Analisis Perencanaan Rute Wisata di Pulau Bali Menggunakan Program Dinamis

Jeremya Dharmawan Raharjo - 13521131
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: 13521131@std.stei.itb.ac.id

***Abstrak*—Pulau Bali terkenal dengan keindahan alamnya, budaya yang kaya, dan pantai yang menarik sehingga dijadikan destinasi tujuan wisata oleh wisatawan dari berbagai belahan dunia. Perencanaan rute wisata di Bali dapat dilakukan dengan metode program dinamis untuk mencari rute termangkus dalam melakukan agenda berwisata. Makalah ini membahas metode-metode yang dapat digunakan untuk melakukan perencanaan rute yang menggunakan persoalan Traveling Salesman Problem dengan Program Dinamis yang akan dikomparasi dengan algoritma *Brute Force*.**

Kata kunci—Bali, Brute Force, Program Dinamis, Travelling Salesman Problem(TSP)

I. PENDAHULUAN

Gambar 1. Ilustrasi Pulau Bali

Bali telah menjadi magnet pariwisata yang terkenal bagi wisatawan lokal maupun mancanegara. Selain pesona alamnya yang menakjubkan, terutama pantai-pantainya, Bali juga terkenal dengan seni dan budayanya yang unik dan mengundang minat. Pusat industri pariwisata terutama berada di wilayah selatan Bali, namun juga ada di beberapa daerah lainnya.

Bali sebagai tempat tujuan wisata yang lengkap dan terpadu memiliki banyak sekali tempat wisata menarik. Lokasi wisata yang utama adalah Kuta dan sekitarnya, selain itu terdapat jenis wisata lain seperti Garuda Wisnu Kencana dan wisata pura di Tanah Lot. Lokasi wisata lain yang terletak di Bali bagian utara antara lain di Bedugul dan Kintamani.

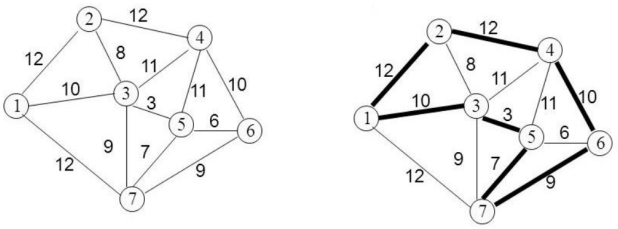
Mengutip laman situs Kementerian Pariwisata dan Ekonomi Kreatif[1] (Kemenparekraf), Bali termasuk 10 destinasi wisata terpopuler di dunia versi tripadvisor. Menurut Menteri Pariwisata dan Ekonomi Kreatif, Sandiaga Salahudin Uno, terjadi peningkatan peringkat Bali dalam industri pariwisata di dunia. Pada tahun 2022, Bali berhasil naik dua peringkat dari peringkat keempat menjadi peringkat kedua. Peringkat ini mengungguli kota London yang turun ke peringkat ketiga, sementara Paris berada di peringkat kelima.

Oleh karena terdapat berbagai destinasi wisata di Pulau Bali, perencanaan rute wisata menjadi hal yang dibutuhkan untuk melakukan agenda berwisata. Persoalan ini perlu dipecahkan agar melakukan agenda berwisata dengan waktu dan jarak yang mangkus.

Makalah ini akan membahas garis besar pemecahan masalah pencarian rute yang mangkus menggunakan algoritma *Brute Force* dan pendekatan dengan Program Dinamis. Bagian pertama dari makalah akan membahas latar belakang dari perencanaan rute di Pulau Bali sebagai destinasi wisata. Bagian kedua dari makalah ini akan membahas landasan teori yang dapat diterapkan untuk memecahkan masalah. Bagian ketiga merupakan metode yang digunakan berdasarkan kajian teoritis pada bagian kedua. Bagian keempat merupakan hasil dan pembahasan dari solusi yang telah dipecahkan berdasarkan metode pada bagian ketiga. Bagian kelima merupakan bagian terakhir dari makalah ini yang memberi kesimpulan.

II. LANDASAN TEORI

2.1 Traveling Salesman Problem (TSP)



Gambar 2. Ilustrasi Traveling Salesman Problem

Traveling Salesman Problem(TSP)[2] merupakan salah satu permasalahan popular pada bidang ilmu komputer untuk mencari sebuah sirkuit hamilton yang memiliki bobot tur minimum. Pada persoalan ini, terdapat sebuah kota awal dan sejumlah n kota yang ingin dikunjungi. Seorang *salesman* (pedagang keliling) diminta untuk memulai perjalanan dari kota A sebagai titik awal menuju semua kota tujuan tepat satu kali. Perjalanan dimulai dan berakhir di kota A, dan tidak diizinkan untuk kembali ke kota awal sebelum semua kota tujuan dikunjungi. Tujuan dari persoalan ini adalah mencari urutan kota yang menghasilkan total jarak perjalanan paling minimum bagi *salesman*.

