiap sisinya diberi nilai atau bobot.
13. Graf Lengkap (*Complete Graph*)
    Graf sederhana yang setiap simpulnya memiliki sisi ke semua simpul lainnya atau saling terhubung.

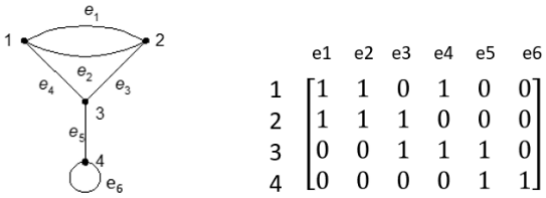
Representasi matriks dari graf adalah cara untuk menyimpan informasi tentang graf dalam bentuk matriks. Ada dua jenis representasi matriks yang umum digunakan, yaitu:

- Matriks ketetanggaan (*Adjacency Matrix*): Matriks ini berukuran $n \times n$, di mana n adalah jumlah simpul (*vertex*) dalam graf. Elemen a_{ij} dari matriks ini menunjukkan apakah ada sisi (*edge*) yang menghubungkan simpul i dan j. Jika ada, maka $a_{ij} = 1$, jika tidak, maka $a_{ij} = 0$. Contoh matriks ketetanggaan dari graf berikut adalah:



Gambar 2.2. Matriks ketetanggaan
Sumber: Munir, R. (2023). Graf Bagian 2.

- Matriks bersisian (*Incidence Matrix*): Matriks ini berukuran $n \times m$, di mana n adalah jumlah simpul dan m adalah jumlah sisi dalam graf. Elemen b_{ij} dari matriks ini menunjukkan apakah simpul i berhubungan dengan sisi j. Jika ya, maka $b_{ij} = 1$, jika tidak, maka $b_{ij} = 0$. Contoh matriks insiden dari graf yang sama dengan sebelumnya adalah:



Gambar 2.3. Matriks bersisian
Sumber: Munir, R. (2023). Graf Bagian 2.

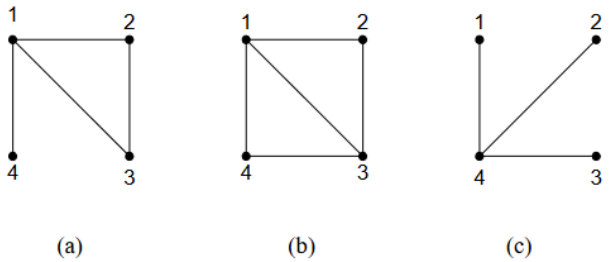
Representasi matriks dari graf dapat digunakan untuk melakukan operasi dan analisis pada graf, seperti mencari jalur terpendek, menentukan konektivitas, menghitung derajat simpul, dan lain-lain.

B. Lintasan dan Sirkuit Hamilton

Lintasan Hamilton merupakan jalur yang melalui setiap simpul dalam graf tepat satu kali. Sebuah Sirkuit Hamilton adalah sirkuit yang melibatkan setiap simpul dalam graf tepat satu kali, kecuali simpul awal (dan sekaligus simpul akhir) yang dilewati dua kali. Sebuah graf yang memiliki sirkuit Hamilton

disebut graf Hamilton, sedangkan yang hanya memiliki lintasan Hamilton disebut graf semi-Hamilton.

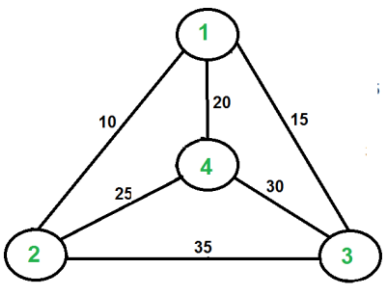
Beberapa teorema tentang graf Hamilton adalah jika graf sederhana G memiliki n simpul ($n \geq 3$) dan setiap simpul memiliki derajat minimal $\frac{n}{2}$, maka G adalah graf Hamilton. Jika graf G adalah graf lengkap, yaitu setiap pasang simpul saling berhubungan, maka G adalah graf Hamilton. Jika graf G adalah graf lengkap dengan n simpul ($n \geq 3$), maka G memiliki $\frac{(n-1)!}{2}$ sirkuit Hamilton yang berbeda. Jika graf G adalah graf lengkap dengan n simpul ($n \geq 3$) dan n ganjil, maka G memiliki $\frac{n-1}{2}$ sirkuit Hamilton yang tidak bersinggungan, yaitu tidak memiliki sisi yang sama. Jika n genap dan $n \geq 4$, maka G memiliki $\frac{n-2}{2}$ sirkuit Hamilton yang tidak bersinggungan.



