Optimasi Reduksi Jaringan Kabel Listrik Perumahan Depok Indah I Menggunakan Algoritma Kruskal

Uktoriko dan Kirso

Universitas Nusa Mandiri
Jl. Kramat Raya No.18 Jakarta Pusat
E-mail: 14230028@nusamandiri.ac.id, 14230033@nusamandiri.ac.id

Abstrak

Algoritma Kruskal digunakan untuk membentuk Minimum Spanning Tree (MST), yaitu suatu metode untuk menghubungkan seluruh titik dalam jaringan dengan total bobot minimum tanpa membentuk siklus. Penelitian ini mengaplikasikan algoritma Kruskal untuk menentukan jalur jaringan listrik yang efisien di Perumahan Depok Indah I. Data berupa koordinat titik dan jarak antar titik dijadikan bobot graf. Proses dilakukan melalui perhitungan manual dan implementasi menggunakan bahasa Python. Hasil algoritma Kruskal kemudian dibandingkan dengan penelitian sebelumnya yang menggunakan algoritma Prim. Kruskal menghasilkan total panjang kabel 115 km, lebih pendek dibandingkan Prim yang membutuhkan 185 km, sehingga terjadi penghematan sebesar 70 km atau sekitar 37,84%. Kesimpulannya, algoritma Kruskal lebih efektif dalam meminimalkan panjang kabel dan dapat menjadi alternatif efisien dalam perencanaan jaringan listrik.

Kata kunci : Kruskal, Minimum Spanning Tree, Jaringan Listrik, Optimasi Graf, Efisiensi Kabel

Pendahuluan

Minimum Spanning Tree (MST) bertujuan untuk membentuk sebuah himpunan bagian dari graf yang menghubungkan seluruh simpul, dengan total bobot sisi seminimal mungkin dan tanpa membentuk siklus [1]. Dimana MST merupakan konsep dasar dalam teori grafik yang memiliki aplikasi signifikan di berbagai domain, termasuk desain jaringan, pengelompokan, dan perutean. Di antara algoritma yang digunakan untuk menemukan MST, algoritma Kruskal menonjol karena efisiensinya dan kesederhanaannya. Algoritma ini beroperasi dibawah strategi greedy dengan membangun pohon secara bertahap dengan menambahkan sisi dengan bobot paling sedikit sambil memastikan bahwa siklus tidak terbentuk [2]. Penerapan MST berkaitan erat dalam kehidupan salah satu contohnya perencanaan jaringan listrik [3] yang dapat meminimalisir penggunaan kabel akan membuat biaya yang minimum. Disisi lain karakteristik algoritma Kruskal untuk meminimalkan overhead komputasi secara efektif [4] [5].

Prinsip-prinsip yang diatur oleh algoritma Kruskal dan Prim memiliki beragam aplikasi praktis. Misalnya, algoritma ini dapat digunakan dalam mengoptimalkan jaringan transportasi, seperti penelitian [6] yang memanfaatkan algoritma Kruskal untuk menentukan rute terpendek di antara berbagai objek wisata, yang menyoroti kemampuannya dalam mengurangi biaya perjalanan dan meningkatkan aksesibilitas. Demikian pula, penelitian lain dalam bidang teknik listrik, yang menunjukkan bagaimana algoritma tersebut dapat membantu dalam konfigurasi ulang jaringan pengumpan yang optimal dalam berbagai kondisi beban, sehingga meningkatkan stabilitas operasional dan mengurangi kehilangan daya [7]. Pada penelitian terkait penyusunan jalur pipa menunjukan kinerja algoritma kruskal, sollin dan prim menghasilkan nilai yang sama dan dapat melakukan penghematan sebesar 717m pipa [8] dan penelitian [9] yang menerapkan algoritma kruskal dan dapat melakukan penghematan sebesar 30,61% , berbeda dengan penelitian [10] dimana hanya menggunakan algoritma prim, meski demikian hasilnya dapat menghemat sepanjang 3.870 meter.

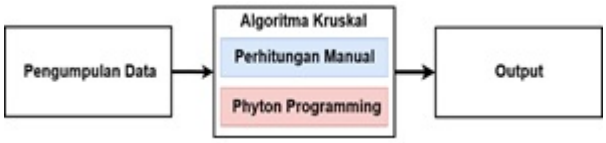
Pada penelitian ini akan dilakukan investigasi terkait penerapan algoritma Kruskal dalam menentukan jalur optimum jaringan listrik di Perumahan Depok Indah I sebagai bahan pertimbangan dalam perencanaan pemasangan kabel listrik. Sebelumnya, penelitian dilakukan dengan menggunakan algoritma Prim, yang menghasilkan total kebutuhan kabel sepanjang 185 km [3]. Harapan dalam penelitian ini yaitu diperolehnya nilai kebutuhan kabel yang lebih minimum dibanding penelitian sebelum-

DOI : http://dx.doi.org/10.32409/jikstik.24.2.3808

nya, sehingga biaya yang efisiensi lebih optimum.