TSP sendiri masih menjadi persoalan yang digolongkan sebagai persoalan yang tergolong pada NP-hard problems pada bidang ilmu komputer. Hal ini karena TSP merupakan persoalan kombinatorik di antara objek-objek kombinatorik seperti permutasi, kombinasi, atau himpunan bagian dari sebuah himpunan. Hingga saat ini, belum terdapat algoritma yang mangkus untuk menyelesaikan persoalan TSP untuk n buah kota.

Seringkali persoalan TSP dapat menjadi cukup kompleks apabila terdapat n buat kota yang cukup banyak. Beberapa algoritma yang dapat digunakan sebagai pendekatan untuk pemecahan persoalan TSP antara lain Brute Force, Greedy, Backtracking, Genetic Algorithm, Program Dinamis, dan Branch and Bound. Menemukan solusi-solusi optimal dari persoalan TSP tidaklah mudah, karena kompleksitas algoritmanya bukan polinomial.

2.2 Algoritma Brute Force

Algoritma *brute force*[2] merupakan pendekatan yang lempang (*straightforward*) untuk memecahkan suatu persoalan. Biasanya algoritma brute force didasarkan pada pernyataan pada persoalan (*problem statement*) beserta definisi/konsep yang dilibatkan dalam permasalahan tersebut. Algoritma brute force melakukan pendekatan yang naif terhadap pemecahan suatu permasalahan. Kelebihan dari algoritma ini meliputi:

1. Algoritma brute force dapat digunakan untuk menyelesaikan berbagai jenis persoalan.
2. Algoritma brute force mudah dipahami dan memiliki konsep yang relatif sederhana.
3. Algoritma ini juga dapat menghasilkan solusi yang layak untuk persoalan seperti pencarian dan pencocokan string.
4. Algoritma brute force digunakan sebagai standar perbandingan dalam komputasi algoritma.

Sementara itu, kelemahan dari algoritma ini adalah:

1. Algoritma brute force tidak selalu menghasilkan solusi yang efisien.
2. Algoritma brute force membutuhkan waktu yang lama untuk menyelesaikan masalah dengan ukuran input yang besar.
3. Algoritma brute force kurang kreatif dalam pemecahan masalah yang lain.

2.3 Program Dinamis

Program dinamis[3] merupakan metode pemecahan masalah dengan cara menguraikan solusi menjadi beberapa tahapan. Solusi yang diuraikan dalam berbagai tahapan membuat solusi persoalan dapat dipandang sebagai serangkaian keputusan yang saling terkait satu sama lain. Program dinamis digunakan untuk menyelesaikan persoalan-persoalan optimasi (maksimasi atau minimisasi).

Program dinamis sendiri menggunakan Prinsip Optimalitas untuk membentuk rangkaian keputusan yang optimal. Prinsip Optimalitas menyatakan bahwa jika solusi keseluruhan adalah optimal, maka setiap bagian solusinya pada setiap tahap juga optimal. Dengan prinsip ini, ketika kita berpindah dari tahap k ke tahap k + 1, kita dapat menggunakan solusi optimal dari tahap k tanpa perlu kembali ke tahap awal.

Adapun karakteristik dari persoalan yang menggunakan program dinamis adalah sebagai berikut:

1. Persoalan dapat dipecah menjadi beberapa tahap, di mana setiap tahap hanya membutuhkan satu keputusan.
2. Setiap tahap memiliki beberapa status yang terkait dengan tahap tersebut. Secara umum, status adalah berbagai kemungkinan masukan pada tahap tertentu.
3. Keputusan yang diambil pada setiap tahap menghasilkan transformasi dari status yang terkait ke status pada tahap berikutnya.
4. Ongkos atau biaya pada setiap tahap meningkat secara teratur seiring bertambahnya jumlah tahap.
5. Ongkos pada setiap tahap bergantung pada ongkos tahap-tahap sebelumnya dan ongkos dari tahap tersebut ke tahap berikutnya.
6. Terdapat hubungan rekursif yang memungkinkan untuk menentukan keputusan terbaik pada setiap status pada tahap k, sehingga memberikan keputusan terbaik pada setiap status pada tahap k + 1.
7. Prinsip optimalitas berlaku pada persoalan tersebut.