Gambar 2.4. Contoh Graf Hamilton
Sumber: Munir, R. (2023). Graf Bagian 3.

C. *Travelling Salesman Problem (TSP)*

Secara umum, *Travelling Salesman Problem* (TSP), diberikan sejumlah himpunan simpul V dan memiliki harga c_{uv} pada perjalanan setiap pasang $u, v \in V$. Sebuah perjalanan adalah sirkuit yang melewati tepat sekali di setiap simpul V. *Travelling Salesman Problem* (TSP) digunakan untuk mencari minimal jumlah harga dalam sebuah perjalanan tersebut. TSP dapat dimodelkan sebagai masalah graf dengan mempertimbangkan sebuah graf lengkap $G = (V, E)$, dan memberikan setiap sisi $u, v \in E$ biaya c_{uv}. Sebuah perjalanan pada sirkuit di G bertemu setiap titik, yang dinamakan sirkuit Hamilton.



Gambar 2.4. Contoh TSP
Sumber: https://www.geeksforgeeks.org/travelling-salesman-problem-using-dynamic-programming/.

Sebagai contoh, gambar di atas memiliki rute TSP yaitu 1 – 2 – 4 – 3 – 1 yang memiliki jumlah harga 80. TSP termasuk dalam kelas permasalahan NP-hard, yang berarti tidak ada algoritma yang dapat menyelesaikan TSP secara efisien untuk semua kasus. Ada berbagai macam algoritma yang dapat digunakan untuk menyelesaikan TSP, diantaranya adalah dengan *Branch and Bound*, *Dynamic Programming*, *Reduced Matrix Method*, *nearest neighbor*, *cheapest insertion*, *greedy*, dan *genetic*.

D. *Penyelesaian TSP dengan Algoritma Dynamic Programming*

Dynamic Programming adalah metode yang sangat cocok untuk menyelesaikan masalah yang memiliki struktur suboptimal, yaitu solusi optimal untuk masalah dapat dibangun dari solusi optimal untuk submasalahnya. Metode ini juga sering digunakan untuk masalah yang memiliki submasalah yang tumpang tindih, yang berarti submasalah yang sama dihadapi berulang kali selama perhitungan.

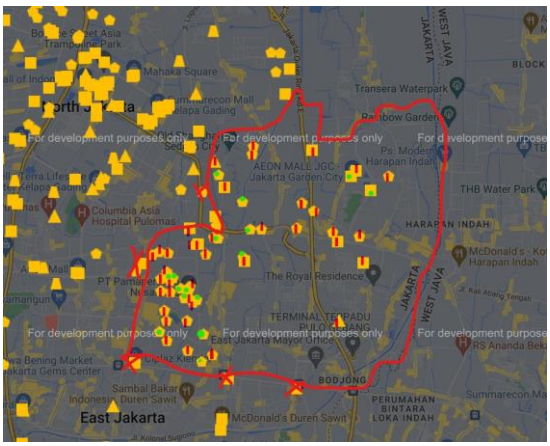
Algoritma ini dapat diterapkan pada TSP dengan menggunakan metode ini, untuk menyelesaikan submasalah ini dengan pemrograman dinamis, kita dapat mendefinisikan fungsi rekursif yang menerima sebagai masukan keadaan saat ini dari masalah (yaitu, kumpulan kota yang belum dikunjungi dan kota saat ini) dan mengembalikan solusi optimal untuk keadaan tersebut (yaitu, rute terpendek yang mengunjungi semua kota yang tersisa dan kembali ke kota awal).

Kemudian dapat menggunakan fungsi rekursif ini untuk menghitung solusi untuk submasalah dan menyimpannya dalam tabel atau array. Tabel ini dapat digunakan untuk mencari solusi untuk submasalah ketika mereka dihadapi lagi, yang dapat meningkatkan efisiensi dari solusi keseluruhan.

