**STRUKTUR DATA DAN ALGORITMA**

**(GRAPH DATA TYPE, TRAVELSALS, SHORTEST PATH TREE)**



**DISUSUN OLEH :**

(G1A024009)

(G1A024011)

(G1A024019)

(G1A024031)

(G1A024043)

(G1A024077)

(G1A024082)

1. Muhammad Yusuf Arrayyan
2. Ariel Tiansyahrullah
3. Tasya Triani
4. Judika Ebenezer Sianturi
5. Farhan Khairullah
6. Muhammad Abel Cakrawangsa
7. Afifah Chairunnisya

**KELAS :**

Informatika (A)

**DOSEN PENGAMPU :**

Arie Vatresia, S.T., M.TI., Ph.D

**PROGRAM STUDI INFORMATIKA**

**FAKULTAS TEKNIK**

**UNIVERSITAS BENGKULU**

**2025**

**DAFTAR ISI**

DAFTAR ISI 1

BAB I PENDAHULUAN 1

* 1. Latar Belakang 1
  2. Rumusan Masalah 2

BAB II PEMBAHASAN 3

* 1. Data type serta peranannya dalam implementasi algoritma Djikstra 3
  2. Traversal Pada Graf Dan Konsep Traversal Dalam

Mendukung Proses Pencarian Jalur Terpendek 4

* 1. Shortest Path Tree Dan Hubungannya dengan Algoritma Dijkstra 6
  2. Implementasi 7

BAB III PENUTUP 13

* 1. Kesimpulan 13
  2. Saran 13

DAFTAR PUSTAKA 14

**BAB 1  
PENDAHULUAN**

**1.1 Latar Belakang**

Seiring dengan pesatnya perkembangan teknologi informasi dan komunikasi, kebutuhan akan sistem yang mampu mengelola dan memproses data dalam bentuk hubungan antar objek semakin meningkat. Salah satu model representasi yang banyak digunakan untuk menggambarkan hubungan tersebut adalah struktur graf. Graf meurpakan struktur matematika yang terdiri atas simpul (node) dan sisi (edge), yang digunakan untuk merepresentasikan keterkaitan antar entitas. Dalam berbagai bidang, seperti transportasi, telekomunikasi, jaringan komputer, manajemen logistik, hingga pengembangan aplikasi berbasis peta digital, graf menjadi model yang sangat penting untuk menunjang berbagai kebutuhan analisis dan optimasi jalur.

Salah satu permasalahan mendasar yang sering muncul dalam penggunaan graf adalah bagaimana menentukan jalur terpendek antara dua buah simpul. Penentuan jalur terpendek ini tidak hanya menjadi masalah teoritis dalam ilmu komputer dan matematika diskrit, tetapi juga memiliki aplikasi luas di dunia nyata. Misalnya, dalam sistem navigasi kendaraan, penentuan jalur terpendek dapat menghemat waktu dan bahan bakar; dalam jaringan komputer, rute data optimal diperlukan untuk meningkatkan efisiensi komunikasi; sementara dalam logistik, jalur terpendek dapat membantu mengurangi biaya distribusi dan meningkatkan kecepatan layanan. Dalam mengatasi masalah jalur terpendek tersebut, telah dikembangkan berbagai macam algoritma. Salah satu algoritma yang paling dikenal dan banyak digunakan adalah algoritmaDijkstra. Algoritma Dijkstra diperkenalkan oleh ilmuwan komputer asal Belanda, Edsger W. Dijkstra, pada tahun 1956, algoritma ini menawarkan solusi yang sistematis untuk menemukan jalur terpendek dari sebuah simpul sumber ke seluruh simpul lain dalam graf berbobot, dengan syarat bahwa bobot sisi-sisi pada graf tidak bernilai negatif. Prinsip kerja algoritma ini didasarkan pada pendekatan greedy, yaitu memilih langkah terbaik secara lokal pada setiap iterasi untuk memperoleh solusi global yang optimal. Algoritma Dijkstra memiliki karakteristik yang membuatnya unggul dalam berbagai aplikasi nyata, antara lain keakuratan dalam menentukan jalur minimum serta efisiensinya dalam memproses graf berskala besar. Selain itu, implementasi algoritma ini relatif sederhana dan dapat dioptimalkan lebih lanjut dengan berbagai struktur data, seperti priority queue berbasis heap, sehingga kinerjanya dapat semakin ditingkatkan dalam situasi graf dengan jumlah simpul dan sisi yang sangat banyak. Dalam konteks graf berbobot, di mana setiap sisi memiliki nilai bobot yang merepresentasikan jarak, biaya, atau waktu, penerapan algoritma Dijkstra menjadi sangat relevan. Algoritma ini memastikan bahwa solusi yang diperoleh benar-benar merupakan jalur minimum yang memperhitungkan keseluruhan bobot lintasan, bukan hanya berdasarkan jumlah simpul atau panjang lintasan. Hal ini menjadikannya sangat efektif untuk diterapkan pada permasalahan nyata yang membutuhkan perhitungan biaya atau waktu minimal secara akurat.

