Aplikasi Graf dalam Penentuan Jalur Teroptimal dalam Rute Kebun Binatang dengan Pendekatan Travelling Salesman Problem

Muhammad Yusuf Rafi- 13522009[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]*author@itb.ac.id*

***Abstract*—Kebun binatang merupakan salah satu tempat rekreasi yang diminati oleh masyarakat karena menawarkan pengalaman yang unik dan edukatif. Dalam kebun binatang, tentu ada banyak habitat dan atraksi hewan yang dihubungkan melalui rute tertentu. Namun, sering kali terdapat rute yang berulang dengan jarak antar satu titik ke titik lain cukup jauh. Maka dari itu, dibutuhkan rute teroptimal yang bisa menghubungkan semua titik atau tempat tepat sekali sebelum kembali ke titik awal dengan jarak antar tempat terminimum. Melalui konsep graf dan *Traveling Salesperson Problem(TSP)*, dapat dikembangkan aplikasi melalui algoritma *Branch and Bound* dalam menentukan rute teroptimal tersebut.**

***Keywords*—Rute Teroptimal, Graf, *Traveling Salesperson Problem,* Algoritma *Branch and Bound*.**

I. Pendahuluan

Dalam era di mana rekreasi dan pengetahuan menjadi bagian dari gaya hidup, destinasi wisata seperti kebun binatang menjadi salah satu tempat yang sangat diminati. Kebun binatang bukan hanya menjadi tempat hiburan, tetapi juga menjadi wahana pembelajaran yang unik bagi pengunjung, khususnya terkait dengan keanekaragaman hayati dan habitatnya. Namun, di balik keragaman dan kompleksitas atraksi yang disajikan kebun binatang, terdapat tantangan yang sering dihadapi, yaitu menemukan rute teroptimal yang dapat menghubungkan berbagai area dengan jarak tempuh minimum.

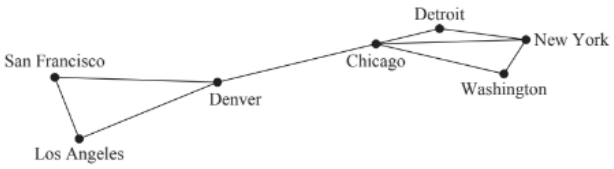
Pentingnya menemukan rute yang efisien dan optimal di dalam kebun binatang menjadi krusial, bukan hanya untuk memaksimalkan pengalaman pengunjung, tetapi juga untuk mengoptimalkan manajemen operasional kebun binatang itu sendiri. Oleh karena itu, dalam konteks penelitian ini, peran konsep-konsep dalam teori graf dan *Traveling Salesperson Problem* (TSP) menjadi sangat relevan. Penggunaan TSP yang cerdas dan strategis di dalam kebun binatang dapat menjadi solusi yang tepat guna untuk mengatasi permasalahan dalam menentukan rute terbaik dengan jarak yang optimal.

Makalah ini bertujuan untuk menguraikan serta menganalisis terkait penerapan algoritma *Branch and Bound* dalam menyelesaikan TSP dapat menghasilkan solusi terbaik dalam menentukan rute teroptimal di dalam kebun binatang. Melalui penelitian yang terstruktur, makalah ini akan membahas secara mendalam konsep-konsep dasar dari graf, aplikasi graf pada penyelesaian TSP, dan algoritma *Branch and Bound* sebagai aplikasi pemrograman dalam mendapatkan solusi tersebut.

II. Landasan Teori

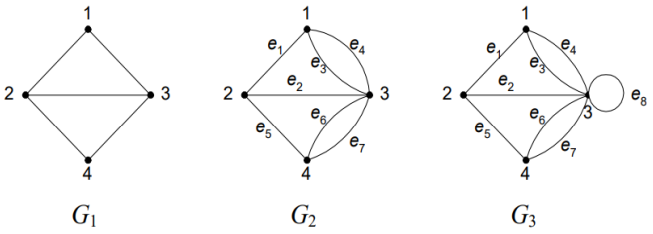
A. Teori Graf

Graf adalah struktur diskret yang terdiri dari simpul (*vertices*) dan sisi (*edges*) yang menghubungkan simpul-simpul tersebut. Graf dapat dilambangkan, G = (V, E), terdiri dari V, sebuah himpunan tak kosong dari simpul dan E, sebuah himpunan sisi-sisi yang menghubungkan sepasang simpul. V = { v1 , v2 , ... , vn } dan E = {e1 , e2 , ... , en }.
Setiap sisi memiliki satu atau dua simpul yang terkait dengannya.



Gambar 1. Representasi Graf dalam Lokasi Jaringan Komputer
Sumber: Discrete Mathematics and Its Application, 7th Ed pp. 603

Sebuah graf di mana setiap sisi menghubungkan dua simpul yang berbeda dan di mana tidak ada dua sisi yang menghubungkan pasangan simpul yang sama disebut sebagai graf sederhana. Berdasarkan orientasi arah pada sisi, graf dibedakan menjadi graf tak berarah dan graph berarah. Graf tak-berarah (*undirected graph*) adalah jenis graf di mana setiap sisi (*edge*) tidak memiliki arah yang ditentukan. Artinya, sisi yang menghubungkan dua simpul (*vertices)* tidak memiliki orientasi tertentu, tidak ada panah yang menunjukkan arah dari satu simpul ke simpul lainnya. Dalam graf tak berarah, hubungan antara dua simpul bersifat saling terhubung tanpa arah tertentu, sehingga jika simpul A terhubung dengan simpul B, maka secara bersamaan simpul B juga terhubung dengan simpul A.

