

**MODEL PEMBELAJARAN DAN LAPORAN AKHIR
PROJECT-BASED LEARNING
MATA KULIAH ALGORITMA DAN PEMROGRAMAN LANJUT
RANCANGAN KODE *SCRIPT* PEMROGRAMAN BERBASIS OOP PADA
STUDI KASUS: GRAPH TRAVERSAL
KELAS A**



**“Pencarian Rute Terpendek Tempat Praktik Mandiri Dokter Umum di
Kecamatan Gunung Anyar Menggunakan Algoritma Breadth First
Search”**

DISUSUN OLEH KELOMPOK IV :

- | | |
|----------------------|---------------------------|
| 1. RIZKI AMANDA | (22083010045) - KETUA |
| 2. ISMI CHOIRUNNISAK | (22083010060) - ANGGOTA |
| 3. ADELIA YUANDHIKA | (22083010066) - ANGGOTA |
| 4. NEZALFA SABRINA | (22083010067) - ANGGOTA |
| 5. JASMINE AULIA | (22083010074) - ANGGOTA |
| 6. CAHYA EKA MELATI | (22083010090) - ANGGOTA |

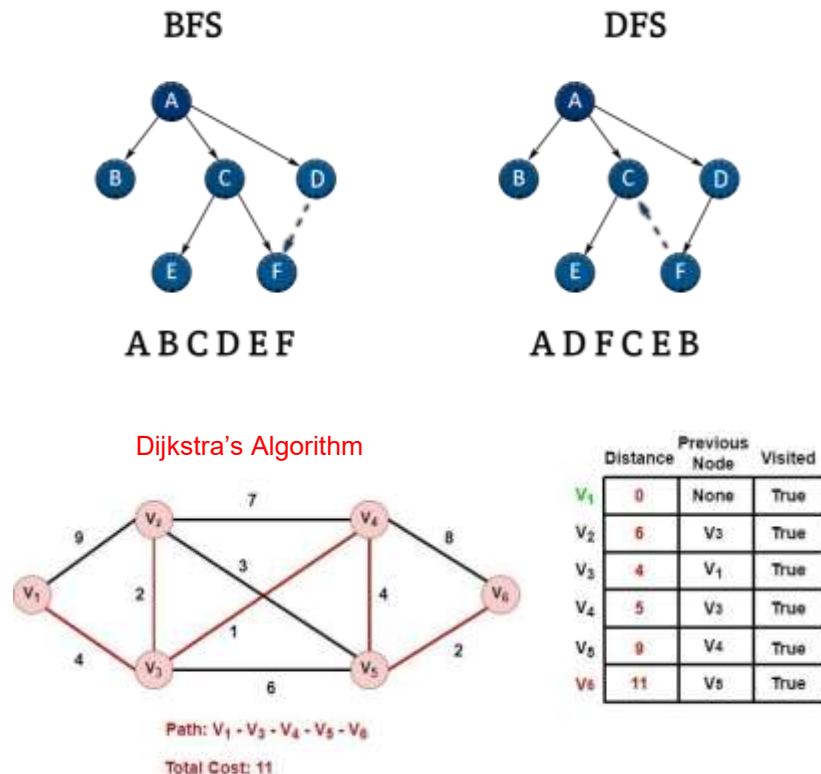
DOSEN PENGAMPU:

TRESNA MAULANA FAHRUDIN, S.S.T., MT (199305012022031007)
AMRI MUHAJIMIN, S.Stat., M.Stat., M.S. (21119950723270)

**PROGRAM STUDI SAINS DATA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS PEMBANGUNAN NASIONAL “VETERAN”
JAWA TIMUR
2023**

MODEL PEMBELAJARAN

1. Buatlah kelompok (1 kelompok sekitar 5-6 mahasiswa), untuk merancang kode *script* pemrograman berbasis OOP pada studi kasus Graph Traversal berikut ini:



Graph traversal adalah proses mengunjungi setiap node atau vertex di dalam sebuah graf atau graph. Tujuannya bisa berbeda-beda tergantung pada keperluan, seperti mencari jalur terpendek antara dua node, menemukan siklus dalam graf, atau mencari node yang memiliki properti tertentu.

Proses graf traversal dilakukan dengan mengunjungi setiap node secara berurutan, dan untuk setiap node yang dikunjungi, mengunjungi semua node tetangganya yang belum pernah dikunjungi sebelumnya. Terdapat beberapa algoritma yang digunakan untuk melakukan graf traversal, seperti

1. Breadth-First Search (BFS)
2. Depth-First Search (DFS)
3. Dijkstra's Algorithm
4. Bellman-Ford
5. Floyd Warshall Algorithm
6. Johnson's Algorithm
7. Uniform Cost Search

Solusi yang diusulkan tidak terbatas pada nama-nama algoritma graph traversal yang disebutkan di atas, kelompok boleh mengusulkan topik lainnya selama dalam ruang lingkup bidang Graph Traversal.

Contoh implementasi Graph Traversal dalam menyelesaikan permasalahan sehari-hari di antaranya:

1. Penerapan Algoritma **BFS** dan **DFS** untuk Penjadwalan Rencana Studi
2. Perbandingan Performansi Terhadap Algoritma Breadth First Search (BFS) & Depth First Search (DFS) Pada Web Crawler
3. Analisis Algoritma **Dijkstra** dan Algoritma Bellman-Ford Sebagai Penentuan Jalur Terpendek Menuju Lokasi Kebakaran (Studi Kasus: Kecamatan Praya Kota)
4. Implementasi Algoritma **Breadth First Search** Pada Pencarian Rute Terpendek Tempat Kos Di Semarang Tengah
5. Implementasi Algoritma **BFS** pada Desain Sistem Pengolahan Temu Kembali Berkas
6. Implementasi Algoritma **BFS** dan **DFS** dalam Penyelesaian Token Flip Puzzle

Penilaian ditekankan pada bagaimana kelompok dapat menyusun kode *script* program berbasis OOP menggunakan bahasa pemrograman Python dengan mengimplementasikan konsep yang sudah dipelajari sebelumnya seperti konsep *class*, *attribute* dan *method*, *inheritance*, *overriding*, *overloading*, *polymorphism*, *modules and packages*, dan materi yang relevan lainnya.

Output dari *project-based learning* juga diminta berupa *executable file* yang dapat dijalankan secara langsung tanpa menjalankan kode *script* melalui *interpreter*. Oleh karena itu, beberapa pustaka untuk membuat aplikasi berbasis Graphical User Interface (GUI) menggunakan tkinter, PyQt5, dan pustaka lainnya.

2. Dokumentasi *project-based learning* didokumentasikan ke dalam laporan yang mencakup:
 - a. Halaman Depan
 - b. Kata Pengantar
 - c. Latar Belakang beserta subbabnya
 - d. Tinjauan Pustaka beserta subbabnya
 - e. Metodologi Penelitian beserta subbabnya
 - f. Hasil dan Pembahasan beserta subbabnya
 - g. Kesimpulan
 - h. Lampiran
3. Video dokumentasi proyek yang menarik dan diunggah di platform Youtube (Lampiran)

KATA PENGANTAR

Segala syukur kami panjatkan ke hadirat Allah SWT atas segala rahmat, taufik, hidayah, inayah-Nya sehingga kami dapat menyelesaikan project based learning dengan judul "Pencarian Rute Terpendek Tempat Praktik Dokter Umum Mandiri di Kecamatan Gunung Anyar Menggunakan Algoritma Breadth First Search" tepat pada waktunya. Project based learning ini disusun untuk memenuhi tugas mata kuliah Algoritma Pemrograman Lanjut. Dalam project ini, kami berkesempatan untuk mempelajari dan mengimplementasikan algoritma Breadth First Search (BFS) dalam pencarian rute terpendek untuk mencari tempat dokter umum di Kecamatan Gunung Anyar. Dalam penyusunan project based learning ini, kami menyadari bahwa project ini masih belum mencapai kata sempurna karena keterbatasan yang kami miliki. Oleh karena itu, kami mengharapkan kritik dan saran yang membangun untuk kesempurnaan project ini.

Maka, dalam kesempatan ini kami menyampaikan ucapan terima kasih yang sebesar-besarnya kepada Bapak Tresna Maulana Fahrudin, S.S.T., M.T dan Bapak Amri Muhaimin, S.Stat., M.Stat., M.S. selaku dosen pengampu mata kuliah Algoritma Pemrograman Lanjut yang telah memberikan arahan, bimbingan, dan masukan yang sangat berharga dalam penyelesaian project ini. Akhir kata, kami berterima kasih atas semua dukungan dan partisipasi yang telah diberikan selama pengerjaan project ini. Kami berharap project ini dapat memberikan manfaat yang nyata bagi masyarakat dalam mencari rute terpendek menuju praktik dokter umum di Kecamatan Gunung Anyar. Semoga dengan adanya project ini dapat menambah pengetahuan dan pemahaman tentang Algoritma Breadth First Search. Terima kasih.

DAFTAR ISI

| | |
|--|----|
| MODEL PEMBELAJARAN..... | 2 |
| DAFTAR ISI | 5 |
| DAFTAR GAMBAR..... | 6 |
| DAFTAR TABEL | 7 |
| BAB I PENDAHULUAN..... | 8 |
| 1.1 Latar Belakang..... | 8 |
| 1.2 Permasalahan | 8 |
| 1.3 Tujuan..... | 9 |
| 1.4 Manfaat | 9 |
| BAB II TINJAUAN PUSTAKA | 10 |
| 2.1 Teori Penunjang..... | 10 |
| 2.2 Penelitian Terkait..... | 14 |
| BAB III METODOLOGI PENELITIAN | 15 |
| 3.1 Struktur Data..... | 16 |
| 3.2 Metode BFS | 19 |
| 3.3 Aplikasi GUI..... | 21 |
| 3.4 Output | 22 |
| BAB IV HASIL DAN PEMBAHASAN..... | 23 |
| 4.1 Penjelasan Kode Script..... | 23 |
| 4.2 Mengubah Kode Script menjadi Executable File | 36 |
| 4.3 Waktu Pemrosesan | 38 |
| 4.4 Akurasi..... | 41 |
| BAB V: KESIMPULAN | 44 |
| DAFTAR PUSTAKA | 45 |
| LAMPIRAN | 47 |

DAFTAR GAMBAR

| | |
|--|----|
| Gambar 1. Anaconda Logo | 11 |
| Gambar 2. Python 3 Logo | 12 |
| Gambar 3. Cara Kerja Algoritma Breadth First Search | 13 |
| Gambar 4. Desain Sistem BFS | 16 |
| Gambar 5. Ilustrasi Graf Rute Tempat Praktik Mandiri Dokter Umum | |
| 17 | |
| Gambar 6. Flowchart Pencarian Rute Terdekat Menggunakan Metode BFS | 20 |
| Gambar 7. Desain Aplikasi GUI | |
| 21 | |
| Gambar 8. Ikon Aplikasi GoingDoc | 38 |

DAFTAR TABEL

| | |
|--|-------|
| Tabel 1. Tempat Praktik Dokter Umum pada Graf | 17- |
| 18 | |
| Tabel 2. Tempat Umum pada Graf | 18-19 |
| Tabel 3. Hasil dan Waktu Pencarian Rute Terpendek | |
| 39-41 | |
| Tabel 4. Persentase Error Data Jarak Aktual dan Jarak Output | 41- |
| 43 | |

BAB I

PENDAHULUAN

1.1 Latar Belakang

Dalam kehidupan sehari-hari, aksesibilitas terhadap pelayanan kesehatan yang baik sangat penting untuk kesejahteraan masyarakat. Salah satu layanan kesehatan yang paling umum dicari oleh masyarakat adalah praktik dokter mandiri. Dilansir dari Dinas Kesehatan Kabupaten atau Kota Tahun 2021 jumlah kunjungan rawat jalan di Klinik Pratama 5.666.267 dan Praktek Mandiri Dokter sejumlah 151.931. Di Kecamatan Gunung Anyar, terdapat sejumlah praktik dokter umum mandiri yang memberikan pelayanan medis kepada penduduk setempat. Namun, berdasarkan survei yang dilakukan oleh Dinas Kesehatan setempat, ditemukan bahwa masyarakat Kecamatan Gunung Anyar menghabiskan waktu perjalanan lebih lama dari yang seharusnya saat mencari rute ke tempat praktik dokter umum mandiri.

