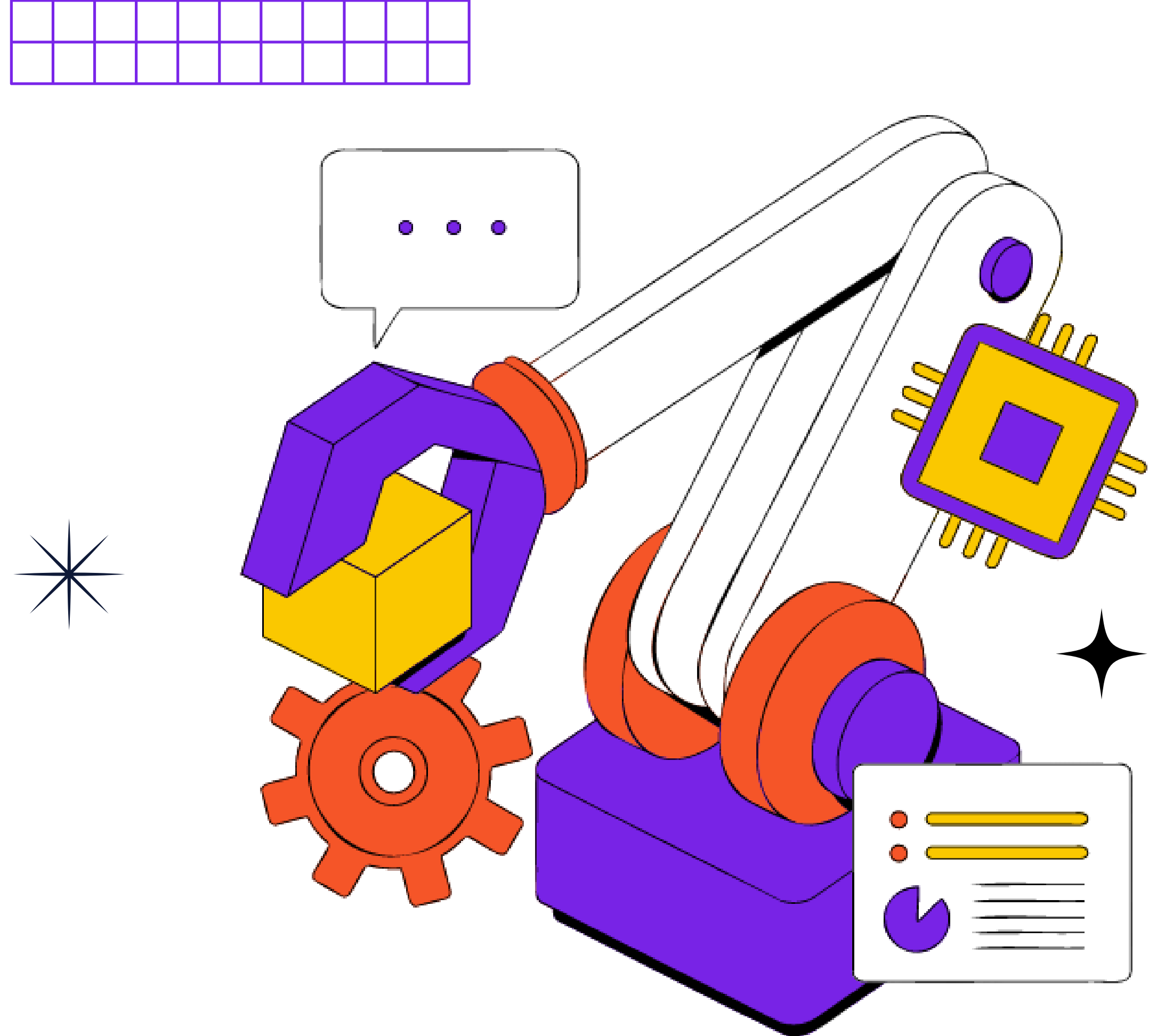


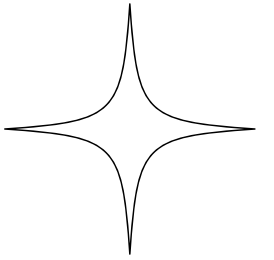
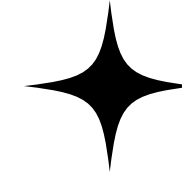
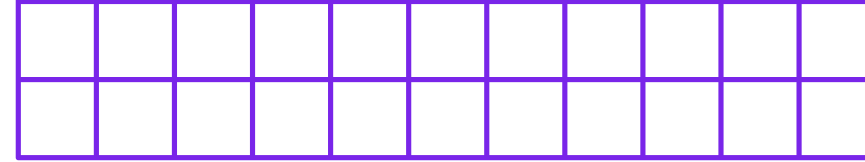
FRANS ALERO HUTAPEA
(2023071026)