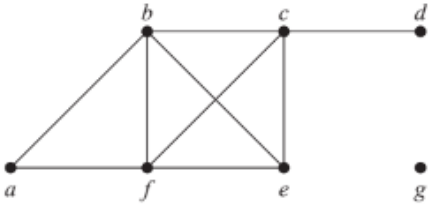


Gambar 2. (a) graf sederhana, (b) graf ganda, dan (c) graf semu
Sumber:https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/19-Graf-Bagian1-2023.pdf

Pada Gambar 2, dapat didefinisikan graf sederhana memiliki 4 simpul dan 4 sisi, V = { 1, 2, 3, 4 } dan E = { (1, 2), (1, 3), (2, 3), (2, 4), (3, 4) }. Adapun graf ganda dan semu termasuk ke dalam graf tak-sederhana yang bisa memiliki sisi ganda yang menghubungkan dua simpul yang sama, yang berarti terdapat lebih dari satu sisi yang menghubungkan sepasang simpul tertentu. Selain itu, graf tak-sederhana juga bisa memiliki simpul yang terhubung dengan dirinya sendiri, membentuk apa yang disebut sebagai loop di dalam graf.

B. Terminologi Graf

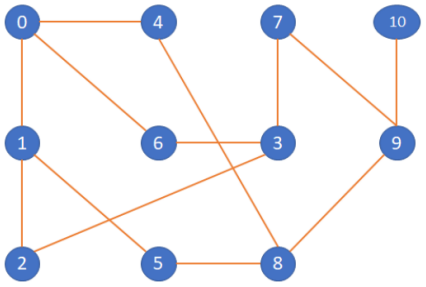
Dua titik atau simpul v1 dan v2 dalam graf tak berarah G dianggap bertetangga (*adjacent*) jika keduanya berbagi sisi atau edge yang sama. Untuk melacak jumlah sisi yang berada pada sebuah simpul, didefinisikan sebuah derajat. Derajat sebuah titik dalam graf tak berarah adalah jumlah dari sisi-sisi yang terhubung ke titik tersebut. Satu pengecualian adalah jika terdapat suatu lingkaran atau loop pada titik tertentu, maka loop tersebut akan dihitung dua kali saat menghitung derajat titik tersebut. Derajat dari sebuah titik v dilambangkan sebagai deg(v). Selain itu, graf juga mempunyai bobot, di mana setiap sisi (edge) memiliki nilai numerik. Bobot ini menunjukkan nilai atau biaya yang terkait dengan setiap sisi dalam graf, mewakili karakteristik khusus seperti jarak, biaya, waktu, atau atribut lainnya yang relevan.



Gambar 3. Terminologi Ketetanggaan (*Adjacent*) dan Derajat pada Graf
Sumber: Discrete Mathematics and Its Application, 7th Ed pp.601-174

Pada Gambar 3, tiap simpul pada graf tersebut memiliki derajat tertentu, deg(a) = 2, deg(b) = deg(c) = deg(f) = 4, deg(d) = 1, deg(e) = 3, dan deg(g) = 0. Ketetanggan (*adjacent*) dari tiap simpul, yakni N (a) = {b, f }, N (b) = {a, c, e, f }, N (c) = {b, d, e, f }, N (d) = {c}, N (e) = {b, c, f }, N (f) = {a, b, c, e}, and N (g) = ∅.

Sebuah graf juga memiliki lintasan yang berhubungan nantinya dengan sirkuit pada graf. Lintasan (*path*) adalah urutan simpul yang berbeda yang dihubungkan oleh sisi-sisi yang berurutan. Dengan kata lain, ini adalah rangkaian simpul yang terhubung secara berurutan melalui sisi-sisi dalam graf tanpa mengunjungi simpul yang sama dua kali, sedemikian sehingga lintasan tertutup dalam graf yang dimulai dan berakhir di simpul yang sama disebut sirkuit.



Gambar 4. Lintasan dan Sirkuit pada Graf
Sumber:https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/19-Graf-Bagian1-2023.pdf

Pada Gambar 4, lintasan 0, 4, 8, 5, 1, 0 adalah sebuah sirkuit. Panjang sirkuit adalah jumlah sisi dalam sirkuit tersebut, sehingga sirkuit pada G memiliki panjang 5.

C. Representasi Graf

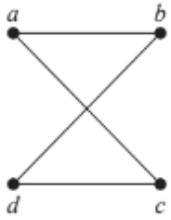
Ada banyak cara yang berguna untuk merepresentasikan graf. Salah satu cara untuk merepresentasikan graf sederhana adalah dengan merepresentasikan graf menggunakan matriks berdasarkan daftar keberdekatan (adjacency lists), yang menentukan simpul-simpul yang bertetangga dengan setiap simpul dalam graf.

