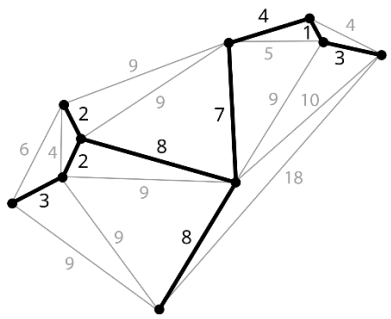
Optimasi Penjadwalan Aktivitas Harian Berbasis Waktu Tempuh Minimum Menggunakan Algoritma Minimum Spanning Tree

Amira Izani- 13523143[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]izani1807@gmail.com, 13523143@std.stei.itb.ac.id

***Abstract*— Penelitian ini membahas penerapan algoritma Minimum Spanning Tree (MST), khususnya algoritma Prim, untuk mengoptimalkan rute perjalanan dengan waktu tempuh minimum dalam aktivitas sehari-hari. Representasi graf berbentuk adjacency matrix digunakan untuk menggambarkan hubungan antar lokasi, seperti Rumah, Pasar, Restoran, Danau, dan Apotek. Hasil implementasi menunjukkan bahwa MST dapat menghasilkan rute optimal, dengan urutan "Rumah - Pasar - Danau - Apotek - Restoran" dan total waktu tempuh minimum sebesar 66 menit. Meskipun MST terbukti efisien, penelitian ini juga menyoroti keterbatasannya, terutama ketika diperlukan rute yang tidak mengulang simpul. Oleh karena itu, lintasan Euler direkomendasikan untuk menyempurnakan solusi dalam kasus tertentu. Penelitian ini menunjukkan potensi MST dalam membantu perencanaan perjalanan harian dan memberikan dasar untuk penelitian lebih lanjut yang mengkombinasikan MST dengan teori graf lainnya.**

***Keywords*— Minimum Spanning Tree, algoritma Prim, lintasan Euler, optimasi perjalanan, graf berbobot.**

I. PENDAHULUAN



Gambar 1.1 Minimum Spanning Tree
(sumber: https://en.wikipedia.org/wiki/Minimum_spanning_tree)

Graf adalah salah satu topik penting dalam matematika diskrit yang mempelajari cara menghubungkan berbagai elemen atau titik dengan garis. Dalam kehidupan sehari-hari, konsep graf dapat digunakan untuk memecahkan masalah praktis, seperti menentukan rute perjalanan tercepat, mengatur jaringan transportasi, atau bahkan menyusun jadwal kegiatan. Di zaman sekarang, efisiensi waktu menjadi hal yang sangat penting. Oleh karena itu, kita memerlukan metode untuk menyelesaikan masalah tersebut secara efektif. Salah satu metode yang bisa digunakan adalah algoritma Minimum Spanning Tree (MST), yang membantu kita menemukan jalur paling efisien untuk menghubungkan titik-titik tersebut dengan waktu tempuh minimum.

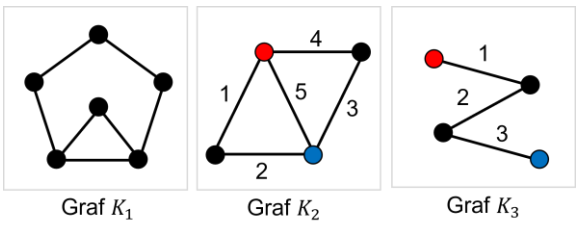
Dalam kehidupan nyata, penerapan teori graf sangat membantu menyelesaikan berbagai masalah. Misalnya, ketika kita ingin mengunjungi beberapa tempat dalam satu hari, graf dapat membantu menentukan urutan tempat yang harus dikunjungi agar perjalanan menjadi lebih cepat. Algoritma MST, seperti algoritma Prim dan Kruskal, digunakan untuk mencari jalur terpendek yang menghubungkan semua titik. Dengan cara ini, kita bisa menghemat waktu dan menjalani aktivitas dengan lebih terorganisir.

