# Implementasi Algoritma A* pada Perencanaan Jalur Transportasi Umum di Kota Bekasi

Anas Ghazi Al Gifari - 13523159
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: anasghassie75@gmail.com , 13523159@std.stei.itb.ac.id

***Abstract*—Permasalahan kemacetan dan kurang optimalnya trayek transportasi umum di Kota Bekasi menuntut adanya sistem perencanaan jalur yang efisien dan adaptif. Makalah ini mengusulkan penerapan algoritma A* untuk perencanaan rute transportasi umum dengan membangun model jaringan jalan menggunakan data OpenStreetMap yang diolah melalui OSMnx dan NetworkX. Sistem yang dikembangkan memetakan titik-titik strategis kota sebagai simpul, lalu menentukan rute terpendek antara dua titik dengan mempertimbangkan bobot jarak tiap ruas jalan. Empat skenario pengujian pada pasangan node berjauhan menunjukkan bahwa algoritma A* mampu menemukan rute optimal yang mengikuti ruas jalan utama secara akurat, konsisten, dan logis secara spasial. Hasil visualisasi peta interaktif memperkuat keunggulan sistem dalam merekomendasikan jalur tercepat dan paling relevan bagi pengguna maupun perencana transportasi. Sistem ini diharapkan dapat menjadi fondasi dalam pengembangan layanan perencanaan rute transportasi umum, serta menjadi referensi untuk implementasi serupa di kota-kota besar lainnya.**

***Keywords—A*, perencanaan rute, jaringan jalan, transportasi umum, Kota Bekasi, OSMnx, NetworkX.***

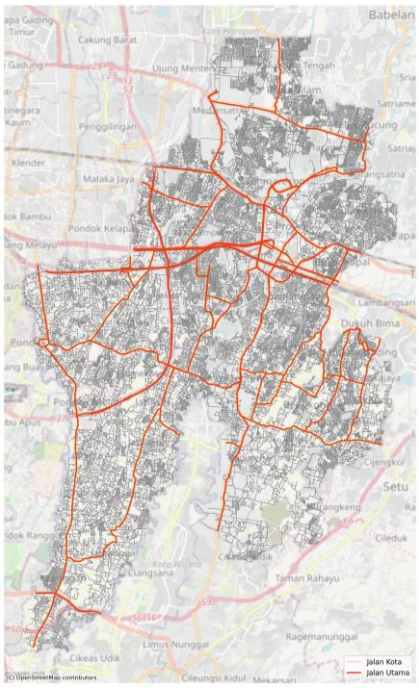
## I. PENDAHULUAN

Kota Bekasi merupakan kawasan perkotaan dengan pertumbuhan penduduk dan ekonomi yang pesat di Indonesia. Sebagai kota penyangga utama Jakarta, Bekasi menghadapi peningkatan kebutuhan mobilitas, tapi perbaikan transportasi umumnya belum memadai. Permasalahan yang muncul antara lain kepadatan lalu lintas, terbatasnya trayek angkutan umum seperti Trans Patriot dan angkot, serta kurangnya integrasi antar moda yang membuat perpindahan rute kurang efisien.

Gambar 1.1. Trans Patriot Bekasi
(Sumber: Trans Patriot Bekasi (Kompas, 2024))

Situasi ini diperparah oleh faktor eksternal seperti cuaca ekstrem yang dapat mengganggu kelancaran lalu lintas serta pertumbuhan wilayah permukiman baru yang belum sepenuhnya terlayani angkutan umum. Akibatnya, banyak warga Bekasi yang terpaksa memilih moda transportasi pribadi, sehingga beban jalan raya pun kian bertambah. Perlu adanya upaya meningkatkan pelayanan transportasi melalui perluasan trayek dan pengembangan sistem angkutan massal, tapi tantangan dalam perencanaan rute yang optimal masih menjadi hambatan besar dalam proses pengambilan keputusan.

Di era digital dan keterbukaan data spasial saat ini, peluang pemanfaatan teknologi untuk mendukung perencanaan transportasi semakin terbuka lebar. Salah satu metode yang efektif dalam pencarian rute terpendek adalah algoritma A*. Dengan ketersediaan data jaringan jalan melalui platform OpenStreetMap dan alat bantu seperti OSMnx, kini dimungkinkan untuk melakukan perencanaan jalur transportasi yang benar-benar mengikuti kondisi jalan di lapangan, bukan sekadar hasil analisis berbasis peta statis atau asumsi trayek.




Gambar 1.2. Peta Jaringan Jalan Kota Bekasi
(Sumber: Dokumentasi Pribadi)

Pengembangan program dalam makalah ini berangkat dari kebutuhan untuk menghadirkan solusi perencanaan jalur transportasi umum yang efisien, adaptif, dan mudah diintegrasikan dengan data spasial terkini di Bekasi. Program yang dikembangkan bertujuan untuk membantu menemukan rute optimal antara dua titik strategis, tidak hanya dari sisi jarak tempuh, tetapi juga dengan mempertimbangkan keterhubungan jaringan jalan dan karakteristik wilayah urban Bekasi. Dengan demikian, hasil program diharapkan dapat digunakan sebagai landasan pengambilan keputusan baik oleh pemerintah, operator transportasi, maupun masyarakat umum yang membutuhkan rekomendasi jalur tercepat dan paling efektif untuk mobilitas harian mereka.

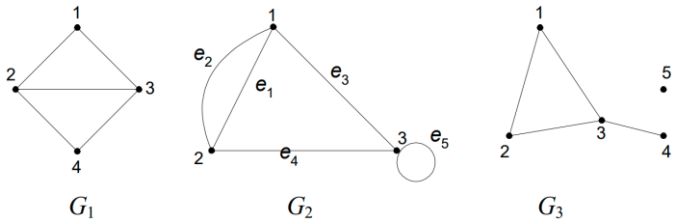
Pada akhirnya, program ini diharapkan dapat menjadi contoh implementasi nyata pemanfaatan algoritma optimasi dalam transportasi perkotaan, serta memberikan kontribusi dalam upaya pengembangan sistem transportasi umum yang terintegrasi, adaptif, dan berkelanjutan di Kota Bekasi maupun kota-kota besar lainnya di Indonesia.

## II. Landasan Teori

### A. *Graf*

Graf merupakan suatu struktur yang menggambarkan penghubungan antara himpunan elemen-elemen tidak kosong berupa titik yang disebut simpul dengan himpunan pasangan tidak terurut titik-titik tersebut yang dihubungkan oleh suatu garis yang disebut sisi. Graf digunakan untuk merepresentasikan objek-objek diskrit dan hubungan-hubungan antarobjek tersebut. Representasi visual dari graf adalah dengan menyatakan objek sebagai simpul dan hubungan antarobjek sebagai sisi.

Graf $G$ didefinisikan sebagai pasangan himpunan $(V, E)$, ditulis dengan notasi $G = (V, E)$, dengan $V$ adalah himpunan tidak kosong dari simpul (*vertex*) dan $E$ adalah himpunan sisi (*edges*) yang menghubungkan sepasang simpul. Definisi tersebut menyatakan bahwa $V$ tidak boleh kosong, sedangkan $E$ boleh. Simpul pada graf dapat dinyatakan dengan huruf, bilangan, atau gabungan keduanya. Sedangkan sisi-sisi yang menghubungkan simpul $u$ dengan simpul $v$ dinyatakan dengan pasangan $(u, v)$ atau dinyatakan dengan lambang $e_1, e_2, e_3$, dan seterusnya. Dengan kata lain, jika $e$ adalah sisi yang menghubungkan simpul $u$ dengan simpul $v$, $e$ dapat dituliskan sebagai $e = (u, v)$.