MUHAMMAD AULIA RIFKI
(2023071030)

PROEJCT UAS

DESAIN DAN ANALISIS
ALGORITMA

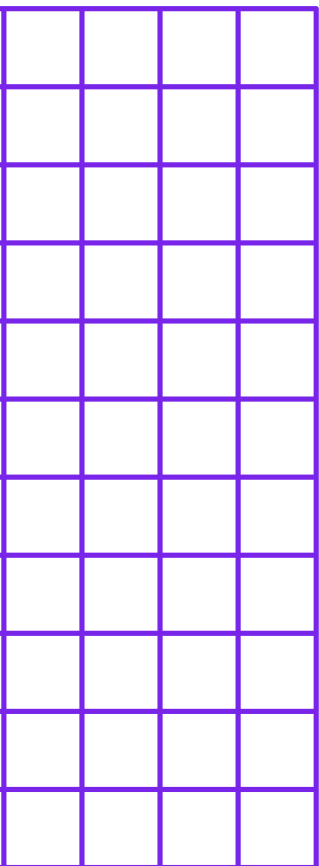
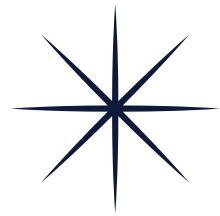
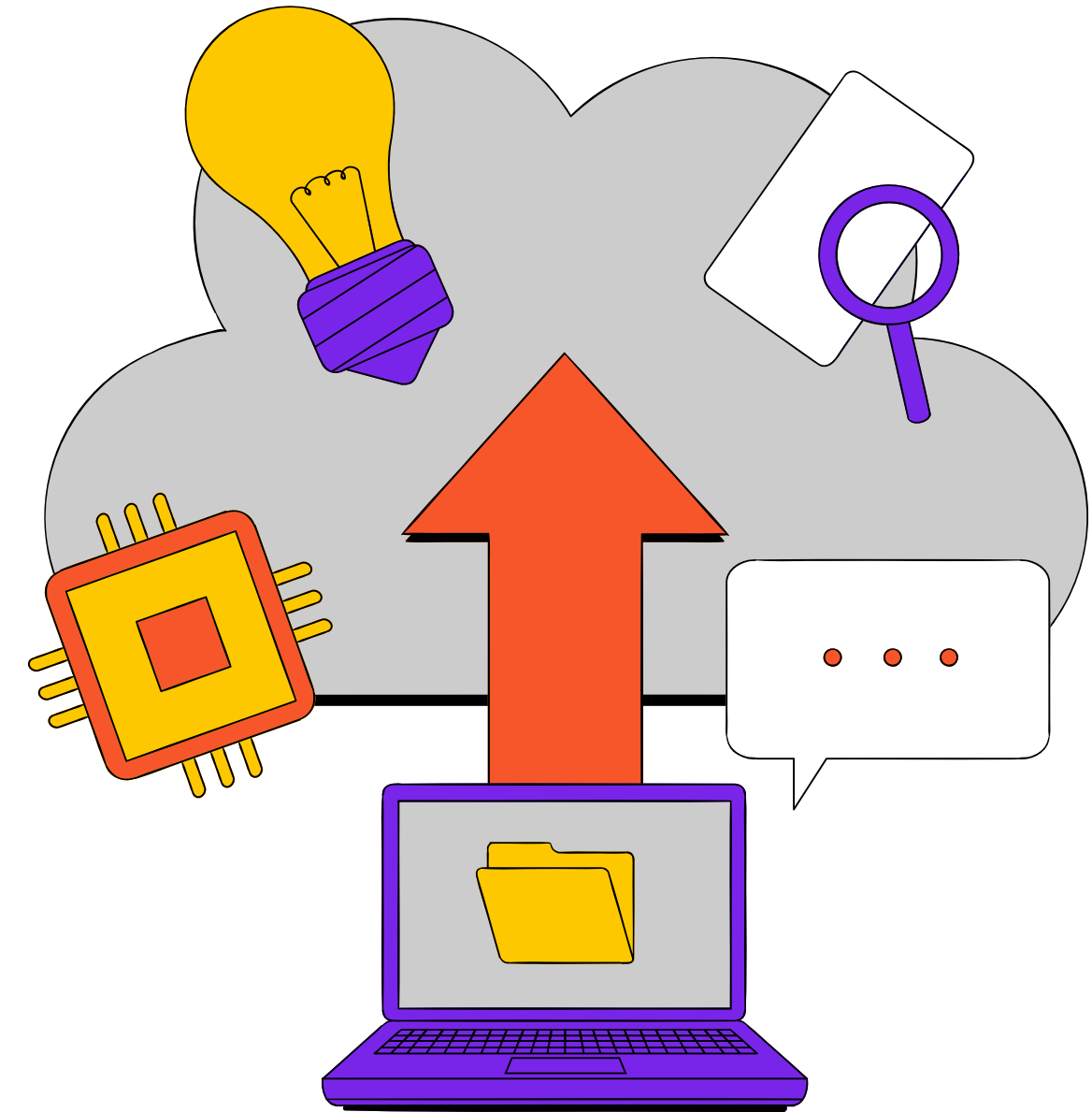