Pencarian rute yang tidak efisien dapat mengakibatkan waktu tempuh yang lebih lama dan meningkatkan tingkat ketidaknyamanan bagi pasien yang membutuhkan perawatan medis dengan segera. Masalah tersebut biasanya terjadi karena tidak adanya papan petunjuk atau tanda yang jelas, serta dikarenakan ketidaktahuan mengenai rute terpendek yang harus mereka tempuh untuk mencapai tujuan. Meskipun saat ini pengguna dapat menggunakan Google Maps atau GPS (Global Positioning System) untuk mencari rute terpendek, GPS yang ada dalam aplikasi digunakan hanya untuk mengukur jarak antara pengguna dan tujuan yang ingin dicapai (Juhara, 2016). Namun, Google Maps belum dapat merekomendasikan rute terpendek secara langsung. Hal ini menunjukkan adanya kebutuhan untuk pendekatan yang lebih efisien dalam menemukan rute terpendek ke tujuan.

Oleh karena itu, pada project based learning ini kami mengusulkan suatu aplikasi yang dapat memudahkan masyarakat dalam mencari rute terpendek menuju tempat Praktik Dokter Umum Mandiri di Kecamatan Gunung Anyar. Aplikasi ini akan menggunakan algoritma Breadth First Search (BFS) untuk mencari rute terpendek dari lokasi pengguna ke Praktik Dokter Umum Mandiri di Kecamatan Gunung Anyar. Dengan aplikasi ini, pengguna dapat memasukkan lokasi awal dan tujuan mereka dan kemudian aplikasi akan menampilkan rute terpendek beserta petunjuk untuk mencapai tujuan tersebut.

1.2 Permasalahan

- Masyarakat kesulitan dalam mencari rute terpendek dan tercepat ke tempat praktik dokter umum di Kecamatan Gunung Anyar.
- Belum banyaknya aplikasi yang memudahkan masyarakat Gunung Anyar dalam mencari praktik dokter umum terdekat.

1.3 Tujuan

- Memudahkan pencarian rute terpendek ke tempat praktik dokter umum di wilayah Kecamatan Gunung Anyar.
- Menyediakan aplikasi yang dapat memudahkan masyarakat sekitar Gunung Anyar dalam mencari praktik dokter umum terdekat menggunakan metode Breadth First Search.

1.4 Manfaat

- Untuk Masyarakat
Memudahkan masyarakat dalam mencari rute terpendek menuju tempat praktik dokter umum di Kecamatan Gunung Anyar. Mengurangi waktu perjalanan dan menghemat energi yang diperlukan untuk mencapai tujuan.
- Untuk Industri
Meningkatkan aksesibilitas dan pelayanan industri kesehatan di Kecamatan Gunung Anyar dan meningkatkan efisiensi operasional dengan mengoptimalkan rute terpendek bagi pasien dan tenaga medis.
- Untuk Perkembangan Ilmu
Meningkatkan pengetahuan dan pemahaman mahasiswa terkait algoritma Breadth First Search dan aplikasinya dalam masalah nyata. Meningkatkan keterampilan pemrograman dan pemecahan masalah mahasiswa melalui implementasi proyek berbasis algoritma.

BAB II

TINJAUAN PUSTAKA

2.1 Teori Penunjang

2.1.1 Praktik Dokter Umum Mandiri

Menurut penjelasan Pasal 4 ayat (1) huruf a Peraturan Pemerintah Republik Indonesia Nomor 47 Tahun 2016 tentang Fasilitas Pelayanan Kesehatan, yang dimaksud dengan “Tempat Praktik Mandiri Tenaga Kesehatan” adalah fasilitas pelayanan kesehatan yang diselenggarakan oleh tenaga kesehatan yang memiliki kewenangan untuk memberikan pelayanan langsung kepada pasien atau klien. Sementara itu, pengertian dari praktik dokter umum sendiri adalah dokter yang fokus dalam mengobati penyakit yang muncul secara tiba-tiba (akut) dan menahun (kronis) atau biasa dikenal sebagai dokter layanan pertama. Dokter umum berbeda dengan dokter spesialis. Dokter spesialis merupakan dokter yang memiliki mengkhususkan diri terhadap suatu bidang atau bagian tubuh tertentu sedangkan dokter umum adalah dokter yang menyediakan pelayanan yang bersifat menyeluruh terhadap pasien. Dokter umum juga memegang peranan penting pada area kedokteran karena mereka seringkali menjadi orang yang pertama berhubungan dengan pasien. Dokter umum tidak terikat untuk mengobati bagian atau organ tubuh tertentu, sehingga mereka memiliki keahlian luas yang membantu mereka untuk menolong pasien pada segala usia, jenis kelamin dan dengan berbagai masalah kesehatan. (Haris, 2019 : 6)

Menurut Peraturan Menteri Kesehatan Nomor 1438/Menkes/Per/I/2010, dokter umum dan tenaga kesehatan lainnya di fasilitas pelayanan kesehatan diwajibkan untuk mengikuti Standar Prosedur Operasional (SPO) sesuai dengan keputusan klinis yang mereka ambil. Dalam praktik mandiri, dokter umum memiliki tanggung jawab penuh dalam pengambilan keputusan medis dan manajemen praktik. Mereka menerima pasien, melakukan pemeriksaan medis, mendiagnosis penyakit, meresepkan obat, dan memberikan perawatan dasar untuk berbagai kondisi kesehatan umum. Pendekatan ini dapat memiliki dampak positif pada kualitas layanan yang diberikan dan dokter juga dapat mengatur jadwal kunjungan pasien dan menentukan jenis layanan yang mereka sediakan. Keuntungan dari tempat praktik dokter umum mandiri ini adalah pasien mendapatkan perawatan yang lebih personal dan cepat dilayani karena antrian praktik mandiri dokter umum biasanya lebih sedikit daripada di rumah sakit.

2.1.2 Anaconda

Anaconda adalah sebuah aplikasi open source gratis yang berfungsi untuk distribusi Python dan R. Python banyak digunakan dalam berbagai perhitungan ilmiah, termasuk dalam bidang machine learning, pengolahan data dalam skala besar, analisis prediksi, dan banyak lagi. Aplikasi ini menyimpan lebih dari 1500 package populer yang dapat diakses di beragam platform sistem operasi seperti Windows, Linux, dan MacOS (Anaconda, 2021). Pada tahun 2012, Peter Wang dan Travis Oliphant mendirikan Anaconda dengan tujuan untuk mengintegrasikan Python dalam analisis data bisnis, yang saat itu sedang mengalami transformasi pesat akibat tren teknologi yang muncul. Anaconda Navigator termasuk sebuah Graphical User Interface (GUI) yang dapat digunakan untuk menjalankan aplikasi serta mengelola packages untuk menggunakan library dalam kode program yang dibutuhkan untuk data learning.

Anaconda menyediakan banyak pustaka dan package yang sudah diinstal sebelumnya. Aplikasi tersebut memiliki package installer yang cukup andal dan dilengkapi dengan package yang lengkap dan terupdate. Beberapa package yang ada dalam Anaconda di antaranya adalah NumPy, SciPy, Pandas, Scikit learn, nltk, dan Jupiter Library. Anaconda memiliki banyak kelebihan diantaranya yaitu mudah dan cepat dalam menginstall package untuk data science, pengguna dapat memilih versi bahasa pemrograman python yang ingin diaplikasikan., lalu library dependensi dan lingkungan juga dapat diatur menggunakan Anaconda. Namun Anaconda juga memiliki kekurangan yaitu pengguna wajib mempunyai pengalaman luas dengan bahasa pemrograman Python, lalu besarnya file installer yang mencapai ratusan MB dapat memakan banyak ruang laptop atau PC pengguna. (idmetafora.com)



Gambar 1. Anaconda Logo

2.1.3 Python 3

Python adalah bahasa pemrograman yang umumnya digunakan dalam pengembangan aplikasi berbasis web. Python merupakan bahasa pemrograman yang bersifat interpretatif

dan memiliki banyak fungsi yang dapat dieksekusi. Guido van Rossum menciptakan Python, sebuah bahasa pemrograman yang pertama kali dirilis pada tahun 1991. Tujuannya adalah untuk memfokuskan bahasa pemrograman tersebut pada tingkat yang tinggi dan memiliki makna semirip mungkin dengan bahasa manusia. Python telah dikenal sebagai bahasa pemrograman tingkat tinggi dengan sintaksis yang mudah dimengerti, menggunakan perintah dalam bahasa Inggris. Secara umum, logika perintah yang ditulis dalam Python lebih mudah digunakan dibandingkan dengan bahasa pemrograman tingkat rendah seperti bahasa assembly (D. Kuhlman, 2013).

Python memiliki sejumlah besar perpustakaan (library) dan kerangka kerja (framework) yang dibuat oleh komunitas yang aktif. Hal ini memungkinkan pengembang untuk memperluas kemampuan Python dalam berbagai bidang. Filosofi desain Python bertujuan untuk membuat kode menjadi mudah dibaca. Dengan demikian, Python mendorong gaya penulisan kode yang bersih dan terstruktur. Python dapat dianggap sebagai bahasa pemrograman yang menggabungkan kemampuan dan kapabilitas dengan sintaksis kode yang jelas. Ini membuatnya menjadi bahasa yang populer di kalangan pengembang. Selain itu, Python juga menyediakan banyak pustaka standar yang kaya dan komprehensif dalam fungsionalitasnya. Pengguna dapat langsung menggunakan pustaka-pustaka ini tanpa perlu menginstal perpustakaan tambahan. Keberadaan perpustakaan dan pustaka standar tersebut membuat Python menjadi pilihan yang kuat untuk pengembangan perangkat lunak (A. N. Syahrudin and T. Kurniawan, 2018).



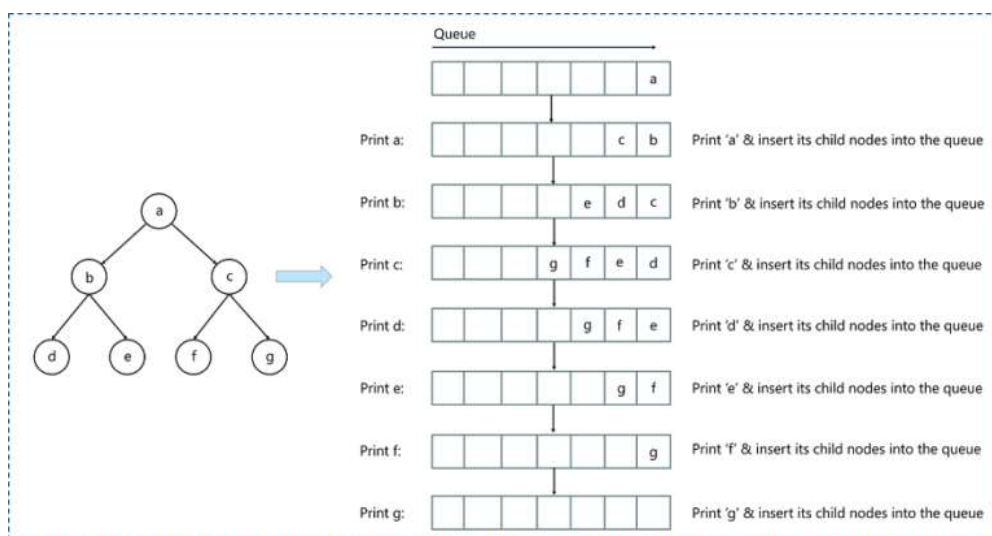
Gambar 2. Python 3 Logo

2.1.4 Algoritma Breadth First Search (BFS)

Penjelajahan graf melibatkan mengunjungi setiap simpul dan sisi tepat satu kali dalam urutan yang terdefinisi dengan baik. Hal ini memastikan bahwa tidak ada simpul yang terlewat atau dikunjungi lebih dari sekali. Saat menggunakan algoritma graf tertentu, sangat penting untuk memastikan bahwa setiap simpul dalam graf dikunjungi tepat satu kali. Hal ini memerlukan perhatian terhadap urutan kunjungan simpul yang dapat bervariasi tergantung pada algoritma atau pertanyaan yang sedang diselesaikan. Dalam penjelajahan graf, penting untuk melacak simpul mana yang telah dikunjungi. Salah satu

cara umum untuk melacak simpul tersebut adalah dengan memberikan tanda pada simpul-simpul yang telah dikunjungi. Melacak simpul yang telah dikunjungi adalah langkah penting dalam penjelajahan graf. Hal ini membantu menghindari pengulangan kunjungan dan memastikan bahwa semua simpul dikunjungi dengan benar. Penjelajahan graf melibatkan urutan kunjungan simpul yang harus dipastikan tepat dan tidak ada simpul yang terlewat. Melacak simpul-simpul yang telah dikunjungi adalah salah satu cara umum untuk menjaga konsistensi dalam proses tersebut. (Sachin Gupta, 2023)

Algoritma Breadth First Search (BFS) adalah metode pencarian yang digunakan dalam teori graf untuk mencari jalur terpendek dari suatu simpul awal menuju simpul tujuan dalam sebuah graf berarah atau tak terarah. Pada algoritma BFS, proses eksplorasi dilakukan dengan mengeksplorasi semua simpul yang terhubung secara sejajar atau berdekatan dengan simpul awal sebelum melanjutkan ke simpul-simpul yang lebih jauh. Pendekatan breadth-first pada BFS menjamin bahwa jalur yang ditemukan pertama kali merupakan jalur terpendek, karena proses eksplorasi dilakukan secara berurutan dan bertahap dari simpul awal ke simpul-simpul terdekat. Langkah pertama dalam algoritma BFS adalah memilih simpul awal sebagai titik awal eksplorasi, yang kemudian dimasukkan ke dalam antrian simpul yang berfungsi sebagai antrian sementara untuk menyimpan simpul-simpul yang akan diperiksa selanjutnya. Setelah simpul awal diproses, algoritma BFS akan memeriksa secara bertahap simpul-simpul tetangga yang belum dikunjungi, dengan mengambil satu simpul dari antrian simpul untuk diperiksa pada setiap langkah.