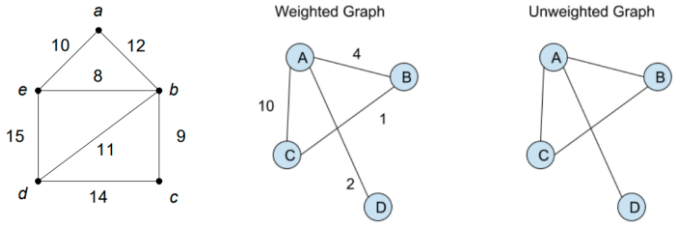

Gambar 2.1. Contoh-Contoh Graf
(Sumber: 20-Graf-Bagian1-2024.pdf)

### B. *Graf Berbobot*

Graf berbobot adalah graf di mana setiap sisi (*edge*) diberikan sebuah nilai atau bobot tertentu. Bobot ini biasanya berupa bilangan real yang merepresentasikan karakteristik hubungan antar simpul, seperti jarak, waktu tempuh, atau biaya.

Dalam banyak aplikasi dunia nyata, penggunaan bobot pada sisi sangat penting untuk mencerminkan parameter yang relevan, misalnya dalam pemodelan peta jalan, bobot dapat berupa jarak fisik antar persimpangan, waktu perjalanan, atau biaya tol. Keberadaan bobot pada sisi memungkinkan algoritma graf mempertimbangkan faktor kuantitatif dalam proses pencarian lintasan terpendek atau perhitungan lainnya. Sebaliknya, graf yang tidak memiliki bobot pada sisi disebut graf tak berbobot, di mana semua sisi dianggap memiliki bobot yang sama.

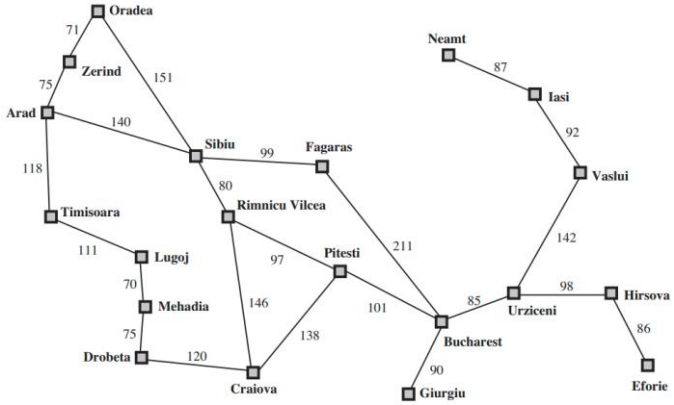
Dengan demikian, graf berbobot menjadi struktur yang sangat relevan dalam berbagai aplikasi perencanaan rute, optimasi jaringan, dan analisis sistem yang memerlukan pengukuran nilai pada hubungan antar simpul.




Gambar 2.2. Ilustrasi Graf Berbobot dan Graf Tak Berbobot
(Sumber: 20-Graf-Bagian1-2024.pdf)

### C. *Perencanaan Rute*

Perencanaan rute adalah proses penentuan jalur terbaik dari satu lokasi ke lokasi lain pada suatu jaringan jalan atau graf transportasi. Secara umum, perencanaan rute bertujuan mencari rute tercepat, terpendek, atau paling efisien sesuai kriteria tertentu (misal waktu tempuh minimum atau jarak terpendek). Dalam konteks komputasi, perencanaan rute dipandang sebagai masalah pencarian lintasan terpendek (shortest path) pada graf berbobot yang merepresentasikan jaringan jalan. Lintasan terpendek didefinisikan sebagai lintasan dengan total bobot minimum yang diperlukan untuk mencapai suatu simpul tujuan dari suatu simpul awal pada graf.



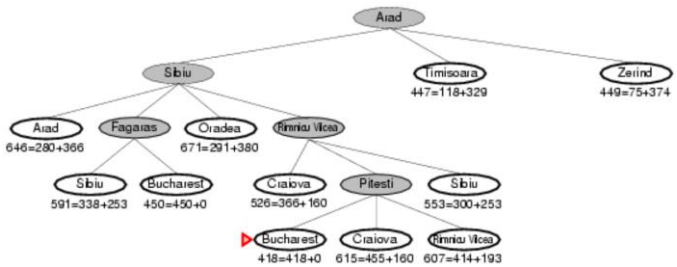

Gambar 2.3. Contoh Graf Kota dan Jarak Antar Simpul
(Sumber: Russell & Norvig, *Artificial Intelligence: A Modern Approach*)

Pada sistem transportasi nyata, graf jalan sering kali berukuran sangat besar dengan ribuan hingga jutaan simpul dan sisi. Oleh karena itu, perencanaan rute membutuhkan algoritma yang efisien. Metode penelusuran sederhana seperti Breadth-First Search atau Depth-First Search tidak cukup karena harus mengabaikan bobot, sementara algoritma seperti Dijkstra dapat menemukan lintasan optimal pada graf berbobot non-negatif. Namun, algoritma Dijkstra (yang ekuivalen dengan Uniform Cost Search) bisa menjadi lambat untuk jaringan yang sangat besar. Di sinilah algoritma A* berperan penting, karena menggabungkan kekuatan penelusuran berbasis biaya dan informasi heuristik untuk memangkas ruang pencarian sehingga lebih efisien.

### *D. Algoritma A**

Algoritma A* (dibaca "A star") merupakan algoritma pencarian jalur terpendek yang terkenal efisien dan banyak digunakan dalam pemetaan dan AI. A* bekerja dengan menghitung suatu fungsi evaluasi $f(n)$ untuk setiap simpul $n$, yang merupakan penjumlahan dari dua komponen: $g(n)$ dan $h(n)$. Komponen $g(n)$ merepresentasikan biaya terakumulasi dari simpul awal hingga simpul $n$ (misalnya, total jarak atau waktu tempuh yang sudah dilalui), sedangkan $h(n)$ adalah estimasi biaya dari simpul $n$ menuju simpul tujuan (nilai heuristik). Secara matematis dinyatakan $f(n) = g(n) + h(n)$. Pada setiap langkah iterasi, A* akan memilih untuk mengembangkan simpul dengan nilai $f(n)$ terendah, karena simpul tersebut dianggap paling menjanjikan berada di jalur terpendek.

Pendekatan ini menggabungkan kelebihan pencarian biaya terkecil dan pencarian heuristik. Apabila fungsi heuristik $h(n)$ diatur selalu nol untuk semua simpul, maka $f(n) = g(n)$ sehingga algoritma A* berperilaku sama dengan algoritma Dijkstra yang menjamin solusi optimal, namun tanpa informasi arah ke tujuan. Sebaliknya, jika $g(n)$ diabaikan dan algoritma hanya mengandalkan $h(n)$, yaitu $f(n) = h(n)$, maka algoritma berubah menjadi Greedy Best-First Search yang lebih mengutamakan kecepatan mencapai tujuan namun tidak optimal secara global. Dengan memadukan keduanya, A* mampu mencapai keseimbangan dengan mencari lintasan dengan mempertimbangkan biaya terpakai sekaligus mengarahkan pencarian ke arah tujuan.