Metode Penelitian

Penelitian ini menggunakan algoritma Kruskal untuk menentukan jalur optimal jaringan listrik di Perumahan Depok Indah I. Data yang digunakan berupa titik-titik lokasi (vertex) serta jarak antar titik yang dijadikan sebagai bobot pada edge (sisi) dalam graf. Algoritma Kruskal diterapkan untuk membentuk MST, yaitu graf berhubungan tanpa siklus yang menghubungkan seluruh titik/vertex dengan total panjang kabel minimum. Hasil yang diperoleh akan dibandingkan dengan penelitian sebelumnya oleh Tania, yang menggunakan algoritma Prim dan menghasilkan kebutuhan kabel sepanjang 185 km. Gambar 1 menyajikan visualisasi sederhana terkait metode yang digunakan dalam penelitian ini.

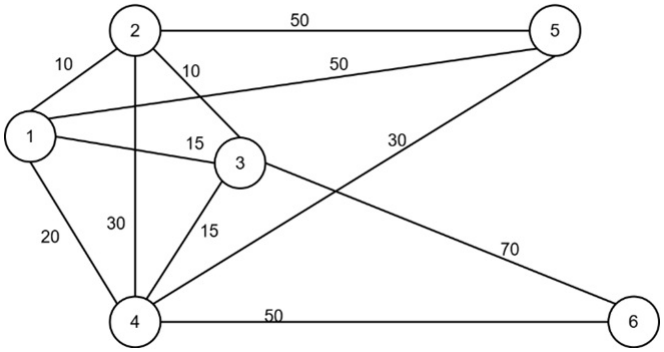


Gambar 1: Metode Penelitian

Berdasarkan Gambar 1, terdapat tiga tahapan utama dalam penelitian ini. Tahap pertama adalah pengumpulan data, yang mencakup pengambilan data titik-titik lokasi (vertex) pada wilayah Perumahan Depok Indah I serta pengukuran jarak antar titik yang akan dijadikan sebagai bobot pada graf. Data ini dapat diperoleh melalui survei lapangan, peta, atau sumber data geospasial lainnya. Tahap kedua adalah implementasi algoritma Kruskal, yang dilakukan melalui dua pendekatan. Pertama, perhitungan dilakukan secara manual untuk memahami mekanisme dasar dari algoritma Kruskal dalam membentuk Minimum Spanning Tree (MST). Kedua, dilakukan implementasi menggunakan bahasa pemrograman Python untuk mengotomatisasi proses dan meningkatkan efisiensi perhitungan, terutama untuk jumlah vertex dan edge yang lebih banyak. Tahap ketiga merupakan tahap analisis output, di mana hasil MST yang diperoleh dari algoritma Kruskal dibandingkan dengan hasil penelitian sebelumnya yang menggunakan algoritma Prim. Perbandingan ini bertujuan untuk melihat efisiensi panjang kabel yang dibutuhkan, dengan fokus utama pada total panjang kabel minimum yang dihasilkan oleh masing-masing metode.

Pengumpulan Data

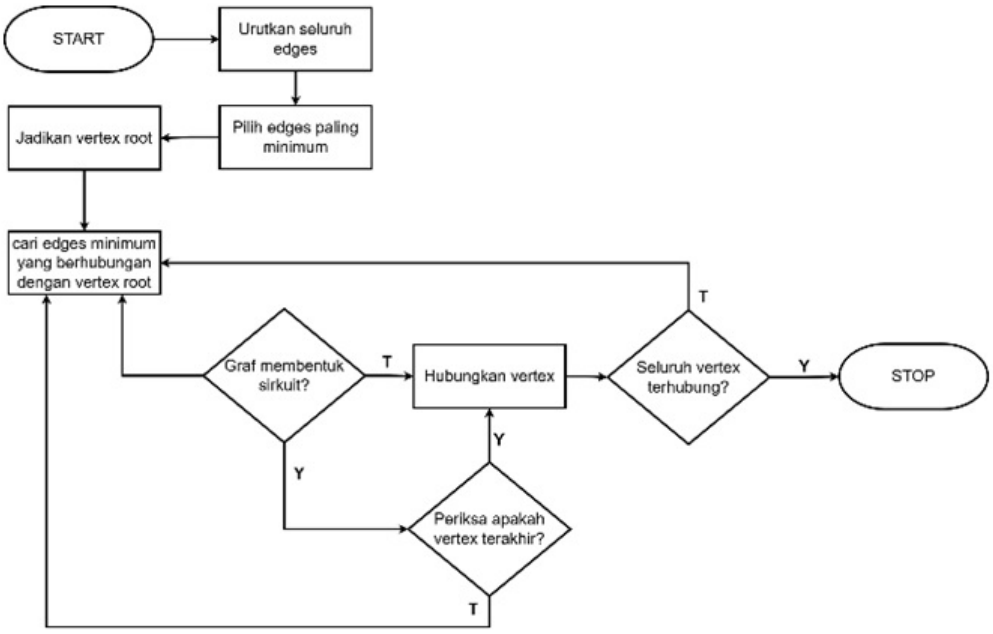
Data yang digunakan dalam penelitian ini mengacu pada sumber [3], yang memuat informasi mengenai jaringan listrik dengan total 6 vertex (titik/simpul), lihat Gambar 2. Setiap vertex merepresentasikan lokasi tertentu dalam kawasan yang dianalisis. Hubungan antar vertex dinyatakan dalam bentuk edge (sisi), dengan bobot berupa jarak antar titik dalam satuan kilometer. Dari data tersebut, diketahui bahwa bobot terkecil berada pada koneksi antara vertex 1–2, 2–1, 2–3, dan 3–2, yang masing-masing memiliki jarak 10 km, menunjukkan jalur terpendek dalam jaringan. Sebaliknya, bobot tertinggi terdapat pada vertex 3–6 dan 6–3, dengan jarak mencapai 70 km, yang merupakan jalur terjauh dalam graf tersebut. Ilustrasi struktur spanning tree yang terbentuk dari data tersebut dapat dilihat pada penelitian sebelumnya [3], yang menjadi referensi awal dan pembanding dalam penelitian ini.