Misalkan G = (V, E) adalah graf sederhana di mana |V| = n. Anggaplah simpul-simpul dari G terdaftar secara sembarang sebagai v1, v2,..., vn. Matriks ketetanggan dari G, terkait dengan pengurutan simpul-simpul ini, adalah matriks n x n berisikan berisi angka 0 dan 1:.

$$a_{ij} = \begin{cases} 1 & \text{if } \{v_i, v_j\} \text{ is an edge of } G, \\ 0 & \text{otherwise.} \end{cases}$$

Gambar 5. Ketentuan Elemen *Adjacency* Matriks
Sumber:Discrete Mathematics and Its Application, 7th Ed pp.601-174

Matriks ketetanggaan dari sebuah graf sederhana bersifat simetris, yaitu aij = aji, karena kedua entri ini bernilai 1 ketika vi dan vj saling berdekatan, dan keduanya bernilai 0 jika tidak. Selain itu, karena graf sederhana tidak memiliki loop, setiap entri aii, i = 1, 2, 3,...,n, adalah 0.



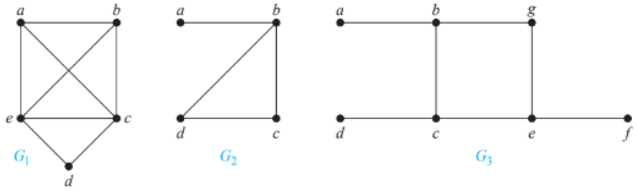
$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

Gambar 6. Representasi Graf dengan *Adjacency* Matriks
Sumber:Discrete Mathematics and Its Application, 7th Ed pp.601-174

Pada Gambar 5, graf tersebut direpresentasikan dalam bentuk matriks ketetanggaan di mana urutan pada baris dan kolom sesuai dengan urutan simpul a, b, c, d. Untuk representasi graf berbobot, secara tidak langsung dapat kita representasikan juga dengan matriks ketetanggan, di mana jika sepasang simpul bertetanggan, dapat kita ganti nilai 1 pada aij dengan nilai bobot pada sisi tersebut.

C. Lintasan dan Sirkuit Hamilton

Dalam sebuah graf G, lintasan sederhana yang melewati setiap simpul tepat sekali disebut lintasan Hamilton, dan sirkuit sederhana yang melewati setiap simpul tepat sekali yang kembali ke simpul awalnya disebut sirkuit Hamilton. Graf yang memiliki sirkuit Hamilton dinamakan graf Hamilton, sedangkan graf yang hanya memiliki lintasan Hamilton disebut graf semi-Hamilton.



Gambar 7. (a) graf hamilton, (b) graf semi-hamilton, dan (c) bukan graf hamilton dan semi-hamilton
Sumber:Discrete Mathematics and Its Application, 7th Ed pp.601-174

G1 memiliki sirkuit Hamilton: a, b, c, d, e, a. Namun, Tidak ada sirkuit Hamilton di G2 (ini terlihat dengan mencatat bahwa setiap sirkuit yang memuat setiap simpul harus memuat sisi {a, b} dua kali), tetapi G2 memiliki lintasan Hamilton, yaitu: a, b, c, d. Sedangkan, G3 tidak memiliki sirkuit Hamilton maupun lintasan Hamilton, karena setiap lintasan yang memuat semua simpul harus memuat salah satu sisi {a, b}, {e, f}, dan {c, d} lebih dari sekali.

Untuk menentukan sebuah graf hamilton, semi-hamilton, atau bukan keduanya, dapat dilandaskan dua teori. Teorema Dirac menyatakan bahwa jika G adalah sebuah graf sederhana dengan n titik, dimana n ≥ 3 dan setiap titik memiliki derajat paling sedikit n/2, maka G memiliki sebuah sirkuit Hamilton.

Teorema Ore menyatakan bahwa jika G adalah sebuah graf sederhana dengan n titik, dimana n ≥ 3 dan untuk setiap pasangan titik non-berdekatan u dan v di G, derajat(u) + derajat(v) ≥ n, maka G memiliki sebuah sirkuit Hamilton. Dalam graf dengan n buah simpul (n 3), terdapat (n – 1)!/2 buah sirkuit Hamilton.

D. The Traveling Salesperson Problem

Traveling Salesperson Problem (TSP) atau Masalah Salesperson yang Bepergian adalah masalah umum dalam teori graf dan komputer sains. Masalah ini mengajukan pertanyaan tentang cara terbaik bagi seorang *salesperson* (penjual) untuk mengunjungi sejumlah kota tertentu dan kembali ke kota asal, dengan meminimalkan total jarak yang harus ditempuh.

Penyelesain TSP secara langsung melibatkan eksplorasi semua sirkuit Hamilton dan memilih sirkuit yang memiliki jumlah bobot terminimum. Namun, jumlah sirkuit yang perlu diperiksa berkembang secara cepat dengan jumlah simpul (n). Misalnya, dengan n simpul, ada (n-1)! sirkuit yang perlu dipertimbangkan setelah memilih titik awal. Maka dari itu, menyelesaikan TSP secara langsung untuk graf-graf dengan banyak simpul menjadi tidak praktis. Sebagai contoh, dengan hanya 25 simpul, sekitar 3.1 × 10^23 sirkuit Hamilton yang berbeda akan perlu dipertimbangkan, membutuhkan waktu yang tidak praktis.

Algoritma yang efisien untuk TSP sangat penting karena relevansinya baik secara praktis maupun teoritis. Salah satu algoritma yang cocok, yakni Algoritma *Branch and Bound*.

D. Algoritma Branch and Bound

Algoritma Branch and Bound adalah strategi pencarian yang secara efektif memperkecil ruang pencarian dengan membaginya menjadi bagian-bagian yang lebih kecil (pencabangan) dan mengevaluasi batas atas dan batas bawah dari setiap bagian ini (bound) untuk memutuskan apakah perlu melanjutkan pencarian atau tidak.