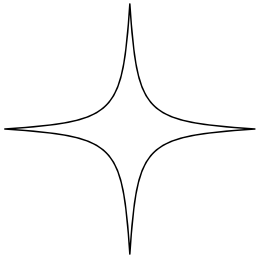
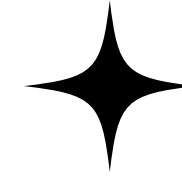
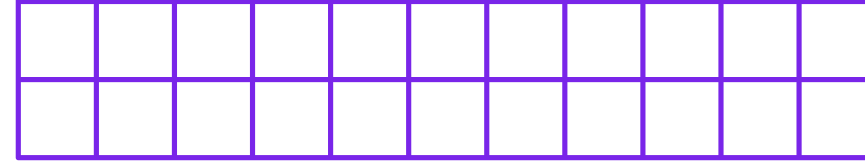


01.

MASALAH DIMASYARAKAT

Kemaacetan lalu lintas sering terjadi di sekitar **kawasan Gelora Bung Karno (GBK)**, terutama saat ada acara besar seperti pertandingan olahraga. Salah satu contohnya adalah pertandingan Timnas Indonesia melawan Australia, yang menarik banyak penonton dan menyebabkan peningkatan volume kendaraan. Kemacetan ini mencakup beberapa ruas jalan utama, seperti Jalan Sudirman, Gatot Subroto, Gerbang Pemuda, dan Asia Afrika .