Gambar 2: Spanning Tree Jaringan Listrik Perumahan Depok Indah I

Implementasi Algoritma Kruskal

Algoritma Kruskal dalam penelitian ini diterapkan melalui dua pendekatan. Pendekatan pertama dilakukan secara manual, baik dalam bentuk perhitungan maupun visualisasi graf. Langkah ini bertujuan untuk memahami alur kerja algoritma secara konseptual, termasuk proses pemilihan edge dengan bobot terendah secara bertahap hingga terbentuk MST tanpa siklus. Pendekatan kedua dilakukan secara komputerisasi dengan memanfaatkan bahasa pemrograman Python melalui platform Google Colab sebagai alat bantu. Implementasi algoritma dalam bentuk program bertujuan untuk mempercepat proses perhitungan, meminimalkan kesalahan manusia, dan memungkinkan analisis terhadap graf yang lebih kompleks atau memiliki jumlah vertex dan edge yang besar. Gambar 3 adalah ilustrasi sederhana yang menggambarkan cara kerja algoritma Kruskal dalam membentuk spanning tree dari graf berbobot.



Gambar 3: Flowchart Algoritma Kruskal

Tabel 1: Bobot/ Jarak antar Vertex

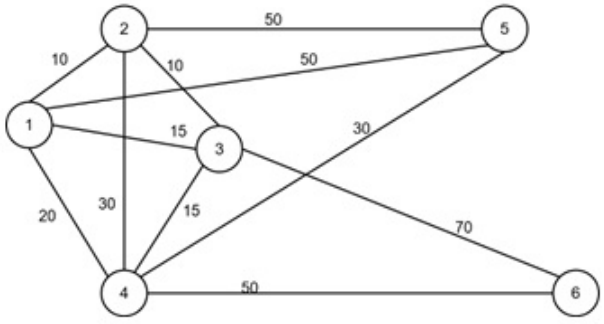
Vertex Awal	Vertex Tujuan	Bobot/ Jarak (km)
1	2	10
1	3	15
1	4	20
1	5	50
2	3	10
2	4	30
2	5	50
3	4	15
3	6	70
4	5	30
4	6	50

Output

Hasil dari pengolahan data menggunakan algoritma Kruskal melalui perhitungan manual akan diverifikasi dengan hasil yang diperoleh melalui proses komputerisasi, yaitu implementasi algoritma yang sama menggunakan bahasa pemrograman Python pada platform Google Colab. Verifikasi ini dilakukan untuk memastikan konsistensi hasil, serta untuk menilai keakuratan dan efisiensi pendekatan berbasis perangkat lunak. Selanjutnya, hasil akhir berupa struktur Minimum Spanning Tree (MST) dan total panjang kabel yang dibutuhkan akan dibandingkan dengan hasil dari penelitian sebelumnya [3], yang menggunakan algoritma Prim. Perbandingan ini bertujuan untuk mengevaluasi efektivitas algoritma Kruskal dalam konteks perencanaan jaringan listrik di kawasan Perumahan Depok Indah I, khususnya dari segi optimalisasi panjang kabel. Hasil dan Pembahasan Berdasarkan data yang diperoleh dari penelitian sebelumnya [3], diketahui bahwa bobot yang dimaksud dalam konteks ini merupakan jarak antar vertex (simpul) dalam graf yang merepresentasikan jaringan distribusi listrik. Setiap vertex menunjukkan lokasi tertentu, sementara bobot pada setiap edge (sisi) menunjukkan panjang kabel atau jarak fisik yang menghubungkan dua vertex tersebut. Adapun daftar jarak antar vertex yang digunakan sebagai bobot dalam graf disajikan pada Tabel 1.

Penghitungan Manual

Pada proses penghitungan manual, algoritma Kruskal dijalankan melalui beberapa tahapan iterasi untuk mendapatkan hasil akhir berupa MST. Setiap iterasi dilakukan dengan memilih edge dengan bobot paling kecil yang belum membentuk siklus dalam graf. Proses ini dilakukan secara berulang hingga seluruh vertex terhubung dalam satu graf tanpa siklus dan dengan total bobot minimum.