Berdasarkan latar belakang tersebut, makalah ini disusun untuk membahas secara mendalam mengenai implementasi algoritma Dijkstra dalam menentukan jalur terpendek pada graf berbobot. Pembahasan akan mencakup prinsip kerja algoritma, langkah-langkah implementasi, analisis performa, serta contoh aplikasi nyata yang relevan. Melalui pembahasan ini, diharapkan pembaca dapat memperoleh pemahaman yang komprehensif mengenai pentingnya penggunaan algoritma Dijkstra dalam berbagai bidang yang memerlukan optimasi jalur.

**1.2 Rumusan Masalah**

1. Apa yang dimaksud dengan data type dan bagaimana peranannya dalam implementasi algoritma Dijkstra?
2. Apa yang dimaksud dengan traversal pada struktur graf, dan bagaimana konsep traversal mendukung proses pencarian jalur terpendek?
3. Apa yang dimaksud dengan shortest path tree, dan bagaimana hubungannya dengan hasil yang diperoleh dari algoritma Dijkstra?
4. Jukida adalah calon mahasiswa yang sedang mengikuti UTBK di sebuah kampus. Ia memulai perjalanan dari gerbang utama untuk menuju gedung D tempat pelaksanaan ujian. Kampus tersebut terdiri dari berbagai gedung dan fasilitas yang dihubungkan oleh jalan-jalan yang dapat direpresentasikan sebagai graf berbobot!

**BAB 2  
PEMBAHASAN**

**2.1 Data Type Serta Peranannya Dalam Implementasi Algoritma Djikstra**

Tipe data (*data type*) merupakan suatu konsep dasar dalam bidang ilmu komputer yang berfungsi untuk menentukan jenis nilai yang dapat disimpan oleh suatu variabel, serta operasi-operasi apa saja yang dapat dilakukan terhadap nilai tersebut. Setiap tipe data memiliki karakteristik spesifik yang membedakannya dari tipe data lainnya, seperti kemampuan untuk merepresentasikan angka, karakter, teks, hingga struktur data yang lebih kompleks. Pemilihan tipe data yang tepat menjadi faktor penting dalam pengembangan algoritma yang efektif dan efisien. Dalam implementasi algoritma Dijkstra, tipe data memiliki peranan yang sangat krusial dalam berbagai aspek, antara lain:

* + 1. Representasi Graf

Graf berbobot yang menjadi objek utama dalam algoritma Dijkstra biasanya direpresentasikan dengan menggunakan struktur data tertentu, seperti *adjacency list* atau *adjacency matrix*. Struktur ini umumnya dibangun dengan tipe data seperti array, list, atau dictionary. Dalam *adjacency list*, setiap simpul menyimpan daftar simpul tetangganya beserta bobot sisi yang menghubungkannya, sedangkan pada *adjacency matrix*, hubungan antar simpul direpresentasikan dalam bentuk matriks dua dimensi bertipe numerik, seperti integer atau float.

* + 1. Penyimpanan Nilai Jarak

Algoritma Dijkstra membutuhkan suatu mekanisme untuk menyimpan jarak minimum sementara dari simpul sumber ke simpul-simpul lainnya. Nilai-nilai jarak ini biasanya disimpan dalam sebuah array bertipe numerik. Pemilihan tipe data numerik, seperti integer untuk bobot yang bulat atau float untuk bobot pecahan,hal ini akan sangat memengaruhi ketepatan hasil serta performa algoritma.

* + 1. Pengelolaan Antrian Prioritas

Dalam mempercepat pemilihan simpul dengan jarak minimum berikutnya, algoritma Dijkstra sering menggunakan struktur data *priority queue*, yang biasanya diimplementasikan dengan heap. Dalam konteks ini, setiap elemen *priority queue* merupakan pasangan nilai (jarak, simpul), yang dapat direpresentasikan menggunakan tuple atau objek terstruktur.

* + 1. Penanda Status Simpul

Selain menyimpan jarak, algoritma juga membutuhkan informasi mengenai status simpul, apakah simpul tersebut sudah diproses atau belum. Informasi ini dapat disimpan dengan menggunakan array bertipe boolean, yang secara sederhana menandai status setiap simpul selama proses eksekusi algoritma.