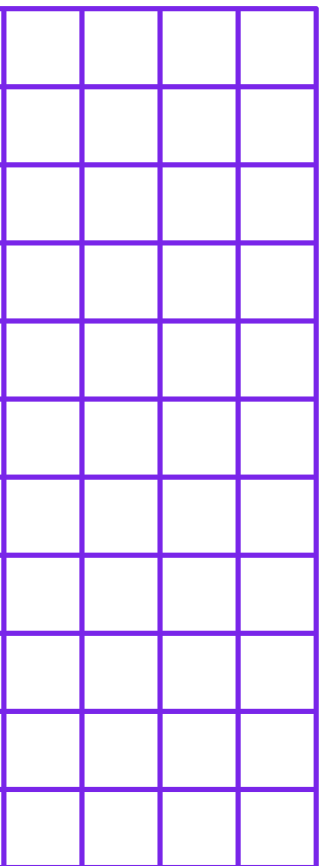
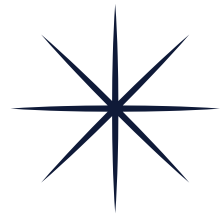
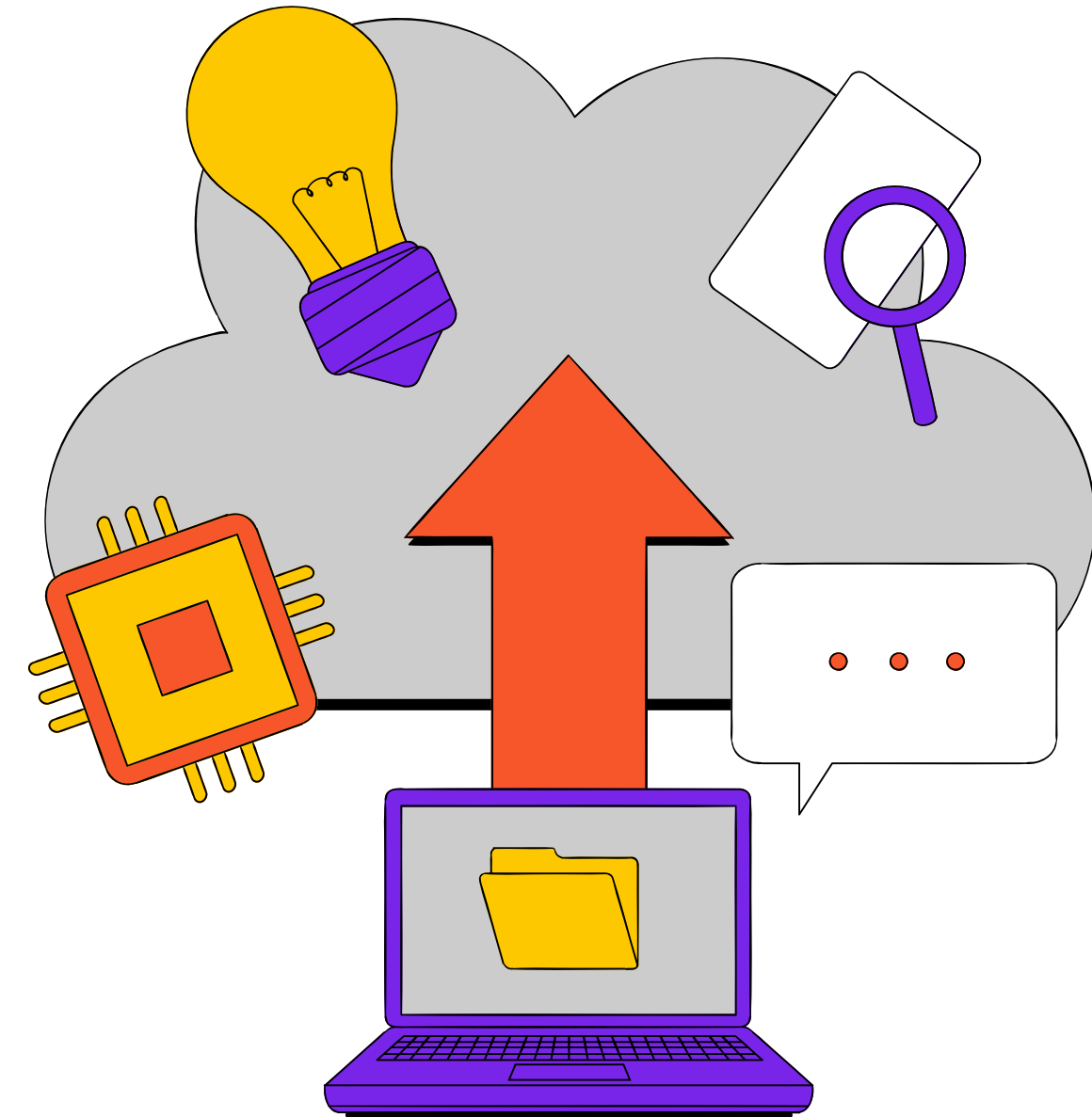