Penelitian ini bertujuan untuk mengaplikasikan algoritma MST dalam kehidupan sehari-hari, terutama untuk merencanakan aktivitas harian dengan waktu tempuh minimum. Penelitian akan menggunakan bahasa pemrograman seperti Python atau Java untuk mengimplementasikan algoritma ini. Algoritma Prim dan Kruskal akan digunakan untuk menentukan jalur yang paling optimal. Selain itu, penelitian ini juga akan mengevaluasi sejauh mana algoritma ini dapat meningkatkan efisiensi dan produktivitas.

Dari berbagai studi yang telah dilakukan, algoritma Minimum Spanning Tree terbukti efektif dalam berbagai bidang, seperti perencanaan rute transportasi, distribusi logistik, dan pengaturan jadwal. Penelitian-penelitian sebelumnya menunjukkan bahwa MST mampu mengurangi waktu tempuh dan biaya secara signifikan, sehingga meningkatkan efisiensi operasional. Berdasarkan hasil kajian ini, algoritma MST memiliki potensi besar untuk diterapkan dalam kehidupan sehari-hari, terutama dalam membantu kita merencanakan aktivitas dengan lebih baik dan produktif.

II. LANDASAN TEORI

1. Teori Graf



Gambar 2.1 Graf (sumber:
https://mathcyber1997.com/teori-graf/)

Teori graf dimulai pada abad ke-18 oleh Leonhard Euler, yang memecahkan “Masalah Tujuh Jembatan Königsberg” pada tahun 1736. Euler membuktikan bahwa tidak mungkin melewati tujuh jembatan tanpa mengulangi jalur. Solusi ini menjadi dasar teori graf. Sejak itu, konsep ini terus berkembang dengan kontribusi dari matematikawan seperti Kirchhoff dan Hamilton.

Graf adalah pasangan himpunan, di mana adalah simpul dan adalah sisi yang menghubungkan simpul (Munir, 2005). Simpul mewakili objek, dan sisi adalah hubungan antar simpul. Contohnya, di jaringan sosial, simpul adalah pengguna, dan sisi menunjukkan hubungan pertemanan.

Berdasarkan struktur, ada:

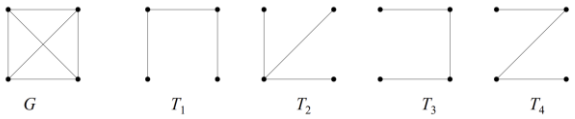
1. Graf sederhana: Tidak ada loop atau sisi ganda.
2. Graf tak-sederhana: Mengandung loop atau sisi ganda.

Berdasarkan arah sisi, ada:

1. Graf tak berarah: Sisi tanpa arah, seperti jaringan telepon.
2. Graf berarah: Sisi dengan arah, seperti peta jalan satu arah.

Ada juga graf berbobot, di mana setiap sisi memiliki nilai, misalnya jarak antar titik pada peta. Graf ini berguna untuk mencari jalur terpendek atau rute paling efisien.

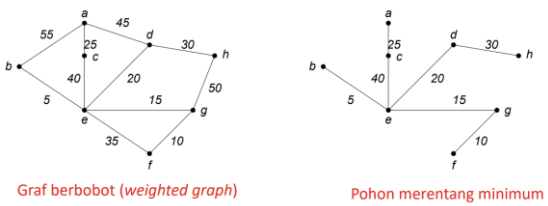
2. Spanning Tree



Gambar 2.2 Spanning Tree (Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/23-Pohon-Bag1-2024.pdf)

Pohon merentang dari sebuah graf terhubung adalah upagraf merentang yang berbentuk pohon. Upagraf merentang, atau spanning subgraph, merupakan bagian dari graf yang mencakup semua simpul di dalam graf tersebut. Untuk mendapatkan pohon merentang dari graf, sirkuit di dalam graf perlu diputus. Setiap graf terhubung minimal memiliki satu pohon merentang. Sebaliknya, jika graf tidak terhubung dan memiliki komponen, maka graf tersebut akan memiliki buah hutan merentang yang disebut sebagai hutan merentang atau spanning forest.