Dengan demikian, disimpulkan bahwa tipe data memegang peranan penting dalam implementasi algoritma Dijkstra, baik dalam aspek representasi struktur graf, penyimpanan informasi jarak, maupun pengelolaan alur eksekusi. Pemilihan tipe data yang kurang tepat dapat menyebabkan inefisiensi, baik dari segi penggunaan memori maupun kecepatan eksekusi algoritma. Oleh karena itu, pemahaman terhadap konsep tipe data dan penerapannya menjadi hal mendasar dalam pengembangan sistem yang menerapkan algoritma Dijkstra untuk penentuan jalur terpendek.

**2.2** **Traversal Pada Graf Dan Konsep Traversal Dalam Mendukung Proses Pencarian Jalur Terpendek**

Traversal pada struktur graf merupakan proses sistematis untuk mengunjungi semua simpul (node) dan sisi (edge) yang terdapat di dalam graf. Traversal bertujuan untuk mengakses, mengevaluasi, atau memproses seluruh elemen yang ada pada graf, sesuai dengan metode atau aturan tertentu. Dalam dunia ilmu komputer, traversal graf menjadi dasar bagi berbagai macam algoritma, baik untuk pencarian, pengurutan, analisis hubungan antar simpul, maupun penyelesaian berbagai permasalahan optimasi, seperti pencarian jalur terpendek.

Terdapat dua metode traversal graf yang paling umum, yaitu Breadth-First Search (BFS) yang melakukan penelusuran secara melebar, dengan mengunjungi semua simpul tetangga dari suatu simpul terlebih dahulu sebelum berpindah ke simpul pada tingkat berikutnya. Sedangkan Depth-First Search (DFS) akan melakukan penelusuran secara mendalam, dengan menelusuri satu jalur sampai ke ujung sebelum kembali dan menelusuri jalur lain.

Dalam konteks pencarian jalur terpendek, khususnya pada graf berbobot, konsep traversal memegang peranan penting. Traversal memungkinkan algoritma dalam mengakses simpul-simpul dalam urutan tertentu yang mendukung evaluasi jarak terpendek dari simpul sumber ke simpul lainnya. Dalam algoritma seperti Dijkstra, traversal tidak dilakukan secara sembarangan, melainkan berdasarkan prioritas jarak minimum. Simpul yang memiliki jarak kumulatif terkecil dari sumber akan diproses lebih dahulu. Konsep traversal mendukung proses pencarian jalur terpendek melalui beberapa mekanisme, antara lain:

2.2.1 Pemilihan Simpul Secara Prioritas

Dalam algoritma jalur terpendek, traversal dilakukan dengan memilih simpul yang memiliki jarak terpendek yang diketahui sejauh ini. Dengan cara ini, jalur optimal dibangun secara bertahap, dan tidak ada jalur lebih pendek yang terlewat.

2.2.2 Pembaruan Jarak dan Rute

Setelah suatu simpul diproses dalam traversal, jarak ke semua tetangga simpul tersebut diperiksa dan diperbarui jika ditemukan jalur yang lebih pendek melalui simpul yang sedang diproses. Hal ini memungkinkan algoritma untuk terus memperbaiki solusi hingga menemukan jalur terpendek akhir.

2.2.3 Efisiensi Eksplorasi Graf

Traversal terarah berdasarkan jarak minimum mencegah algoritma memproses jalur-jalur yang lebih panjang secara tidak perlu. Hal ini meningkatkan efisiensi waktu eksekusi dan mengoptimalkan penggunaan sumber daya memori.

Dengan demikian, traversal pada struktur graf bukan hanya sekadar proses penelusuran, tetapi menjadi bagian integral dari mekanisme pencarian jalur terpendek. Traversal yang tepat memungkinkan algoritma seperti Dijkstra bekerja secara optimal dalam menemukan solusi tercepat dan paling efisien di antara seluruh kemungkinan jalur yang tersedia dalam graf berbobot.

**2.3 Shortest Path Tree Dan Hubungannya dengan Algoritma Dijkstra**

Shortest Path Tree (SPT) adalah sebuah struktur pohon yang dibentuk dari graf berbobot, di mana pohon tersebut merepresentasikan jalur terpendek dari satu simpul sumber ke seluruh simpul lainnya dalam graf. Setiap jalur dari simpul sumber ke simpul lain dalam SPT merupakan jalur dengan total bobot minimum dibandingkan dengan semua jalur alternatif yang tersedia di dalam graf. Dalam shortest path tree, simpul sumber bertindak sebagai akar (*root*), dan setiap simpul lainnya memiliki satu jalur unik yang menghubungkannya ke akar melalui lintasan dengan bobot total terkecil. Struktur ini penting dalam berbagai aplikasi, seperti dalam sistem navigasi, perencanaan jaringan, dan pengoptimalan logistik, di mana diperlukan jalur tercepat, terpendek, atau paling efisien.