Dalam Branch and Bound, bagian yang menantang adalah menemukan cara untuk menghitung batasan pada solusi terbaik yang mungkin. Pada masalah minimisasi, batas bawah memberikan informasi mengenai solusi minimum yang mungkin jika kita mengikuti simpul tersebut.

Cara kerja algoritma ini, yakni:

1. Mulai dari simpul "0" atau simpul awal untuk mulai dengan menghitung batasan terhadap solusi terbaik yang mungkin. Dalam kasus ini, batasan terhadap solusi terbaik merupakan batas bawah dari biaya rute yang akan dihasilkan nanti dengan rumus (1/2) * Σ (jumlah biaya dua sisi minimum yang bertetangga pada tiap simpul).
2. Selanjutnya, mulai mengeksplorasi setiap simpul dalam graf. untuk menemukan semua kemungkinan jalur menggunakan pendekatan rekursif, setiap langkah rekursi, algoritma mengevaluasi semua jalur yang mungkin dan memperbarui batasan bawah berdasarkan sisi minimum dan kedua terkecil serta berat jalur yang ditambahkan pada perjalanan TSP saat melewati simpul tertentu. Tentunya, Algoritma

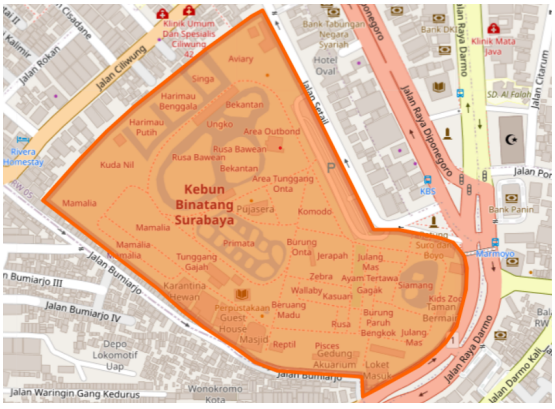
memilih sisi dengan bobot terkecil untuk ditambahkan pada rute yang sedang dibangun. Namun, pemilihan sisi ini tidak dilakukan begitu saja, melainkan mempertimbangkan batasan terhadap solusi terbaik (batas bawah) yang telah dihitung sebelumnya.

3. Perhitungan untuk simpul 1, karena bergerak dari 0 ke 1, rute sekarang telah mencakup sisi 0-1. Hal ini memungkinkan kita untuk melakukan perubahan yang diperlukan pada batasan bawah simpul awal sebagai berikut:
   Batas bawah untuk simpul 1 = Batas bawah lama - ((biaya sisi minimum 0 + biaya sisi minimum 1) / 2) + (biaya sisi 0-1).
4. Selanjutnya, untuk simpul-simpul lain yang mungkin. Setelah simpul 1, kita bisa memeriksa simpul 2, 3, dan seterusnya. Misal, ketika berpindah dari simpul 1 ke simpul 2, kita memasukkan sisi 1-2 ke dalam rute. Untuk memperbarui batas bawah untuk simpul 2 (simpul> 1),rumusnya adalah:
   Batas bawah(simpul > 1) = Batas bawah lama - ((biaya sisi minimum kedua dari (simpul-1) + biaya sisi minimum dari simpul) / 2) + biaya sisi ((simpul-1)-simpul).
5. Algoritma akan terus berlanjut hingga terbentuk suatu sirkuit Hamilton, yaitu suatu rute yang melalui setiap kota tepat satu kali dan kembali ke kota awal. Proses ini terus dilakukan dengan mengoptimalkan pemilihan sisi-sisi yang ditambahkan selama pembentukan rute, tetapi dengan memperhatikan batasan bawah yang telah dihitung sebelumnya.

III. Metode

A. Data Peta Kebun Binatang

Data yang digunakan dalam riset ini berasal dari *openstreetmap.org* yang merupakan *open source* berisikan tata letak berbagai wilayah di dunia.



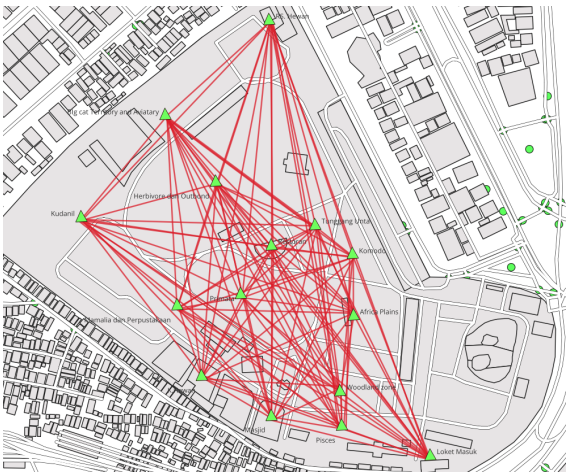
Gambar 7. Peta Kebun Binatang Surabaya
Sumber:https://www.openstreetmap.org/way/279576577#map=17/-7.29568/112.73817

Pada Gambar 7, Peta tersebut merupakan tata letak wilayah pada Kebun Binatang Surabaya pada tahun 2021. Peta tersebut menyediakan informasi rinci tentang rute jalan dan penempatan para satwa di tiap wilayah lengkap dengan tempat restoran, musholla, perpustakaan, dan area rekreasi .

Adapun, agar ruang lingkup tidak terlalu luas, saya membatasi cakupan dari penelitian ini dengan mengambil rute dari loket masuk dan bagian wilayah kiri dari tempat loket masuk..