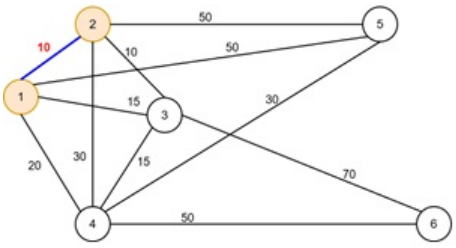


Gambar 4: Posisi Awal

Sebelum proses iterasi dimulai, algoritma Kruskal berada pada tahap inisialisasi. Pada tahap ini, seluruh vertex dalam graf dianggap sebagai komponen yang berdiri sendiri dan belum saling terhubung, lihat Gambar 4. Semua edge (sisi) dalam graf dikumpulkan dan diurutkan berdasarkan bobotnya dari yang terkecil hingga terbesar. Belum ada edge yang dipilih untuk dimasukkan ke dalam MST, sehingga struktur graf masih terpisah-pisah dan belum membentuk satu kesatuan. Keadaan awal ini menjadi dasar bagi proses seleksi edge yang akan dilakukan secara bertahap melalui iterasi, dengan tujuan akhir membentuk graf yang terhubung secara menyeluruh tanpa siklus dan memiliki total bobot minimum.

Iterasi 1 (Menentukan Vertex Root / Titik Awal)

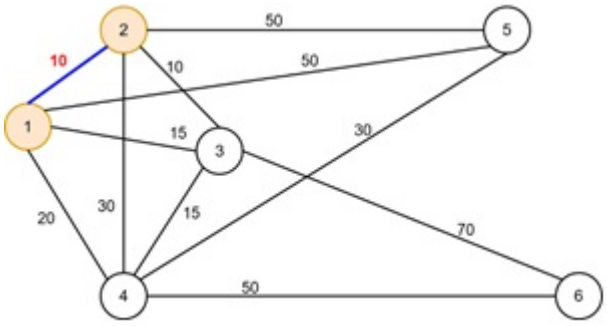
Setelah dilakukan pengurutan edge berdasarkan bobot terkecil, diperoleh bahwa edge antara vertex 1–2 dan 2–3 memiliki bobot paling rendah. Oleh karena itu, vertex 1, 2, atau 3 dapat dipilih sebagai titik awal atau root. Pada penelitian ini, diputuskan untuk menggunakan vertex 1 sebagai root. Karena pada tahap ini seluruh vertex dalam graf belum saling terhubung, proses iterasi dilanjutkan untuk menambahkan edge berikutnya yang memenuhi kriteria pembentukan MST tanpa menciptakan siklus, lihat Gambar 5.



Gambar 5: Hasil Iterasi 1

Iterasi 2 (Menentukan Vertex ke-2)

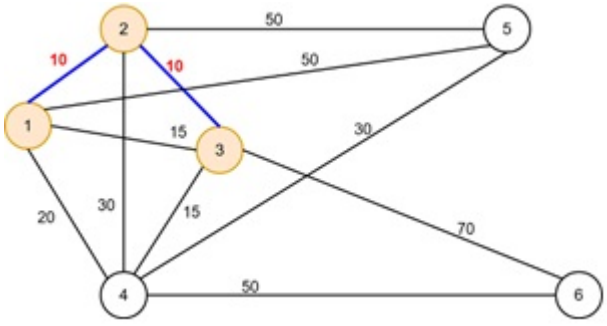
Setelah vertex 1 ditetapkan sebagai root pada iterasi sebelumnya, langkah selanjutnya adalah menentukan vertex ke-2 yang akan dihubungkan. Dilakukan pengurutan ulang terhadap edge untuk mencari sisi dengan bobot terkecil yang terhubung langsung dengan vertex 1. Diperoleh bahwa edge antara vertex 1–2 memiliki bobot paling minimum dan memenuhi syarat tersebut. Oleh karena itu, edge 1–2 dipilih dan ditambahkan ke dalam MST, sehingga vertex 2 kini terhubung dengan vertex 1. Karena masih terdapat vertex lain yang belum terhubung, proses iterasi dilanjutkan ke tahap berikutnya, yaitu iterasi ke-3.



Gambar 6: Hasil Iterasi 2

Iterasi 3 (Menentukan Vertex ke-3)

Setelah vertex 1 dan 2 berhasil dihubungkan pada iterasi sebelumnya, proses dilanjutkan untuk menentukan vertex ke-3. Pengurutan ulang terhadap edge dilakukan kembali untuk mencari sisi dengan bobot paling kecil yang terhubung dengan vertex yang sudah tergabung dalam MST, yaitu vertex 1 dan 2. Diketahui bahwa edge antara vertex 2–3 memiliki bobot terkecil dan memenuhi syarat tersebut. Oleh karena itu, edge 2–3 dipilih dan ditambahkan ke dalam MST, sehingga vertex 3 kini terhubung. Karena masih terdapat vertex yang belum terhubung secara keseluruhan, proses dilanjutkan ke iterasi berikutnya, yaitu iterasi ke-4.