III. METODOLOGI

A. *Pengumpulan Data*

Data-data titik pengangkutan sampah diambil dari Portal Resmi Unit Pengelola Sampah Terpadu Dinas Lingkungan Hidup Provinsi DKI Jakarta, kemudian memfilter untuk wilayah Kecamatan Cakung, Kota Jakarta Timur. Diperoleh informasi nama lokasi, alamat, dan status dari setiap titik. Didapatkan sejumlah 57 titik, namun hanya terdapat 14 titik yang memiliki status valid (yang berwarna hijau) seperti pada gambar berikut, sehingga penulis hanya ingin menganalisis rute pada 14 titik tersebut.



Gambar 3.1. Titik pengangkutan sampah di Kecamatan Cakung, Jakarta Timur
Sumber: https://upstdlh.id/wastemanagement/data.

Penulis mengambil lokasi PT. Nobi Putra Angkasa yang memiliki alamat Jl. Pulobuaran Raya Kav. III Blok FF5 sebagai lokasi awal sekaligus lokasi akhir dalam sebuah rute yang akan ditentukan. Namun, perlu diperhatikan bahwa penulis hanya meninjau total jarak yang ditempuh dari sebuah rute dan faktor lain seperti kemacetan, kondisi jalan, waktu tempuh diabaikan.

B. Perancangan Solusi

Dari 14 titik tersebut akan dibuat graf komplit yang mana setiap dua titik pada graf saling terhubung sehingga ada $\binom{14}{2} = 91$ sisi. Banyaknya rute yang mungkin yaitu $\frac{(n-1)!}{2} = \frac{13!}{2} = 3113510400$ rute. Namun, dengan menerapkan pendekatan dynamic programming, efisiensi perhitungan dapat ditingkatkan karena memiliki kompleksitas $O(n^2 2^n)$.

Sekarang masing-masing titik akan dinamakan seperti tabel di bawah.

No.	Alamat	Nama Titik
1.	**PT. Nobi Putra Angkasa** Alamat : Jl. Pulobuaran Raya Kav. III Blok FF5	LOC1
2.	**PT. Yamaha Music Manufacturing Indonesia** Alamat : Jl. Pulobuaran Raya No.1	LOC2
3.	**PT. Fajar Abadi Masindo** Alamat : Kaw. Industri Pulogadung, Blok 3. R. 10-11 No. 4, Jl. Rawagatel	LOC3
4.	**FSCM Manufacturing Indonesia** Alamat : Jl. Raya Pulogadung No. 30, Kawasan Industri Pulogadung	LOC4
5.	**PT. PPG Coating Indonesia** Alamat : Jl. Rawa Gelam 3, RW.9	LOC5
6.	**PT. PAMAPERSADA NUSANTARA** Alamat : Jl. Rawagelam I No.9, Jatinegara, Kec. Cakung	LOC6
7.	**PT. JIEP** Alamat : Kawasan Industri Pulo Gadung	LOC7
8.	**PT. Sari Melati Kencana** Alamat : Jl. Rawa Terate III, RW.9	LOC8
9.	**PT. The Master Steel** Alamat : Jalan Raya Bekasi KM.21	LOC9
10.	**PT. United Tractors** Alamat : jl. Raya Bekasi KM 22 Rt.1/Rw.1 Cakung barat	LOC10
11.	**PT. Jakarta Cakra Tunggal Steel Mills** Alamat : Jalan Raya Bekasi No.km. 21 - 22 Pulogadung	LOC11
12.	**PT. Mitsubishi Krama Yudha Motor and Manufacturing** Alamat : Jl. Raya Bekasi Km 21	LOC12
13.	**Mall AEON Jakarta Garden City (JGC)** Alamat : Jl. Raya Cakung Cilincing KM. 0,5	LOC13
14.	**MCD JAKARTA GARDEN CITY (REKSO NASIONAL FOOD, PT)** Alamat : Komplek Garden City, Jl. Cakung Cilincing Raya Cacing Raya No.KM, RW.5	LOC14