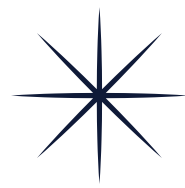


02.

MASALAH DIMASYARAKAT

Kemacetan lalu lintas merupakan masalah utama yang dihadapi masyarakat Jakarta. Setiap hari, terutama saat jam sibuk seperti pagi dan sore, banyak ruas jalan dipadati kendaraan. Meskipun terdapat berbagai pilihan transportasi umum, seperti MRT, LRT, TransJakarta, dan KRL Commuter Line, tingkat kemacetan tetap tinggi. Hal ini disebabkan oleh beberapa faktor, seperti ketergantungan masyarakat pada kendaraan pribadi, kapasitas jalan yang tidak memadai, perilaku pengguna jalan yang kurang disiplin, serta minimnya integrasi transportasi umum.

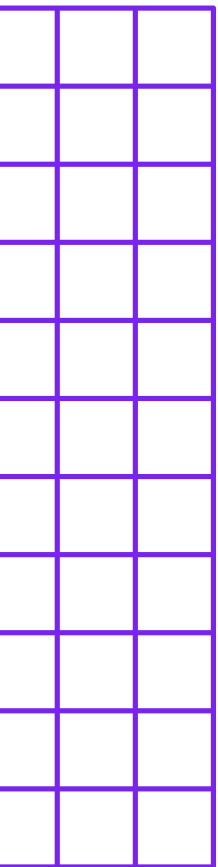


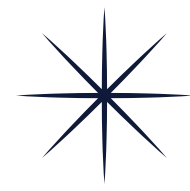


03. ALGORITMA DIJKSTRA

Algoritma Dijkstra adalah metode yang digunakan untuk menentukan jalur terpendek dalam graf berbobot non-negatif. Prinsip algoritma ini adalah memilih jalur dengan bobot terkecil pada setiap langkahnya secara iteratif untuk mencapai solusi optimal. Berdasarkan berbagai jurnal, algoritma ini telah diterapkan dalam berbagai konteks, seperti pencarian rute terpendek ke tempat wisata, rumah sakit, atau fasilitas umum lainnya. Algoritma ini efektif dalam menyelesaikan masalah pencarian jalur terpendek karena bekerja dengan prinsip greedy, yaitu memilih solusi lokal terbaik pada setiap langkah.

Penggunaannya mampu menghemat waktu dan biaya perjalanan, yang sangat penting dalam skenario seperti pariwisata atau layanan darurat. Proses kerja algoritma dimulai dengan menentukan simpul awal dan tujuan, membandingkan bobot jalur dari simpul awal ke setiap tetangga, serta memperbarui jalur jika ditemukan bobot yang lebih kecil. Aplikasi algoritma ini mencakup sektor pariwisata untuk membantu wisatawan menemukan rute tercepat menuju destinasi, dan sektor kesehatan untuk memastikan layanan darurat mencapai lokasi dengan efisiensi maksimal.