3. Algoritma Minimum Spanning Tree



Gambar 2.3 Minimum Spanning Tree (Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/23-Pohon-Bag1-2024.pdf)

Graf terhubung berbobot dapat memiliki lebih dari satu pohon merentang. Dari berbagai pohon merentang tersebut, salah satu yang memiliki total bobot terkecil disebut pohon merentang minimum atau Minimum Spanning Tree (MST). Pohon merentang minimum digunakan untuk mencari solusi optimal dalam berbagai aplikasi, seperti perencanaan jaringan telekomunikasi, distribusi logistik, dan optimasi rute transportasi. Algoritma seperti Prim dan Kruskal sering digunakan untuk menentukan MST dengan efisien. Algoritma Prim bekerja dengan membangun MST secara bertahap dari satu simpul awal, sementara algoritma Kruskal memilih sisi-sisi dengan bobot terkecil secara berurutan, memastikan tidak ada sirkuit yang terbentuk. Dengan MST, masalah optimasi yang kompleks dapat diselesaikan dengan cara yang sederhana dan efisien, sehingga sangat berguna dalam dunia nyata. Ada dua Algoritma yang biasa digunakan untuk mencari minimum spanning tree dari weighted graph:

1. Algoritma Prim

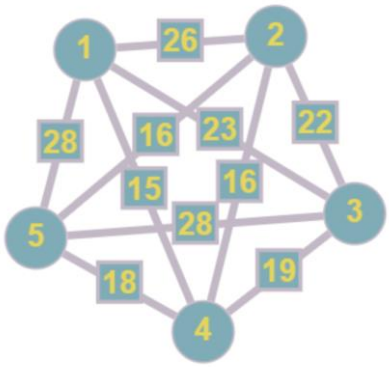
Algoritma Prim bekerja dengan memilih satu simpul sebagai titik awal. Dari simpul ini, sisi dengan bobot terkecil yang terhubung ke simpul lain dipilih dan ditambahkan ke dalam pohon . Proses ini berlanjut dengan menambahkan sisi dengan bobot terkecil yang terhubung ke simpul-simpul di dalam , asalkan sisi tersebut tidak membentuk sirkuit. Proses ini diulang sampai semua simpul dalam graf terhubung. Algoritma Prim cocok digunakan pada graf yang padat atau memiliki banyak sisi karena prosesnya dimulai dari satu simpul dan berkembang secara lokal.

2. Algoritma Kruskal

Algoritma Kruskal dimulai dengan mengurutkan semua sisi dalam graf berdasarkan bobotnya dari yang terkecil hingga terbesar. Pada awalnya, pohon masih kosong. Selanjutnya, sisi dengan bobot terkecil dipilih dan ditambahkan ke , asalkan penambahan tersebut tidak membentuk sirkuit. Proses ini diulang hingga pohon memiliki sisi, di mana adalah jumlah simpul dalam graf. Algoritma Kruskal sangat efektif untuk graf yang jarang atau memiliki sedikit sisi, karena pendekatannya tidak bergantung pada simpul awal, tetapi pada urutan sisi yang dipilih.

III. IMPLEMENTASI

1. Mengubah Graf Berbobot Menjadi Adjacency Matrix



Gambar 3.1 Data waktu tempuh dalam bentuk Graf Berbobot

Pada kasus ini, data waktu tempuh antar lokasi seperti Rumah, Danau, Restoran, Pasar, dan Apotek digunakan sebagai dasar untuk implementasi algoritma graf. Data ini menggambarkan waktu tempuh (dalam menit) antar lokasi yang direpresentasikan dalam bentuk tabel. Untuk mempermudah implementasi algoritma, data tersebut diubah menjadi adjacency matrix, yaitu sebuah matriks dua dimensi di mana setiap baris dan kolom merepresentasikan lokasi, dan nilai pada matriks menunjukkan waktu tempuh antar lokasi tersebut.