Program dinamis sendiri memiliki beberapa teknik implementasi[4]:

- Pendekatan *top-down* merupakan salah satu pendekatan dalam mengimplementasikan program dinamis secara rekursif. Pendekatan top-down adalah cara sederhana untuk mengimplementasikan program dinamis karena prinsipnya serupa dengan pendekatan brute force yang menggunakan rekursi. Pendekatan ini menyelesaikan masalah dengan inisiasi pemecahan masalah dari kasus yang besar dan kemudian memecah kasus besar tersebut menjadi kasus yang lebih kecil secara rekursif.
- Pendekatan *bottom-up* merupakan pendekatan yang melibatkan pembuatan solusi secara iteratif dari kasus yang paling kecil hingga kasus yang paling besar.

Pendekatan ini bertolak dari pemecahan submasalah yang lebih kecil dan menggunakan hasilnya untuk membangun solusi secara bertahap. Pendekatan ini dimulai dengan menyelesaikan upapersoalan terkecil terlebih dahulu. Kemudian, hasil dari upapersoalan yang lebih kecil digunakan untuk memecahkan upapersoalan yang lebih besar, dan seterusnya, hingga mencapai solusi akhir dari persoalan utama.

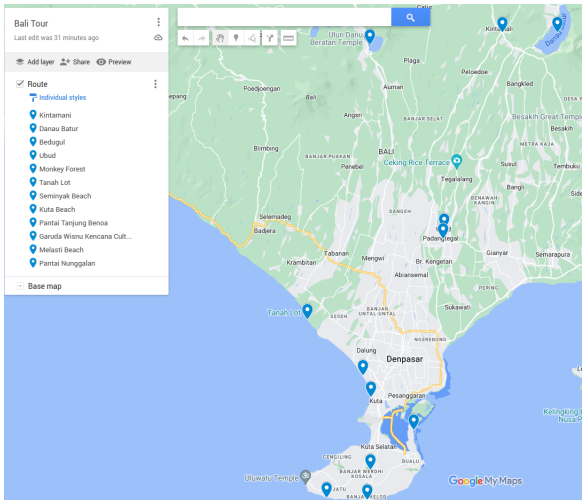
III. METODE

3.1 Penentuan Lokasi Destinasi Wisata

Penulis berhasil mengidentifikasikan beberapa objek wisata di Bali yang terkenal di kalangan para wisatawan.Berikut merupakan daftar destinasi wisata unggulan di Bali versi penulis:

1. Kintamani
2. Danau Batur
3. Bedugul
4. Ubud
5. Monkey Forest
6. Tanah Lot
7. Pantai Seminyak
8. Pantai Kuta
9. Pantai Tanjung Benoa
10. Garuda Wisnu Kencana
11. Pantai Melasti
12. Pantai Nunggalan

Penulis mengurutkan lokasi wisata berdasarkan posisi yang paling utara pada peta, kemudian posisi paling barat pada peta. Penulis menggunakan aplikasi Google Map untuk melakukan penandaan rute dan destinasi-destinasi wisata seperti pada gambar di bawah ini.



Gambar 3. Peta Lokasi Wisata di Pulau Bali
(Sumber: maps.google.com)

Adapun untuk menentukan rute wisata yang mangkus, mencari tur TSP pada destinasi wisata yang sudah ditandai adalah metode yang penulis gunakan dalam penyelesaian persoalan. Tempat-tempat wisata tersebut akan direpresentasikan sebagai simpul dari graf.

Berdasarkan tempat-tempat wisata di atas, dapat dibentuk graf lengkap berbobot yang merepresentasikan jarak antar tempat wisata (dalam kilometer) yang dapat direpresentasikan dalam bentuk matriks ketetanggaan pada Tabel 1 sebagai berikut:

TABEL 1. MATRIKS KETETANGGAAN DESTINASI WISATA DI PULAU BALI

No	1	2	3	4	5	6	7	8	9	10	11	12
1	0	12.9	56.2	33.5	33.3	61.7	58.5	72.7	81.4	81.4	88.6	88.8
2	11.8	0	68.0	38.7	43.1	72.8	79	76.2	85.4	84.9	92.1	92.3
3	53.5	62.7	0	42.1	41.8	48.7	51.5	57.0	72.1	68.5	74.1	75.9
4	33.5	40.7	45.3	0	2.2	33.0	30.3	34.8	44.5	44.0	51.2	51.4
5	35.1	41.4	46.9	1.6	0	34.6	31.3	37.1	46.2	45.6	51.3	53.0
6	60.7	73.1	51.5	32.4	31.5	0	16.7	20.1	36.9	32.9	38.6	40.3
7	58.4	69.5	55.0	29.8	28.8	17.0	0	2.9	22.5	17.7	23.3	25.0
8	60.1	80.6	57.8	36.2	35.3	19.9	2.7	0	19.4	15.8	19.5	23.2
9	76.7	88.3	75.5	45.0	44.0	37.7	21.0	16.8	0	15.0	17.0	22.4
10	75.6	88.7	71.7	43.9	42.9	33.5	16.8	13.0	15.4	0	7.4	9.1
11	82.8	95.5	81.6	51.1	50.1	39.1	22.8	18.2	16.8	7.4	0	11.2
12	89.2	95.6	79.1	51.3	50.3	40.8	24.2	19.9	21.9	9.1	10.3	0