Gambar 3. Cara Kerja Algoritma Breadth First Search (sumber : edureca.co)

Pemrosesan simpul-simpul dilakukan secara berurutan sesuai dengan urutan penambahan ke dalam antrian simpul. Dalam algoritma BFS, simpul-simpul yang lebih

dekat dengan simpul awal akan diproses terlebih dahulu sebelum simpul-simpul yang lebih jauh. Algoritma ini terus melanjutkan langkah-langkah tersebut sampai antrian simpul kosong, yang menunjukkan bahwa semua simpul yang terhubung langsung atau tidak langsung dengan simpul awal telah diperiksa. Pada setiap tingkat eksplorasi, algoritma BFS memastikan bahwa semua simpul pada tingkat tersebut diperiksa sebelum memeriksa simpul pada tingkat selanjutnya. Hal ini memastikan bahwa simpul-simpul pada jarak yang sama dari simpul awal akan diproses sebelum simpul-simpul pada jarak yang lebih jauh. Dengan demikian, algoritma BFS dapat menemukan jalur terpendek berdasarkan simpul-simpul yang dekat dengan simpul awal.

Dengan menggunakan algoritma BFS, pengguna dapat efisien mencari jalur terendek dalam graf. Algoritma ini mempertimbangkan keterhubungan antar simpul-simpul dan menjaga urutan kunjungan secara sistematis. Prinsip kerja BFS melibatkan eksplorasi secara berurutan dari simpul awal ke simpul-simpul terdekat. Dalam proses eksplorasi, simpul-simpul yang terhubung secara langsung dengan simpul awal diperiksa terlebih dahulu sebelum menjelajahi simpul-simpul yang lebih jauh (Delima Zai, Haeni Budiati, dan Sunneng Sandino Berutu 2016). Dengan demikian, algoritma BFS memastikan bahwa jalur terpendek ditemukan dengan mempertimbangkan simpul-simpul yang dekat dengan simpul awal. Pengguna dapat memanfaatkan algoritma BFS ini untuk mencari jalur terpendek dalam sebuah graf dengan efisien dan secara sistematis.

2.2 Penelitian Terkait

2.2.1. Erniyati, M. Kom, Mulyati, M. Kom (2018) dalam penelitiannya yang berjudul “Pencarian Jalur Terdekat Menuju Rumah Sakit di Kota Bogor dengan Menggunakan Algoritma A*” menerapkan algoritma A* (*A-Star*) untuk mencari rute Rumah Sakit di kota Bogor berbasis android. Penelitian tersebut dibangun menggunakan *software Android Studio* dengan bahasa pemrograman java dan *XML* untuk *user interface*. Metode penerapannya dilakukan secara bertahap menggunakan model *System Development Life Cycle* (SDLC). SDLC mengusulkan model pendekatan perkembangan perangkat lunak yang sistematis dengan tahapan perencanaan sistem, analisis sistem, perancangan, implementasi, dan uji coba. Hasil dari penelitian tersebut adalah sebuah aplikasi pencarian jalur terdekat menuju rumah sakit di Kota Bogor yang memiliki output berupa rekomendasi rumah sakit yang terdekat dengan pengguna. Aplikasi tersebut dapat dijalankan pada sistem android sehingga dapat mempermudah user untuk mengakses dimanapun dan kapanpun.

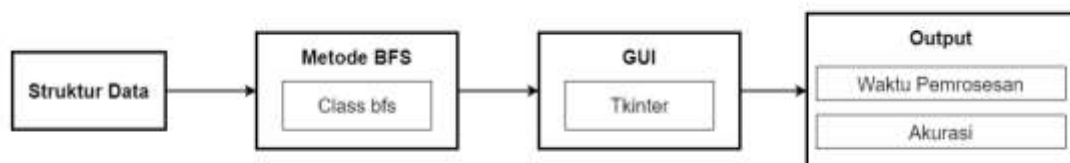
2.2.2 Hari Toha Hidayat (2019) dalam penelitiannya yang berjudul “Sistem Penunjang Keputusan Pencarian Jarak Terpendek Menuju Rumah Sakit dan Puskesmas dengan Metode Dijkstra” membahas tentang pentingnya sistem informasi interaktif untuk membantu masyarakat mencari rute terdekat ke rumah sakit dan puskesmas. Penelitian ini menggunakan algoritma Dijkstra (Shortest Path Algorithm) untuk menemukan jarak terpendek antara lokasi awal dan rumah sakit atau puskesmas. Tujuannya adalah mempermudah masyarakat dalam mencari informasi rute terdekat. Jurnal ini juga mengulas penggunaan algoritma Dijkstra dalam pencarian rute terpendek untuk wisata dan aplikasi sejenis. Metodologi penelitian melibatkan penggunaan sistem pendukung keputusan (DSS) untuk membantu pengambilan keputusan dalam situasi semi terstruktur. DSS bertujuan untuk meningkatkan efektivitas, produktivitas, dan kualitas keputusan. Jurnal ini relevan dalam memanfaatkan teknologi informasi untuk membantu masyarakat mencari rute terdekat ke rumah sakit dan puskesmas.

2.2.3 Haris Muhammad Syarif (2019) dalam penelitiannya yang berjudul “Aplikasi Sistem Layanan Periksa Dokter pada Dokter Praktek Umum Mandiri” membahas pada pengembangan aplikasi berbasis *cloud computing* (*Software as a Service*) yang digunakan untuk membantu mengelola rekam medis dan proses layanan pada praktek dokter umum mandiri.. Aplikasi ini dirancang untuk bertemu dokter dan pasien secara online, memungkinkan pasien untuk mendaftar dan mengisi data medis, dokter dapat menjadwalkan praktik serta melihat dan menambahkan rekam medis pasien. Dalam majalah tersebut dijelaskan bahwa aplikasi tersebut dapat membantu meningkatkan efisiensi pelayanan, mengurangi biaya kertas dan alat tulis, serta memudahkan dokter dalam mengelola rekam medis. Diharapkan kedepannya aplikasi ini dapat memenuhi kebutuhan akan sumber daya IT di industri medis khususnya pada praktek dokter umum mandiri.

BAB III

METODOLOGI PENELITIAN

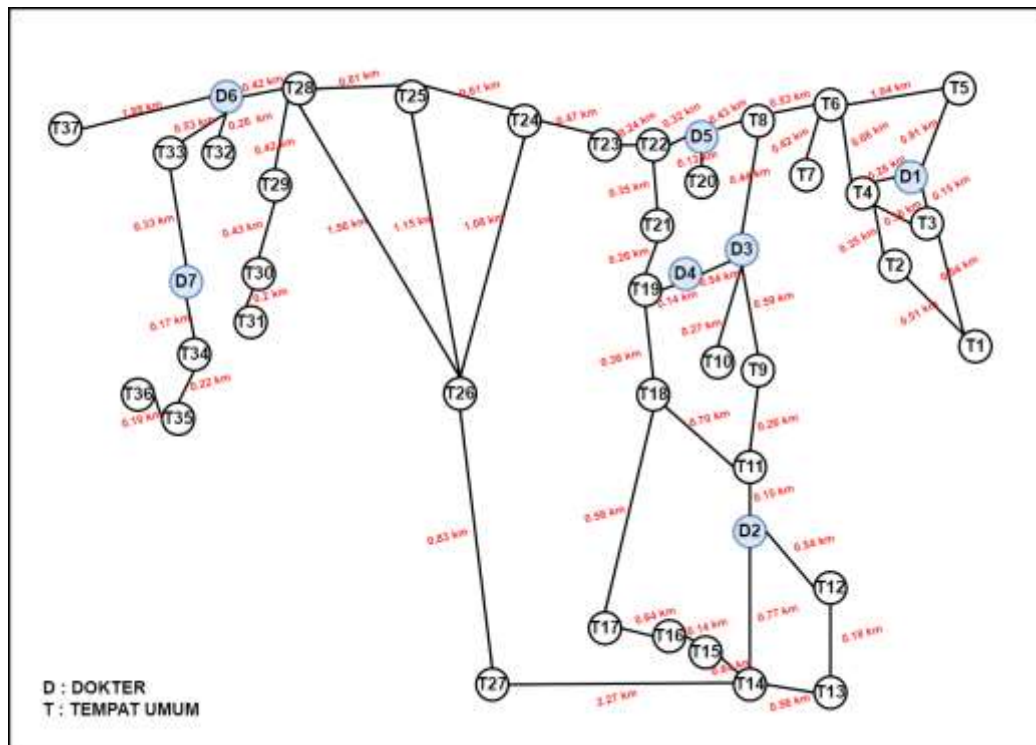
Dalam penelitian ini, peneliti ingin mengembangkan sebuah aplikasi yang dapat memberikan solusi dalam mengatasi permasalahan pencarian rute terdekat dokter praktik mandiri. Untuk mencapai tujuan tersebut, desain sistem yang efektif dan efisien sangat penting. Desain ini melibatkan penggunaan beberapa komponen kunci, termasuk struktur data graf, metode BFS, aplikasi GUI, dan pengukuran output dalam bentuk waktu pemrosesan dan akurasi. Dengan menggunakan desain sistem ini diharapkan dapat menyediakan solusi yang efektif dan efisien dalam pembuatan aplikasi untuk pencarian rute terpendek tempat praktik dokter mandiri di Kecamatan Gunung Anyar.



Gambar 4. Desain Sistem

3.1 Struktur Data

Metode penelitian yang digunakan dalam penelitian ini adalah melalui penggunaan graf sebagai struktur data. Graf digunakan sebagai langkah penting dalam pengembangan sistem pencarian rute terpendek tempat praktik mandiri dokter umum di Kecamatan Gunung Anyar. Desain graf mencerminkan hubungan antara tempat praktik mandiri dokter umum dan rute-rute yang ada di Kecamatan Gunung Anyar. Graf disini merupakan struktur data yang terdiri dari node (tempat praktik mandiri dokter umum) yang saling terhubung oleh edge (rute). Setiap node dalam graf memiliki identitas unik, seperti nomor atau nama tempat praktik mandiri dokter umum. Setiap edge dalam graf memiliki angka yang merepresentasikan jarak antara dua tempat praktik mandiri dokter umum yang terhubung.



Gambar 5: Ilustrasi Graf Rute Tempat Praktik Mandiri Dokter Umum Kecamatan Gunung Anyar

Pada ilustrasi graf di atas, terdapat dua kode yang memiliki arti tertentu. Kode D merepresentasikan lokasi tempat praktik mandiri dokter umum di Kecamatan Gunung Anyar, sementara kode T merepresentasikan lokasi umum di Kecamatan Gunung Anyar. Dalam graf tersebut, terdapat rute yang menghubungkan simpul-simpul tersebut, dan rute-rute tersebut diberi label angka yang mengindikasikan jarak atau bobot perpindahan antara tempat praktik mandiri dokter umum yang terhubung. Ilustrasi graf ini memberikan gambaran tentang bagaimana algoritma BFS dapat diimplementasikan untuk mencari rute terpendek antara dua tempat praktik mandiri dokter umum di Kecamatan Gunung Anyar. Dengan menggunakan ilustrasi ini sebagai acuan, implementasi algoritma BFS dapat dilakukan secara efisien dan sistematis untuk menemukan rute terpendek antara kedua tempat tersebut.