B. Pembentukan Graf

Langkah pertama yang diperlukan, yakni pembentukan graf dari peta yang telah tersedia. Maka dari itu, saya menggunakan aplikasi QGIS(aplikasi pembuatan peta) untuk pembentukan simpul dan sisi pada setiap wilayah yang berada di area kebun binatang.



Gambar 8. Representasi Graf pada Kebun Binatang Surabaya
Sumber:hak cipta penulis

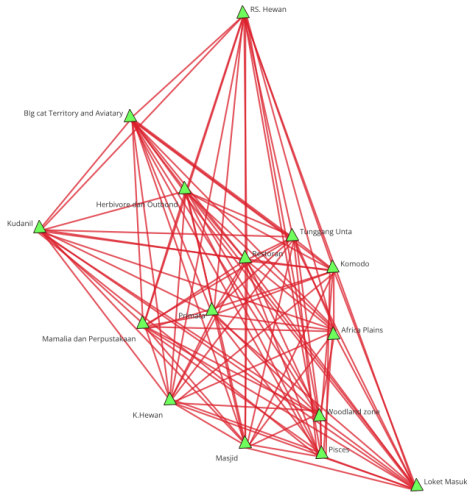
Pada Gambar 7, graf tersebut merupakan graf sederhana dan tak-berarah yang memiliki 15 simpul dengan masing-masing mempunyai 14 sisi yang terhubung dengan simpul lainnya. Urutan simpul dari 1 hingga 15 disusun berdasarkan representasi dari tiap wilayah sebagai berikut:

1. LM : Loket Masuk
2. P : Pisces
3. WZ : *Woodland zone*
4. AP : *Africa Plains*
5. KO : Komodo
6. M ; Masjid
7. KH : Karantina Hewan
8. MP : Mamalia dan Perpustakaan
9. PI : Primata
10. R : Restoran
11. TU : Tunggang Unta
12. KU : Kudanil
13. HO : *Herbivore and Outbond*
14. BA : Big cat Territory and Aviatary
15. RH : RS. Hewan

Graf ini merupakan graf hamilton karena dengan 15 titik, dimana titik ≥ 3 dan setiap titik memiliki derajat paling sedikit 15/2 sesuai dengan teorema Dirac.

C. Graf Berbobot

Setelah pembentukan graf sederhana selesai, diperlukan pembentukan graf berbobot yang merepresentasikan nilai numerik, seperti jarak, biaya, waktu, atau atribut lainnya yang relevan.graf pada setiap sisi (edge). Cakupan dari penelitian ini bertujuan untuk menentukan rute teroptimal atau rute dengan jarak yang tidak terlalu jauh dan tidak berulang, maka dari itu, saya memasukkan nilai jarak dalam meter pada setiap sisi di graf.



Gambar 9. Graf Kebun Binatang Surabaya
Sumber:hak cipta penulis

Nilai jarak pada setiap sisi merepresentasikan jarak dari titik satu ke titik lainnya sehingga informasi ini berguna untuk keberjalanan algoritma *Branch and Bound* dalam menentukan rute optimal. Pendataan jarak di tiap sisi disimpan sebagai berikut:

Tabel 1. Daftar Jarak antar Titik pada Graf Kebun Binatang Surabaya

Sisi	Jarak (meter)	Sisi	Jarak (meter)	Sisi	Jarak (meter)
LM -P	91.931	WZ-KU	316.085	KH--PI	91.539
LM-W	109.802	WZ-HO	244.178	KH--R	149.846
LM-AP	162.251	WZ-BA	331.410	KH--TU	192.986
LM-KO	219.654	WZ-RH	383.980	KH--KU	204.460
LM-M	166.976	AP-KO	56.711	KH-HO	199.389
LM-KH	245.098	AP-M	135.581	KH--BA	270.308
LM-MP	299.602	AP-KH	170.465	KH--RH	372.283
LM-PI	253.316	AP-MP	177.365	MP--PI	64.341
LM-R	269.174	AP-PI	114.218	MP--R	113.429
LM-TU	263.173	AP-R	108.153	MP--TU	164.845
LM-KU	430.802	AP-TU	96.121	MP--KU	133.682
LM-HO	357.260	AP-KU	295.473	MP-HO	134.190
LM-BA	438.359	AP-HO	194.135	MP--BA	193.891
LM-RH	475.017	AP-BA	275.197	MP-RH	308.561
P-WZ	29.108	AP-RH	310.815	PI--R	57.888
P-M	67.977	KO-M	183.477	PI--TU	105.077
P-AP	107.135	KO-KH	198.052	PI--KU	182.313
P-KH	149.920	KO-MP	186.481	PI-HO	119.281
P-KO	176.143	KO-PI	122.801	PI--BA	199.585
P-MP	208.475	KO-R	83.178	PI-RH	282.355
P-PI	166.776	KO-TU	49.251	R--TU	48.919
P-R	195.228	KO-KU	281.016	R--KU	198.010
P-TU	207.483	KO-HO	157.735	R-HO	87.698
P-KU	338.891	KO-BA	241.808	R--BA	174.602
P-HO	279.542	KO-RH	256.532	R-RH	232.081
P-BA	363.673	M-KH	84.493	TU--KU	239.681
P-RH	421.352	M--MP	148.005	TU-HO	109.603
WZ-AP	72.139	M--PI	128.292	TU--BA	190.042
WZ-KO	137.804	M--R	177.578	TU-RH	218.259
WZ-M	72.033	M--TU	203.829	KU-HO	141.650
WZ-KH	136.499	M--KU	281.061	KU--BA	135.134
WZ-MP	185.641	M--HO	246.108	KU-RH	282.089
WZ-PI	137.603	M--BA	326.046	HO--BA	84.970
WZ-R	160.156	M--RH	408.740	HO-RH	175.612
WZ-TU	171.254	KH--MP	75.644	BA-RH	144.728

D. Implementasi Algoritma Branch and Bound

Konsep Algoritma *Branch and Bound* saya implementasikan dalam bahasa pemrograman, yakni python untuk mendapatkan hasil yang cepat dan akurat tanpa harus dilakukan perhitungan secara manual.