Hubungan antara shortest path tree dan algoritma Dijkstra sangat erat. Algoritma Dijkstra secara sistematis membangun shortest path tree selama proses eksekusinya yaitu sebagai berikut:

* + 1. Proses Pembentukan Shortest Path Tree

Saat algoritma Dijkstra berjalan, ia akan memulai dari simpul sumber dan secara bertahap menentukan jalur terpendek ke simpul lain satu per satu. Setiap kali jarak terpendek ke suatu simpul ditemukan, simpul tersebut akan ditambahkan ke dalam struktur pohon yang sedang dibentuk. Hubungan antar simpul ditentukan berdasarkan jalur dengan bobot kumulatif terkecil yang ditemukan selama traversal.

* + 1. Pencatatan Jalur Optimal

Dalam membangun shortest path tree, algoritma Dijkstra tidak hanya menyimpan informasi tentang jarak minimum, tetapi juga mencatat dari simpul mana jalur tersebut berasal, sering disebut sebagai *predecessor* atau *parent*. Dengan demikian, setelah seluruh proses selesai, hubungan antar simpul membentuk pohon jalur terpendek yang dapat dilacak dari setiap simpul kembali ke simpul sumber.

* + 1. Karakteristik Pohon

Karena shortest path tree dihasilkan dari proses pemilihan jalur terpendek secara bertahap dan sistematis, pohon ini memiliki sifat tidak mengandung siklus (seperti halnya pohon dalam teori graf) dan mencakup semua simpul yang dapat dijangkau dari simpul sumber. Jalur dari simpul sumber ke setiap simpul lain adalah unik dan optimal dalam hal total bobot.

Dengan demikian, hasil utama dari algoritma Dijkstra bukan hanya sekadar menemukan jarak minimum antara simpul sumber dan simpul lain, melainkan juga membangun sebuah struktur shortest path tree yang menggambarkan jalur-jalur minimum tersebut secara lengkap. Struktur ini kemudian dapat digunakan untuk berbagai kebutuhan, seperti menampilkan rute optimal, mengoptimalkan jaringan, atau mendukung keputusan strategis dalam sistem berbasis graf.

**2.4 Implementasi**

Permasalahan:

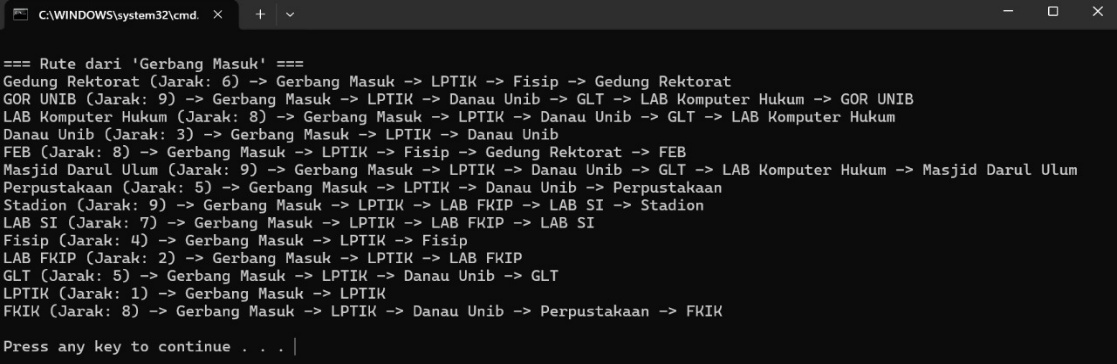
Jukida adalah calon mahasiswa yang sedang mengikuti UTBK di sebuah kampus. Ia memulai perjalanan dari gerbang utama untuk menuju gedung D tempat pelaksanaan ujian. Kampus tersebut terdiri dari berbagai gedung dan fasilitas yang dihubungkan oleh jalan-jalan yang dapat direpresentasikan sebagai graf berbobot!