Tabel 1. Tempat Praktik Dokter Umum pada Graf

| No | Tempat Praktik Dokter Umum | Kode Dokter |
|----|----------------------------|-------------|
| 1 | Dr. Abdul | D1 |
| 2 | Dr. Fitri Yuda Mayasari | D2 |

| | | |
|---|-------------------|----|
| 3 | Dr. Ani Budiwati | D3 |
| 4 | Dr. Eko Iswahyudi | D4 |
| 5 | Dr. Jimmy S. | D5 |
| 6 | Dr. Luksadi | D6 |
| 7 | Dr. Brana Pradata | D7 |

Tabel 2. Tempat Umum pada Graf

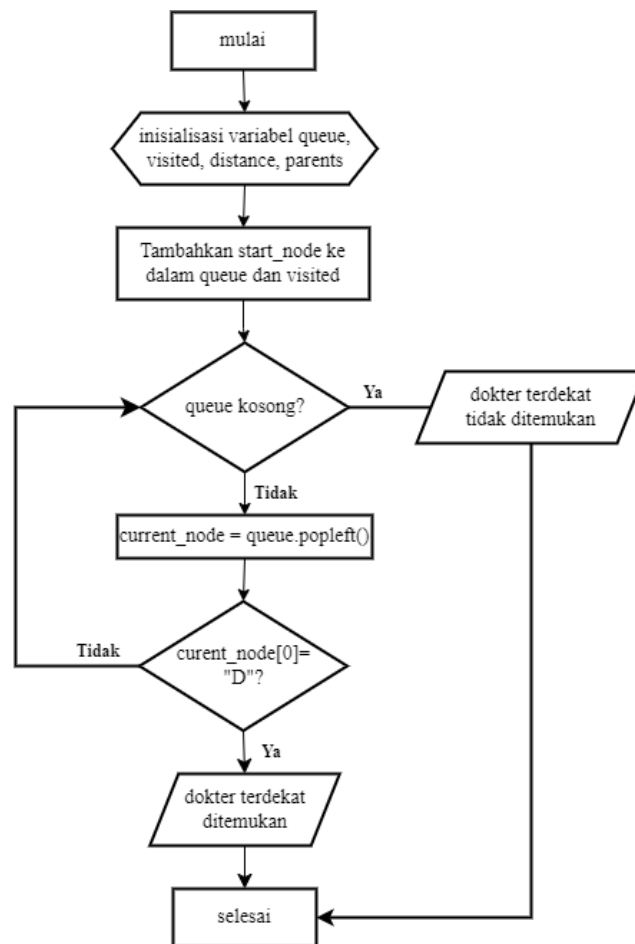
| No | Tempat Umum | Kode Tempat |
|----|---|-------------|
| 1 | Mushola & TPQ Jl. Wisma Tirta Agung Asri | T1 |
| 2 | Perumahan Wisma Indah | T2 |
| 3 | Musholla Al Fattah Puri Gununganyar Regency | T3 |
| 4 | Taman Gunung Anyar | T4 |
| 5 | Rusunawa Gunung Anyar | T5 |
| 6 | Musholla Hubbul Wathon | T6 |
| 7 | Masjid Al Muttaqin, Gunung Anyar Emas | T7 |
| 8 | Bengkel cat mobil Grand Star | T8 |
| 9 | DAMKAR GUNUNG ANYAR | T9 |
| 10 | Masjid Nur Madrikah | T10 |
| 11 | Panti Asuhan Ulul Azmi | T11 |
| 12 | SD Budi Mulia Islamic School | T12 |
| 13 | Gunung Anyar Town House | T13 |
| 14 | Masjid Roudlotul Mukminin | T14 |
| 15 | Gunung Anyar Asri | T15 |
| 16 | PERUM BUMI PRATAMA ASRI | T16 |
| 17 | Politeknik Pelayaran Surabaya | T17 |
| 18 | Perumahan Gunung Anyar Permai | T18 |
| 19 | Masjid al Mubarak | T19 |
| 20 | PT. Berhasil Sinar Gemilang Abadi | T20 |

| | | |
|----|--|-----|
| 21 | Apotek Doa Sehat | T21 |
| 22 | SPBU Pertamina - Gunung Anyar | T22 |
| 23 | Fotocopy Alfin | T23 |
| 24 | UPN "Veteran" Jawa Timur | T24 |
| 25 | Mr. Suprek Rungkut | T25 |
| 26 | Masjid Assalam Purimas | T26 |
| 27 | UINSA Kampus 2 | T27 |
| 28 | McDonald's - Rungkut Madya | T28 |
| 29 | SPBU Pertamina 54.601.99 Rungkut Mapan | T29 |
| 30 | SDN Rungkut Mananggal I | T30 |
| 31 | Masjid Al-Muslimun | T31 |
| 32 | Kantor Kelurahan Rungkut Tengah | T32 |
| 33 | Masjid At-Taibin | T33 |
| 34 | Kantor Kelurahan Rungkut Mananggal | T34 |
| 35 | Pondok Pesantren Manbaul Falah | T35 |
| 36 | Taman Rungkut Mananggal | T36 |
| 37 | Masjid Baiturrozaq SIER Surabaya | T37 |

3.2 Metode BFS

Metode yang digunakan untuk mengatasi permasalahan pencarian rute terpendek tempat praktik mandiri dokter umum di Kecamatan Gunung Anyar adalah metode BFS (Breadth First Search). BFS adalah salah satu algoritma yang digunakan dalam pemrosesan graf atau struktur data berbasis graf. Metode ini digunakan untuk melakukan pencarian secara melebar pada graf, yaitu dengan mengunjungi semua simpul (node) pada level yang sama sebelum melanjutkan ke level selanjutnya. BFS menggunakan struktur data antrian (queue) untuk melacak simpul-simpul yang harus dikunjungi. Ketika simpul dikunjungi, simpul tersebut akan dimasukkan ke dalam antrian, dan pengunjungannya akan dilanjutkan pada simpul-simpul yang berada di antrian. Prinsip dalam antrian queue adalah FIFO (*first*

in first out), yang berarti elemen yang pertama kali dimasukkan akan menjadi elemen yang pertama kali dikeluarkan.

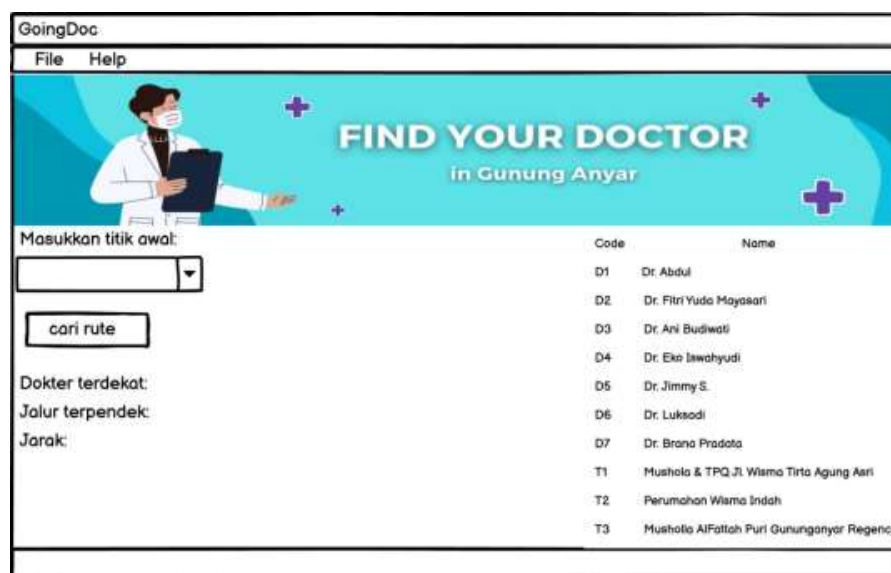


Gambar 6: Flowchart Pencarian Rute Terdekat Menggunakan Metode BFS

Pada Gambar 6 terdapat flowchart yang menggambarkan langkah-langkah algoritma BFS yang digunakan untuk menemukan rute terpendek dari titik awal ke lokasi praktik mandiri dokter umum di kecamatan Gunung Anyar. Flowchart ini berfungsi sebagai panduan visual yang menjelaskan urutan langkah-langkah yang harus diikuti dalam proses pencarian. Proses pencarian dimulai dengan memasukkan titik awal ke dalam antrian (queue). Dengan menggunakan konsep FIFO, algoritma BFS berulang kali dieksekusi hingga semua simpul yang terhubung dengan simpul awal telah dikunjungi atau rute terpendek menuju praktik dokter mandiri telah ditemukan. Prinsip FIFO penting dalam memastikan bahwa algoritma menjelajahi graf secara bertahap sesuai tingkat eksplorasi, sehingga rute terpendek dapat ditemukan dengan akurat. Dengan demikian, penggunaan antrian pada flowchart BFS ini membantu mengatur dan mengendalikan urutan pengolahan simpul-simpul yang terlibat dalam pencarian rute terpendek.

3.3 Aplikasi GUI

Penerapan algoritma BFS dalam penelitian ini dilakukan dengan mengembangkan sebuah aplikasi berbasis Graphical User Interface (GUI). Aplikasi GUI dipilih sebagai pilihan untuk menunjukkan penerapan langsung dari metode BFS. Penelitian ini akan menggunakan library Tkinter dalam pembuatan aplikasi GUI. Tkinter merupakan perpustakaan standar GUI widget yang digunakan dalam pembuatan interface Python menggunakan Toolkit GUI Tk. Tkinter menyediakan berbagai elemen GUI seperti tombol, scrollbar, listbox, check button, radio button, label teks, dan lain sebagainya. Dengan menggunakan Tkinter, pengembang dapat dengan mudah membuat elemen-elemen GUI seperti jendela (window), tombol (button), label, input teks (entry), daftar (list), dan berbagai elemen GUI lainnya. Selain itu, Tkinter juga mendukung tata letak (layout) seperti grid, pack, dan place untuk mengatur penempatan dan penampilan elemen-elemen GUI.



Gambar 7: Desain Aplikasi GUI

Gambar 7 menunjukkan desain aplikasi GUI yang mengacu pada mockup telah disusun sebelumnya (lihat lampiran 3 untuk gambar mockup lengkap). Desain ini memberikan gambaran visual yang jelas mengenai tampilan dan tata letak antarmuka pengguna. Aplikasi ini akan memiliki komponen seperti kolom input untuk titik awal, tombol pencarian rute, tabel informasi kode node dan namanya, serta tampilan hasil pencarian rute. Selain itu, aplikasi juga akan menyediakan fitur pembantu seperti menu fitur pembantu seperti menu file dan menu help. Dengan desain aplikasi yang telah disusun, peneliti dapat melanjutkan ke tahap implementasi dengan lebih terarah dan efisien.

3.4 Output

A. Waktu Pemrosesan

Waktu proses merupakan selisih waktu mulai eksekusi hingga output ditampilkan. Waktu pemrosesan dicari dengan menggunakan library time dalam Python yang menyediakan berbagai fungsi terkait waktu. Modul ini termasuk dalam modul utilitas standar Python. Metode `time.time()` dalam library time digunakan untuk mendapatkan waktu dalam detik sejak epoch. Penanganan detik kabisat tergantung pada platform. Library time dalam Python juga memiliki kemampuan untuk mengakses waktu sistem operasi, mendapatkan informasi seperti tanggal, waktu saat ini, zona waktu, dan lain sebagainya. Dengan menggunakan library time, kami dapat mengukur dan menganalisis waktu pemrosesan algoritma BFS secara objektif. Hasil pengukuran waktu ini akan menjadi bagian penting dalam mengevaluasi kinerja algoritma BFS dalam mencari rute terpendek ke tempat praktik mandiri dokter umum di Kecamatan Gunung Anyar.

B. Akurasi

Akurasi output aplikasi diuji melalui perhitungan persentase akurasi. Persentase akurasi dihitung dengan membandingkan jarak yang ditampilkan oleh aplikasi dengan jarak aktual yang didapatkan melalui Google Maps. Untuk menghitung persentase akurasi, digunakan rumus sebagai berikut:

$$Akurasi = 100\% - error\%$$

Error adalah selisih dari data jarak yang ditampilkan oleh aplikasi dengan jarak aktual.

Rumus untuk menghitung `error%` adalah sebagai berikut:

$$error\% = \frac{|jarak\ aktual - jarak\ pada\ aplikasi|}{jarak\ aktual} \times 100\%$$

Dengan menggunakan rumus tersebut, akan diperoleh persentase akurasi yang menggambarkan seberapa dekat jarak yang ditampilkan oleh aplikasi dengan jarak aktual yang sebenarnya. Hasil akurasi data ini akan memberikan informasi penting mengenai performa aplikasi dan tingkat keefektifan hasil pencarian rute terpendek dalam konteks praktik mandiri dokter umum di Kecamatan Gunung Anyar. Dengan mengukur akurasi output aplikasi, peneliti dapat mengevaluasi sejauh mana aplikasi dapat memberikan hasil yang akurat dan berguna bagi pengguna.