	Rumah	Danau	Restoran	Pasar	Apotek
Rumah	0	26	23	15	28
Danau	26	0	22	16	16
Restoran	23	22	0	19	28
Pasar	15	16	19	0	18
Apotek	28	16	28	18	0

Sebagai contoh, elemen matriks matrix[0][1]=26 menunjukkan waktu tempuh dari Rumah ke Danau adalah 26 menit. Karena graf ini tidak berarah, waktu tempuh dari Danau ke Rumah juga sama. Representasi adjacency matrix mempermudah penerapan algoritma seperti Minimum Spanning Tree (MST) untuk mencari rute dengan waktu tempuh paling singkat. Data ini diambil dari aktivitas sehari-hari, seperti perjalanan dari Rumah ke berbagai tempat, sehingga hasilnya dapat membantu merencanakan rute yang lebih efisien.

2. Implementasi Algoritma Minimum Spanning Tree

Program ini dibuat menggunakan Python dan modul sys untuk membantu mencari solusi terbaik dalam menyelesaikan masalah jaringan atau rute dengan algoritma Prim. Algoritma Prim digunakan untuk menemukan Minimum Spanning Tree (MST), yaitu cara menghubungkan semua lokasi (simpul) dalam graf dengan bobot total terkecil. Bobot di sini bisa berupa waktu tempuh, jarak, atau biaya.

```python
import sys

class Graph:
    def __init__(self, vertices):
        self.V = vertices
        self.graph = [[0 for column in range(vertices)]
                      for row in range(vertices)]

    def printMST(self, parent):
        total_weight = 0
        mst = {}
        for i in range(1, self.V):
            total_weight += self.graph[i][parent[i]]
            mst.setdefault(parent[i], []).append(i)
        return total_weight, mst

    def minKey(self, key, mstSet):
        min_val = sys.maxsize
        min_index = -1
        for v in range(self.V):
            if key[v] < min_val and not mstSet[v]:
                min_val = key[v]
                min_index = v
        return min_index
```

Program ini terdiri dari beberapa fungsi utama. Pertama, init__ digunakan untuk menyiapkan graf dan jumlah lokasinya. Lalu, printMST menghitung total bobot perjalanan dan mencatat hubungan antar lokasi yang membentuk pohon dengan bobot terkecil. Ada juga minKey, yang berfungsi memilih lokasi berikutnya dengan bobot terkecil untuk dimasukkan ke dalam pohon.

```python
    def primMST(self):
        key = [sys.maxsize] * self.V
        parent = [None] * self.V
        key[0] = 0
        mstSet = [False] * self.V
        parent[0] = -1

        for cout in range(self.V):
            u = self.minKey(key, mstSet)
            mstSet[u] = True
            for v in range(self.V):
                if self.graph[u][v] > 0 and not mstSet[v] and key[v] > self.graph[u][v]:
                    key[v] = self.graph[u][v]
                    parent[v] = u

        return self.printMST(parent)

    def getTraversalOrder(self, mst, start):
        visited = set()
        order = []

        def dfs(node):
            if node not in visited:
                visited.add(node)
                order.append(node)
                for neighbor in mst.get(node, []):
                    dfs(neighbor)

        dfs(start)
        return order
```

primMST, menjalankan algoritma Prim. Algoritma ini bekerja dengan memulai dari satu lokasi awal (misalnya Rumah) dan secara bertahap memilih lokasi berikutnya dengan bobot terkecil yang belum dikunjungi, hingga semua lokasi terhubung.