Penulis akan melakukan pencarian rute TSP dengan melakukan komputasi terhadap kedua belas lokasi dan hanya sembilan buah lokasi sebagai pembanding(tanpa memasukkan Garuda Wisnu Kencana Cultural Park, Pantai Melasti, dan Pantai Nunggalan sebagai destinasi wisata yang ingin dikunjungi). Berikut Merupakan implementasi matriks di atas dalam bahasa c++ :

```
vector<vector<float>> bali_tour = {
    {0, 12.9, 56.2, 33.5, 33.3, 61.7, 58.5, 72.7, 81.4, 81.4, 88.6, 88.8},
    {11.8, 0, 68.0, 38.7, 43.1, 72.8, 79, 76.2, 85.4, 84.9, 92.1, 92.3},
    {53.5, 62.7, 0, 42.1, 41.8, 48.7, 51.5, 57.0, 72.1, 68.5, 74.1, 75.9},
    {33.5, 40.7, 45.3, 0, 2.2, 33.0, 30.3, 34.8, 44.5, 44.0, 51.2, 51.4},
    {35.1, 41.4, 46.9, 1.6, 0, 34.6, 31.3, 37.1, 46.2, 45.6, 51.3, 53.0},
    {60.7, 73.1, 51.5, 32.4, 31.5, 0, 16.7, 20.1, 36.9, 32.9, 38.6, 40.3},
    {58.4, 69.5, 55.0, 29.8, 28.8, 17.0, 0, 2.9, 22.5, 17.7, 23.3, 25.0},
    {60.1, 80.6, 57.8, 36.2, 35.3, 19.9, 2.7, 0, 19.4, 15.8, 19.5, 23.2},
    {76.7, 88.3, 75.5, 45.0, 44.0, 37.7, 21.0, 16.8, 0, 15, 17, 22.4},
    {75.6, 88.7, 71.7, 43.9, 42.9, 33.5, 16.8, 13.0, 15.4, 0, 7.4, 9.1},
    {82.8, 95.5, 81.6, 51.1, 50.1, 39.1, 22.8, 18.2, 16.8, 7.4, 0, 11.2},
    {82.9, 95.6, 79.1, 51.3, 50.3, 40.8, 24.2, 19.9, 21.9, 9.1, 10.3, 0}
};
```

Gambar 4. Matriks ketetanggaan pertama dalam c++
(Sumber: Dokumentasi pribadi)

```
vector<vector<float>> bali_tour_2= {
    {0, 12.9, 56.2, 33.5, 33.3, 61.7, 58.5, 72.7, 81.4},
    {11.8, 0, 68.0, 38.7, 43.1, 72.8, 79, 76.2, 85.4},
    {53.5, 62.7, 0, 42.1, 41.8, 48.7, 51.5, 57.0, 72.1},
    {33.5, 40.7, 45.3, 0, 2.2, 33.0, 30.3, 34.8, 44.5},
    {35.1, 41.4, 46.9, 1.6, 0, 34.6, 31.3, 37.1, 46.2},
    {60.7, 73.1, 51.5, 32.4, 31.5, 0, 16.7, 20.1, 36.9},
    {58.4, 69.5, 55.0, 29.8, 28.8, 17.0, 0, 2.9, 22.5},
    {60.1, 80.6, 57.8, 36.2, 35.3, 19.9, 2.7, 0, 19.4},
    {76.7, 88.3, 75.5, 45.0, 44.0, 37.7, 21.0, 16.8, 0},
    };
```

Gambar 5. Matriks ketetanggaan kedua dalam c++
(Sumber: Dokumentasi pribadi)

3.2 Implementasi Pencarian Rute dengan Brute Force

Dengan menggunakan algoritma Brute Force, apabila kita memiliki n buah simpul pada graf lengkap, terdapat (n-1)! kombinasi dari rute TSP. Kombinasi-kombinasi ini dapat dihitung berdasarkan konsep kombinatorial tentang bagaimana menyusun urutan-urutan simpul untuk membentuk tur. Pendekatan dengan algoritma Brute Force melakukan permutasi dan kombinasi pada array untuk menemukan solusi optimal. Kombinasi urutan tur pada array dapat dilakukan secara rekursif.