BAB IV

HASIL DAN PEMBAHASAN

4.1 Penjelasan Kode Script

4.1.1 Modul ver2.py

Pada pembuatan aplikasi ini, dibentuk file ver2.py. File ini dijadikan sebagai modul berisi kelas graf dan kelas bfs beserta fungsi-fungsi yang dapat dimanfaatkan dalam pembuatan aplikasi.

```
import networkx as nx
import matplotlib.pyplot as plt
from collections import deque
```

Modul NetworkX diperlukan untuk memanipulasi, menganalisis, dan memvisualisasikan struktur jaringan (grafik). Sedangkan modul Matplotlib dipergunakan untuk membuat visualisasi data dan deque dari modul collections merupakan struktur antrian data yang akan digunakan dalam pengimplementasian Breadth First Search.

```
class Graph:
    def __init__(self):
        self.G = nx.Graph()
        self.codes = []
        self.names = []
```

Class Graph adalah kelas yang berisi fungsi-fungsi untuk membuat dan memanipulasi graf yang akan digunakan pada permasalahan ini. Graf yang digunakan berisi kode dokter atau tempat sebagai node dan jarak sebagai edge. Fungsi init merupakan fungsi khusus yang disebut konstruktor. Metode ini akan dijalankan secara otomatis ketika objek dari kelas Graph dibuat. `G=nx.Graph()` membuat objek graf menggunakan `nx.Graph()` dari library networkx dan menyimpannya dalam variabel G. Objek ini akan digunakan untuk merepresentasikan graf dalam kelas Graph. Codes dan names adalah list kosong yang akan digunakan untuk menyimpan kode dari setiap node atau edge dalam graf.

```
def add_node(self, code, name, latitude, longitude):
    self.G.add_node(code, name=name, latitude=latitude, longitude=longitude)
    self.codes.append(code)
    self.names.append(name)
```

Fungsi `add_node` pada class `graph` digunakan untuk menambahkan simpul pada graf. Parameter yang digunakan untuk menambahkan node adalah `code`, `name`, `latitude`, dan `longitude`. Parameter `code` mewakili kode node yang akan ditambahkan ke graf, `name` mewakili nama node, `latitude` mewakili koordinat lintang dari lokasi node, dan `longitude` mewakili koordinat bujur dari lokasi node. Fungsi ini dipanggil dengan menambahkan nilai-nilai parameter sebagai argumen kemudian ditambahkan ke dalam objek graf `G`.

```
def add_edge(self, node1, node2, weight=0):
    self.G.add_edge(node1, node2, weight=weight)
```

Pada class diatas menggunakan metode `add_adge`, yang dimana metode ini digunakan untuk menambah sisi baru diantara dua node dalam graph yang direpresentasikan dengan atribut `G`. kode yang terdapat pada baris `self.G.add_edge(node1, node2, weight=weight)` yang dimana sebuah edge baru ditambahkan ke dalam graf yang telah direpresentasikan oleh atribut `G`, edge tersebut menghubungkan dua node yaitu `node1` dan `node 2`, `weight` disini juga dapat diberikan sebagai argumen opsional. Nilai `weight` ini juga akan diteruskan sebagai argumen ke metode `add_edge` pada objek `G`.

```
def create_graph(self):
    # Menambahkan node
    nodes = [("D1", "Dr. Abdul ", -7.33416651, 112.81106319),
             ("D2", "Dr. Fitri Yuda Mayasari", -7.34177390, 112.80172376),
             ("D3", "Dr. Ani Budiwati", -7.33597052, 112.80002715),
             ("D4", "Dr. Eko Iswahyudi", -7.33652257, 112.79632167),
             ("D5", "Dr. Jimmy S.", -7.33331395, 112.79686486),
             ("D6", "Dr. Luksadi", -7.33149401, 112.77200001),
             ("D7", "Dr. Brana Pradata", -7.33603313, 112.76843121),
             ("T1", "Mushola & TPQ Jl. Wisma Tirta Agung Asri", -7.33716645,
              112.81448358),
             ("T2", "Perumahan Wisma Indah", -7.33607377, 112.81059970),
             ("T3", "Musholla AlFattah Puri Gununganyar Regency", -7.33533380,
              112.81152717),
             ....

    for node in nodes:
        code, name, latitude, longitude = node
        self.add_node(code, name, latitude, longitude)
```

Fungsi `create_graph` digunakan untuk membuat graf. Pada fungsi ini, kita dapat memanggil fungsi `add_node` dan `add_edge` untuk menambahkan simpul dan sisi. Nodes adalah list yang berisi data simpul yang akan ditambahkan. Proses *looping* atau perulangan akan dilakukan melalui setiap elemen yang ada dalam 'nodes' untuk

menambahkan setiap data simpul pada list ke dalam graf dengan menggunakan fungsi `add_node`.

```
# Menambahkan edge
edges = [("T1", "T2", 0.51),
         ("T1", "T3", 0.54),
         ("T2", "T4", 0.35),
         ("T3", "T4", 0.36),
         ("T3", "D1", 0.15),
         ("T4", "D1", 0.28),
         ("T5", "D1", 0.91),
         .....
```

Edge akan menjadi penghubung antar node dari node-node yang telah dibuat sebelumnya. Dalam masing-masing edge yang dibuat dilengkapi dengan jarak sebenarnya antar node sehingga diharapkan dapat lebih membantu dalam memperkirakan jarak yang harus ditempuh untuk mencapai tempat praktik dokter umum yang ada di Kecamatan Gunung Anyar.

```
lge in edges:
    de1, node2, weight = edge
    f.add_edge(node1, node2, weight)
```

Proses *looping* atau perulangan akan dilakukan melalui setiap elemen yang ada dalam ‘edges’ yang setiap ‘edges’nya berisi informasi edge dalam graf. Baris kedua akan melakukan *unpacking* ‘edge’ yang mana elemen pertama akan diatribusikan ke variabel ‘node1’, elemen kedua ke ‘node2’, dan elemen ketiga ke ‘weight’ sehingga akan mempermudah dalam mengakses setiap bagian dari ‘edge’. Selanjutnya metode ‘`add_edge()`’ digunakan untuk menambahkan sisi(edge) ke graf menggunakan informasi dari ‘edge’.

```
_nodes_codes(self):
    1 self.codes
```

Fungsi ‘`get_nodes_codes()`’ akan menerima `self` sebagai parameternya. Fungsi tersebut akan mengembalikan nilai atribut ‘codes’. Metode tersebut akan memberikan akses ke nilai atribut ‘codes’ dari luar kelas yang telah didefinisikan.

```
_nodes_names(self):
    1 self.names
```

Fungsi ‘`get_nodes_names()`’ menerima `self` sebagai parameter dan akan mengembalikan nilai dari atribut ‘names’. Akibatnya nilai atribut ‘names’ dapat diakses

dari luar kelas.

```
w_graph(self):
: {node: (data['longitude'], data['latitude']) for node, data in
G.nodes(data=True)}
s = {node: node for node in self.G.nodes()}
_labels = {(u, v): f"{weight:.2f}" for u, v, weight in self.G.edges(data='weight')}
figure(figsize=(15,10))
aw_networkx(self.G, pos=pos, with_labels=False, node_size=200,
_color='skyblue')
aw_networkx_labels(self.G, pos=pos, labels=labels, font_size=8,
_color='black')
aw_networkx_edge_labels(self.G, pos=pos, edge_labels=edge_labels,
_size=8)
axis('off')
show()
```

Fungsi tersebut bertujuan untuk menggambar graf menggunakan modul NetworkX dan matplotlib. Fungsi tersebut mengambil 'self' sebagai parameter. Variabel 'pos' berisikan koordinat setiap node dalam graf yang diperoleh dengan mengiterasi setiap simpul 'self.G' dan mendapatkan atribut 'longitude' dan 'latitude' dari data terkait. Berikutnya 'labels' akan memetakan setiap simpul dalam graf ke simpul itu sendiri. Baris selanjutnya akan menghasilkan 'edge_labels' yang berisi label untuk setiap sisi edge dalam graf yang didapatkan dari iterasi melalui setiap sisi 'self.G' dan mendapatkan atribut 'weight'. Visualisasi graf akan memiliki ukuran 15 x 10. Label simpul tidak ditampilkan yang dikendalikan oleh 'with_labels=False'. Ukuran simpulnya adalah 200 dan warna simpulnya adalah 'skyblue'. Ukuran *font* ditentukan sebagai 8. Selanjutnya 'plt.axis('off')' akan menghilangkan sumbu pada gambar yang dihasilkan. Menggunakan 'plt.show()' visualisasi berupa gambar yang telah dihasilkan akan ditampilkan.

```
class bfs:
def __init__(self, graph):
self.G = graph.G
```

Pada kode diatas merupakan implementasi dari kelas "bfs" yang digunakan untuk melakukan algoritma Breadth-First Search (BFS) pada sebuah graf, yang dimana nantinya graf tersebut disimpan dan diberikan dalam atribut "G"

```
def shortest_path(self, start_node):  
    try:  
        if start_node not in self.G.nodes:  
            raise ValueError("Invalid start node")
```

Berdasarkan kode skrip diatas merupakan implementasi dari metode “shortest_path” yang dimana metode ini digunakan untuk mencari jalur terpendek. Yang mempunyai parameter “start_node” yang merupakan simpul awal “self G”, jadi kode skrip ini bertujuan untuk memeriksa apakah simpul awal yang diberikan valid dalam graf yang direpresentasikan oleh objek “self.G”. Jika simpulan tersebut tidak valid, maka kode tersebut akan mengangkat pengecualian “ValueError”.

```
queue = deque()  
visited = set()  
distance = {}  
parents = {}
```

Berdasarkan kode script tersebut, digunakan struktur data deque (double-ended queue) untuk mewakili antrian (queue) yang akan digunakan dalam suatu algoritma atau operasi. Selain itu, digunakan set (visited) untuk melacak node yang telah dikunjungi, dictionary (distance) untuk menyimpan jarak dari node awal ke setiap node lainnya, dan dictionary (parents) untuk menyimpan informasi tentang parent node dari setiap node.

```
queue.append(start_node)  
visited.add(start_node)  
distance[start_node] = 0  
parents[start_node] = None
```

Berdasarkan kode skrip diatas ada beberapa variabel yang akan digunakan dalam algoritma dalam pencarian rute terdekat seperti queue disini digunakan untuk menginisialisasikan sebuah antrian kosong yang menggunakan objek “deque()”, sedangkan “visited” disini digunakan untuk menginisialisasikan sebuah himpunan kosong menggunakan objek “set”, sedangkan “distance” disini digunakan untuk menginisialisasi sebuah kamus(dictionary) kosong sama seperti “parents” yang digunakan untuk menyimpan simpul-simpul “parents” dari setiap simpul dalam jalur terpendek yang dalam

kontek untuk pencarian jalur terpendek, ini akan membantu melacak jalur yang mengarah ke simpul tertentu.

```
while queue:
    current_node = queue.popleft()
```

Berdasarkan kode skrip diatas pernyataan loop “while” yang dimana akan dijalankan selama antrian “queue” tidak kosong. Loop ini akan terus berjalan hingga semua simpul yang relevan diproses, kita menggunakan metode “popleft()” dari objek “queue” yang dimana akan mengambil simpul pertama dari antrian.

```
if current_node[0] == 'D':
    # Jika simpul dengan indeks pertama "D" ditemukan, dokter terdekat
    ditemukan
    path = []
    parent = parents[current_node]
    path.append(current_node)
```

Berdasarkan kode skrip diatas digunakan untuk menemukan dokter terdekat dalam graf yang sedang dicari rute terpendeknya, jika simpul dengan indeks pertama “D” ditemukan, maka jalur mundur dari simpul terdekat ke simpul awal sedang dibangun dengan menyimpan simpul-simpul dalam daftar “path”

```
while parent:
    path.append(parent)
    parent = parents[parent]
```

Berdasarkan kode skrip diatas kita menggunakan loop “while”, “append()” dan juga “parents”. Yang dimana loop “while” akan dijalankan selama “parent” tidak bernilai “None”.

```
path.reverse()
distance = distance[current_node]
return path, distance
```

Berdasarkan kode skrip diatas kita menggunakan “reverse()” yang digunakan untuk membalik urutan elemen dalam daftar jalur “path”. sedangkan pada variabel “distance” digunakan untuk mengupdate nilai dengan jarak terpendek dari simpul awal.

```
for neighbor in self.G[current_node]:
    if neighbor not in visited:
        queue.append(neighbor)
        visited.add(neighbor)
        distance[neighbor] = distance[current_node] +
self.G.edges[current_node, neighbor]["weight"]
        parents[neighbor] = current_node
```

Berdasarkan kode diatas,baris pertama akan melakukan iterasi pada semua tetangga (neighbor)dari semua (node) dan mengacu pada daftar tetangga simpul “current_node” dalam objek graf”self.G”,baris-baris pada kode selanjutnya akan diulang untuk setiap tetangga yang belum dikunjungi simpul “current_node”. Dengan melakukan iterasi melalui tetangga-tetangga secara bertahap,algoritma BFS dapat menjelajahi graf secara meluas dari simpul awal ke simpul-simpul lainnya.

```
# Jika dokter tidak ditemukan, kembalikan None
return "Dokter terdekat tidak ditemukan"
except ValueError as e:
    print(str(e))
```

Berdasarkan kode diatas jika tidak ada dokter yang memenuhi kriteria yang ditentukan sebelumnya,kode ini akan mencetak pesan pengecualian yang memberikan informasi lebih lanjut tentang pengecualian tersebut.

```
def print_result(self, start_node, path, distance):
    if path:
        print("Jalur terdekat:", " -> ".join(path))
        print(f"Jarak terdekat dari {start_node} ke dokter terdekat: {distance:.2f}
km")
    else:
        print("Tidak ada dokter yang ditemukan")
```

Berdasarkan kode script diatas kita mempunyai tiga parameter yakni “start_node”,”path”, dan”distance”. Yang dimana kondisi ini akan mengevaluasi apakah “path” sendiri memiliki nilai atau bukan,jika “path” tidak kosong maka terdapat jalur yang

ditemukan. Fungsi “join” diatas juga digunakan untuk menggabungkan elemen-elemen dalam “path” dengan string yang sebagai pemisah antara setiap elemen.