Pertama, saya buat terlebih dahulu definisi fungsi untuk mendapatkan nilai bobot terminimum dan nilai terminimum kedua untuk perhitungan batasan bawah sebagai berikut:

```
def firstMinimum(adj, i):
    minimum = maxsize
    for j in range(len(adj)):
        if adj[i][j] < minimum and i != j:
            minimum = adj[i][j]
    return minimum

def secondMinimum(adj, i):
    first, second_minimum = maxsize, maxsize
    for j in range(len(adj)):
        if i != j:
            if adj[i][j] <= first:
                second_minimum = first
                first = adj[i][j]
            elif adj[i][j] <= second_minimum and adj[i][j] != first:
                second_minimum = adj[i][j]
    return second_minimum
```

Gambar 10. Algoritma Penentuan Bobot Terminimum
Sumber:hak cipta penulis

Selanjutnya, saya membuat fungsi rekursif untuk pencarian semua kemungkinan pada setiap simpul dalam graf mulai dari simpul 0 untuk menemukan semua kemungkinan jalur hingga terbentuk sirkuit hamilton yang kembali ke simpul 0 dengan bobot terminimum sebagai berikut:

```python
def Finalpath(curr_path, final_path, adj):
    for i in range(len(adj)):
        final_path[i] = curr_path[i]
    final_path[len(adj)] = curr_path[0]

def BranchandBoundRec(adj, curr_bound, curr_weight, level, curr_path, visited):
    global final_cost
    if level == len(adj):
        if adj[curr_path[level - 1]][curr_path[0]] != 0:
            curr_res = curr_weight + adj[curr_path[level - 1]][curr_path[0]]
            final_cost = min(final_cost, curr_res)
            Finalpath(curr_path, final_path, adj)  # Memperbarui jalur terbaik
        return

    for i in range(len(adj)):
        if adj[curr_path[level - 1]][i] != 0 and not visited[i]:
            temp = curr_bound
            curr_weight += adj[curr_path[level - 1]][i]

            if level == 1:
                curr_bound -= ((firstMinimum(adj, curr_path[level - 1]) +
                                firstMinimum(adj, i)) // 2)
            else:
                curr_bound -= ((secondMinimum(adj, curr_path[level - 1]) +
                                firstMinimum(adj, i)) // 2)

            if curr_bound + curr_weight < final_cost:
                curr_path[level] = i
                visited[i] = True
                BranchandBoundRec(adj, curr_bound, curr_weight, level + 1, curr_path, visited)

            curr_weight -= adj[curr_path[level - 1]][i]
            curr_bound = temp
            visited = [False] * len(visited)
            for j in range(level):
                if curr_path[j] != -1:
                    visited[curr_path[j]] = True
```

Gambar 11. Algoritma Penentuan Sirkuit Hamilton dengan Bobot Terminimum
Sumber:hak cipta penulis

Pada Gambar 11, fungsi BranchandBoundRec() merupakan fungsi rekursif untuk penentuan rute teroptimalnya dan jalur rutenya nanti akan disimpan dalam variabel list final_path menggunakan fungsi Finalpath(). Fungsi BranchandBoundRec() bisa memanggil fungsi Finalpath() berkali-kali untuk mengubah rutenya jika terdapat rute yang lebih minimum.