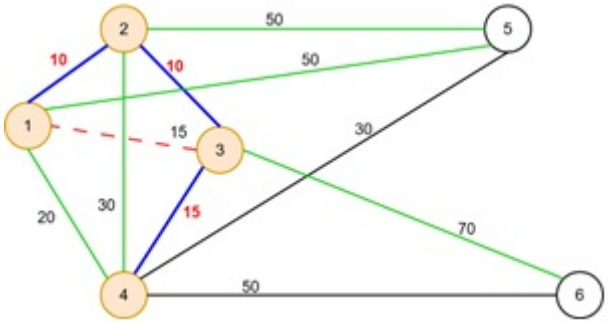


Gambar 7: Hasil Iterasi 3

Iterasi 4 (Menentukan Vertex ke-4)

Setelah vertex 1, 2, dan 3 berhasil terhubung dalam MST, proses dilanjutkan dengan menentukan vertex ke-4. Dilakukan pengurutan ulang terhadap edge untuk mencari sisi dengan bobot paling kecil yang menghubungkan vertex yang telah tergabung (1, 2, dan 3) dengan vertex lain yang belum terhubung. Pada tahap ini, terdapat beberapa kandidat, seperti vertex 4 dan 5. Diperoleh bahwa edge 1–3 dan edge 3–4 memiliki bobot paling kecil di antara opsi yang ada. Namun, karena edge 1–3 akan membentuk siklus—karena vertex 1 dan 3 sudah

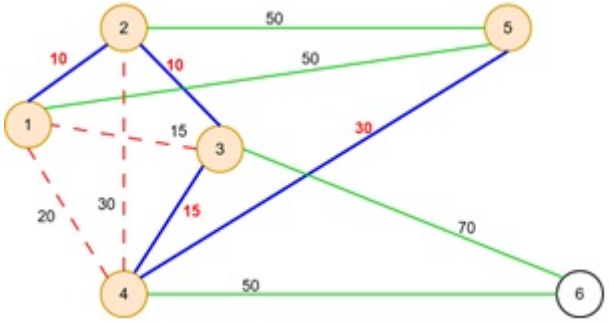
terhubung sebelumnya—maka edge tersebut harus diabaikan sesuai prinsip algoritma Kruskal yang tidak memperbolehkan terbentuknya siklus. Dengan demikian, edge 3–4 dipilih dan ditambahkan ke dalam MST. Karena masih terdapat vertex yang belum terhubung, proses dilanjutkan ke iterasi ke-5.



Gambar 8: Hasil Iterasi 4

Iterasi 5 (Menentukan Vertex ke-5)

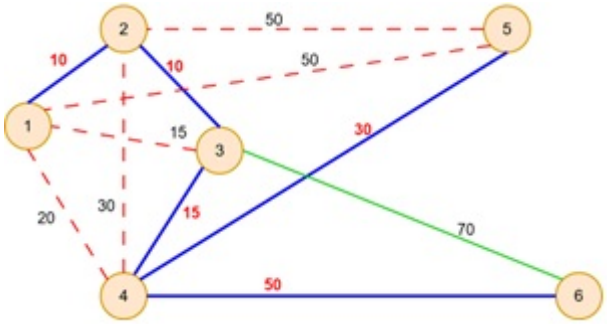
Setelah vertex 1, 2, 3, dan 4 berhasil terhubung dalam MST, langkah selanjutnya adalah menentukan vertex ke-5. Dilakukan pengurutan ulang terhadap edge untuk menemukan sisi dengan bobot paling kecil yang menghubungkan vertex yang sudah tergabung (1 hingga 4) dengan vertex yang belum terhubung, serta tidak membentuk siklus. Pada tahap ini, terdapat beberapa opsi, yaitu vertex 5 dan 6. Dari hasil pengurutan, diketahui bahwa edge antara vertex 4–5 memiliki bobot paling kecil di antara opsi yang tersedia dan tidak menimbulkan siklus. Oleh karena itu, edge 4–5 dipilih dan ditambahkan ke dalam MST. Karena masih ada vertex yang belum terhubung, proses dilanjutkan ke iterasi ke-6.



Gambar 9: Hasil Iterasi 5

Iterasi 6 (Menentukan Vertex ke-6)

Setelah vertex 1, 2, 3, 4, dan 5 berhasil terhubung dalam MST, proses dilanjutkan untuk menghubungkan vertex terakhir, yaitu vertex 6. Pada tahap ini, hanya tersisa satu vertex yang belum terhubung, namun terdapat dua opsi jalur penghubung, yaitu melalui vertex 3 dengan bobot 70 atau melalui vertex 4 dengan bobot 50. Karena algoritma Kruskal selalu memilih edge dengan bobot paling kecil yang tidak membentuk siklus, maka dipilih edge antara vertex 4–6. Dengan demikian, vertex 6 berhasil dihubungkan ke MST melalui vertex 4. Seluruh vertex kini telah terhubung tanpa membentuk siklus, sehingga proses iterasi dinyatakan selesai dan MST telah terbentuk secara lengkap.



Gambar 10: Hasil Iterasi 6

Hasil Akhir Perhitungan Manual:

Berdasarkan hasil perhitungan manual menggunakan algoritma Kruskal, jalur MST yang terbentuk terdiri dari edge 1–2, 2–3, 3–4, 4–5, dan 5–6. Total bobot dari jalur-jalur tersebut adalah $10 + 10 + 15 + 30 + 50 = 115$. Dengan demikian, dapat disimpulkan bahwa jumlah kabel yang dibutuhkan untuk pemasangan jaringan listrik di kawasan Perumahan Depok Indah I adalah sepanjang 115 km. Hasil ini menunjukkan penggunaan kabel yang efisien karena seluruh titik terhubung tanpa adanya pemborosan atau pengulangan jalur (siklus).