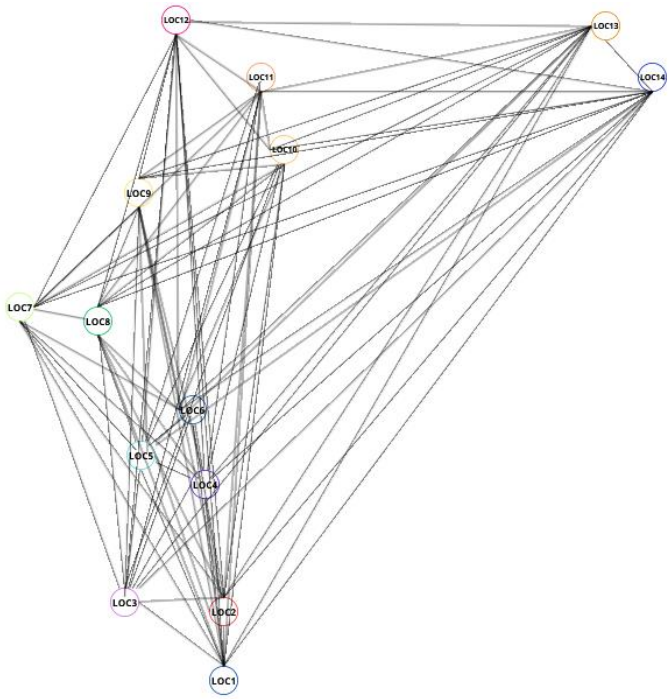
Tabel 3.2. Alamat lokasi pengangkutan sampah
Sumber: https://upstdlh.id/wastemanagement/data.

Berdasarkan data yang sudah diambil dari Google Maps diperoleh jarak antar setiap titik yaitu pada tabel berikut.

	LOC1	LOC2	LOC3	LOC4	LOC5	LOC6	LOC7	LOC8	LOC9	LOC10	LOC11	LOC12	LOC13	LOC14
LOC1	0	0,9	2,3	1,7	2,3	2,1	3,1	2,8	5	5,4	6	5,4	9,4	9,1
LOC2	0,9	0	2,2	1,4	2	1,8	2,8	2,6	4,8	4,5	5	4,8	8,5	8,2
LOC3	2,3	2,2	0	2	2,2	1,8	2	1,8	5,4	5,8	6,3	5,7	9,8	9,5
LOC4	1,7	1,4	2	0	0,6	2,1	2	1,8	3,7	5,7	6,3	5,7	9,8	10
LOC5	2,3	2	2,2	0,6	0	2,1	2	1,8	3,7	5,7	6,2	5,7	9,8	9,5
LOC6	2,1	1,8	1,8	2,1	2,1	0	1,3	1,1	3,7	4,1	4,5	4,1	8,1	7,8
LOC7	3,1	2,8	2	2	2	1,3	0	0,5	1,9	3,6	4,1	3,6	7,6	7,3
LOC8	2,8	2,6	1,8	1,8	1,8	1,1	0,5	0	2,2	4,1	4,6	4,1	8,1	7,8
LOC9	5	4,8	5,4	3,7	3,7	3,7	1,9	2,2	0	3,6	4,1	3,6	7,6	7,3
LOC10	5,4	4,5	5,8	5,7	5,7	4,1	3,6	4,1	3,6	0	2,4	2,3	5,4	5,1
LOC11	6	5	6,3	6,3	6,2	4,5	4,1	4,6	4,1	2,4	0	3	6,2	5,9
LOC12	5,4	4,8	5,7	5,7	5,7	4,1	3,6	4,1	3,6	2,3	3	0	6,5	6,2
LOC13	9,4	8,5	9,8	9,8	9,8	8,1	7,6	8,1	7,6	5,4	6,2	6,5	0	2,7
LOC14	9,1	8,2	9,5	10	9,5	7,8	7,3	7,8	7,3	5,1	5,9	6,2	2,7	0

Tabel 3.3. Jarak antar setiap dua lokasi
Sumber: https://www.google.com/maps.

Dari tabel di atas, maka akan dibuat graf seperti di bawah.



Gambar 3.4. Ilustrasi graf komplit dengan 14 titik lokasi pengangkutan sampah
Sumber: Dokumen Penulis.