Gambar 2.4. Ilustrasi Algoritma A*
(Sumber: 22-Route-Planning-(2025)-Bagian2.pdf)

Algoritma A* akan menemukan lintasan terpendek yang optimal asalkan fungsi heuristik yang digunakan memenuhi sifat tertentu. Secara khusus, heuristik harus *admissible* dan *consistent*. Heuristik dikatakan *admissible* jika untuk setiap simpul $n$, nilai $h(n)$ tidak pernah melebihi biaya sebenarnya dari $n$ ke tujuan ($h(n) \leq h^*(n)\ \forall n$). Dengan kata lain, heuristik selalu optimis atau tidak terlalu besar (sering kali heuristik berupa perkiraan jarak terdekat yang tidak boleh melebihi jarak sesungguhnya). Contoh heuristik yang *admissible* dalam konteks jaringan jalan adalah jarak garis lurus (straight-line distance) ke tujuan, karena jarak garis lurus tidak akan melebihi jarak rute jalan sebenarnya. Selain itu, heuristik sebaiknya juga konsisten (monotonik), artinya memenuhi kondisi $h(n) \leq c(n,n') + h(n')$ untuk setiap edge $(n,n')$, sehingga estimasi biaya memenuhi segitiga. Jika heuristik *admissible* dan konsisten, maka algoritma A* terjamin akan menemukan solusi lintasan terpendek yang optimal dan lengkap. Dalam situasi heuristik yang tidak admissible, A* mungkin lebih cepat namun berisiko gagal menemukan solusi optimal.

### *E. Fungsi Heuristik*

Fungsi heuristik adalah komponen kunci dalam algoritma A*. Secara umum, heuristik memberikan estimasi atau perkiraan biaya dari suatu simpul tertentu ke simpul tujuan. Dalam AI, heuristik berfungsi sebagai "pengetahuan tambahan" yang memandu pencarian ke arah yang lebih menjanjikan. Fungsi heuristik $h(n)$ dapat dianggap sebagai aturan atau metode untuk menilai seberapa dekat suatu keadaan (simpul $n$) dengan keadaan tujuan. Menurut literatur, kata heuristic sendiri berasal dari bahasa Yunani heuriskein yang berarti "menemukan" atau "mencari" solusi. Dalam konteks pencarian, heuristik didefinisikan sebagai suatu fungsi yang menghasilkan nilai berupa biaya perkiraan (estimasi) dari sebuah simpul menuju solusi atau tujuan.

Dalam penerapan algoritma A*, fungsi heuristik harus dirancang sesuai dengan permasalahan. Misalnya, pada perencanaan rute di peta, heuristik yang lazim digunakan adalah estimasi jarak lurus (Euclidean) atau jarak Manhattan dari lokasi sekarang ke lokasi tujuan, tergantung karakteristik jaringan jalan. Heuristik ini memberikan nilai perkiraan jarak yang harus ditempuh, tanpa memperhitungkan rintangan atau keterbatasan jalan sebenarnya. Meskipun demikian, apabila heuristik dirancang dengan baik (admissible dan konsisten seperti dibahas sebelumnya), A* akan tetap menghasilkan lintasan terpendek. Sebaliknya, heuristik yang buruk (misal terlalu besar melebihi jarak sebenarnya) dapat membuat A* kehilangan jaminan optimalitas, sedangkan heuristik yang terlalu kecil (misal selalu nol) membuat A* kehilangan keunggulan efisiensinya dibanding algoritma pencarian buta. Oleh karena itu, perancangan fungsi heuristik merupakan aspek krusial dalam sistem perencanaan rute berbasis A*.

### *F. Jaringan Jalan Berskala Besar*

Salah satu tantangan dalam perancangan sistem perencanaan jalur transportasi adalah menangani jaringan jalan berskala besar. Graf jalan untuk level kota, provinsi, hingga negara biasanya memiliki jumlah simpul dan sisi yang sangat banyak sedemikian sehingga dapat mencapai ratusan ribu hingga jutaan. Kalkulasi lintasan terpendek pada graf sebesar

ini memerlukan algoritma yang sangat efisien baik dalam waktu maupun penggunaan memori.

Algoritma A* secara signifikan mempercepat pencarian dengan memotong cabang pencarian yang kurang menjanjikan menggunakan heuristik. Namun, untuk skala yang sangat besar (misal jaringan jalan seluruh benua), bahkan A* pun perlu didukung teknik tambahan. Riset terbaru di bidang route planning menunjukkan bahwa dengan berbagai optimisasi, perhitungan rute pada jaringan jalan berskala kontinental dapat dijawab dalam hitungan milisekon saja. Beragam teknik seperti prapemrosesan graf (contohnya Contraction Hierarchies, Transit Nodes, dan Landmark-based A seperti ALT) telah dikembangkan untuk memangkas waktu query hingga beberapa orde magnitudo dibandingkan algoritma klasik. Beberapa algoritma mampu menjawab kueri pencarian rute hampir seketika (di bawah satu milisekon) dengan memanfaatkan data prapemrosesan yang cukup besar. Tentu, ada pertukaran antara waktu prapemrosesan, ruang penyimpanan, dan kecepatan jawaban kueri.

Untuk konteks tugas perancangan sistem yang berbasis algoritma A* murni tanpa optimisasi tambahan, kinerja pada graf berskala besar tetap perlu diperhatikan. Pendekatan-pendekatan seperti pembatasan ruang lingkup pencarian dengan heuristik yang agresif, atau membagi masalah menjadi sub-masalah (misalnya dengan hirarki jalan utama vs jalan lokal), sering digunakan untuk menangani skala besar. Meskipun A* optimal secara waktu pencarian dibandingkan algoritma uninformed, kompleksitas terburuknya masih eksponensial terhadap panjang solusi dalam graf umum. Namun, pada graf jalan yang memiliki struktur khusus (sparsity tinggi, derajat rata-rata rendah, dan adanya pola geografis), A* dengan heuristik geografis (jarak udara) biasanya dapat bekerja sangat efektif mendekati garis lurus ke tujuan, sehingga hanya mengeksplor area lokal di sekitar lintasan optimal.

### *G. NetworkX dan OSMnx*

NetworkX adalah pustaka (library) Python yang populer untuk representasi dan analisis graf dan jaringan. Pustaka ini menyediakan struktur data untuk graf serta fungsi-fungsi algoritmik yang berhubungan dengan graf. NetworkX bersifat open-source dan banyak digunakan dalam riset maupun pengembangan aplikasi karena kemudahannya dalam memodelkan graf kompleks. Menurut dokumentasi resminya, "*NetworkX is a Python library for studying graphs and networks*". Pustaka ini mendukung berbagai tipe graf, termasuk graf tak berarah, berarah, multigraf, dan graf dengan bobot. Pengguna dapat dengan mudah membangun graf dari data, menambahkan simpul dan sisi, serta menyimpan atribut (seperti bobot, label, koordinat, dll.) pada elemen graf tersebut.

Salah satu alasan menggunakan NetworkX dalam perancangan sistem perencanaan rute adalah ketersediaan algoritma pencarian lintasan terpendek secara siap pakai. NetworkX memiliki implementasi algoritma klasik seperti Dijkstra untuk graf berbobot, dan bahkan telah menyertakan implementasi algoritma A* sejak versi 1.0. Dengan demikian, pengembang dapat memanfaatkan fungsi pustaka, misalnya networkx.astar_path(G, source, target, heuristic), untuk langsung menghitung rute terpendek berbasis A* pada graf $G$ yang merepresentasikan jaringan jalan. NetworkX juga menyediakan berbagai fungsi analisis graf lainnya (degree, centrality, connected components, dll.) yang dapat membantu dalam memahami struktur jaringan jalan yang ditangani. Meskipun NetworkX ditulis murni dalam Python (sehingga mungkin kurang efisien untuk graf yang sangat besar dibanding pustaka C/C++), kemudahannya menjadikannya pilihan tepat untuk prototipe. Untuk skala data yang lebih besar, integrasi dengan pustaka lain juga dimungkinkan, misalnya NetworkX 3.x mendukung penggunaan backend GraphBLAS atau cuGraph (GPU) untuk mempercepat komputasi graf berat.