Penghitungan Komputerisasi

Penghitungan menggunakan metode komputerisasi dilakukan untuk mempercepat dan mempermudah proses penentuan MST. Dalam penelitian ini, implementasi algoritma Kruskal dilakukan menggunakan bahasa pemrograman Python. Proses komputerisasi mencakup beberapa tahapan, mulai dari representasi graf dalam bentuk struktur data, pengurutan edge berdasarkan bobot, hingga proses iteratif untuk membentuk MST secara otomatis tanpa menciptakan siklus. Pendekatan ini tidak hanya meningkatkan efisiensi penghitungan, tetapi juga memungkinkan verifikasi hasil manual dengan lebih akurat dan konsisten.

Tahapan 1: Representasi Graf

```
# Daftar sisi (edges) berupa tuple: (simpul_awal, simpul_tujuan, bobot)
edges = [
    (1, 2, 10), (1, 3, 15), (1, 4, 20), (1, 5, 50),
    (2, 3, 10), (2, 4, 30), (2, 5, 50),
    (3, 4, 15), (3, 6, 70),
    (4, 5, 30), (4, 6, 50),
]
```

Gambar 11: Representasi Graf

Pada tahap awal ini, graf direpresentasikan dalam bentuk daftar edge (sisi) dengan menggunakan struktur data list of tuples pada bahasa pemrograman Python. Setiap tuple dalam daftar tersebut terdiri dari tiga elemen utama, yaitu simpul awal (vertex asal), simpul tujuan (vertex tujuan), dan bobot yang menunjukkan jarak atau panjang kabel antara dua simpul tersebut. Representasi ini memudahkan proses pengolahan data secara komputerisasi, karena seluruh informasi mengenai koneksi antar simpul telah tersusun secara sistematis dan siap untuk diproses lebih lanjut dalam tahapan berikutnya, seperti pengurutan dan pemilihan edge untuk membentuk Minimum Spanning Tree (MST).

Tahapan 2: Inisialisasi Struktur Union-Find

```
# Daftar sisi (edges) berupa tuple: (simpul_awal, simpul_tujuan, bobot)
edges = [
    (1, 2, 10), (1, 3, 15), (1, 4, 20), (1, 5, 50),
    (2, 3, 10), (2, 4, 30), (2, 5, 50),
    (3, 4, 15), (3, 6, 70),
    (4, 5, 30), (4, 6, 50),
]
```

Gambar 12: Inisiasi Struktur Parent - Union Find

Struktur ini digunakan untuk melacak "parent" dari setiap simpul dan berperan penting dalam proses deteksi siklus selama pembentukan MST. Nantinya, setiap simpul akan menjadi bagian dari suatu himpunan yang diperbarui seiring proses berjalan.

Tahapan 3: Inisiasi Fungsi

```
# Fungsi pencarian root dari suatu simpul
def find(u):
    if parent[u] != u:
        parent[u] = find(parent[u])  # Path compression
    return parent[u]

# Fungsi untuk menggabungkan dua himpunan
def union(u, v):
    root_u = find(u)
    root_v = find(v)
    if root_u != root_v:
        parent[root_v] = root_u
        return True
    return False
```

Gambar 13: Inisiasi Fungsi

Pembuatan fungsi find(u) digunakan untuk menemukan root dari suatu simpul dengan pendekatan path compression, yang membuat pencarian lebih efisien. Sementara itu, fungsi union(u, v) digunakan untuk menggabungkan dua himpunan berbeda jika mereka belum terhubung, dan akan mengembalikan True jika penggabungan tidak membentuk siklus.

Tahapan 4: Ekstraksi Semua Simpul

```
# Ambil semua simpul dari daftar sisi
nodes = set()
for u, v, _ in edges:
    nodes.add(u)
    nodes.add(v)
```

Gambar 14: Ekstraksi Simpul

Dari daftar edge yang telah didefinisikan, seluruh simpul unik dikumpulkan ke dalam satu set bernama nodes. Tujuan dari tahap ini adalah untuk mengetahui seluruh simpul yang terlibat dalam graf, sehingga bisa digunakan untuk inisialisasi dan penghitungan jumlah edge yang dibutuhkan.

Tahapan 5: Inisialisasi Parent

```
# Inisialisasi setiap simpul sebagai parent dirinya sendiri
for node in nodes:
    parent[node] = node
```

Gambar 15: Inisiasi Parent

Setiap simpul dalam graf diinisialisasi sebagai parent dari dirinya sendiri. Hal ini menandakan bahwa pada awalnya, semua simpul merupakan bagian dari himpunan terpisah. Inisialisasi ini penting untuk proses union pada algoritma Kruskal agar dapat bekerja dengan benar.

Tahapan 6: Pengurutan Edge Berdasarkan Bobot

```
# Urutkan sisi berdasarkan bobot (ascending)
edges.sort(key=lambda x: x[2])
```

Gambar 16: Pengurutan edge

Daftar edge diurutkan berdasarkan nilai bobot secara ascending (dari kecil ke besar). Ini dilakukan

karena algoritma Kruskal selalu memilih edge dengan bobot paling kecil terlebih dahulu dalam proses pembentukan MST untuk menjamin hasil yang optimal.