C. Implementasi Algoritma Dynamic Programming dalam TSP

Program ini mengimplementasikan algoritma Travelling Salesman Problem (TSP) dengan bahasa C. Kode ini menggunakan pendekatan rekursif dengan memoisasi untuk meningkatkan efisiensi perhitungan.

```
#define N 14

int tsp(int dist[N][N], int mask, int pos, int n, int
dp[N][1 << N], int parent[N][1 << N]) {
    if (mask == (1 << n) - 1) {
        return dist[pos][0];
    }

    if (dp[pos][mask] != -1) {
        return dp[pos][mask];
    }

    int min = INT_MAX;

    for (int city = 0; city < n; city++) {
        if ((mask & (1 << city)) == 0) {
            int newDist = dist[pos][city] + tsp(dist, mask
| (1 << city), city, n, dp, parent);
            if (newDist < min) {
                min = newDist;
                parent[pos][mask] = city;
            }
        }
    }

    dp[pos][mask] = min;
    return min;
}
```

Fungsi tsp merupakan inti dari penyelesaian TSP. Dengan menggunakan rekursi dan dynamic programming, fungsi ini secara efisien mengeksplorasi semua kemungkinan rute yang mengunjungi setiap kota tepat satu kali. Parameter `dist` menyimpan jarak antar kota, mask merepresentasikan himpunan kota yang sudah dikunjungi, `pos` adalah kota saat ini, `n` adalah jumlah total kota, `dp` adalah matriks dinamis untuk menyimpan hasil perhitungan yang telah dilakukan sebelumnya, dan `parent` adalah matriks yang menyimpan informasi kota parent pada setiap langkah untuk merekonstruksi rute optimal. Pada setiap langkah rekursif, fungsi mengevaluasi kemungkinan rute yang belum dieksplorasi dengan mempertimbangkan kota-kota yang belum dikunjungi. Nilai `dp` digunakan untuk menghindari perhitungan berulang, sehingga meningkatkan efisiensi algoritma. Hasil akhir dari fungsi ini adalah jarak minimum yang harus ditempuh untuk menyelesaikan TSP, mengunjungi setiap kota tepat satu kali, dan kembali ke kota awal.

```
void printPath(int parent[N][1 << N], int pos, int mask) {
    if (mask == (1 << N) - 1) {
        return;
    }

    int nextPos = parent[pos][mask];
    printf("LOC%d - ", nextPos + 1);
    printPath(parent, nextPos, mask | (1 << nextPos));
}
```

Prosedur `printPath` digunakan untuk merekonstruksi dan mencetak rute optimal berdasarkan informasi yang telah disimpan dalam matriks parent. Dengan menggunakan rekursi, fungsi ini membangun langkah-langkah rute terpendek dengan mengikuti kota parent dari setiap langkah hingga mencapai kota awal. Fungsi ini mencetak setiap langkah dalam rute optimal, memberikan gambaran visual tentang jalur terpendek yang harus diikuti untuk menyelesaikan TSP.

```
int main() {
    int dist[N][N] = {
        {0, 900, 2300, 1700, 2300, 2100, 3100, 2800, 5000,
5400, 6000, 5400, 9400, 9100},
```

```
        {900, 0, 2200, 1400, 2000, 1800, 2800, 2600, 4800,
4500, 5000, 4800, 8500, 8200},
        {2300, 2200, 0, 2000, 2200, 1800, 2000, 1800, 5400,
5800, 6300, 5700, 9800, 9500},
        {1700, 1400, 2000, 0, 600, 2100, 2000, 1800, 3700,
5700, 6300, 5700, 9800, 10000},
        {2300, 2000, 2200, 600, 0, 2100, 2000, 1800, 3700,
5700, 6200, 5700, 9800, 9500},
        {2100, 1800, 1800, 2100, 2100, 0, 1300, 1100, 3700,
4100, 4500, 4100, 8100, 7800},
        {3100, 2800, 2000, 2000, 2000, 1300, 0, 500, 1900,
3600, 4100, 3600, 7600, 7300},
        {2800, 2600, 1800, 1800, 1800, 1100, 500, 0, 2200,
4100, 4600, 4100, 8100, 7800},
        {5000, 4800, 5400, 3700, 3700, 3700, 1900, 2200,
0, 3600, 4100, 3600, 7600, 7300},
        {5400, 4500, 5800, 5700, 5700, 4100, 3600, 4100,
3600, 0, 2400, 2300, 5400, 5100},
        {6000, 5000, 6300, 6300, 6200, 4500, 4100, 4600,
4100, 2400, 0, 3000, 6200, 5900},
        {5400, 4800, 5700, 5700, 5700, 4100, 3600, 4100,
3600, 2300, 3000, 0, 6500, 6200},
        {9400, 8500, 9800, 9800, 9800, 8100, 7600, 8100,
7600, 5400, 6200, 6500, 0, 2700},
        {9100, 8200, 9500, 10000, 9500, 7800, 7300, 7800,
7300, 5100, 5900, 6200, 2700, 0}
    };

    int dp[N][1 << N];
    int parent[N][1 << N];

    for (int i = 0; i < N; i++) {
        for (int j = 0; j < (1 << N); j++) {
            dp[i][j] = -1;
            parent[i][j] = -1;
        }
    }

    printf("\n|----------------------------------------
------------------------------------------------------
-------------------|\n");
    printf("|
                                                RUTE
OPTIMAL
      |");
    printf("\n|----------------------------------------
------------------------------------------------------
-------------------|\n");
```

```
                printf("|               Titik       mulai:
LOC1
                                                |\n");
    int res = tsp(dist, 1, 0, N, dp, parent);
             printf("|            Jarak      minimum:      %d
m
                                          |\n", res);
    printf("|   Rute: LOC1 - ");
    printPath(parent, 0, 1);
    printf("LOC1    |\n");
    printf("|----------------------------------------
------------------------------------------------------
-----------------|\n\n");

    return 0;
}
```