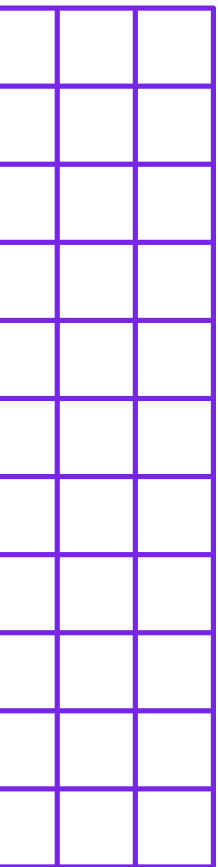


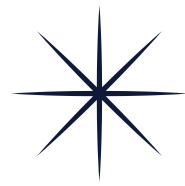
04.

RELATED WORK

PERENCANAAN RUTE WISATA DI TEMANGGUNG DENGAN JALUR TERPENDEK.

Penelitian ini, membandingkan algoritma Dijkstra dan Floyd-Warshall untuk menentukan rute terpendek antar rumah sakit di Kota Kediri. Meskipun kedua algoritma menghasilkan hasil rute yang sama, algoritma Floyd-Warshall dianggap lebih efektif karena mampu menghitung semua pasangan simpul sekaligus, sedangkan algoritma Dijkstra lebih sesuai untuk pencarian rute dari satu simpul ke simpul lainnya



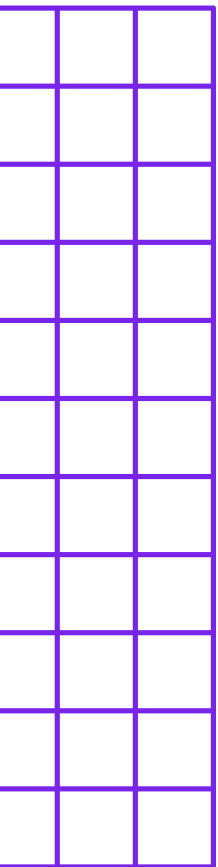


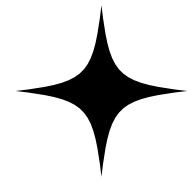
05.

RELATED WORK

IMPLEMENTASI ALGORITMA DIJKSTRA UNTUK MENENTUKAN RUTE TERPENDEK MENUJU PELAYANAN KESEHATAN

Penelitian oleh ini, mengimplementasikan algoritma Dijkstra untuk menentukan rute terpendek menuju fasilitas kesehatan. Studi ini menunjukkan keunggulan algoritma Dijkstra dalam memproses graf berbobot positif untuk menemukan jalur dengan total bobot terkecil. Prosesnya melibatkan iterasi dari simpul awal ke simpul tujuan dengan prinsip greedy untuk memilih bobot terkecil pada setiap langkah





06.

IMPLEMENTASI

```
def dijkstra(graph, start):
    queue = [(0, start)]
    distances = {start: 0}
    previous_nodes = {start: None}
    visited = set()

    while queue:
        current_distance, current_node = heapq.heappop(queue)

        if current_node in visited:
            continue

        visited.add(current_node)

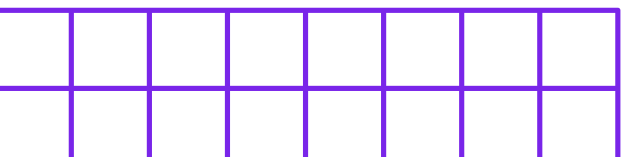
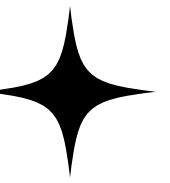
        for neighbor, weight in graph.get(current_node, []):
            if neighbor in visited:
                continue

            new_distance = current_distance + weight

            # Update jarak terpendek jika ditemukan yang lebih pendek
            if neighbor not in distances or new_distance < distances[neighbor]:
                distances[neighbor] = new_distance
                previous_nodes[neighbor] = current_node
                heapq.heappush(queue, (new_distance, neighbor))

    return distances, previous_nodes
```