2.4.1 Penyelesaian jalur BFS

****

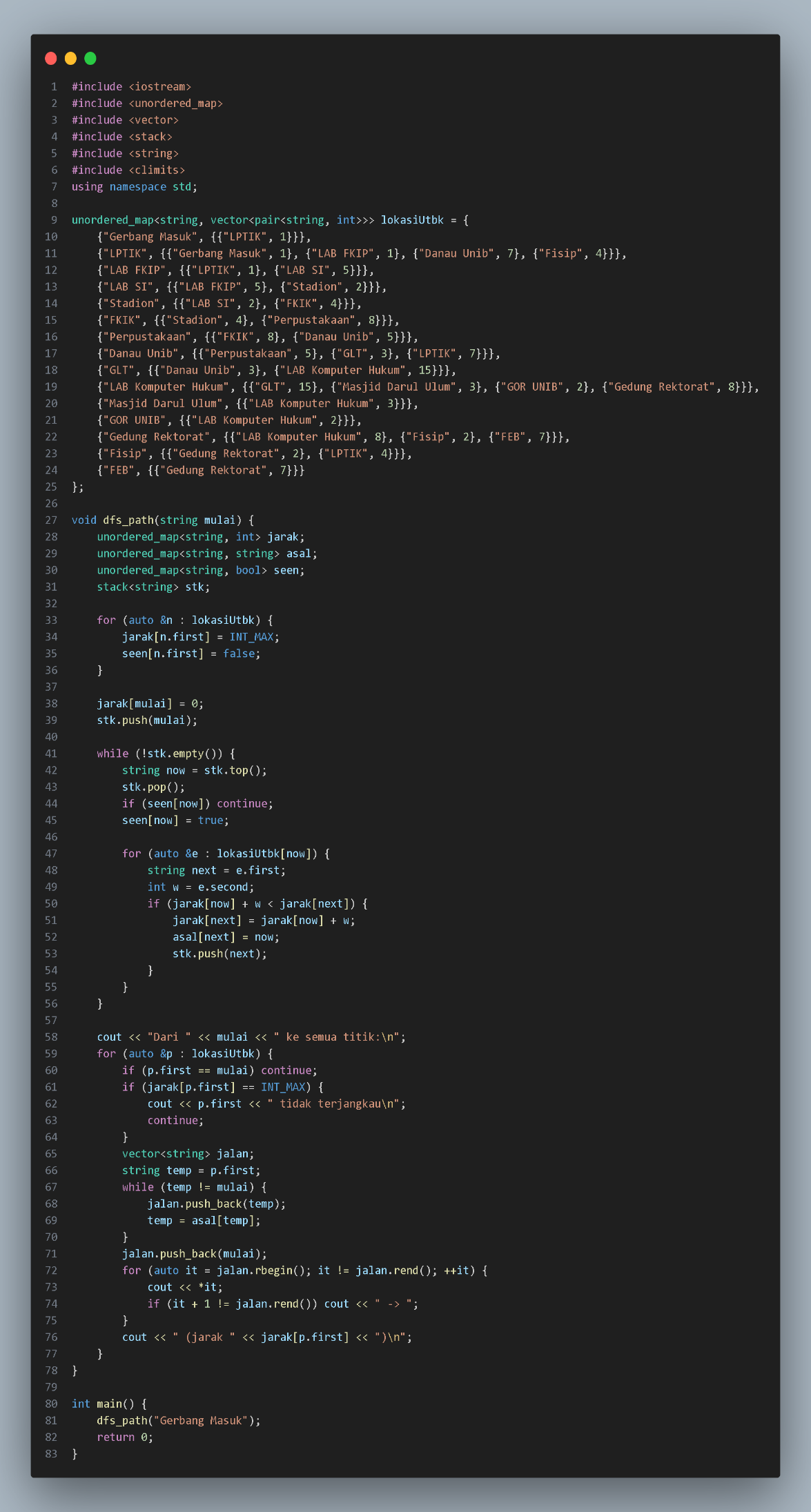
Untuk menyelesaikan masalah ini, kita membuat sebuah program dengan menggunakan bahasa pemrograman C++.