Berikut merupakan implementasi algoritma brute force dalam mencari tur TSP dalam bahasa c++ versi penulis:

```
void BruteForce::calculate(vector<int>* best_combination,
vector<int>& combination_arr, int start, vector<vector<float>>
matrix){
    //Prosedur menghitung semua permutasi rute TSP sebanyak
    (n-1)!

    if (start == combination_arr.size() - 1) { //basis
        if(weight(combination_arr,matrix) < //ketika bobot lebih
kecil                        weight(*best_combination,matrix)){
            *best_combination = combination_arr;
        }
        return;
    }

    for (int i = start; i < combination_arr.size(); i++) {
    //rekurens
        swap(combination_arr[start], combination_arr[i]);
        calculate(best_combination, combination_arr, start + 1,
                  matrix);
        swap(combination_arr[start], combination_arr[i]);
    }
}

float BruteForce::weight(vector<int> combination,
vector<vector<float>> matrix){
    //Kalkulasi bobot tur berdasarkan rute kombinasi
    float weight = 0;
    for(int i = 0; i < combination.size(); i++){
        if(i == combination.size()-1){
            weight += matrix[combination[i]][0];
        }
        else {
            weight += matrix[combination[i]][combination[i+1]];
        }
    }
    return weight;
};
```

Prosedur `calculate` di atas akan melakukan permutasi terhadap `combination_arr` secara rekursif. Tiap-tiap kombinasi dari `array [0…n-1]` akan dievaluasi dan dipilih bobot terkecil dari sirkuit hamilton dengan bobot terkecil. Rute yang memiliki bobot terkecil akan dikembalikan pada variabel `best_combination`.

3.3 Implementasi Pencarian Rute dengan Program Dinamis

Pencarian tur TSP menggunakan program dinamis dapat dilakukan dengan tahapan berikut:

1) Karakteristikkan struktur solusi optimal

Apabila G = (V, E) merupakan sebuah graf lengkap berarah yang memiliki sisi-sisi dengan bobot c_{ij} yang lebih besar dari 0. Anggaplah V memiliki n simpul, dimana n lebih besar dari 1. Setiap simpul diberikan nomor 1, 2, ..., n. Dalam asumsi bahwa tur dimulai dari simpul pertama, setiap tur akan memiliki tepat satu sisi dari simpul k ke 1 untuk setiap $k \in V - \{1\}$. Dengan menggunakan prinsip optimalitas, jika tur tersebut optimal, maka jalur dari simpul k ke simpul 1 juga menjadi jalur terpendek dari simpul k ke 1 yang melalui simpul-simpul di dalam V – {1, k}.

2) Definisikan secara rekursif nilai solusi optimal

Misalkan f(i, S) adalah bobot lintasan terpendek yang berawal dari simpul i, yang melalui semua simpul di dalam S dan berakhir pada simpul 1.Secara rekursif, hubungan dapat dilakukan dengan cara berikut ini:

$$f(1, V - \{1\}) = min_{2 \le k \le n}\{c_{1k} + f(k, V - \{1, k\})\}$$

dengan merampatkan persamaan di atas diperoleh:

$f(i, \emptyset) = c_{i,1}$ **Basis**

$f(i, S) = min_{j \in S}\{c_{ij} + f(j, S - \{j\})\}$ **Rekurens**

3) Hitung nilai solusi optimal secara maju

Pada bagian ini, lakukan kalkulasi tahap demi tahap dimulai dari tiap upahimpunan kombinasi lokasi. Dengan menggunakan pendekatan *bottom-up*, buat matriks program dinamis dengan ukuran 2^n x n (inisiasikan elemen dengan nilai tertinggi) untuk melakukan komputasi program dinamis tahap demi tahap. Kemudian, lakukan perbandingan pada tiap-tiap upahimpunan untuk mencari bobot minimum sehingga dipilih bobot minimum. Lakukan hal tersebut untuk membuat upahimpunan yang lebih besar hingga mencapai solusi akhir.

4) Rekonstruksi solusi optimal

Setelah mendapatkan nilai minimum dari TSP yang dihitung, lakukan rekonstruksi balik dari simpul k dengan bobot minimum tersebut dan matriks program dinamis hingga

mencapai elemen pertama. Rekonstruksi tersebut akan menghasilkan tur TSP optimal.