Fungsi Dijkstra dalam kode ini digunakan untuk mencari jalur terpendek dari simpul awal ke semua simpul lainnya dalam graf berbobot. Fungsi menerima dua parameter: graph, yang merepresentasikan graf dalam bentuk dictionary, dan start, yaitu simpul awal. Fungsi menggunakan **priority queue** (queue) untuk memproses simpul berdasarkan jarak terpendek yang ditemukan. Variabel distances mencatat jarak terpendek dari simpul awal ke setiap simpul, sedangkan **previous_nodes** melacak jalur dengan mencatat simpul sebelumnya. Proses dimulai dengan mengambil simpul dengan jarak terpendek dari queue. Untuk setiap tetangga simpul yang sedang diproses, jarak baru dihitung, dan jika lebih pendek dari jarak sebelumnya yang tercatat, jarak diperbarui, dan simpul ditambahkan kembali ke **queue**. Fungsi ini terus berjalan hingga semua simpul diproses. Outputnya adalah **distances**, yang berisi jarak terpendek ke semua simpul, dan **previous_nodes**, yang mencatat jalur terpendek dari simpul awal ke simpul tujuan. Fungsi ini merupakan inti dari algoritma Dijkstra dan sangat efisien untuk menyelesaikan masalah jalur terpendek.





07.

IMPLEMENTASI



```
# Representasi graf jalan di kawasan GBK
graph = {
    "Sudirman": [("Gatot Subroto", 4), ("Asia Afrika", 6)],
    "Gatot Subroto": [("Sudirman", 4), ("Gerbang Pemuda", 4)],
    "Asia Afrika": [("Sudirman", 6), ("Gerbang Pemuda", 5)],
    "Gerbang Pemuda": [("Gatot Subroto", 4), ("Asia Afrika", 5)],
}
```

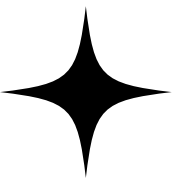
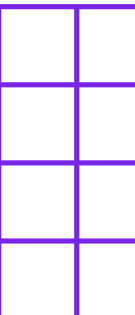
Graf direpresentasikan sebagai dictionary, di mana kunci (key) adalah nama simpul, dan nilainya adalah daftar tuple berisi tetangga simpul dan bobot jalur. Misalnya, "Sudirman": [("Gatot Subroto", 4), ("Asia Afrika", 6)] menunjukkan bahwa simpul "Sudirman" terhubung ke "Gatot Subroto" dengan bobot 4 dan ke "Asia Afrika" dengan bobot 6. Representasi ini digunakan sebagai input untuk algoritma Dijkstra untuk menghitung jalur dan jarak terpendek.

```
def get_shortest_path(previous_nodes, start, goal):
    path = []
    current_node = goal

    while current_node is not None:
        path.append(current_node)
        current_node = previous_nodes[current_node]

    path.reverse()
    return path
```

Fungsi ini digunakan untuk merekonstruksi jalur terpendek dari simpul tujuan (*goal*) ke simpul awal (*start*) berdasarkan *dictionary previous_nodes*, yang berisi informasi tentang simpul sebelumnya di jalur terpendek. Fungsi menambahkan simpul dari tujuan ke jalur (*path*) dengan menelusuri simpul sebelumnya hingga mencapai simpul awal, lalu membalik daftar tersebut menggunakan *reverse()* untuk menghasilkan jalur dalam urutan yang benar. Outputnya adalah daftar berisi jalur terpendek dari awal ke tujuan.





08.