Setelah itu, getTraversalOrder digunakan untuk menentukan urutan perjalanan berdasarkan hasil MST. Fungsi ini memastikan perjalanan dilakukan secara efisien tanpa kembali ke lokasi yang sama.

```
locations = ['Rumah', 'Danau', 'Restoran', 'Pasar', 'Apotek']
adj_matrix_random = [
    [0, 26, 23, 15, 28],  # Rumah
    [26, 0, 22, 16, 16],  # Danau
    [23, 22, 0, 19, 28],  # Restoran
    [15, 16, 19, 0, 18],  # Pasar
    [28, 16, 28, 18, 0],  # Apotek
]


g = Graph(len(locations))
g.graph = adj_matrix_random

# algoritma prim
total_weight, mst = g.primMST()

# traversal order from MST
traversal_order_indices = g.getTraversalOrder(mst, start=0)
traversal_order = [locations[i] for i in traversal_order_indices]

# Output
print("Urutan koneksi destinasi:")
print(" - ".join(traversal_order))
print(f"\nTotal Waktu Tempuh Minimum: {total_weight} menit")
```

```
Urutan koneksi destinasi:
Rumah - Pasar - Danau - Apotek - Restoran

Total Waktu Tempuh Minimum: 66 menit
```

Output program ini menunjukkan rute perjalanan yang paling efisien dari Rumah ke semua lokasi lainnya. Urutan koneksi destinasi yang ditampilkan adalah "Rumah - Pasar - Danau - Apotek - Restoran", yang berarti perjalanan dimulai dari Rumah, kemudian menuju Pasar, dilanjutkan ke Danau, Apotek, dan akhirnya Restoran. Urutan ini dipilih berdasarkan jalur dengan waktu tempuh terkecil di setiap langkah.

Selain urutan perjalanan, program juga menampilkan Total Waktu Tempuh Minimum, yaitu 66 menit. Angka ini adalah total waktu yang dibutuhkan untuk mengunjungi semua lokasi sesuai dengan rute yang dihasilkan.

IV. ANALISIS DAN PEMBAHASAN

Hasil implementasi algoritma Prim menunjukkan bahwa MST dapat secara efektif mengoptimalkan rute perjalanan. Dengan representasi graf berbobot dalam bentuk adjacency matrix, algoritma Prim menghasilkan rute perjalanan dengan total waktu tempuh minimum sebesar 66 menit. Urutan perjalanan yang diperoleh adalah "Rumah - Pasar - Danau - Apotek - Restoran." Rute ini mencerminkan efisiensi dalam menghubungkan semua lokasi dengan bobot terkecil, tanpa membentuk siklus.

Namun, meskipun MST efektif untuk menghubungkan semua simpul, ada keterbatasan jika perjalanan membutuhkan lintasan yang tidak mengulangi simpul. Dalam situasi ini, lintasan Euler dapat digunakan untuk melengkapi perencanaan rute. Misalnya, jika diperlukan perjalanan yang tidak boleh melewati simpul yang sama, MST saja tidak cukup. Kombinasi MST dan lintasan Euler dapat memberikan solusi yang lebih menyeluruh dalam mengatasi masalah perjalanan kompleks.

Penggunaan MST pada aktivitas sehari-hari, seperti perjalanan dari Rumah ke berbagai lokasi, membuktikan bahwa algoritma ini mampu menghemat waktu dan meningkatkan efisiensi. Penelitian ini juga menunjukkan bahwa teori graf lainnya dapat digunakan untuk menyempurnakan hasil, khususnya pada situasi dengan batasan rute tertentu.

Algoritma Minimum Spanning Tree (MST) tidak hanya berguna dalam penelitian, tetapi juga memiliki banyak manfaat dalam kehidupan sehari-hari. Salah satu contohnya adalah perencanaan rute perjalanan untuk mengunjungi beberapa tempat dengan waktu tempuh minimum, seperti yang dilakukan dalam penelitian ini. Di luar itu, MST juga sering digunakan dalam membangun jaringan telekomunikasi, di mana kabel harus dipasang dengan biaya serendah mungkin untuk menghubungkan semua titik.