Berikut merupakan implementasi pencarian rute optimal menggunakan program dinamis dalam bahasa c++:

```
// Fungsi untuk mencari rute TSP dengan pendekatan Bottom-up
vector<int> ProgramDinamis(const vector<vector<float>>& graph) {
    int n = graph.size(); // Jumlah simpul
    int numSubsets = 1 << n; // Jumlah subset yang mungkin (2^n)

    // Inisialisasi tabel DP dengan nilai MAX_VAL
    vector<vector<float>> dp(numSubsets, vector<float>(n,
                             MAX_VAL));

    // Kasus dasar
    dp[1][0] = 0;

    // Mengisi tabel DP secara bottom-up
    for (int subset = 1; subset < numSubsets; subset++) {
        for (int curr = 0; curr < n; curr++) {
            if ((subset & (1 << curr)) != 0) {
              // Memeriksa apakah simpul "curr" ada dalam subset
                for (int prev = 0; prev < n; prev++) {
                    if (prev != curr &&
                        (subset & (1 << prev)) != 0) {
                 // Memeriksa apakah simpul "prev" ada dalam subset dan
                    tidak sama dengan "curr"

                    // Menghitung biaya untuk mencapai simpul "curr" dari
                       simpul "prev"
                        float cost = dp[subset ^ (1 <<
                             curr)][prev] + graph[prev][curr];
                        // Memilih biaya minimum antara biaya
                          sebelumnya dengan biaya baru
                        dp[subset][curr] = min(dp[subset][curr],
                                               cost);
                    }
                }
            }
        }
    }

    // Mencari rute TSP optimal
    float minCost = MAX_VAL;
    int lastNode = -1;
    for (int i = 1; i < n; i++) {
        // Menghitung biaya total dari simpul terakhir kembali ke
          simpul awal
        float cost = dp[numSubsets - 1][i] + graph[i][0];
        if (cost < minCost) {
            minCost = cost;
            lastNode = i;
        }
    }

    // Rekonstruksi rute TSP dari belakang
    vector<int> tspRoute;
    int subset = numSubsets - 1;
    while (lastNode != 0) { // Lakukan runut balik pada tabel DP
        tspRoute.push_back(lastNode);
        int prevNode = -1;
        for (int prev = 0; prev < n; prev++) {
            if (prev != lastNode && (subset & (1 << prev)) != 0
                &&
                dp[subset][lastNode] == dp[subset ^ (1 <<
                    lastNode)][prev] + graph[prev][lastNode])
            {
                // Jika Node sebelumnya
                prevNode = prev;
                break;
            }
        }
        subset = subset ^ (1 << lastNode);
        lastNode = prevNode;
    }

    // Menambahkan simpul awal ke rute TSP
    tspRoute.push_back(0);

    // Membalikkan rute TSP
    reverse(tspRoute.begin(), tspRoute.end());

    return tspRoute;
}
```

IV. Hasil dan Pembahasan

4.1 Hasil

Uji coba terhadap algoritma brute force dan program dinamis telah berhasil dilakukan untuk menguji performa dan hasil pencarian rute wisata di pulau Bali. Rute yang telah dihasilkan dapat digunakan oleh wisatawan untuk melakukan perjalanan wisata dengan jarak yang optimum. Indeks matriks secara inklusif dimulai dari 0 sebagai indeks pertama.

Eksperimen perencanaan rute menggunakan algoritma brute force untuk n = 12 tempat wisata (menggunakan matriks ketetanggan pertama) menghasilkan rute sebagai berikut:

```
Rute Tur Terbaik: 0 -> 2 -> 5 -> 6 -> 7 -> 9 -> 11 -> 10 -> 8 -> 4 -> 3 -> 1 -> 0
Bobot rute: 274.6
Waktu komputasi: 335.142 detik
jerry@jery201:~/Documents/c++/IF2211-Makalah$
```

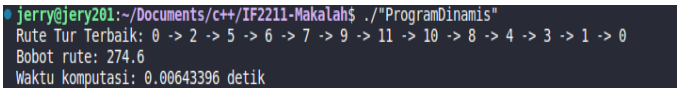
Gambar 6. Solusi TSP matriks ketetanggaan pertama dengan Brute-Force
(Sumber: Dokumentasi pribadi)

Sedangkan untuk n = 9 tempat wisata (menggunakan matriks ketetanggaan kedua) menghasilkan rute sebagai berikut:

```
Rute Tur Terbaik: 0 -> 1 -> 3 -> 4 -> 8 -> 7 -> 6 -> 5 -> 2 -> 0
Bobot rute: 241.5
Waktu komputasi: 0.29342 detik
jerry@jery201:~/Documents/c++/IF2211-Makalah$
```

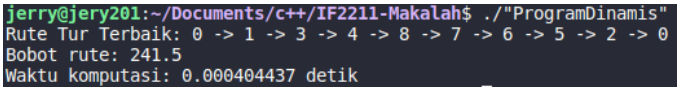
Gambar 7. Solusi TSP matriks ketetanggaan kedua dengan Brute-Force
(Sumber: Dokumentasi pribadi)