Sementara itu, OSMnx adalah pustaka Python yang dirancang untuk mempermudah pengambilan dan manipulasi data jaringan jalan dari OpenStreetMap. Pustaka ini berguna dalam konteks perencanaan rute karena dapat secara otomatis mengunduh peta jalan suatu wilayah dan membentuk graf jalan yang siap digunakan untuk analisis atau pencarian rute. Menurut dokumentasi resminya, "*OSMnx is a Python package to easily download, model, analyze, and visualize street networks and other geospatial features from OpenStreetMap.*". Dengan OSMnx, pengguna cukup memasukkan nama kota atau koordinat geografis, dan pustaka ini akan mengambil data OpenStreetMap untuk wilayah tersebut, kemudian membangunnya menjadi graf NetworkX dengan simpul-simpul sebagai persimpangan jalan dan sisi-sisi sebagai segmen jalan. Bobot seperti panjang jalan atau estimasi waktu tempuh juga dapat ditambahkan sebagai atribut sisi, karena OSMnx menghitung jarak setiap segmen jalan secara otomatis.

Pustaka OSMnx juga menyediakan fungsi-fungsi untuk melakukan analisis jaringan jalan, seperti menghitung statistik jaringan (panjang total jalan, jumlah persimpangan), melakukan *graph simplification* (menggabungkan node-node derajat 2 untuk menyederhanakan graf), serta kemampuan visualisasi peta jaringan. Untuk keperluan perencanaan rute, OSMnx sering dipasangkan dengan NetworkX. Setelah graf jalan diperoleh melalui OSMnx, algoritma NetworkX (misalnya A* atau Dijkstra) dapat diaplikasikan pada graf tersebut untuk menemukan rute terpendek. Contohnya, OSMnx memiliki fungsi ox.shortest_path() yang di balik layar memanfaatkan algoritma NetworkX untuk mencari jalur terpendek pada graf jalan yang diekstrak. Dengan kombinasi OSMnx dan NetworkX, proses pengambilan data peta, pembentukan graf, hingga pencarian rute dapat dilakukan secara utuh dalam bahasa Python dengan relatif sedikit kode.

## III. ANALISIS DAN PEMBAHASAN

### *A. Identifikasi Permasalahan*

Permasalahan transportasi umum di Kota Bekasi semakin kompleks seiring dengan pertumbuhan kota yang pesat. Jalanan utama di Bekasi semakin padat, terutama pada jam-jam sibuk, sehingga mengakibatkan kemacetan yang kian parah. Selain itu, cuaca ekstrem seperti hujan lebat juga kerap memperburuk

situasi lalu lintas, menambah ketidakpastian waktu tempuh bagi pengguna transportasi umum.

Moda transportasi Trans Patriot sebagai salah satu tulang punggung angkutan umum di Bekasi masih menghadapi keterbatasan, baik dari jumlah trayek yang tersedia maupun cakupan wilayah pelayanannya. Di sisi lain, sistem transportasi umum di Bekasi masih kurang terintegrasi, sehingga penumpang sering kali harus berpindah moda dengan jalur yang tidak efisien.

### *B. Studi Kasus Kota Bekasi*

Kota Bekasi mengalami pertumbuhan pesat dalam beberapa dekade terakhir. Sebagai kota penyangga utama Jakarta, arus migrasi ke Bekasi terus meningkat, didorong oleh pembangunan perumahan, pusat bisnis, dan kawasan industri. Lonjakan jumlah penduduk ini berimplikasi langsung pada peningkatan mobilitas masyarakat, sehingga permintaan terhadap sistem transportasi umum yang andal juga meningkat.

Namun, infrastruktur transportasi di Bekasi belum sepenuhnya mampu mengimbangi laju pertumbuhan ini. Rute angkutan umum masih belum merata, terutama ke wilayah-wilayah permukiman baru. Hal ini menyebabkan aksesibilitas ke fasilitas penting seperti stasiun, terminal, pusat perbelanjaan, dan rumah sakit menjadi tidak optimal bagi sebagian besar masyarakat.

Makalah ini mengambil Kota Bekasi sebagai studi kasus untuk menunjukkan bagaimana algoritma A* dapat membantu mengatasi masalah perencanaan jalur transportasi di wilayah urban yang sedang berkembang pesat.

### *C. Pemodelan Graf Jalur Transportasi Umum*

Dalam makalah ini, jaringan transportasi umum di Kota Bekasi dimodelkan sebagai graf berbobot. Setiap simpul pada graf mewakili titik-titik penting, seperti halte bus Trans Patriot, stasiun KRL, terminal angkot, pusat perbelanjaan, rumah sakit, kawasan permukiman, dan fasilitas publik lainnya.

Heuristik yang digunakan dalam A* adalah estimasi jarak terpendek dari simpul saat ini ke tujuan akhir, yang dalam kasus ini dapat dihitung menggunakan jarak Euclidean (lurus). Pemodelan graf ini memungkinkan algoritma A* untuk mencari rute optimal di antara berbagai moda transportasi dan jalur yang tersedia.

### *D. Desain dan Tahapan Implementasi Algoritma A**

Desain implementasi algoritma A* untuk perencanaan jalur transportasi umum di Kota Bekasi meliputi beberapa tahapan sebagai berikut:

1. Pengumpulan Data
   Tahap awal melibatkan pengumpulan data terkait peta jalan.
2. Pembangunan Graf
   Data yang terkumpul kemudian dipetakan menjadi graf, di mana setiap simpul mewakili lokasi strategis, dan sisi mewakili jalur transportasi yang dapat dilalui.
3. Penentuan Fungsi Heuristik
   Fungsi heuristik ($h(n)$) dirancang untuk mengestimasi waktu tempuh tersingkat dari suatu simpul ke tujuan akhir. Estimasi ini dapat menggunakan jarak lurus antar titik.
4. Implementasi Algoritma A*
   Algoritma A* dijalankan dengan memulai pencarian dari simpul asal (misal, rumah sakit atau pusat perbelanjaan) ke simpul tujuan (misal, stasiun atau kawasan industri). Algoritma secara iteratif memilih simpul dengan nilai $f(n)$ terendah, di mana $f(n) = g(n) + h(n)$, sampai tujuan tercapai.
5. Visualisasi dan Analisis Hasil Jalur
   Jalur optimal yang dihasilkan divisualisasikan pada peta kota Bekasi untuk melihat kepraktisan dan efektivitasnya. Analisis dilakukan dengan membandingkan hasil algoritma terhadap rute yang biasa ditempuh secara manual oleh pengguna transportasi umum.

### *E. Pemilihan Titik Lokasi*

Perencanaan jalur transportasi umum di wilayah perkotaan memerlukan pemilihan titik-titik strategis sebagai simpul (*nodes*) dalam jaringan. Di Kota Bekasi, pemilihan titik strategis harus mempertimbangkan pemerataan wilayah yang mewakili sebagian besar kecamatan, fungsi penting (pusat pendidikan, kesehatan, ekonomi, pemerintahan, transportasi), serta relevansi terhadap integrasi antar moda (bus kota Trans Patriot, angkutan kota/angkot, KRL Commuter Line, bus antarkota termasuk DAMRI). Titik-titik ini berperan sebagai hub transportasi atau destinasi utama yang melayani pergerakan penumpang dalam kota maupun dari/ke wilayah Jabodetabek. Berikut disajikan daftar titik strategis di Kota Bekasi, dikelompokkan per kecamatan, beserta alasan pemilihan setiap titik.

- Pasar Bantargebang
- Perumahan Vida Bekasi
- Terminal Bekasi
- Simpang Bulak Kapal
- Summarecon Bekasi
- Perumahan Taman Wisma Asri
- Stasiun Kranji
- Pasar Sumber Arta
- Kantor Walikota Kota Bekasi
- RSUD dr. Chasbullah Abdulmadjid
- Grand Galaxy Park
- Pasar Rawalumbu
- Perumnas III
- Harapan Indah
- Pondok Ungu Permai
- Komsen Jatiasih
- Kawasan Kranggan
- Pondok Gede Permai

- Pasar Pondok Gede
- Pasar Kecapi
- Kawasan Jatiwarna

## IV. Implementasi dan Pengujian

### A. *Desain dan Arsitektur Program*

Dalam membangun sistem perencanaan jalur transportasi umum berbasis algoritma A*, digunakan struktur program yang modular dan terpisah ke dalam beberapa berkas utama. Pemilihan struktur ini dilakukan agar setiap bagian program memiliki tanggung jawab yang jelas, mudah dikembangkan, serta dapat diuji secara terpisah. Modularitas ini memudahkan penggantian atau penambahan fitur di masa mendatang tanpa mengganggu keseluruhan sistem.