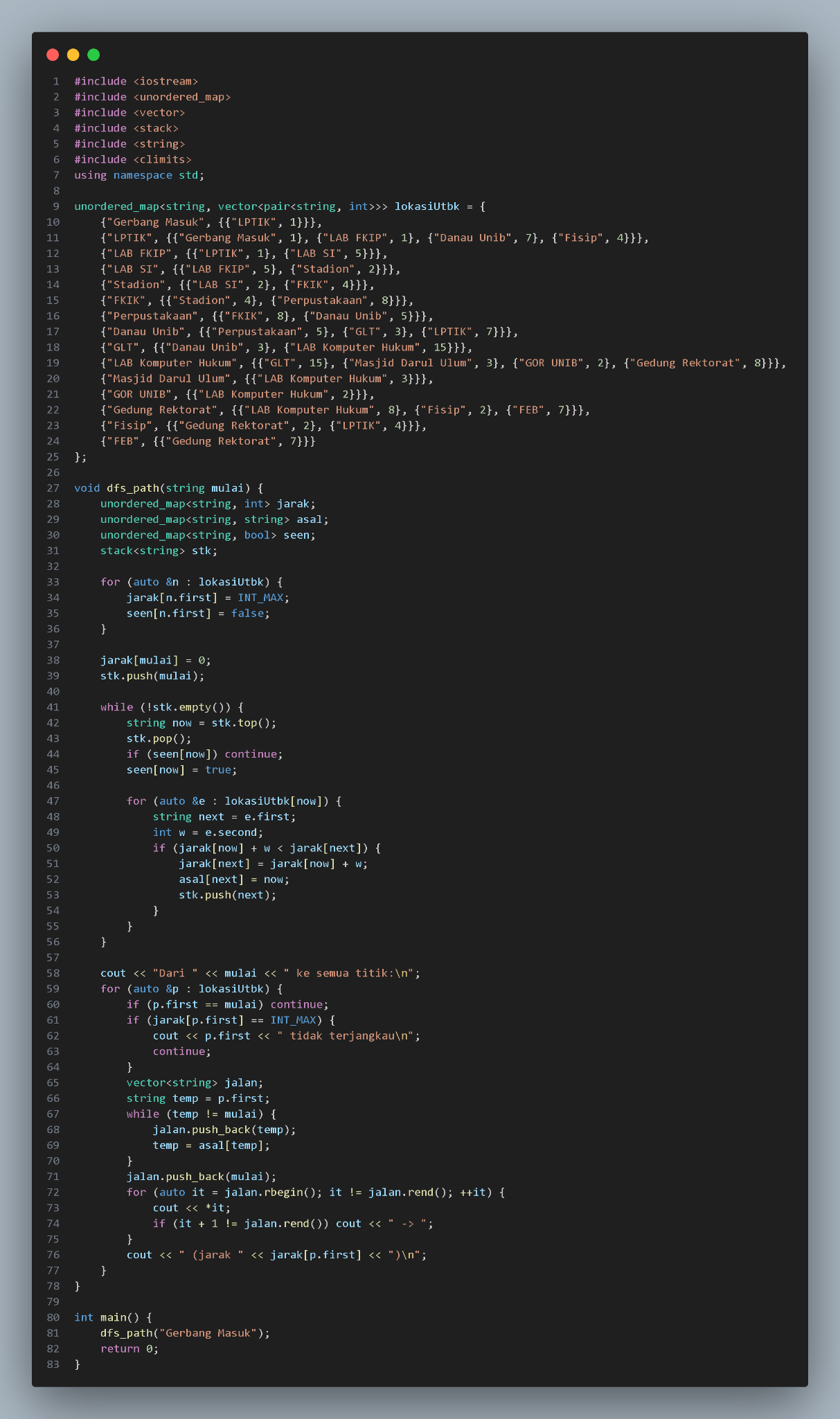
1. Dalam masalah ini, LokasiUTBK sebagai tempat awal serta bobot sebagai nilai dalam vector.
2. Lalu jarak dan asal berfungsi sebagai Array untuk menyimpan nilai.
3. Memanggil Algoritma BFS untuk memulai pencarian jalur dari Gerbang Masuk ke tujuan yang ingin dieksekusi.
4. Mengeksekusi BFS, untuk menemukan jalur terpendek ke lokasi lain, dengan mengambil node terdepan dari antrian, di sini program akan membaca semua lokasi dan jarak yang tersimpan.
5. Menggunakan antrian untuk memeriksa lokasi yang sedang diperiksa.
6. Mengupdate jarak dan jalur saat menemukan lokasi yang lebih dekat dari Gerbang Masuk.
7. Setelah peengsekusian BFS selesai, algoritma memeriksa setiap lokasi untuk mencetak jalur, dan mencetak jalur dan jarak dari titik awal ke tujuan.

Hasil:

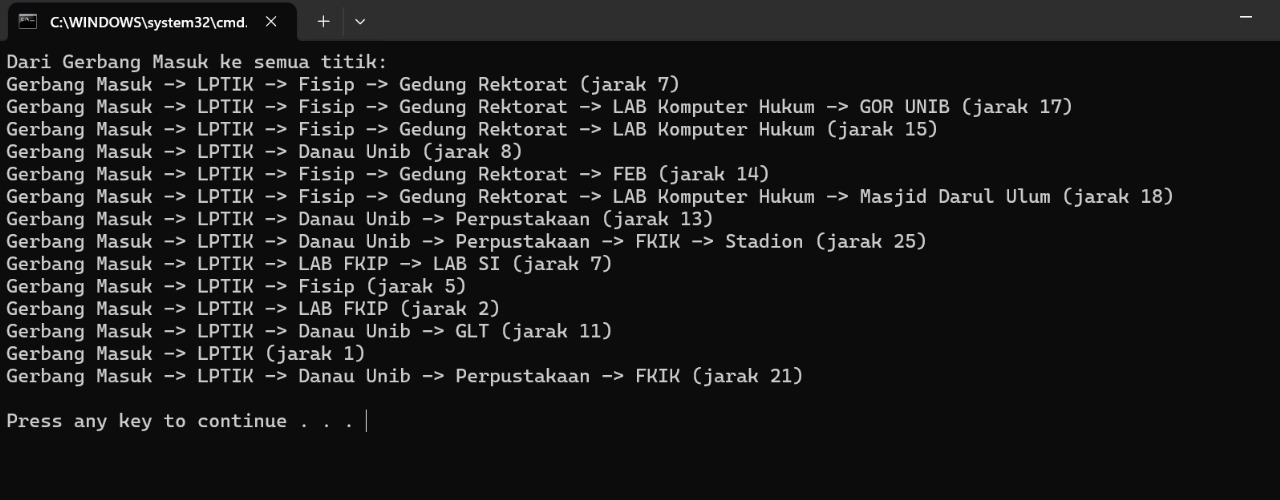
Output tersebut menunjukkan rute lokasi awal dari Gerbang Masuk menuju beberapa tujuan di lingkungan kampus. Setiap jalur memberikan tahapan untuk menuju ke lokasi tujuan. Algoritma BFS di sini digunakan untuk mencari rute rute tersebut secara terstruktur dari awal lokasi sampai tujuan, memastikan semua kemungkinan jalur yang berada di lingkungan kampus ternavigasi dengan efisien.

2.4.2 Penyelesaian jalur DFS:



****

1. Di sini menggunakan struktur data unordered\_map untuk menyimpan data, dan vector di dalamnya untuk menyimpan lokasi-lokasi dan jarak nya.
2. Adapun stack yang berguna untuk menyimpan lokasi yang akan dikunjungi
3. Pengeksekusian dimulai pada string mulai, lalu jika stack tidak kosong algortima akan mengambil lokasi teratas dari stack untuk setiap lokasi yang terhubung ke lokasi, jika jarak lebih pendek, maka jarak dan asal akan diupdate, lalu algoritma akan menambahkan lokasi baru ke dalam stack.
4. Jika loop telah selesai, program akan menampilkan semua titik dan jarak dari titik awal sampai ke tujuan, lalu menggunakan vector ‘jalan’ untuk menyimpan jalur dari titik awal ke lokasi lain, dan program akan menampilkan hasil jarak dari ‘Gerbang Masuk’ ke semua titik lokasi yang dapat dijangkau, beserta jalur yang dilewati.

****Hasil:

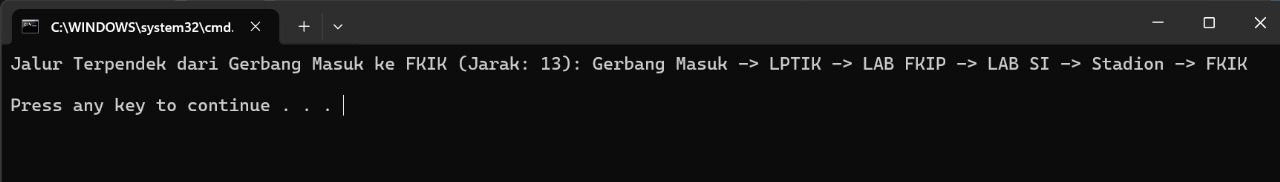
Output yang dihasilkan oleh Algoritma DFS merupakan implementasi pencarian jalur pada graf atau lokasi-lokasi yang akan dilewati pada lingkungan kampus, seperti output di atas yang menunjukkan rute perjalanan dari ‘Gerbang Masuk’ ke berbagai titik lokasi di kampus. Setiap jalur dimulai dari ‘Gerbang Masuk’ dan mengarah ke berbagai lokasi di kampus untuk sampai ke tujuan dan menunjukkan rute seberapa jauh lokasi tersebut dari ‘Gerbang Masuk’. Algoritma ini menunjukkan graf lokasi kampus lebih mudah dan juga terarah.

2.4.3 Penyelesaian jalur Algoritma Djikstra:



****

1. Di sini menggunakan struktur data unordered\_map untuk menyimpan data, dan vector di dalamnya untuk menyimpan lokasi & jarak ke lokasi lainnya.
2. Fungsi void Dijkstra di sini sebagai algoritma program untuk mencari jalur terpendek, lalu priority\_queue untuk memilih node yang berbobot paling kecil atau yang akan mengelola jarak yang sedang diproses.
3. String mulai, string tujuan berfungsi untuk mencari & menampilkan jalur terpendek dari titik awal ke tujuan. Pengeksekusian dengan menetapkan jarak dari setiap node ke int\_max kecuali titik awal yang diatur ke 0.
4. Saat pengeksekusian program menggunakan loop yaitu loop while untuk menjelajahi node sampai mencapai tujuan. Lalu node dengan bobot yang paling kecil diambil dan jaraknya akan dihitung. Kemudian jika jarak baru lebih kecil, maka jarak akan diupdate dan node akan dimasukkan ke dalam antrian. Jika jalur ditemukan, hasilnya akan disiapkan.
5. Jika menemukan jalur, maka Array akan menyimpan jalur yang ditemukan dari node tujuan ke node awal. Maka dijkstra mulai, tujuan akan dipanggil untuk menemukan dan mencetak jalur terpendek yang sudah dieksekusi.