Setelah pembuatan program selesai, graf Kebun Binatang Surabaya, saya representasikan dengan *Adjacency* matriks. Urutan baris dan kolom pada matriks saya urutkan sesuai simpul yang telah saya definisikan dengan *index* 0,1,2,3,....n. Selain itu, saya membuat sedikit perubahan dimana jika sepasang simpul bertetanggaan, pada matriks akan saya isi dengan bobot jarak pada sisi yang dibentuk sepasang sisi tersebut. Tentunya, elemen matriks pada simpul yang sama akan saya isi dengan 0, karena tidak ada sisi yang mengarah kepada dirinya sendiri.

```python
adj = [
    [0.0, 91.931, 109.802, 162.251, 219.654, 166.976, 245.098, 299.602, 253.316, 269.174, 263.173, 430.802, 357.260, 438.359, 475.017],
    [91.931, 0.0, 29.108, 67.977, 107.135, 149.920, 176.143, 208.475, 166.776, 195.228, 207.483, 338.891, 279.542, 363.673, 421.352],
    [109.802, 29.108, 0.0, 72.139, 137.804, 72.033, 136.499, 185.641, 137.603, 160.156, 171.254, 316.085, 244.178, 331.410, 383.980],
    [162.251, 107.135, 72.139, 0.0, 56.711, 135.581, 170.465, 177.365, 114.218, 108.153, 96.121, 295.473, 194.135, 275.197, 310.815],
    [219.654, 176.143, 137.804, 56.711, 0.0, 183.477, 198.052, 186.481, 122.801, 83.178, 49.251, 281.016, 157.735, 241.808, 256.532],
    [166.976, 67.977, 72.033, 135.581, 183.477, 0.0, 84.493, 148.005, 128.292, 177.578, 203.82, 281.061, 246.108, 326.046, 408.740],
    [245.098, 149.920, 136.499, 170.465, 198.052, 84.493, 0.0, 75.644, 91.539, 149.846, 192.986, 204.460, 199.389, 270.308, 372.283],
    [299.602, 208.475, 185.641, 177.365, 186.481, 148.005, 75.644, 0.0, 64.341, 113.429, 164.845, 133.682, 134.190, 193.891, 308.561],
    [253.316, 166.776, 137.603, 114.21, 122.801, 128.292, 91.539, 64.341, 0.0, 57.888, 105.077, 182.313, 119.281, 199.585, 282.355],
    [269.174, 195.228, 160.156, 108.153, 83.178, 177.578, 149.846, 113.429, 57.888, 0.0, 48.919, 198.010, 87.69, 174.602, 232.881],
    [263.173, 207.483, 171.254, 96.121, 49.251, 203.829, 192.986, 164.845, 105.077, 48.919, 0.0, 239.681, 109.603, 190.042, 218.259],
    [430.802, 338.891, 316.085, 295.473, 281.016, 281.061, 204.460, 133.682, 182.313, 198.010, 239.681, 0.0, 141.650, 135.134, 282.089],
    [357.260, 279.542, 244.178, 194.135, 157.735, 246.108, 199.389, 134.190, 119.281, 87.698, 109.603, 141.650, 0.0, 84.970, 175.612],
    [438.359, 363.673, 331.410, 275.197, 241.808, 326.046, 270.308, 193.891, 199.585, 174.602, 190.042, 135.134, 84.970, 0.0, 175.612],
    [475.017, 421.352, 383.980, 310.815, 256.532, 408.740, 372.283, 308.561, 282.355, 232.081, 218.259, 282.089, 175.612, 144.728, 0.0]
]
```

Gambar 12. Representasi Graf dengan *Adjacency* Matriks
Sumber:hak cipta penulis

Matriks ketetanggan tersebut telah disimpan dalam variabel. Selanjutnya, dilakukan inisialisasi program dan pemanggilan fungsi rekursif, lalu hasil rute serta total bobot terminimumnya akan ditampilkan.

```python
final_cost = maxsize
final_path = [-1] * (len(adj) + 1)
curr_path = [-1] * (len(adj) + 1)
visited = [False] * len(adj)
curr_bound = 0
for i in range(len(adj)):
    curr_bound += (firstMinimum(adj, i) + secondMinimum(adj, i))

if (curr_bound % 2 == 1):
    curr_bound = curr_bound // 2
else:
    curr_bound = curr_bound// 2 + 1

visited[0] = True
curr_path[0] = 0
BranchandBoundRec(adj, curr_bound, 0, 1, curr_path, visited)

print("Total Jarak Terminimum :", final_cost)
print("Rute Teroptimal:", end=' ')
for i in range(len(adj) + 1):
    print(final_path[i], end=' ')
```

Gambar 13. Inisialisasi Program
Sumber:hak cipta penulis

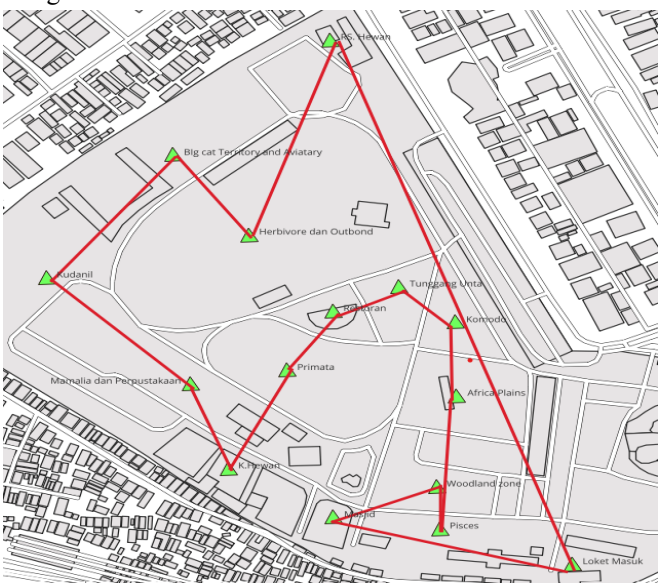
Selesai program dijalankan, akan ditampilkan hasil keluarannya sebagai berikut:

```
PS C:\Users\ASUS\Documents\Matdis> python -u "c:\Users\ASUS\Documents\Matdis\bound.py"
Total Jarak Terminimum : 1413.8429999999996
Rute Teroptimal: 0 5 2 1 3 4 10 9 8 6 7 11 13 12 14 0
PS C:\Users\ASUS\Documents\Matdis>
```

Gambar 14. Keluaran Program
Sumber:hak cipta penulis

Pada Gambar 13, dapat dilihat bahwa pada rute optimal terbentuk sirkuit hamilton yang dimulai dari simpul 0 atau simpul awal, lalu melewati semua simpul tepat sekali hingga nantinya kembali lagi ke titik awal. Total jarak terminimum merupakan hasil yang didapat selama pencarian rekursif yang memiliki nilai 1413.843 meter.