Setiap modul memiliki fungsi khusus: data.py menyimpan basis data titik-titik strategis, config.py sebagai pusat pengaturan node asal dan tujuan, graph_utils.py untuk menangani pemrosesan jaringan jalan serta perhitungan heuristik, astar_solver.py menjalankan algoritma A* dan konversi hasil jalur ke bentuk koordinat, serta visualisasi.py yang menampilkan hasil rute pada peta interaktif berbasis Folium. Sementara itu, seluruh alur dikendalikan oleh main.py, yang bertugas mengorkestrasi eksekusi dari setiap modul. Dengan pendekatan seperti ini, program diharapkan mampu memberikan hasil yang presisi dan transparan, serta dapat diadaptasi untuk studi kota lain atau pengembangan fitur lebih lanjut, seperti integrasi moda transportasi khusus atau analisis beban lalu lintas.

```python
nodes = {
  'Pasar Bantar Gebang': (-6.313491, 106.987108),
  'Perumahan Vida': (-6.320985, 107.000294),
  'Terminal Bekasi': (-6.249158, 107.013482),
  'Simpang Bulak Kapal': (-6.260884, 107.018686),
  'Summarecon Bekasi': (-6.225566, 107.002893),
  'Perumahan Taman Wisma Asri': (-6.209952, 107.026655),
  'Stasiun Kranji': (-6.224633, 106.979595),
  'Pasar Sumber Arta': (-6.24972, 106.945719),
  'Kantor Walikota Kota Bekasi': (-6.235976, 106.993468),
  'RSUD dr. Chasbullah Abdulmadjid': (-6.241748, 107.000417),
  'Grand Galaxy Park': (-6.270829, 106.971745),
  'Pasar Rawalumbu': (-6.271255, 107.001885),
  'Perumnas III': (-6.242825, 107.036179),
  'Harapan Indah': (-6.186043, 106.975038),
  'Pondok Ungu Permai': (-6.180097, 107.00837),
  'Komsen Jatiasih': (-6.296509, 106.95894),
  'Kawasan Kranggan': (-6.365413, 106.920265),
  'Pondok Gede Permai': (-6.299932, 106.970615),
  'Pasar Pondok Gede': (-6.285634, 106.91158),
  'Pasar Kecapi': (-6.305986, 106.934111),
  'Kawasan Jatiwarna': (-6.31034, 106.925168),
}
```

Gambar 4.1. Data Koordinat Node Lokasi pada file data.py
(Sumber: Dokumentasi Pribadi)

```python
from data import nodes

ASAL_NAMA = 'Pasar Bantar Gebang'
TUJUAN_NAMA = 'Harapan Indah'

ORIGIN_POINT = nodes[ASAL_NAMA]
DESTINATION_POINT = nodes[TUJUAN_NAMA]
GRAPH_PLACE = "Bekasi, Jawa Barat, Indonesia"
```

Gambar 4.2. Inisialisasi Parameter Pencarian Jalur pada file config.py
(Sumber: Dokumentasi Pribadi)

```python
import osmnx as ox
from math import radians, cos, sin, asin, sqrt

def load_graph(place, network_type='drive'):
    """
    Download graf jalan OSMnx.
    """
    print(f"Scraping jaringan jalan untuk area: {place} ...")
    return ox.graph_from_place(place, network_type=network_type)

def nearest_node(G, point):
    """
    Cari node terdekat di graf dari (lat, lon)
    """
    lon, lat = point[1], point[0]
    return ox.nearest_nodes(G, lon, lat)

def haversine(coord1, coord2):
    """
    Hitung jarak haversine dalam kilometer.
    """
    lat1, lon1 = coord1
    lat2, lon2 = coord2
    lon1, lat1, lon2, lat2 = map(radians, [lon1, lat1, lon2, lat2])
    dlon = lon2 - lon1
    dlat = lat2 - lat1
    a = sin(dlat/2)**2 + cos(lat1) * cos(lat2) * sin(dlon/2)**2
    c = 2 * asin(sqrt(a))
    r = 6371
    return c * r

def heuristic(u, v, G):
    """
    Fungsi heuristik A*: jarak haversine antar node graf OSMnx (km).
    """
    coord_u = (G.nodes[u]['y'], G.nodes[u]['x'])
    coord_v = (G.nodes[v]['y'], G.nodes[v]['x'])
    return haversine(coord_u, coord_v)
```

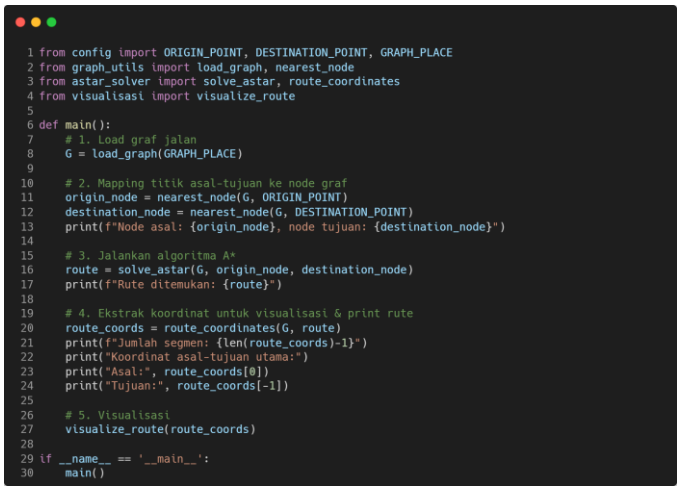
Gambar 4.3. Pemrosesan Jaringan Jalan dan Perhitungan Jarak pada file graph_utils.py
(Sumber: Dokumentasi Pribadi)

```python
import networkx as nx
from graph_utils import heuristic

def solve_astar(G, origin_node, destination_node):
    """
    Jalankan A* dan ambil rute optimal (berupa list node OSM).
    """
    print("Mencari rute optimal A* ...")
    route = nx.astar_path(G, origin_node, destination_node,
                          heuristic=lambda u,v: heuristic(u, v, G),
                          weight='length')
    return route

def route_coordinates(G, route):
    """
    Ubah node rute jadi list koordinat (lat, lon).
    """
    return [(G.nodes[node]['y'], G.nodes[node]['x']) for node in route]
```

Gambar 4.4. Implementasi Fungsi Algoritma A dan Ekstraksi Koordinat Rute pada file astar_solver.py
(Sumber: Dokumentasi Pribadi)

```python
import folium
import os
from config import ASAL_NAMA, TUJUAN_NAMA

def visualize_route(route_coords, filename=ASAL_NAMA+" ke "+TUJUAN_NAMA+".html"):
    """
    Plot rute pada peta Folium, simpan sebagai HTML.
    """
    center_lat = sum([lat for lat, lon in route_coords]) / len(route_coords)
    center_lon = sum([lon for lat, lon in route_coords]) / len(route_coords)
    m = folium.Map(location=[center_lat, center_lon], zoom_start=13)

    # Garis rute
    folium.PolyLine(route_coords, color='blue', weight=5, opacity=0.7).add_to(m)

    # Marker asal dan tujuan
    folium.Marker(route_coords[0], popup='Asal', icon=folium.Icon(color='green')).add_to(m)
    folium.Marker(route_coords[-1], popup='Tujuan', icon=folium.Icon(color='red')).add_to(m)

    os.makedirs("test", exist_ok=True)
    m.save(os.path.join("test", filename))
    print(f"Peta rute A* tersimpan di '{filename}'")
```