4.1.2 Kode Script Aplikasi GUI

```
import networkx as nx
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import tkinter as tk
from tkinter import ttk
from tkinter import messagebox
from ver2 import Graph, bfs
from PIL import Image, ImageTk`
```

Berdasarkan kode script di atas beberapa pustaka dan modul yang diperlukan diimpor untuk mengembangkan aplikasi GUI dengan kemampuan grafik dan visualisasi. Baris ini mengimpor beberapa library yang akan digunakan dalam aplikasi GUI. networkx digunakan untuk memanipulasi grafik, matplotlib digunakan untuk visualisasi grafik, tkinter adalah library GUI untuk membuat tampilan aplikasi, PIL digunakan untuk memanipulasi gambar.

```
# Inisialisasi graf
graph = Graph()
graph.create_graph()
bfs = bfs(graph)
```

Membuat objek graph menggunakan kelas Graph dan memanggil metode create_graph() untuk membuat grafik. Kemudian, objek graph digunakan untuk membuat objek bfs menggunakan fungsi bfs() yang telah didefinisikan sebelumnya.

```
# Akses nilai code dan name dari node
codes = graph.get_nodes_codes()
names = graph.get_nodes_names()
```

Dua variabel codes dan names diisi dengan nilai kode dan nama dari node-node graf yang diperoleh melalui metode get_nodes_codes() dan get_nodes_names() dari objek graf.


```

def bfs_shortest_path():
    start_node = start_combobox.get()

    result = bfs.shortest_path(start_node)
    path = " -> ".join(result[0])
    distance = str(result[1]) + " km"
    nearest_doctor = result[0][-1]

    # Tampilkan hasil pada GUI
    nearest_doctor_label.configure(text=f"Dokter terdekat : {nearest_doctor}",
font='normal 10')
    path_label.configure(text=f"Jalur terpendek : {path}", font='normal 10')
    distance_label.configure(text=f"Jarak: {distance}", font='normal 10')

```

Fungsi `bfs_shortest_path()` digunakan untuk mencari jalur terpendek menggunakan algoritma BFS. Node awal diambil dari `start_combobox`, kemudian metode `shortest_path()` dari objek `bfs` (yang merupakan hasil pemanggilan `bfs(graph)`) digunakan untuk mendapatkan jalur terpendek dan jaraknya. Hasil tersebut kemudian ditampilkan pada GUI dengan mengubah teks pada label `nearest_doctor_label`, `path_label`, dan `distance_label`.

```

def show_graph_gui():

    plt.close()
    # Tampilkan graf di frame
    G = graph.G
    pos = {node: (data['longitude'], data['latitude']) for node, data in
G.nodes(data=True)}

    # Perbarui warna edge yang dilalui hasil rute terdekat menjadi merah
    shortest_path = bfs.shortest_path(start_combobox.get())[0]
    edge_colors = ['red' if (u, v) in zip(shortest_path, shortest_path[1:]) else 'gray' for u,
v, _ in G.edges(data=True)]

    # Tampilkan graf dengan perubahan warna edge
    nx.draw(G, pos=pos, with_labels=True, node_color='lightblue', node_size=200,
edge_color=edge_colors, linewidths=1, font_size=8)
    nx.draw_networkx_edges(G, pos, edgelist=[(shortest_path[i], shortest_path[i+1])
for i in range(len(shortest_path)-1)],
edge_color='red', width=4)

    fig = plt.gcf()
    fig.set_size_inches(8, 3.5)
    graph_canvas = FigureCanvasTkAgg(fig, master=graphFrame)
    graph_canvas.draw()
    graph_canvas.get_tk_widget().grid(row=2, column=0, sticky="nsew")
    graph_displayed = True

```

Fungsi `show_graph_gui()` digunakan untuk menampilkan graf pada GUI. Graf yang ditampilkan didasarkan pada objek `graph.G` yang merupakan graf yang telah dibuat sebelumnya. Posisi node-node pada graf ditentukan menggunakan koordinat longitude dan latitude. Warna edge yang dilalui oleh jalur terpendek ditandai dengan warna merah. Graf tersebut kemudian ditampilkan pada frame `graphFrame` menggunakan metode `draw()` dan `draw_networkx_edges()` dari `networkx`, serta `FigureCanvasTkAgg` dan `gcf()` dari `matplotlib`. Gambar graf kemudian diubah menjadi widget Tkinter dengan `get_tk_widget()` dan ditampilkan di frame.

```
def run_function():
    bfs_shortest_path()
    show_graph_gui()
```

Fungsi `run_function()` memanggil fungsi `bfs_shortest_path()` dan `show_graph_gui()` untuk menjalankan proses pencarian jalur terpendek dan menampilkan graf pada GUI.

```
def reset_display():
    # Menghapus tampilan hasil rute
    nearest_doctor_label.configure(text="Dokter terdekat: ", font='normal 10')
    path_label.configure(text="Jalur terpendek: ", font='normal 10')
    distance_label.configure(text="Jarak: ", font='normal 10')

    # Menghapus gambar graf
    for widget in graphFrame.winfo_children():
        widget.destroy()
```

Fungsi `reset_display()` digunakan untuk menghapus tampilan hasil rute terpendek dan gambar graf pada GUI. Label `nearest_doctor_label`, `path_label`, dan `distance_label` diatur kembali menjadi teks kosong, sedangkan gambar graf dihapus dengan menghancurkan semua widget yang ada di dalam `graphFrame`.

```
#panduan
def show_user_guide():
    guide = """
    Panduan Penggunaan Mencari Rute Dokter Terdekat:

    1. Pilih titik awal menu drop-down.
    2. Klik tombol 'Cari rute'.
    3. Hasil pencarian dan graf akan ditampilkan.
    4. Reset tampilan, pilih menu "File" dan klik "Reset"

    Catatan: Pastikan memilih node yang valid dan tersedia dalam graf.
    """
    messagebox.showinfo("Panduan Penggunaan", guide)
```

Fungsi `show_user_guide()` menampilkan panduan penggunaan program dengan menggunakan `messagebox.showinfo()` dari modul `tkinter`. Panduan tersebut berisi langkah-langkah penggunaan program, yaitu memilih titik awal, mencari rute, menampilkan hasil, dan mereset tampilan.

```
# Membuat GUI
root = tk.Tk()
root.title("GoingDoc")
```

Pembuatan GUI dilakukan dengan menggunakan modul `tkinter`. Objek `root` dibuat sebagai jendela utama dengan judul "GoingDoc" menggunakan `tk.Tk()`. Ini adalah root window yang berfungsi sebagai container untuk semua elemen GUI.

```
# Mengatur style frame
s = ttk.Style()
s.configure("mainFrame.TFrame", background="white")
s.configure("inputFrame.TFrame", background="white")
s.configure("graphFrame.TFrame", background="white")
s.configure("infoFrame.TFrame", background='white')
```

Style dari modul `ttk` digunakan untuk mengatur gaya frame dalam GUI. Beberapa gaya seperti "mainFrame.TFrame", "inputFrame.TFrame", "graphFrame.TFrame", dan "infoFrame.TFrame" diatur dengan latar belakang putih.

```

# Membuat frame
mainFrame = ttk.Frame(root, width=250, height=250, style="mainFrame.TFrame")
mainFrame.grid(row=0, column=0, sticky="NSEW")
root.grid_rowconfigure(0, weight=1)
root.grid_columnconfigure(0, weight=1)

bannerFrame = ttk.Frame(mainFrame, width=250, height=40)
bannerFrame.grid(row=0, column=0, sticky="NEW", columnspan=3)

inputFrame = ttk.Frame(mainFrame, style="inputFrame.TFrame", width=50,
height=210)
inputFrame.grid(row=1, column=0, sticky="NSEW")

graphFrame = ttk.Frame(mainFrame, style="graphFrame.TFrame", width=150,
height=210)
graphFrame.grid(row=1, column=1, sticky="NSEW")

infoFrame = ttk.Frame(mainFrame, style="infoFrame.TFrame", width=50,
height=210)
infoFrame.grid(row=1, column=2, sticky="NSE")

mainFrame.grid_rowconfigure(0, weight=1)
mainFrame.grid_rowconfigure(1, weight=1)
mainFrame.grid_columnconfigure(0, weight=1)
mainFrame.grid_columnconfigure(1, weight=1)

```

Beberapa frame dibuat sebagai container untuk elemen-elemen GUI. mainFrame merupakan frame utama yang menampung semua elemen GUI. bannerFrame digunakan untuk menampilkan gambar latar belakang. inputFrame digunakan untuk menampung elemen-elemen input dan tombol. graphFrame digunakan untuk menampilkan graf. infoFrame digunakan untuk menampilkan tabel code dan nama dari node-node graf.

```

# Mengatur gambar latar belakang
image = Image.open("doctor1.png") # Ganti dengan path ke gambar yang ingin
Anda gunakan
resized_image = image.resize((1360, 260), Image.LANCZOS)

# Konversi gambar ke format yang dapat ditampilkan di Tkinter
background_image = ImageTk.PhotoImage(resized_image)
background_label = tk.Label(bannerFrame, image=background_image,
highlightthickness=0)
background_label.grid(padx=0, pady=0, sticky="nsew")

```

Elemen gambar latar belakang dibuat menggunakan modul PIL. Gambar tersebut diubah menjadi format yang dapat ditampilkan di Tkinter menggunakan

ImageTk.PhotoImage dan kemudian ditempatkan di dalam label background_label di bannerFrame.

```
# Membuat Menu Bar
menubar = tk.Menu(root)
root.config(menu=menubar)

# Menu File
file_menu = tk.Menu(menubar, tearoff=0)
menubar.add_cascade(label="File", menu=file_menu)
file_menu.add_command(label="Reset", command=reset_display)

# Menu Help
help_menu = tk.Menu(menubar, tearoff=0)
menubar.add_cascade(label="Help", menu=help_menu)
help_menu.add_command(label="User Guide", command=show_user_guide)
```

Membuat menu bar menggunakan tk.Menu(root). Dua menu utama yaitu "File" dan "Help" dibuat menggunakan tk.Menu(menubar, tearoff=0). Kemudian menu-menu tersebut diatur dan ditambahkan ke menu bar menggunakan add_cascade().

```
# Membuat label dan combobox untuk memilih node awal dan akhir
start_label = ttk.Label(inputFrame, text="Masukkan titik awal: ")
start_label.configure(background="white", foreground="black", font=("Cambria", 15, "bold"))
start_label.grid(row=0, column=0, padx=5, pady=5, sticky="nsew")

start_combobox = ttk.Combobox(inputFrame, values=codes)
start_combobox.grid(row=1, column=0, padx=5, pady=5, sticky="nsw")
```

Elemen-elemen input seperti label dan combobox dibuat menggunakan ttk.Label dan ttk.Combobox. Label "Masukkan titik awal:" ditampilkan pada start_label, sedangkan start_combobox digunakan untuk memilih titik awal dari combobox dengan nilai-nilai kode yang didapat dari codes.

```
# Membuat tombol untuk mencari rute terpendek
search_button = ttk.Button(inputFrame, text="Cari rute", command=run_function)
search_button.grid(row=2, column=0, padx=5, pady=5, sticky="nsw")
```