Rute tersebut akan direpresentasikan dengan graf berbobot sebagai berikut:



Gambar 15. Representasi Graf pada Rute Teroptimal
Sumber:hak cipta penulis

Pada Gambar 14, ditampilkan rute terefisien dalam

pemetaan Kebun Binatang Surabaya, di mana rute tersebut telah melewati semua titik tepat sekali dan kembali ke titik awal dengan total jarak terminimum. Dengan demikian, didapatkan rute terefisien yang tentunya bisa membantu dalam manajemen operasional pihak kebun binatang dan juga bisa memberikan pengalaman yang terbaik kepada pengunjung.

IV. Pembahasan

Diimplementasikan algoritma *Branch and Bound* dalam bahasa pemrograman untuk menemukan sirkuit hamilton pada graf dengan bobot terminimum secara efisien dengan hasil seoptimal mungkin. Input pada pemrograman ini berupa *adjacency* matriks yang merupakan representasi dari suatu graf dengan modifikasi isi elemen pada tiap matriks merupakan berupa nilai jarak (dalam meter) pada tiap sisi untuk sepasang simpul dan titik yang dipasangkan dengan titik sendiri bernilai 0 karena graf berbentuk sederhana jadi tidak ada suatu lingkaran atau *loop* pada titik tertentu.

Ada banyak cara dalam merepresentasikan suatu graf dengan struktur data lainnya, namun saya memilih matriks ketetanggan karena struktur data ini sangat cukup untuk merepresentasikan graf yang sedang diteliti. Tentu, jika menggunakan struktur data lain, maka pemrograman juga perlu diubah sesuai dengan pengoperasian struktur data tersebut.

Tantangan dari penyelesaian *Traveling Salesperson Problem* (TSP), berkaitan erat dengan eksplorasi semua sirkuit Hamilton dengan jumlah sirkuit yang perlu diperiksa berkembang secara cepat, sebanyak (n-1)! sirkuit. Maka dari itu, hasil dari proses pemrograman ini juga tidak sangat cepat seiring dengan banyaknya simpul. Namun, pada penelitian ini dengan 15 simpul, hasil pemrograman cukup cepat dengan rata-rata waktu eksekusi 2 sampai 2,3 detik .

Adapun, optimalisasi pada pemograman ini, yakni bisa menerima graf dengan struktur data yang berbeda-beda yang tentunya bisa memenuhi kebutuhan setiap pengguna.

V. Kesimpulan

Melalui eksplorasi dalam penerapan aplikasi graf dan implementasi algoritma *Branch and Bound* dalam menyelesaikan Traveling Salesman Problem (TSP) dalam konteks penentuan rute teroptimal di kebun binatang, penelitian ini memberikan hasil yang cukup memuaskan. Metode ini berhasil memberikan solusi yang efisien dan efektif dalam mencari rute terpendek yang menghubungkan berbagai titik penting di dalam kebun binatang. Pendekatan *Branch and Bound* juga dapat memberikan solusi yang optimal dengan walaupun dengan ukuran matriks ketetanggan yang cukup besar.

Selain itu, penelitian ini bisa juga menerapkan algoritma lainnya selain *Branch and Bound karena* dalam penyelesaian TSP terdapat banyak sekali caranya. Namun, tetap yang diprioritaskan pertama adalah hasil solusi yang optimal dan efisien.

VI. Ucapan Terima kasih

Pertama-tama, Saya ucapkan rasa syukur kepada Tuhan Yang Maha Esa atas berkat-Nya dan keberkahan-Nya saya bisa menyelesaikan penulisan makalah ini. Tidak lupa juga, Saya ucapkan terima kasih kepada Ibu Dr. Nur Ulfa Maulidevi, S.T, M.Sc., atas bimbingannya selama pembelajaran mata kuliah IF2120 Matematika Diskrit. Tentunya, penyelesaian makalah ini bukan karena usaha Saya sendiri, tetapi juga pihak-pihak lain yang telah mendukung saya selama penulisan makalah. Terakhir, Saya berharap penelitian ini bisa bermanfaat untuk para pembaca dan bisa menjadi panduan yang bermanfaat bagi para pengelola kebun binatang dan pihak terkait untuk meningkatkan efisiensi operasional dan memberikan pengalaman terbaik bagi pengunjung.

References

[1] Rosen, K.H., "Discrete Mathematics and Its Applications," 7th Ed. New York, NY: McGraw-Hill Education, 2011. pp. 641–717.

[2] *Rinaldi Munir, "Graf (Bagian 1)," 2023. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/19-Graf-Bagian1-2023.pdf. [Diakses pada 30 November 2023]*

[3] *Rinaldi. Munir, "Graf Bagian 3 - 2023," ITB, Indonesia, 2023. [Online]. Available:https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/21-Graf-Bagian3-2023.pdf [Diakses pada 30 November 2023]*

[4] E. Lehman, T. Leighton, and A. Meyer, Mathematics for Computer Science. Cambridge, MA: Electrical Engineering and Computer Science Press, 2013. pp.121-196

[5] GeeksforGeeks, "Traveling Salesman Problem using Branch and Bound,".[Online].Available: https://www.geeksforgeeks.org/traveling-salesman-problem-using-branch-and-bound-2/. [*Diakses pada 1 Desember 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, 12 Desember 2023

Muhammad Yusuf Rafi/13522009