Gambar 4.5. Implementasi Visualisasi Rute A dengan Folium pada file visualisasi.py
(Sumber: Dokumentasi Pribadi)



```python
from config import ORIGIN_POINT, DESTINATION_POINT, GRAPH_PLACE
from graph_utils import load_graph, nearest_node
from astar_solver import solve_astar, route_coordinates
from visualisasi import visualize_route

def main():
    # 1. Load graf jalan
    G = load_graph(GRAPH_PLACE)

    # 2. Mapping titik asal-tujuan ke node graf
    origin_node = nearest_node(G, ORIGIN_POINT)
    destination_node = nearest_node(G, DESTINATION_POINT)
    print(f"Node asal: {origin_node}, node tujuan: {destination_node}")

    # 3. Jalankan algoritma A*
    route = solve_astar(G, origin_node, destination_node)
    print(f"Rute ditemukan: {route}")

    # 4. Ekstrak koordinat untuk visualisasi & print rute
    route_coords = route_coordinates(G, route)
    print(f"Jumlah segmen: {len(route_coords)-1}")
    print("Koordinat asal-tujuan utama:")
    print("Asal:", route_coords[0])
    print("Tujuan:", route_coords[-1])

    # 5. Visualisasi
    visualize_route(route_coords)

if __name__ == '__main__':
    main()
```

Gambar 4.6. Kode Utama Proses Pencarian dan Visualisasi Rute pada file main.py
(Sumber: Dokumentasi Pribadi)

### *B. Integrasi Data Titik Strategis*

Pengambilan data titik strategis dilakukan berdasarkan identifikasi kebutuhan mobilitas masyarakat dan titik-titik pusat aktivitas di Kota Bekasi. Setiap node yang dimasukkan ke dalam data.py mewakili pusat permukiman, fasilitas transportasi umum utama (seperti terminal dan stasiun), pusat ekonomi, kawasan perumahan besar, dan fasilitas layanan publik. Data ini berbentuk dictionary berisi nama titik dan koordinat GPS-nya. Dengan pendekatan ini, seluruh wilayah administratif Kota Bekasi dapat diwakili secara merata dan komprehensif.

Untuk membangun model jaringan jalan Kota Bekasi, digunakan library OSMnx yang memungkinkan pengunduhan dan ekstraksi data jalan dari OpenStreetMap secara otomatis dan akurat. Proses ini dilakukan dengan perintah graph_from_place() pada OSMnx, dengan parameter network_type='drive', sehingga seluruh ruas jalan yang dapat dilalui kendaraan bermotor—baik jalan besar maupun jalan lingkungan—akan terpetakan ke dalam graf berbasis simpul (*nodes*) dan ruas (*edges*). Setiap node merepresentasikan simpul jaringan jalan, sedangkan edge merupakan ruas jalan antar node dengan atribut panjang. Dengan demikian, hasil graf yang didapat mencerminkan kondisi jalan di lapangan, bukan sekadar garis lurus antar titik.

Selanjutnya, setiap titik strategis yang ada dalam basis data akan dipetakan ke node terdekat dalam graf jalan menggunakan fungsi nearest_nodes() OSMnx. Proses ini sangat krusial agar rute yang dihasilkan benar-benar dimulai dan diakhiri pada jalan terdekat dari lokasi strategis, sehingga setiap rekomendasi jalur yang dihasilkan dapat langsung diterapkan pada kondisi nyata di lapangan.

### *C. Pemilihan Node Ujung dan Skenario Pengujian*

Proses pemilihan node asal dan tujuan dalam pengujian program dilakukan berdasarkan kebutuhan analisis skenario perjalanan yang merepresentasikan rute-rute penting di Kota Bekasi. Setiap node yang digunakan sebagai titik ujung merupakan hasil seleksi dari daftar titik strategis yang telah diidentifikasi sebelumnya. Pemilihan dua node strategis dilakukan dengan mempertimbangkan jarak geografis, keterwakilan antar kecamatan, dan variasi karakteristik kawasan, seperti dari permukiman padat ke kawasan bisnis, atau dari terminal ke perumahan baru. Tujuannya adalah untuk memastikan bahwa program diuji pada kasus nyata yang umum terjadi dalam mobilitas harian warga Bekasi.

Beberapa pasangan node yang digunakan untuk skenario pengujian antara lain:

- Pasar Bantar Gebang ke Harapan Indah: Skenario perjalanan lintas timur ke barat laut, menempuh jarak maksimum dan melewati beberapa simpul utama.
- Kawasan Kranggan ke Pondok Ungu Permai: Uji lintas barat daya ke utara, merepresentasikan perjalanan antarkecamatan ujung kota.
- Komsen Jatiasih ke Summarecon Bekasi: Skenario selatan ke utara yang menantang karena harus melewati kawasan tengah kota yang padat.
- Perumahan Vida ke Pasar Sumber Arta: Menguji efektivitas program dalam mencari rute dari kawasan permukiman baru ke pusat ekonomi di perbatasan kota.

Dalam tiap skenario, proses yang dilakukan meliputi pemetaan titik strategis ke node graf jalan, menjalankan algoritma A* untuk pencarian rute, dan analisis hasil baik dalam bentuk rute (daftar node dan koordinat) maupun visualisasi interaktif. Pengujian dengan berbagai pasangan node yang berjauhan ini bertujuan untuk menilai konsistensi dan keandalan algoritma dalam berbagai kondisi geografis serta pola permukiman Kota Bekasi.

### *D. Implementasi Algoritma A* secara Teknis*

Implementasi algoritma A* pada jaringan jalan dilakukan dengan beberapa tahapan teknis terstruktur. Pertama, seluruh jaringan jalan Kota Bekasi diunduh secara otomatis dari OpenStreetMap melalui library OSMnx. Data yang diambil meliputi ribuan *node* (persimpangan dan titik-titik ruas jalan) beserta *edge* yang menghubungkannya (ruas jalan dengan atribut seperti panjang, nama jalan, dan jenis jalan). Graf yang dihasilkan bersifat *directed* (arah ruas jalan dapat dua arah atau satu arah sesuai kondisi di lapangan) dan telah dibobot berdasarkan panjang jalan (atribut 'length' dalam meter).

Setelah graf jalan terbentuk, titik asal dan tujuan dari daftar node strategis dipetakan ke node graf OSM terdekat menggunakan fungsi ox.nearest_nodes() supaya pencarian rute selalu dimulai dan diakhiri pada node jaringan jalan, bukan sekadar pada titik koordinat acak.

Algoritma A* dijalankan dengan fungsi nx.astar_path() dari NetworkX, di mana bobot edge didasarkan pada panjang ruas jalan, dan heuristik yang digunakan adalah jarak lintas udara (haversine) antar node. Pemilihan heuristik ini didasarkan pada efisiensi komputasi dan relevansi spasial: jarak haversine memberikan estimasi jarak terdekat yang logis antar dua titik di

permukaan bumi tanpa harus menghitung seluruh bobot edge secara *exhausti*.

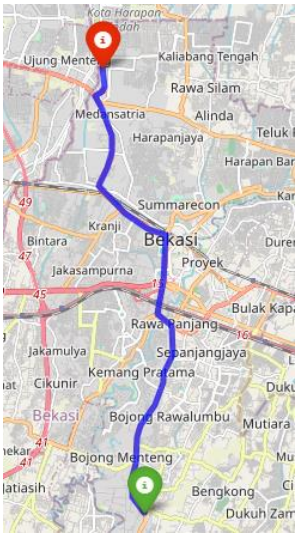
Setelah rute optimal ditemukan, hasilnya diekstrak dalam bentuk daftar node OSM yang kemudian dikonversi ke list koordinat GPS untuk keperluan visualisasi. Visualisasi jalur dilakukan menggunakan library Folium. Marker khusus ditambahkan pada titik asal dan tujuan dan seluruh jalur disimpan ke dalam file HTML.