Tombol "Cari rute" dibuat menggunakan ttk.Button dengan teks "Cari rute" dan diposisikan di inputFrame. Ketika tombol ditekan, fungsi run_function() akan dijalankan.

```
# Membuat label untuk menampilkan hasil
nearest_doctor_label = ttk.Label(inputFrame,background="white", text="Dokter
terdekat: ", font='normal 10')
nearest_doctor_label.grid(row=3, column=0, padx=5, pady=5, sticky="nsew")

path_label = ttk.Label(inputFrame, background="white", text="Jalur terpendek: ",
font='normal 10')
path_label.grid(row=4, column=0, padx=5, pady=5, sticky="nsew")

distance_label = ttk.Label(inputFrame, background="white", text="Jarak: ",
font='normal 10')
distance_label.grid(row=5, column=0, padx=5, pady=5, sticky="nsew")
```

Label `nearest_doctor_label`, `path_label`, dan `distance_label` dibuat menggunakan `ttk.Label` di `inputFrame`. Awalnya, label tersebut memiliki teks kosong dan akan diubah saat fungsi `bfs_shortest_path()` dijalankan.

```
# Membuat widget untuk menampilkan tabel code dan nama dari nodes dengan
kemampuan scrolling
nodes_treeview = ttk.Treeview(infoFrame, height=19, show="headings")
nodes_treeview['columns'] = ('code', 'name')
nodes_treeview.heading('code', text='Code')
nodes_treeview.heading('name', text='Name')
nodes_treeview.column('code', width=100)
nodes_treeview.column('name', width=200)

scrollbar = ttk.Scrollbar(infoFrame, orient='vertical',
command=nodes_treeview.yview)
scrollbar.grid(row=0, column=1, sticky="ns")
nodes_treeview.configure(yscrollcommand=scrollbar.set)
```

Tabel `nodes_treeview` dibuat menggunakan `ttk.Treeview` di `infoFrame` dengan tinggi 19 baris dan hanya menampilkan kepala (headings). Kolom-kolom yang ditampilkan adalah "Code" dan "Name" dengan lebar masing-masing 100 dan 200 piksel. Tabel tersebut diisi dengan kode dan nama node dari variabel `codes` dan `names` menggunakan metode `insert()`. Untuk mengaktifkan fungsi tampilan scroll pada tabel, scrollbar dibuat menggunakan `ttk.Scrollbar` dengan orientasi vertikal. Scrollbar tersebut dikaitkan dengan tabel menggunakan metode `configure()`.

```
# Mengisi tabel code dan nama dari nodes
for code, name in zip(codes, names):
    nodes_treeview.insert("", 'end', values=(code, name))

nodes_treeview.grid(row=0, column=0, pady=10, sticky="nseW")

root.mainloop()
```

Mengatur tata letak grid dengan `grid()` untuk semua frame dan elemen-elemen di dalamnya, termasuk pengaturan berat baris dan kolom agar dapat menyesuaikan ukuran jendela. Terakhir, menjalankan event loop utama menggunakan `root.mainloop()` untuk menampilkan GUI dan merespons interaksi pengguna.

4.2 Mengubah Kode Script menjadi Executable File

Untuk memudahkan pengguna dalam menggunakan aplikasi GUI yang telah dibuat, file python harus diubah menjadi file yang dapat dijalankan tanpa interpreter. Salah satu cara untuk melakukannya adalah dengan menggunakan auto-py-to-exe untuk mengubah file py menjadi exe. Berikut adalah langkah-langkah untuk mengubah file py menjadi exe:

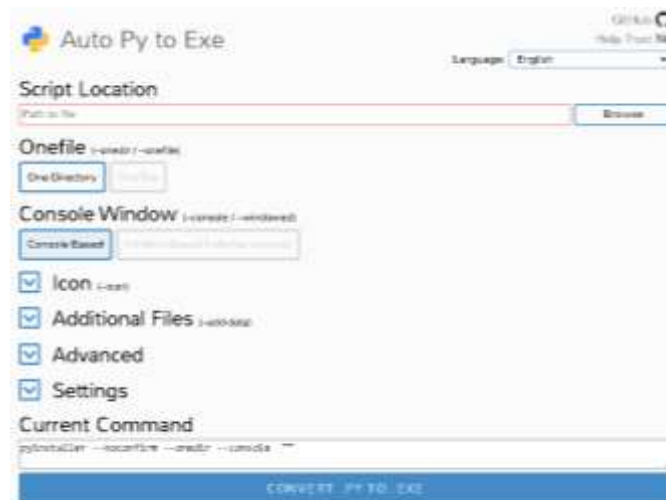
1. Install auto-py-to-exe melalui terminal dengan menggunakan perintah yang sesuai.

```
PS C:\Users\ismic\Documents\File Project Alpro> pip install auto-py-to-exe
```

2. Setelah instalasi selesai, buka auto-py-to-exe melalui terminal dengan mengetikkan perintah yang tepat.

```
PS C:\Users\ismic\Documents\File Project Alpro> auto-py-to-exe
```

3. Setelah aplikasi terbuka, akan terlihat tampilan auto-py-to-exe yang akan digunakan untuk melakukan konversi file.



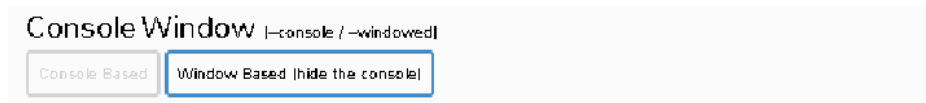
4. Pada langkah ini, pilih lokasi file py utama yang ingin diubah menjadi file exe.



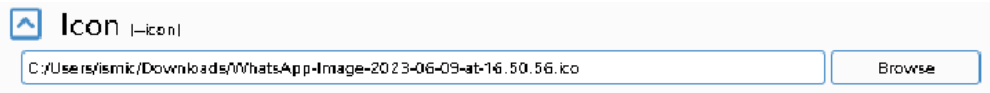
5. Kemudian tentukan jenis file apakah ingin one directory atau one file. Jika memilih one directory, hasil konversi akan menghasilkan output dalam bentuk satu directory yang berisi file-file terkait. Sedangkan jika memilih one file, hasil konversi akan menghasilkan satu file exe tunggal.



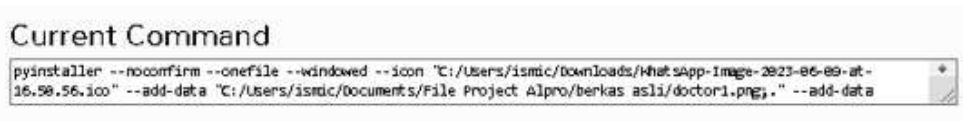
6. Selain itu, tentukan jenis aplikasi yang ingin dihasilkan, apakah berbasis konsol atau berbasis window. Pilihan ini akan mempengaruhi tampilan dan perilaku aplikasi yang dihasilkan.



7. Pilih gambar berekstensi ico yang akan digunakan sebagai ikon aplikasi “GoingDoc”.



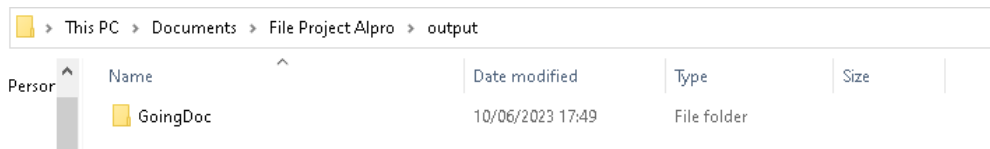
8. Setelah menyelesaikan pengaturan yang diperlukan, perintah pengaturan yang telah diatur akan ditampilkan pada current command di auto-py-to-exe.



9. Klik tombol convert py to exe untuk memulai proses konversi, Auto-py-to-exe akan memulai proses mengubah file py menjadi file exe berdasarkan pengaturan yang ditentukan sebelumnya. Kemudian tunggu hingga proses konversi selesai.



10. Setelah proses konversi selesai, auto-py-to-exe akan menghasilkan file output yaitu direktori file “GoingDoc” sesuai pengaturan yang ditentukan sebelumnya.



11. Berikut adalah tampilan aplikasi “GoingDoc” yang telah menjadi executable file.



GoingDoc

Gambar 8. Ikon Aplikasi GoingDoc

Dengan mengubah file aplikasi menjadi file executable, pengguna dapat dengan mudah menggunakan aplikasi tersebut untuk mencari rute terdekat ke tempat praktik mandiri dokter umum di kecamatan Gunung Anyar. Hal ini akan mengatasi permasalahan masyarakat yang kesulitan dalam mencari rute terpendek dan tercepat ke tempat praktik mandiri dokter umum di daerah tersebut. Saat ini, belum banyak aplikasi yang memudahkan masyarakat Gunung Anyar dalam menemukan praktik dokter umum terdekat, sehingga transformasi file menjadi executable akan memberikan solusi praktis bagi mereka.

4.3 Waktu Pemrosesan

```
ie  
= time.time()
```

Berdasarkan kode script diatas, modul time diperlukan untuk fungsi waktu dalam Python. Tujuan dari kode tersebut adalah untuk memulai penghitungan waktu sebelum eksekusi suatu blok kode. Dengan menyimpan waktu awal dalam variabel start_time, kita dapat mengukur durasi eksekusi dengan menghitung selisih antara waktu akhir dengan waktu awal.

```
asi graf  
raph()  
ite_graph()  
graph)  
st_path("T31")
```

Berdasarkan kode script diatas, langkah pertama adalah membuat objek graf dan menginisialisasi graf kosong menggunakan metode `create_graph()`. Setelah graf diinisialisasi, objek BFS (Penjelajahan Terlebar) dibuat dengan menggunakan graf tersebut. Setelah objek BFS dibuat, metode `shortest_path()` dipanggil untuk mencari jalur terpendek dari simpul awal ke simpul tujuan yang ditentukan. Dalam contoh ini, simpul tujuan adalah "T31".

```
= time.time()
_time = end_time - start_time
```

```
ktu pemrosesan.", execution_time, "detik")
```

Proses ini melibatkan pengukuran waktu awal dan akhir menggunakan fungsi `time.time()` serta perhitungan waktu eksekusi dengan mengurangi waktu akhir dari waktu awal. Hasilnya adalah waktu pemrosesan dalam detik, yang kemudian ditampilkan sebagai output.

Tabel 3. Hasil dan Waktu Pencarian Rute Terpendek

| Titik Awal | Dokter Terdekat | Rute Terdekat | Jarak Terpendek | Waktu Pencarian |
|------------|-----------------|---------------|-----------------|-----------------|
| T1 | D1 | T1→T3→D1 | 0,69 km | 0.010s |
| T2 | D1 | T2→T4→D1 | 0,63 km | 0,000s |
| T3 | D1 | T3→D1 | 0,15 km | 0,005s |
| T4 | D1 | T4→D1 | 0,28 km | 0,004s |
| T5 | D1 | T5→D1 | 0,91 km | 0,006s |
| T6 | D1 | T6→T4→D1 | 0,94 km | 0,004s |
| T7 | D1 | T7→T6→T4→D1 | 1,36 km | 0,000s |
| T8 | D3 | T8→D3 | 0,44 km | 0,000s |

| | | | | |
|-----|----|----------------|---------|--------|
| T9 | D3 | T9→D3 | 0,59 km | 0,006s |
| T10 | D3 | T10→D3 | 0,27 km | 0,000s |
| T11 | D2 | T11→D2 | 0,15 km | 0,005s |
| T12 | D2 | T12→D2 | 0,54 km | 0,000s |
| T13 | D2 | T13→T12→D2 | 0,72 km | 0,000s |
| T14 | D2 | T14→D2 | 0,77 km | 0,000s |
| T15 | D2 | T15→T14→D2 | 1,41 km | 0,005s |
| T16 | D2 | T16→T15→T14→D2 | 1,55 km | 0,000s |
| T17 | D2 | T17→T18→T11→D2 | 1,41 km | 0,000s |
| T18 | D2 | T18→T11→D2 | 0,85 km | 0,006s |
| T19 | D4 | T19→D4 | 0,14 km | 0,000s |
| T20 | D5 | T20→D5 | 0,13 km | 0,003s |
| T21 | D4 | T21→T19→D4 | 0,40 km | 0,000s |
| T22 | D5 | T22→D5 | 0,32 km | 0,006s |
| T23 | D5 | T23→T22→D5 | 0,56 km | 0,001s |
| T24 | D5 | T24→T23→T22→D5 | 1,03 km | 0,000s |
| T25 | D6 | T25→T28→D6 | 1,23 km | 0,000s |
| T26 | D6 | T26→T28→D6 | 1,98 km | 0,000s |
| T27 | D2 | T27→T14→D2 | 3,04 km | 0,000s |

| | | | | |
|----------------------------|----|--------------------|---------|--------|
| T28 | D6 | T28→D6 | 0,42 km | 0,003s |
| T29 | D6 | T29→T28→D6 | 0,84 km | 0,005s |
| T30 | D6 | T30→T29→T28→D6 | 1,27 km | 0,004s |
| T31 | D6 | T31→T30→T29→T28→D6 | 1,47 km | 0,005s |
| T32 | D6 | T32→D6 | 0,26 km | 0,003s |
| T33 | D6 | T33→D6 | 0,53 km | 0,008s |
| T34 | D7 | T34→D7 | 0,17 km | 0,000s |
| T35 | D7 | T35→T34→D7 | 0,39 km | 0,000s |
| T36 | D7 | T36→T35→T34→D7 | 0,49 km | 0,002s |
| T37 | D6 | T37→D6 | 1,88 km | 0,000s |
| Rata-rata waktu pemrosesan | | | | 0,003 |

Pada analisis waktu pemrosesan output, dilakukan pengukuran untuk mengevaluasi kecepatan dan efisiensi aplikasi dalam menampilkan output. Pengukuran ini melibatkan pengamatan terhadap waktu yang diperlukan aplikasi untuk melakukan proses pengolahan dan perhitungan sebelum menghasilkan output akhir. Hasil pengukuran menunjukkan bahwa rata-rata waktu pemrosesan output oleh aplikasi sangat cepat, yaitu sekitar 0,003 detik. Waktu yang singkat ini menunjukkan bahwa aplikasi memiliki kinerja yang sangat cepat dalam menghasilkan output. Dengan demikian, aplikasi berhasil mencapai tingkat kecepatan yang tinggi dalam pemrosesan output sehingga pengguna dapat mendapatkan hasil dengan cepat dan efisien.