Pertama-tama, matriks jarak antar kota `dist[N][N]` diinisialisasi dengan nilai-nilai yang merepresentasikan jarak antar kota. Selanjutnya, dilakukan inisialisasi matriks dinamis `dp` dan `parent` untuk menyimpan hasil perhitungan dan informasi rute optimal. Setelah inisialisasi, program memanggil fungsi `tsp` dengan parameter awal yang sesuai dan menyimpan hasilnya dalam variabel `res`. Hasil ini mencerminkan jarak minimum yang harus ditempuh untuk menyelesaikan TSP, dan rute optimalnya direkonstruksi menggunakan prosedur printPath. Selanjutnya, program mencetak informasi tentang rute optimal, termasuk titik awal, jarak minimum, dan urutan kota yang harus dikunjungi.

IV. HASIL DAN PEMBAHASAN

Setelah program dijalankan, maka akan menghasilkan output seperti berikut.

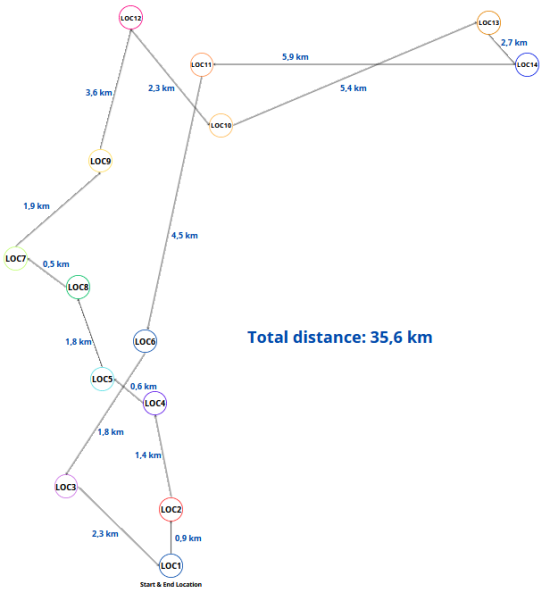
Gambar 4.1. Hasil program TSP dengan dynamic programming
Sumber: Dokumen Penulis.

Sehingga diperoleh total jarak optimal adalah 35600 m atau 35,6 km dengan detail jarak dari setiap titik yang dikunjungi yaitu:

1. Jarak LOC1 - LOC2 : 900 m
2. Jarak LOC2 - LOC4 : 1400 m
3. Jarak LOC4 - LOC5 : 600 m
4. Jarak LOC5 - LOC8 : 1800 m
5. Jarak LOC8 - LOC7 : 500 m
6. Jarak LOC7 - LOC9 : 1900 m
7. Jarak LOC9 - LOC12 : 3600 m
8. Jarak LOC12 - LOC10 : 2300 m
9. Jarak LOC10 - LOC13 : 5400 m
10. Jarak LOC13 - LOC14 : 2700 m

11. Jarak LOC14 - LOC11 : 5900 m
12. Jarak LOC11 - LOC6 : 4500 m
13. Jarak LOC6 - LOC3 : 1800 m
14. Jarak LOC3 - LOC1 : 2300 m