Seluruh tahapan implementasi ini dilakukan dalam *pipeline* yang terintegrasi pada berkas main.py dengan setiap fungsi penting terpisah dalam modul sendiri untuk memudahkan debugging, pengembangan, dan integrasi ke sistem lain.

### *E. Hasil Uji Coba*

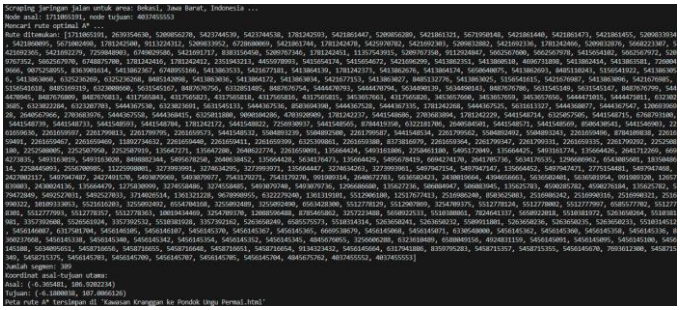
Implementasi dan pengujian program dilakukan pada empat skenario utama, dengan hasil visualisasi rute menggunakan peta interaktif berbasis Folium. Setiap pengujian menunjukkan bahwa algoritma A* mampu menghasilkan rute terpendek yang realistis, melewati ruas-ruas jalan di Kota Bekasi.

Gambar 4.7. Hasil Output Terminal Proses Pencarian Rute A* dari Pasar Bantar Gebang ke Harapan Indah
(Sumber: Dokumentasi Pribadi)

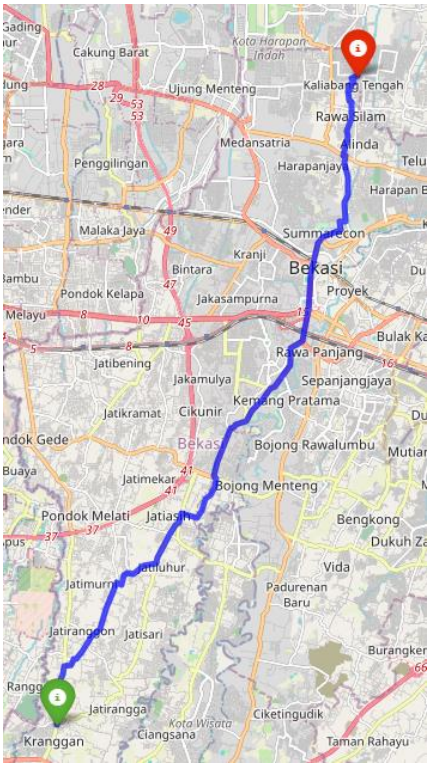



Gambar 4.8. Visualisasi Rute Terpendek Hasil Algoritma A* dari Pasar Bantar Gebang ke Harapan Indah
(Sumber: Dokumentasi Pribadi)

Pada pengujian ini, titik asal berada di Pasar Bantar Gebang (ujung timur Bekasi) dan titik tujuan di Harapan Indah (barat laut). Berdasarkan output terminal dan peta HTML, rute yang dihasilkan benar-benar mengikuti jaringan jalan utama, dimulai dari kawasan Bantar Gebang, melewati koridor jalan besar di pusat kota, dan berakhir di kawasan Harapan Indah. Jalur yang dipilih algoritma tidak sekadar menghubungkan kedua titik dengan garis lurus, melainkan memanfaatkan ruas jalan yang paling efisien dari sisi jarak dan keterhubungan. Visualisasi pada peta juga menunjukkan bahwa setiap simpul rute mengikuti ruas jalan utama, persimpangan, dan menghindari jalur yang kecil atau memutar, menegaskan keunggulan algoritma A* dalam konteks graf jaringan jalan.



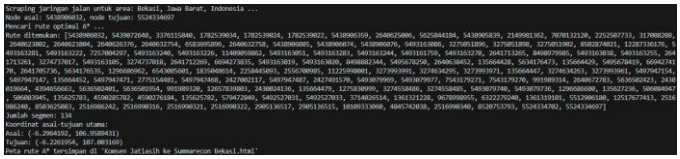
Gambar 4.9. Hasil Output Terminal Proses Pencarian Rute A* dari Kawasan Kranggan ke Pondok Ungu Permai
(Sumber: Dokumentasi Pribadi)




Gambar 4.10. Visualisasi Rute Terpendek Hasil Algoritma A* dari Kawasan Kranggan ke Pondok Ungu Permai
(Sumber: Dokumentasi Pribadi)

Skenario kedua menguji perjalanan panjang dari Kawasan Kranggan (barat daya) ke Pondok Ungu Permai (utara). Output menunjukkan rute yang berkelok sesuai jaringan jalan, menempuh ruas-ruas jalan besar dan tetap menghindari jalur lingkungan yang tidak perlu. Peta interaktif memperlihatkan bahwa jalur yang dipilih tetap logis dan efisien, memperhitungkan titik persimpangan penting dan mengurangi jumlah segmen jalan yang tidak relevan. Hal ini menunjukkan

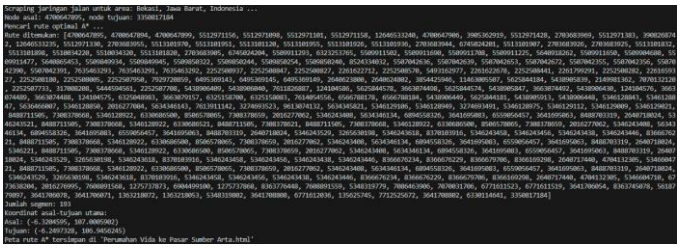
keunggulan pemodelan graf dan keakuratan pemetaan node strategis ke jaringan jalan, sehingga algoritma A* benar-benar dapat merepresentasikan perjalanan nyata.



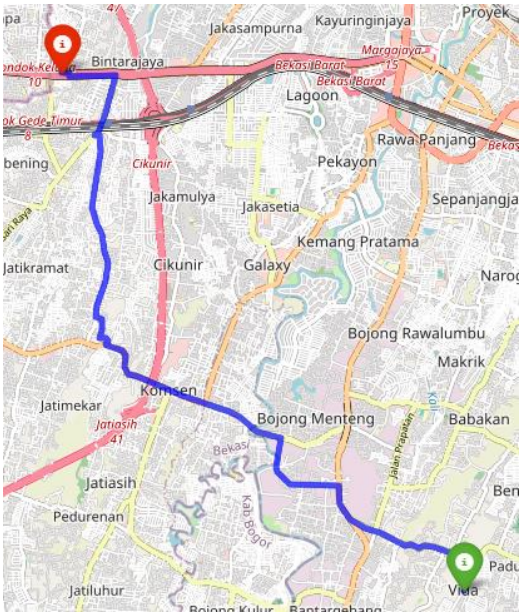
Gambar 4.11. Hasil Output Terminal Proses Pencarian Rute A* dari Komsen Jatiasih ke Summarecon Bekasi
(Sumber: Dokumentasi Pribadi)


Gambar 4.12. Visualisasi Rute Terpendek Hasil Algoritma A* dari Komsen Jatiasih ke Summarecon Bekasi
(Sumber: Dokumentasi Pribadi)

Pada pengujian ketiga, algoritma A* diterapkan untuk rute dari Komsen Jatiasih ke Summarecon Bekasi. Hasil visualisasi peta HTML memperlihatkan jalur optimal yang sebagian besar mengikuti ruas jalan utama di Kota Bekasi. Namun, terdapat anomali pada hasil rute, di mana jalur akhir memotong langsung area Summarecon Mall dan terdapat segmen rute yang keluar jalur saat melewati flyover. Hal ini menunjukkan adanya keterbatasan pada data jaringan jalan OpenStreetMap, khususnya dalam representasi struktur jalan bertingkat atau area privat seperti pusat perbelanjaan. Kesalahan pemetaan node tujuan yang berada di dalam area mall, serta keterhubungan antar-ruas jalan pada data OSM yang kurang akurat, dapat menyebabkan algoritma A* menghasilkan rute yang secara spasial benar tetapi secara nyata tidak dapat dilalui kendaraan. Oleh karena itu, meskipun secara umum algoritma A* mampu menemukan rute terpendek berbasis data jaringan jalan, hasilnya tetap sangat tergantung pada kelengkapan dan akurasi data spasial yang digunakan.