Tahapan 7: Proses Seleksi MST

```
# Urutkan sisi berdasarkan bobot (ascending)
edges.sort(key=lambda x: x[2])

# Proses utama algoritma Kruskal
mst = []             # Daftar sisi yang masuk ke MST
total_weight = 0     # Total bobot MST

for u, v, w in edges:
    if union(u, v):  # Jika tidak membentuk siklus
        mst.append((u, v, w))
        total_weight += w
    if len(mst) == len(nodes) - 1:# MST selesai jika n-1 sisi sdh dipilih
        break
```

Gambar 17: Proses MST

Proses utama Kruskal dilakukan dengan menelusuri edge satu per satu dari yang berbobot terkecil. Setiap edge yang tidak membentuk siklus ditambahkan ke dalam MST. Proses ini berlanjut hingga jumlah edge dalam MST mencapai n-1, di mana n adalah jumlah simpul, yang menandakan MST sudah lengkap.

Tahapan 8: Menampilkan Hasil

```
# Tampilkan hasil MST
print("Hasil MST:")
for u, v, w in mst:
    print(f"{u}-{v} (Bobot: {w})")

# Tampilkan total bobot dari MST
print(f"\nTotal bobot MST: {total_weight}")
```

Gambar 18: Menampilkan Hasil

Setelah MST terbentuk, program menampilkan daftar edge yang terpilih beserta bobotnya. Selain itu, total bobot dari seluruh edge dalam MST juga ditampilkan. Hasil ini menjadi indikator jumlah total kabel yang diperlukan dalam konteks implementasi jaringan, misalnya pemasangan kabel listrik.

```
Hasil MST:
1-2 (Bobot: 10)
2-3 (Bobot: 10)
3-4 (Bobot: 15)
4-5 (Bobot: 30)
4-6 (Bobot: 50)

Total bobot MST: 115
```

Gambar 19: Hasil Output

Berdasarkan hasil penghitungan menggunakan algoritma Kruskal yang dikomputasikan dalam bahasa Python, didapatkan Minimum Spanning Tree (MST) dengan jalur sebagai berikut: simpul 1 terhubung ke simpul 2 dengan bobot 10, simpul 2 ke simpul 3 dengan bobot 10, simpul 3 ke simpul 4 dengan bobot 15, simpul 4 ke simpul 5 dengan bobot 30, dan simpul 4 ke simpul 6 dengan bobot 50. Seluruh simpul dalam graf berhasil terhubung tanpa membentuk siklus. Total bobot dari seluruh sisi dalam MST ini adalah 115. Hasil ini menunjukkan jumlah total kabel yang dibutuhkan untuk menghubungkan seluruh titik dalam jaringan, misalnya untuk instalasi kabel listrik di kawasan Perumahan Depok Indah I, yaitu sepanjang 115 km.

Diskusi

Pada penelitian sebelumnya menggunakan algoritma Prim. Algoritma Prim merupakan salah satu algoritma populer yang digunakan untuk membentuk Minimum Spanning Tree (MST), yaitu struktur jaringan yang menghubungkan seluruh simpul dalam graf berbobot tanpa membentuk siklus, dengan total bobot minimum. Algoritma ini bekerja dengan pendekatan greedy, dimulai dari satu simpul awal yang kemudian diperluas secara bertahap dengan menambahkan sisi (edge) berbobot terkecil yang menghubungkan simpul yang sudah ada dalam MST ke simpul yang belum terhubung. Dalam implementasinya, algoritma Prim bersifat lokal dan berbasis simpul, karena setiap iterasi hanya mempertimbangkan sisi-sisi yang berasal dari simpul yang telah masuk ke dalam MST. Dalam penelitian sebelumnya, algoritma Prim diterapkan dengan menentukan vertex 1 sebagai simpul awal, kemudian secara bertahap memilih sisi dengan bobot terkecil yang terhubung ke simpul yang sudah ada di MST. Hasilnya, terbentuk jalur 1–2–5–4–6, dengan simpul 3 sebagai jalur alternatif yang dapat dihubungkan melalui simpul 1 atau 4. Total panjang kabel yang diperlukan sebesar 185 km. Meskipun algoritma Prim efektif dalam membentuk MST, pendekatannya yang berbasis pada simpul awal dapat menghasilkan jalur yang kurang optimal dalam kondisi tertentu, seperti pada studi kasus ini. Di sisi lain, pada penelitian ini yang menggunakan algoritma Kruskal, jalur yang terbentuk adalah 1–2–3–4–5–6 dengan total panjang kabel yang lebih efisien, yaitu 115 km. Dengan demikian, terdapat selisih 70 km lebih pendek dibandingkan hasil dari algoritma Prim. Jika dihitung secara prosentase, maka penggunaan algoritma Kruskal mampu meningkatkan efisiensi penggunaan kabel sebesar sekitar 37,84%. Hal ini menunjukkan bahwa algoritma Kruskal lebih optimal dalam menghasilkan jaringan dengan total bobot minimum, karena fokus pada pemilihan sisi dengan bobot terendah secara global tanpa bergantung pada titik awal tertentu seperti pada algoritma Prim.