Sehingga rute optimalnya adalah **PT. Nobi Putra Angkasa - PT. Yamaha Music Manufacturing Indonesia - FSCM Manufacturing Indonesia - PT. PPG Coating Indonesia - PT. Sari Melati Kencana - PT. JIEP - PT. The Master Steel - PT. United Tractors - Mall AEON Jakarta Garden City (JGC) - MCD JAKARTA GARDEN CITY - PT. Jakarta Cakra Tunggal Steel Mills - PT. PAMAPERSADA NUSANTARA - PT. Fajar Abadi Masindo - PT. Nobi Putra Angkasa** dengan jarak 35,6 km**.** Jika diilustrasikan pada graf, maka rute optimalnya adalah pada gambar di bawah.



Gambar 4.2. Hasil graf dengan rute optimal
Sumber: Dokumen Penulis.

V. KESIMPULAN

Algoritma dynamic programming dapat mengurangi jumlah perhitungan ulang dan secara signifikan meningkatkan efisiensi solusi dalam menyelesaikan Travelling Salesman Problem. Sehingga diperoleh rute pengangkutan sampah di Kecamatan Cakung, Jakarta Timur yang optimal yaitu **PT. Nobi Putra Angkasa - PT. Yamaha Music Manufacturing Indonesia - FSCM Manufacturing Indonesia - PT. PPG Coating Indonesia - PT. Sari Melati Kencana - PT. JIEP - PT. The Master Steel - PT. United Tractors - Mall AEON Jakarta Garden City (JGC) - MCD JAKARTA GARDEN CITY - PT. Jakarta Cakra Tunggal Steel Mills - PT. PAMAPERSADA NUSANTARA - PT. Fajar Abadi Masindo - PT. Nobi Putra Angkasa** dengan jarak 35,6 km**.**

VI. UCAPAN TERIMA KASIH

Penulis ingin mengucapkan puji syukur kepada Allah SWT karena atas nikmat dan rahmat-Nya, penulis dapat menyelesaikan makalah ini yang berjudul "Aplikasi *Travelling Salesman Problem* dalam Rute Pengangkutan Sampah di Kecamatan Cakung, Jakarta Timur" dengan baik. Tak lupa, penulis ingin menyampaikan terima kasih yang tulus kepada kedua orang tua penulis yang telah memberikan dukungan tanpa henti, doa, dan cinta kasih selama perjalanan penulisan makalah ini. Kehadiran mereka menjadi sumber inspirasi dan kekuatan penulis. Selain itu, penulis ingin menyampaikan rasa terima kasih kepada dosen mata kuliah Matematika Diskrit, Ibu Dr. Fariska Zakhralativa Ruskanda, S.T., Pak Dr. Ir. Rinaldi Munir, M. T., dan Ibu Dr. Nur Ulfa Maulidevi, S. T, M. Sc. yang telah memberikan bimbingan selama perkuliahan ini. Tak lupa juga, penulis ingin mengucapkan terima kasih kepada teman-teman penulis yang telah memberikan dukungan moral dan semangat selama pembuatan makalah ini.

REFERENSI

[1] Munir, R. (2023). Graf Bagian 1. Diakses pada 7 Desember 2023 dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/19-Graf-Bagian1-2023.pdf.
[2] Munir, R. (2023). Graf Bagian 2. Diakses pada 7 Desember 2023 dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/20-Graf-Bagian2-2023.pdf.
[3] Munir, R. (2023). Graf Bagian 3. Diakses pada 7 Desember 2023 dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/21-Graf-Bagian3-2023.pdf.
[4] Cook, S. A. (1971). The Complexity of Theorem-Proving Procedures. Diakses pada 7 Desember 2023 dari https://math.mit.edu/~goemans/18433S15/TSP-CookCPS.pdf, pp. 241-242.
[5] Portal Resmi Unit Pengelola Sampah Terpadu Dinas Lingkungan Hidup Provinsi DKI Jakarta. Diakses pada 8 Desember 2023, dari https://upstdlh.id/wastemanagement/data.
[6] GeeksforGeeks. Travelling Salesman Problem using Dynamic Programming. Diakses pada 8 Desember 2023, dari https://www.geeksforgeeks.org/travelling-salesman-problem-using-dynamic-programming/

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, 10 Desember 2023

Randy Verdian 13522067