Gambar 4.13. Hasil Output Terminal Proses Pencarian Rute A* dari Perumahan Vida ke Pasar Sumber Arta
(Sumber: Dokumentasi Pribadi)




Gambar 4.14. Visualisasi Rute Terpendek Hasil Algoritma A* dari Perumahan Vida ke Pasar Sumber Arta
(Sumber: Dokumentasi Pribadi)

Skenario keempat, yakni dari Perumahan Vida ke Pasar Sumber Arta, memperkuat analisis program dalam memilih jalur terpendek pada jaringan jalan yang kompleks. Output pada peta menunjukkan rute yang menghubungkan kawasan permukiman baru di timur Bekasi dengan pusat ekonomi di barat, melewati jalan utama dan titik-titik transit strategis. Hasil pengujian menegaskan bahwa pemetaan koordinat ke node graf dan algoritma A* bekerja dengan baik, tidak sekadar menghubungkan dua titik secara ideal, tetapi mengikuti aturan dan pola jaringan jalan Kota Bekasi yang sebenarnya.

### *F. Evaluasi*

Evaluasi dari hasil implementasi menunjukkan bahwa program perencanaan jalur berbasis algoritma A* pada jaringan jalan nyata Kota Bekasi umumnya sangat efektif dalam mencari rute terpendek dan efisien, baik dari segi jarak tempuh maupun kelogisan perjalanan. Visualisasi hasil pada peta interaktif memberikan nilai tambah dalam memudahkan pemahaman, validasi, dan komunikasi hasil analisis kepada pengguna maupun pihak terkait lainnya.

Kelebihan utama dari sistem ini terletak pada kemampuannya memanfaatkan data spasial real-time dan

pemetaan presisi ke jaringan jalan aktual, sehingga rute yang dihasilkan pada dasarnya dapat diikuti dalam kondisi nyata di lapangan. Struktur program yang modular juga memungkinkan pengembangan atau integrasi lebih lanjut, seperti penyesuaian bobot edge berdasarkan waktu tempuh atau data trayek transportasi umum.

Namun demikian, terdapat beberapa keterbatasan yang perlu dicatat. Hasil pengujian memperlihatkan adanya anomali pada rute di area tertentu, seperti jalur yang secara visual menembus area bangunan (misal, pusat perbelanjaan) atau "keluar" dari jalur jalan utama, khususnya saat melalui flyover atau struktur jalan bertingkat. Hal ini umumnya disebabkan oleh keterbatasan atau ketidakakuratan data jaringan jalan pada OpenStreetMap, terutama dalam hal pemetaan level jalan, aksesibilitas kendaraan, serta penempatan node asal/tujuan yang kurang tepat. Selain itu, program masih mengasumsikan semua ruas jalan dapat dilalui seluruh jenis kendaraan, dan bobot edge hanya berdasarkan panjang ruas, tanpa mempertimbangkan faktor kemacetan, waktu tempuh aktual, atau kondisi jalan yang dinamis.

## V. Kesimpulan

Berdasarkan hasil implementasi dan pengujian sistem perencanaan jalur transportasi umum berbasis algoritma A* pada jaringan jalan Kota Bekasi, dapat disimpulkan beberapa hal penting sebagai berikut:

1. Algoritma A* efektif dalam menemukan rute terpendek dan efisien pada jaringan jalan Kota Bekasi. Pada berbagai skenario pengujian antara titik-titik strategis, rute yang dihasilkan umumnya optimal secara matematis dan cukup logis secara praktik di lapangan.
2. Pemanfaatan data jaringan jalan OpenStreetMap (OSM) yang diproses dengan OSMnx memberikan tingkat akurasi dan relevansi tinggi karena perencanaan rute sepenuhnya mengikuti struktur jalan mulai dari jalan arteri, persimpangan, hingga lingkungan permukiman. Namun, kualitas hasil sangat tergantung pada kelengkapan dan akurasi data OSM, terutama pada area dengan struktur jalan bertingkat atau akses privat.
3. Visualisasi rute pada peta interaktif (HTML Folium) sangat membantu dalam interpretasi, validasi, dan komunikasi hasil kepada pengguna maupun stakeholder. Fitur ini mendukung proses pengambilan keputusan serta edukasi publik terkait rute transportasi.
4. Pengujian pada beberapa skenario utama menunjukkan algoritma A* dapat bekerja secara konsisten dalam menemukan rute optimal pada jaringan jalan perkotaan yang kompleks. Namun, ditemui juga keterbatasan pada pemilihan rute di area tertentu, seperti rute yang memotong area privat atau flyover akibat keterbatasan pemetaan *node* dan *edge* pada data OSM.

## VI. Appendix

Program yang digunakan dalam makalah ini dapat dilihat pada tautan berikut:

https://github.com/csans13/Bekasi-Astar-Routing

## VII. Ucapan Terima Kasih

Segala puji penulis panjatkan ke hadirat Tuhan Yang Maha Esa atas segala rahmat dan petunjuk-Nya, sehingga makalah ini dapat diselesaikan dengan baik. Penulis juga ingin menyampaikan terima kasih kepada Bapak Monterico Adrian atas kepercayaan dan kesempatan yang telah diberikan untuk mengerjakan makalah ini, serta atas arahan dan masukan yang sangat berarti selama mengikuti perkuliahan Strategi Algoritma. Semoga makalah ini dapat memberikan manfaat dan inspirasi bagi para pembaca, serta menjadi kontribusi kecil dalam pengembangan ilmu pengetahuan.

## Referensi

[1] R. Munir, "Graf (Bagian 1)," Kuliah Matematika Diskrit, Program Studi Informatika, STEI ITB, 2024. Diakses dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf, pada 24 Juni 2025.

[2] R. Munir, "Penentuan Rute (Route/Path Planning) – Bagian 2: Algoritma A*," Kuliah Strategi Algoritma, Program Studi Informatika, STEI ITB, 2025. Diakses dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/22-Route-Planning-(2025)-Bagian2.pdf, pada 24 Juni 2025.

[3] H. Bast, D. Delling, A. Goldberg, M. Müller-Hannemann, T. Pajor, P. Sanders, D. Wagner, and R. F. Werneck, "Route Planning in Transportation Networks," arXiv:1504.05140, 2015. Diakses dari https://arxiv.org/abs/1504.05140, pada 24 Juni 2025.

[4] "NetworkX – Network Analysis in Python (Documentation)," NetworkX v3.5, 2025. Diakses dari https://networkx.org/documentation/stable/, pada 24 Juni 2025.

[5] "NetworkX – Network Analysis in Python (Documentation)," NetworkX v3.5, 2025. Diakses dari https://networkx.org/documentation/stable/, pada 24 Juni 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.

Bandung, 24 Juni 2025

Anas Ghazi Al Gifari
13523159