Adapun Eksperimen perencanaan rute menggunakan program dinamis untuk n = 12 tempat wisata (menggunakan matriks ketetanggan pertama) menghasilkan rute sebagai berikut:



Gambar 8. Solusi TSP matriks ketetanggaan pertama dengan Program Dinamis
(Sumber: Dokumentasi pribadi)

Sedangkan untuk n = 9 tempat wisata (menggunakan matriks ketetanggaan kedua) menghasilkan rute sebagai berikut:



Gambar 9. Solusi TSP matriks ketetanggaan kedua dengan Program Dinamis
(Sumber: Dokumentasi pribadi)

Kedua algoritma memiliki hasil akhir yang sama namun kompleksitas waktunya berbeda. Detail lengkap dari hasil eksperimen di atas dapat dilihat pada tabel berikut:

TABEL 2. HASIL EKSPERIMEN PENCARIAN RUTE WISATA

n = 12			
No	***Parameter***	***Algoritma***	
		Brute Force	***Program Dinamis***
1	Rute tur terbaik	Kintamani -> Bedugul - Tanah Lot -> Pantai Seminyak -> Pantai Kuta -> Garuda Wisnu Kencana -> Pantai Nunggalan -> Pantai Melasti -> Pantai Tanjung Benoa -> Monkey Forest -> Ubud -> Danau Batur -> Kintamani	
2	Bobot rute	274.6 km	
3	Waktu komputasi	335.142 detik	0.00643 detik
n = 9			
No	***Parameter***	***Algoritma***	
		Brute Force	***Program Dinamis***
1	Rute tur terbaik	Kintamani -> Danau Batur -> Ubud -> Monkey Forest -> Pantai Tanjung Benoa -> Pantai Kuta -> Pantai Sanur -> Tanah Lot -> Bedugul -> Kintamani	
2	Bobot rute	241.5 km	
3	Waktu komputasi	0.2934 detik	4.044×10^{-4} detik

4.2 Pembahasan

Eksperimen dalam membuat perencanaan rute menunjukan hasil yang berbeda pada kedua kasus uji yang menggunakan algoritma Brute Force dan Program Dinamis. Terdapat kesamaan pada solusi optimal rute TSP dengan menggunakan kedua jenis algoritma tersebut, namun terdapat beda signifikan terhadap waktu pencarian solusi.

Pada algoritma Brute Force, pencarian memang menghasilkan hasil yang optimal, namun algoritma yang digunakan sebagai acuan algoritma di bidang ilmu komputer sebagai perbandingan algoritma-algoritma lainnya merupakan algoritma yang tidak mangkus untuk persoalan yang kompleks seperti TSP. Berdasarkan perancangan matriks ketetanggaan pada bagian sebelumnya, untuk n = 12 destinasi wisata, terdapat (12-1)! = 11! = 39,916,800 kombinasi tur. Sedangkan untuk n =9, terdapat (9-1)! = 8! = 40,320 kombinasi tur. Dapat disimpulkan bahwa kompleksitas perhitungan untuk algoritma Brute Force adalah $O(n!)$.

Teknik Program Dinamis dapat mereduksi kompleksitas dari algoritma Brute Force. Dengan menggunakan program dinamis, tahapan demi tahapan dapat dikalkulasi sebanyak n-1 buah tahapan. Kemudian, dari tiap tahapan diperlukan upahimpunan dari himpunan sisi-sisi pada graf yang akan dicari tur TSP-nya, yang mana sebanyak 2^n. Selain itu, untuk tiap tahapan dilakukan perbandingan paling banyak n buah kali untuk mencari nilai minimum berdasarkan tahap sebelumnya, sehingga kompleksitas waktu TSP dengan Program Dinamis adalah $O(n^2 \, x \, 2^n)$.

Perbedaan waktu komputasi ketika banyaknya simpul atau tempat wisata (jumlah n) diubah menunjukan perubahan yang lebih signifikan pada algoritma Brute Force dibandingkan Program Dinamis. Pada algoritma Brute Force dengan n = 12 waktu komputasi TSP dilakukan sekitar 5 menit menunjukan perubahan yang signifikan ketika n = 9 dimana waktu komputasinya 0.29 detik. Sedangkan untuk Program Dinamis, waktu komputasi jauh lebih rendah dibandingkan algoritma Brute Force, namun komputasi dengan metode program dinamis membutuhkan array 2 dimensi sebesar n x 2^n, yang berarti kompleksitas ruang untuk Program dinamis adalah $O(n \;\; x \, 2^n)$.