Penutup

Berdasarkan hasil penelitian ini, penerapan algoritma Kruskal dalam menentukan jalur optimal untuk jaringan listrik di Perumahan Depok Indah I menunjukkan hasil yang lebih efisien dibandingkan pendekatan sebelumnya yang menggunakan algoritma Prim. Dengan total panjang kabel sebesar 115 km, pendekatan Kruskal berhasil mengurangi kebutuhan kabel hingga 70 km atau setara dengan sekitar 37,84% dari total panjang kabel yang dibutuhkan pada penelitian sebelumnya. Efisiensi ini menunjukkan keunggulan algoritma Kruskal dalam meminimalkan total bobot graf tanpa membentuk siklus, terutama pada kasus graf yang memiliki banyak jalur alternatif.

Implementasi algoritma secara manual dan komputerisasi menggunakan bahasa Python memberikan pemahaman menyeluruh tentang cara kerja algoritma serta fleksibilitas dalam penerapannya. Hasil ini tidak hanya mendukung pemilihan algoritma Kruskal sebagai solusi perencanaan jaringan listrik yang lebih hemat biaya, tetapi juga menunjukkan potensi besar penerapannya di bidang-bidang lain yang membutuhkan optimasi jaringan, seperti sistem transportasi, telekomunikasi, dan distribusi logistik. Oleh karena itu, penelitian ini dapat dijadikan sebagai acuan dalam pengambilan keputusan teknis untuk proyek-proyek serupa di masa mendatang.

Ucapan Terimakasih

Penulis menyampaikan terima kasih yang sebesar-besarnya kepada Judith Tania, Dede Firza, dan Iman Nur Cahyadi atas penelitian sebelumnya yang telah dilakukan. Hasil dan pendekatan yang mereka gunakan menjadi landasan penting serta inspirasi berharga dalam pelaksanaan penelitian ini. Tanpa kontribusi mereka, arah dan pengembangan studi ini tidak akan terbentuk sebagaimana adanya saat ini. Semoga penelitian ini dapat menjadi kelanjutan dan pelengkap dari upaya yang telah dilakukan sebelumnya.

Daftar Pustaka

[1] S. Dhouib, "Innovative method to solve the minimum spanning tree problem: The Dhouib-Matrix-MSTP (DM-MSTP)," Results in Control and Optimization, vol. 14, no. https://doi.org/10.1016/j.rico.2023.100359, p. 100359, 2024.

[2] P. Paryati and K. Salahddine, "The Implementation of Kruskal's Algorithm for Minimum Spanning Tree in a Graph," E3S Web of Conferences 297, vol. 6, no. https://doi.org/10.1051/e3sconf/202129701062, p. 01062, 2021.

[3] J. Tania, D. Firza dan I. N. Cahyadi, "Penerapan Minimum Spanning Tree Pada Pengoptimalan Jaringan Listrik Di Perumahan Depok Indah I," Bullwtin of Applied Industrial Engineering Theory, no. https://jim.unindra.ac.id/index.php/baiet/article/view/5861, pp. 85-90, 2021.

[4] S. Li, "The Application of Kruskal's Algorithm in Bus Route Planning and the Influence of Different Factors on It," Applied and Computational Engineering, vol. 2, no. https://doi.org/10.54254/2755-2721/2/20220619, pp. 398-402, 2023.

[5] B. F. Melnikov and Y. Y. Terentyeva, "Building communication networks: on the application of the Kruskal's algorithm in the problems of large dimensions," IOP Conf. Series: Materials Science and Engineering, no. https://doi.org/10.1088/1757-899x/1047/1/012089, pp. 1-7, 2020.

[6] R. Rismunanda dan N. Napitupulu, "The Application of Cruscal Algorithm in Determining the Shortest Routes Tourism in Medan," FARABI: Jurnal Matematika Dan Pendidikan Matematika, vol. 5(2), no. https://doi.org/10.47662/farabi.v5i2.390, pp. 116-120, 2022.

[7] Y. Odyuo, D. Sarkar dan L. Sumi , "Optimal feeder reconfiguration in distributed generation environment under time-varying loading condition," Sn Applied Sciences, vol. 3(26), no. https://doi.org/10.1007/s42452-021-04557-w, 2021.

[8] K. A. Wirakusuma dan K. D. Diatmika, "Analisis Perbandingan Kompleksitas Waktu Algoritma Mst DalamPenyusunan Jaringan Pipa Air Bersih," Jurnal Teknologi Dan Sistem Informasi Bisnis, vol. Vol. 7 No. 1, no. https://doi.org/10.47233/jteksis.v7i1.1813, pp. 172-179, 2025.

[9] M. M. Manik dan C. Sormin, "Optimasi Jaringan Pipa PDAM Tirta Sanjung Buana Di Perumahan Salasah Indah Menggunakan Algoritma Kruskal," http://dx.doi.org/10.30829/jistech.v9i2.22728, vol. 9(2), no. http://dx.doi.org/10.30829/jistech.v9i2.22728, pp. 249-257, 2024.

[10] S. Z. Faizah, D. L. Kartika dan A. Winarni, "Implementasi Algoritma PRIM untuk Optimasi Panjang PIPA pada Sistem Distribusi Air Bersih di Desa Banteran ," Proximal: Jurnal Penelitian Matematika dan Pendidikan Matematika, no. https://doi.org/10.30605/proximal.v8i1.5353, pp. 431-441, 2025.