Selain itu, MST membantu dalam perencanaan jaringan listrik agar seluruh rumah di suatu area dapat terhubung dengan efisien tanpa menghabiskan biaya kabel yang besar. Di bidang logistik, MST juga digunakan untuk menentukan rute pengiriman barang yang paling efisien, sehingga perusahaan jasa kurir dapat menghemat waktu dan biaya operasional. Bahkan, MST dapat diterapkan dalam analisis data, misalnya untuk mengelompokkan data secara lebih terstruktur dan efisien.

V. KESIMPULAN

Penelitian ini berhasil mengimplementasikan algoritma Prim untuk membangun Minimum Spanning Tree (MST) dalam perencanaan rute perjalanan sehari-hari. Hasilnya menunjukkan bahwa MST dapat menghasilkan rute perjalanan optimal dengan waktu tempuh minimum, sehingga efisiensi aktivitas dapat ditingkatkan. Meskipun demikian, MST memiliki keterbatasan dalam kasus tertentu, terutama jika perjalanan tidak boleh mengulangi simpul yang sama. Oleh karena itu, lintasan Euler disarankan untuk melengkapi MST dalam menyelesaikan masalah rute yang lebih kompleks. Penelitian selanjutnya diharapkan dapat mengeksplorasi kombinasi MST dengan lintasan Euler untuk menyelesaikan masalah optimasi perjalanan dengan batasan yang lebih beragam.

Ke depan, penelitian ini dapat dikembangkan untuk skala yang lebih besar. Misalnya, MST dapat diterapkan pada pengaturan jadwal transportasi umum, optimasi rute drone untuk pengiriman barang, atau perancangan sistem distribusi air di perkotaan. Kombinasi MST dengan metode lain dari teori graf akan membuka peluang solusi yang lebih efisien untuk berbagai kebutuhan nyata di masyarakat.

VI. PENUTUP

Puji dan syukur dipanjatkan ke hadirat Allah SWT atas rahmat dan karunia-Nya sehingga penulis dapat menyusun makalah ini. Penulis juga menyampaikan terima kasih kepada Pak Arrival Dwi Sentosa, S.Kom., M.T., selaku dosen mata kuliah IF1220 Matematika Diskrit, yang telah memberikan bimbingan selama semester ini. Selain itu, penulis juga berterima kasih kepada Pak Rinaldi Munir yang telah membangun dan mengelola situs web Matematika Diskrit. Situs ini sangat membantu para mahasiswa IF1220 dalam mengakses materi kuliah serta berbagai keperluan yang menunjang pembelajaran Matematika Diskrit.

REFERENCES

[1] Telkom University, "Teori graf: Sejarah, manfaat, dan aplikasinya," [Online]. Tersedia: https://surabaya.telkomuniversity.ac.id/teori-graf-sejarah-manfaat-dan-aplikasinya/. [Diakses: 08-Jan-2025].

[2] H. G. Fadli, "Studi Minimum Spanning Tree dengan Algoritma Prim dan Kruskal," Program Studi Teknik Informatika, Institut Teknologi Bandung, Bandung, Indonesia.

[3] R. Munir, *Pohon (Bag. 1)*, Institut Teknologi Bandung, Bandung, Indonesia, 2024. [Online]. Tersedia: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/23-Pohon-Bag1-2024.pdf. [Diakses: 08-Jan-2025].

[4] R. Munir, *Graf (Bag. 3)*, Institut Teknologi Bandung, Bandung, Indonesia, 2024. [Online]. Tersedia: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/22-Graf-Bagian3-2024.pdf [Diakses: 08-Jan-2025].

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.

Jatinangor, 8 Januari 2025

Amira Izani-13523143