Hasil:

Output yang dihasilkan dari Algoritma Dijkstra merupakan hasil dari pencarian jalur terpendek dari ‘Gerbang Masuk’ menuju lokasi, pengeksekusian Algoritma ini dimulai dengan pengecekan pada setiap lokasi dan juga jarak agar mendapatkan jarak terpendek dan grafik yang terarah. Output di atas menampilkan jalur terpendek dari ‘Gerbang Masuk’ ke ‘FKIK’ dengan melewati LPTIK > LAB SI > Stadion > FKIK. Maka, itu jalur terpendek dari Gerbang Masuk menuju FKIK.

**BAB 3**

**PENUTUP**

**3.1 Kesimpulan**

Berdasarkan studi kasus mengenai perjalanan Jukida, seorang calon mahasiswa yang mengikuti Ujian Tulis Berbasis Komputer di sebuah kampus, dapat disimpulkan bahwa permasalahan penentuan rute dari gerbang utama menuju gedung D tempat pelaksanaan ujian dapat direpresentasikan menggunakan konsep graf tak berarah dan berbobot. Dalam representasi ini, setiap gedung atau fasilitas di lingkungan kampus dipandang sebagai simpul (node) dalam graf, sedangkan jalan-jalan yang menghubungkan antar gedung berfungsi sebagai sisi (edge) yang memiliki bobot tertentu.

Representasi graf berbobot ini sangat relevan dalam menggambarkan situasi nyata di lapangan, di mana tidak semua hubungan antar lokasi memiliki karakteristik yang seragam. Hal ini menunjukkan bahwa dalam konteks graf berbobot, pencarian jalur terpendek tidak hanya mempertimbangkan jumlah simpul yang dilalui, melainkan juga memperhitungkan bobot total sepanjang jalur tersebut. Penerapan konsep ini sangat penting, tertama pada studi kasus diatas, penggunaan algoritma pencarian jalur terpendek seperti algoritma Dijkstra menjadi metode yang efektif untuk menentukan rute tercepat menuju gedung D. Dengan demikian, representasi graf berbobot tidak hanya memberikan gambaran struktur jaringan kampus secara menyeluruh, tetapi juga mendukung proses pengambilan keputusan berbasis data yang akurat.

**3.2 Saran**

Terdapat beberapa saran yang dapat disampaikan untuk pengembangan studi maupun implementasi ke depan yaitu, dalam membangun model graf berbobot untuk lingkungan nyata seperti kampus, disarankan untuk melakukan pengumpulan data bobot yang akurat dan representatif. Kemudian untuk mendukung pengambilan keputusan yang lebih efektif, penggunaan algoritma pencarian jalur seperti Dijkstra atau algoritma alternatif berbasis heuristik perlu dipertimbangkan dan diuji dalam berbagai skenario. Selanjutnya disarankan agar mahasiswa memperdalam pemahaman tidak hanya pada aspek teoritis graf berbobot, tetapi juga pada implementasinya melalui proyek berbasis kasus nyata.

**DAFTAR PUSTAKA**

Ardiansyah, A. H., & Wirawan, K. (2023). *Implementasi Algoritma Dijkstra untuk Menentukan Rute Terpendek pada Peta Digital*. Makalah Matematika Diskrit, Institut Teknologi Bandung.

Bondy, J. A., & Murty, U. S. R. (2008). *Graph theory* (Vol. 244). Springer.

Ginting, S. H. N., Effendi, H., Kumar, S., Marsisno, W., Sitanggang, Y. R. U., Anwar, K., ... & Smrti, N. N. E. (2024). Pengantar Struktur Data. *Penerbit Mifandi Mandiri Digital*, *1*(01).

Klarlund, N., & Schwartzbach, MI (1993, Maret). Jenis grafik. Dalam *Prosiding symposium ACM SIGPLAN-SIGACT ke-20 tentang Prinsip bahasa pemrograman* (hlm. 196-205).

Rodriguez, MA, & Neubauer, P. (2012). Pola traversal grafik. Dalam *Manajemen data grafik: Teknik dan aplikasi* (hlm. 29-46). IGI global.