Karena persoalan TSP merupakan permasalahan yang tergolong NP-hard, penentuan rute wisata di Bali akan semakin kompleks ketika jumlah tempat wisata (simpul) lebih banyak. Untuk algoritma Brute Force tidak mungkin untuk diterapkan pada jumlah n yang sangat besar. Untuk n = 12, hasil eksperimen sudah menunjukan waktu komputasi dalam menit menggunakan bahasa c++. Sedangkan untuk program dinamis, komputasi masih mungkin dilakukan pada rentang waktu singkat apabila n = 12.

V. KESIMPULAN

Pulau Bali merupakan salah satu industri pariwisata yang terkemuka di dunia. Tak sedikit wisatawan, baik dari mancanegara, maupun dari dalam negeri merencanakan berlibur di Bali. Wisatawan perlu merencanakan rute yang baik untuk berwisata di Bali mengingat banyaknya objek wisata yang ditawarkan.

Perencanaan rute berwisata dapat dilakukan dengan mengimplementasikan algoritma-algoritma tertentu dalam bidang ilmu komputer. Salah satu cara merencanakan rute perjalanan wisata dengan menggunakan konsep permasalahan TSP yang dapat diselesaikan dengan algoritma Brute Force

dan Program Dinamis. Algoritma Brute Force merupakan algoritma yang tidak mangkus secara kompleksitas waktunya untuk menyelesaikan permasalahan TSP, sedangkan dengan Program Dinamis persoalan TSP dapat mereduksi kompleksitas waktu yang jauh lebih cepat daripada algoritma Brute Force. TSP tergolong permasalahan dengan kompleksitas waktu yang tinggi untuk mencari solusi optimal karena belum ada algoritma polinomial yang dapat memberikan solusi optimal.

VI. Saran

Saran bagi penelitian selanjutnya adalah melakukan analisis lebih lanjut terhadap parameter lain selain jarak untuk menemukan solusi optimal dalam penentuan rute wisata. Pencarian rute optimal dapat mempertimbangkan parameter seperti ongkos, jenis kendaraan, bobot popularitas tempat wisata dan lain-lain. Penyelesaian TSP dengan Program Dinamis juga dapat dilakukan reduksi pada kompleksitas ruang yang digunakan. Selain itu, dapat mempertimbangkan alternatif penyelesaian persoalan TSP dapat menggunakan algoritma lain seperti Greedy, Branch and Bound, Genetic Algorithm, dan Backtracking.

Selain itu, penentuan rute wisata dapat dipandang dengan sudut pandang permasalahan lain seperti Robot Coin Problem dan pencarian rute antara dua lokasi. Dengan memandang dari sudut pandang permasalahan lain, persoalan rute wisata dapat menggunakan jenis algoritma lain pada bidang ilmu komputer seperti A-star.

Video Link Youtube

https://youtu.be/e_gGq12m7-k

Ucapan Terima Kasih

Pertama-tama, Saya mengucapkan syukur kepada Tuhan Yang Maha Esa karena dengan rahmatNya, Makalah berjudul “Analisis Perencanaan Rute Wisata di Pulau Bali Menggunakan Program Dinamis ” dapat saya selesaikan dengan baik dan tanpa hambatan sedikit pun. Terima kasih kepada Bapak Dr. Ir. Rinaldi Munir, MT, sebagai pengampu mata kuliah IF2211 Strategi Algoritma dan membimbing saya selama satu semester ini. Terima kasih kepada segala pihak yang tidak dapat disebutkan satu persatu sehingga turut membantu terselesaikannya makalah ini.

Daftar Pustaka

[1] https://www.kemenparekraf.go.id/hasil-pencarian/siaran-pers-bali-masuk-10-destinasi-terpopuler-dunia-versi-tripadvisor-ungguli-london-dan-paris (Diakses 21 Mei 2023)

[2] Rinaldi Munir, “Algoritma Brute Force (Bag. 1)”, https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-(2022)-Bag1.pdf (Diakses 21 Mei 2023)

[3] Rinaldi Munir, “Program Dinamis (Bag. 2)”, https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Program-Dinamis-2020-Bagian2.pdf (Diakses 21 Mei 2023)

[4] Rinaldi Munir, “Teori P, NP, dan NP-Complete (Bag. 2)”, https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2019-2020/Teori-P-NP-dan-NPC-(Bagian%202).pdf (Diakses 21 Mei 2023)

[5] https://www.enjoyalgorithms.com/blog/top-down-memoization-vs-bottom-up-tabulation(Diakses 21 Mei 2023)

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, 22 Mei 2023

Jeremya Dharmawan Raharjo

13521131