IMPLEMENTASI



```
G = nx.Graph()

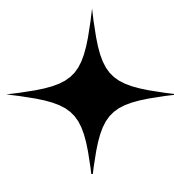
# Menambahkan simpul dan bobotnya
for node, neighbors in graph.items():
    for neighbor, weight in neighbors:
        G.add_edge(node, neighbor, weight=weight)

# Visualisasi graf
pos = nx.spring_layout(G)
plt.figure(figsize=(8, 6))

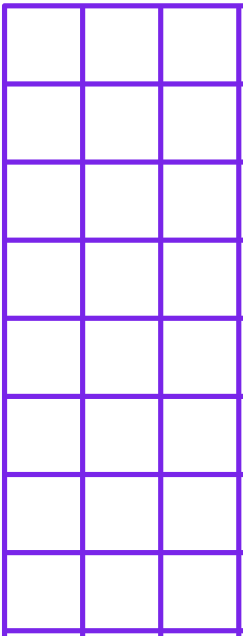
# Gambar graf
nx.draw(G, pos, with_labels=True, node_size=3000, node_color='skyblue', font_size=12, font_weight='bold', edge_color='gray')

# Menambahkan label bobot pada sisi
edge_labels = nx.get_edge_attributes(G, 'weight')
nx.draw_networkx_edge_labels(G, pos, edge_labels=edge_labels)

# Menyimpan gambar
plt.title("Visualisasi Graf Rute Kawasan GBK")
plt.show()
```



Kode ini memanfaatkan pustaka **NetworkX** dan **Matplotlib** untuk memvisualisasikan graf berbobot. Graf dibuat menggunakan ***nx.Graph()*** dan simpul beserta bobot jalurnya ditambahkan menggunakan ***add_edge***. Tata letak graf diatur dengan ***spring_layout*** untuk menampilkan simpul secara estetis. Graf digambar menggunakan ***nx.draw***, dengan simpul berwarna biru muda, label simpul, dan jalur digambarkan. Bobot jalur ditampilkan di sisi graf menggunakan ***draw_networkx_edge_labels***. Visualisasi ini menampilkan struktur graf dengan rute dan bobot jalurnya.



08. HASIL DAN PEMBAHASAN

Contoh Data yang Diuji dalam Algoritma

Dataset Graf

Graf yang digunakan merepresentasikan hubungan antar lokasi di kawasan GBK. Data mencakup simpul (nodes) dan bobot (edges), yang menunjukkan jarak antar simpul.

Simpul Awal	Simpul Tujuan	Bobot (Jarak dalam menit)
Sudirman	Gatot Subroto	4
Sudirman	Asia Afrika	6
Gatot Subroto	Gerbang Pemuda	4
Asia Afrika	Gerbang Pemuda	5

Hasil Pengujian Algoritma Dijkstra

Hasil pengujian Algoritma Dijkstra dengan data di atas menghasilkan jarak terpendek sebagai berikut:

Simpul Awal	Simpul Tujuan	Jarak Terpendek (menit)	Jalur Terpendek
Sudirman	Gerbang Pemuda	8	Sudirman -> Gatot Subroto -> Gerbang Pemuda

Graf merepresentasikan hubungan antar lokasi di kawasan GBK dengan simpul sebagai lokasi dan bobot berupa jarak dalam menit. Dataset mencakup jalur dari Sudirman ke Gatot Subroto (4 menit), Sudirman ke Asia Afrika (6 menit), Gatot Subroto ke Gerbang Pemuda (4 menit), dan Asia Afrika ke Gerbang Pemuda (5 menit). Hasil pengujian algoritma Dijkstra menunjukkan jarak terpendek dari Sudirman ke Gerbang Pemuda adalah 8 menit melalui jalur Sudirman → Gatot Subroto → Gerbang Pemuda.

09.

KESIMPULAN

Algoritma Dijkstra terbukti efektif dalam menentukan rute tercepat dan alternatif untuk mengatasi kemacetan di kawasan GBK. Dengan memanfaatkan data real-time, algoritma ini dapat diintegrasikan ke dalam sistem navigasi dan manajemen lalu lintas pintar untuk meningkatkan mobilitas di kawasan perkotaan. Penelitian lanjutan dapat dilakukan untuk menerapkan algoritma ini pada skala yang lebih besar, seperti seluruh jaringan jalan di Jakarta, dengan integrasi data lalu lintas real-time dari sensor jalan, guna menciptakan sistem transportasi yang lebih efisien dan responsif.





THANK
YOU