4.4 Akurasi

Dalam analisis hasil akurasi aplikasi, dilakukan evaluasi terhadap kemampuan aplikasi dalam menampilkan jarak terpendek rute pencarian tempat praktik mandiri dokter

umum di Kecamatan Gunung Anyar. Perhitungan dilakukan dengan menggunakan rumus yang telah dideskripsikan sebelumnya pada metodologi penelitian.

Tabel 4. Persentase Error Data Jarak Aktual dan Jarak Output

| No | Jarak aktual (km) | Jarak output (km) | Error | Error% |
|----|-------------------|-------------------|-------|--------|
| 1 | 1,20 | 0,69 | 0,51 | 42,50 |
| 2 | 0,61 | 0,63 | 0,02 | 3,27 |
| 3 | 0,50 | 0,15 | 0,35 | 70,00 |
| 4 | 0,26 | 0,28 | 0,02 | 7,69 |
| 5 | 2,30 | 0,91 | 1,39 | 60,43 |
| 6 | 1,26 | 0,94 | 0,32 | 25,40 |
| 7 | 1,76 | 1,36 | 0,40 | 22,73 |
| 8 | 0,45 | 0,44 | 0,01 | 2,22 |
| 9 | 0,90 | 0,59 | 0,31 | 34,44 |
| 10 | 0,28 | 0,27 | 0,01 | 3,57 |
| 11 | 0,16 | 0,15 | 0,01 | 6,25 |
| 12 | 0,50 | 0,54 | 0,04 | 8,00 |
| 13 | 0,75 | 0,72 | 0,03 | 4,00 |
| 14 | 0,75 | 0,77 | 0,02 | 2,67 |
| 15 | 1,35 | 1,41 | 0,06 | 4,44 |
| 16 | 1,48 | 1,55 | 0,07 | 42,57 |
| 17 | 1,66 | 1,41 | 0,25 | 91,57 |
| 18 | 0,96 | 0,85 | 0,11 | 86,45 |
| 19 | 0,17 | 0,14 | 0,03 | 135,20 |

| | | | | |
|-----------------|------|------|------|--------|
| 20 | 0,12 | 0,13 | 0,01 | 8,33 |
| 21 | 0,44 | 0,40 | 0,04 | 9,09 |
| 22 | 1,30 | 0,32 | 0,98 | 75,38 |
| 23 | 1,65 | 0,56 | 1,09 | 66,06 |
| 24 | 2,15 | 1,03 | 1,12 | 52,09 |
| 25 | 1,45 | 1,23 | 0,22 | 15,17 |
| 26 | 2,55 | 1,98 | 0,57 | 22,35 |
| 27 | 1,00 | 3,04 | 2,04 | 204,00 |
| 28 | 0,65 | 0,42 | 0,23 | 35,38 |
| 29 | 1,20 | 0,84 | 0,36 | 30,00 |
| 30 | 3,00 | 1,27 | 1,73 | 57,66 |
| 31 | 3,26 | 1,47 | 1,79 | 54,90 |
| 32 | 0,28 | 0,26 | 0,02 | 7,14 |
| 33 | 0,85 | 0,53 | 0,32 | 37,64 |
| 34 | 0,16 | 0,17 | 0,01 | 6,25 |
| 35 | 0,22 | 0,39 | 0,17 | 77,27 |
| 36 | 0,57 | 0,49 | 0,08 | 14,03 |
| 37 | 2,10 | 1,88 | 0,22 | 10,47 |
| Rata-rata error | | | | 38,62% |

$$Akurasi = 100\% - 38,62\% = 61,38\%$$

Setelah dilakukan dengan akurasi dengan menggunakan rumus yang telah disebutkan sebelumnya, hasil menunjukkan bahwa jarak yang ditampilkan oleh aplikasi memiliki

tingkat kesalahan 38,62 persen dibandingkan dengan jarak aktual yang diperoleh melalui Google Maps. Dengan demikian, akurasi yang didapatkan yaitu sebesar 61,38 persen. Hal ini menunjukkan bahwa meskipun aplikasi berhasil mencapai tingkat akurasi yang cukup tinggi, terdapat potensi untuk melakukan perbaikan lanjut guna meningkatkan ketepatan dalam menampilkan jarak terpendek rute pencarian tempat praktik mandiri dokter umum di Gunung Anyar.

BAB V

KESIMPULAN

Dalam proyek ini, peneliti mengidentifikasi permasalahan yang dihadapi oleh masyarakat Gunung Anyar dalam mencari rute terpendek dan tercepat ke tempat praktik mandiri dokter umum. Peneliti melihat bahwa belum banyak aplikasi yang dapat memudahkan masyarakat dalam mencari praktik dokter umum terdekat. Untuk mengatasi masalah ini, peneliti mengusulkan pendekatan menggunakan metode graf transversal BFS dalam mencari rute terpendek ke tempat praktik mandiri dokter umum di Gunung Anyar. BFS adalah metode efektif dalam pencarian rute terpendek dalam graf. Dalam proyek ini, peneliti menggunakan BFS untuk menjelajahi graf praktik dokter umum secara sistematis dan efisien.

Metode BFS kemudian dikembangkan menjadi sebuah aplikasi bernama “GoingDoc”. Aplikasi ini dapat digunakan untuk mencari dokter terdekat, rute terpendek, dan jarak yang harus ditempuh. Hal ini dapat membantu masyarakat Gunung Anyar untuk mencari rute terpendek menuju tempat praktik mandiri dokter umum. Waktu pemrosesan pencarian output ini rata-rata hanya 0,003 detik, menunjukkan bahwa aplikasi memiliki kinerja yang sangat cepat dalam menghasilkan output. Selain itu, aplikasi ini memiliki tingkat akurasi sebesar 61,38 persen.. Hal ini menunjukkan bahwa meskipun aplikasi berhasil mencapai tingkat akurasi yang cukup tinggi, terdapat potensi untuk melakukan perbaikan lanjut guna meningkatkan ketepatan dalam menampilkan jarak terpendek rute pencarian tempat praktik mandiri dokter umum di Gunung Anyar.

Penyusunan Kode Script untuk implementasi metode BFS dan GUI dilakukan menggunakan pendekatan Object-Oriented Programming (OOP). Kode script ini terdiri dari dua modul, yaitu “ver2.py” dan “GoingDoc.py”. Tujuannya adalah untuk menjaga mencegah kode yang terlalu panjang sehingga kode tetap tampak rapi. Selain itu, keuntungan menggunakan pendekatan ini adalah kemudahan dalam penggunaan ulang, terutama modul “ver2.py” yang berisi class Graph dan Class bfs yang dapat digunakan untuk membuat dan memanipulasi graf pada masalah yang serupa. Secara keseluruhan, rancangan kode script pemrograman berbasis OOP ini memiliki tingkat keringkasan yang jelas, kerapian yang terjaga, dan memungkinkan penggunaan kembali yang efektif. Dengan menggunakan pendekatan ini, pengembangan aplikasi dapat dilakukan dengan lebih efektif di masa depan.

DAFTAR PUSTAKA

- A. Cooper, R. Reimann, D. Cronin, and C. Noessel. 2014. "About face: the essentials of interaction design." *John Wiley & Sons*.
- A. L. d. S. Lima and C. Gresse von Wangenheim. 2022. "Assessing the Visual Esthetic of User Interfaces: A Ten-Year Systematic Mapping." *International Journal of Human-Computer Interaction* 38:144-164. no 2.
- Anaconda. 2021. "Anaconda | The World's Most Popular Data Science Platform." Anaconda | The World's Most Popular Data Science Platform. <https://www.anaconda.com/>.
- Anwari, Hozairi. 2019. "Perbandingan Algoritma Breadth First Search dan Dijkstra untuk Penentuan Rute Terpendek Pengiriman Barang Unilever." *Perbandingan Algoritma Breadth First Search*.
- A. S. Putra. 2018. "2018 Artikel Struktur Data, Audit Dan Jaringan Komputer."
- Badri, Imam I. 2019. "Contoh Perhitungan Uji Akurasi (Pengolahan Data)." TeachMeSoft. <https://www.teachmesoft.com/2019/12/contoh-perhitungan-uji-akurasi.html>.
- Daud, J. 2005. "Studi Efektifitas Penggunaan jalan Kota Medan." *Jurnal Sistem Teknik Industri* 6. No 3.
- Delima Zai, Haeni Budiati, and Sunneng S. Berutu. 2016. "Simulasi Rute Terpendek Lokasi Pariwisata di Nias dengan Metode Breadth First Search dan Tabu Search."
- D. Kuhlman. 2013. "A Python Book." *A Python B*, 1-227.
- Erniyati, M. K., and M. K. Mulyati. 2018. "Pencarian Jalur Terdekat Menuju Rumah Sakit di Kota Bogor dengan Menggunakan Algoritma A*."
- Fajri, T. Rahmad Effendi, Nurul Fadillah. 2020. "Sistem Absensi Berbasis Pengenalan Wajah Secara Real Time menggunakan Metode Fisherface." 351.
- Hari, Toha H. 2019. "Sistem Penunjang Keputusan Pencarian Jarak Terpendek Menuju Rumah Sakit dan Puskesmas dengan Metode Dijkstra."
- Haris Muhammad Syarif. 2019. "Aplikasi Sistem Layanan Periksa Dokter pada Dokter Praktek Umum Mandiri."
- I. Frank. 2006. "Calculating Geographic Distance: Concepts and Methods." (Canadian Institute for Health Information, Toronto, Ontario, Canada), 2006.
- Juhara, and Zamrony P. 2016. "Panduan Lengkap Pemrograman Android."

- “Mengenal Lebih Jauh Mengenai Anaconda Python.” n.d. Software ERP Indonesia - Web & ERP Developer Indonesia - Jasa Pembuatan ERP Sistem Cloud. Accessed Mei 23, 2023. <http://idmetafora.com>.
- M Rizal. 2016. “Analisa Komparasi Queue Simple dan Queue Tree pada Manajemen Bandwith Mikrotik di SMKN 5 Mataram.” *Naskah Publikasi STIMIK AMIKOM Yogyakarta*.
- “Pasal 4 ayat (1) huruf a Peraturan Pemerintah Nomor 47 Tahun 2016 tentang Fasilitas Pelayanan Kesehatan.” n.d.
- “Permenkes No. 1438/Menkes/ Per/IX/2010 tentang Standar Pelayanan Kedokteran/03/14/peraturan-menteri-kesehatan-republik-indonesia-nomor-1438menkesperix2010-tentang-standar-pelayanan-kedokteran.html.” 2017. Inspektorat Jenderal Kemhan RI. <https://www.kemhan.go.id/itjen/2017/03/14/peraturan-menteri-kesehatan-republik-indonesia-nomor-1438menkesperix2010-tentang-standar-pelayanan-kedokteran.html>.
- “Python | time.time() method.” 2019. GeeksforGeeks. <https://www.geeksforgeeks.org/python-time-time-method/>.
- Sayed Fachrurrazi. 2018. “Sistem Penentuan Rute Yang Tepat Dalam Sebuah Labirin Dengan Menerapkan Algoritma Prim.” 60.
- Syahrudin, A. N., and T. Kurniawan. 2018. “Input Dan Output Pada Bahasa.” *J. Dasar Pemrograman Python STMIK*, 1-7.
- T. Schlatter and D. Levinson. 2013. “Visual usability: Principles and practices for designing digital applications.” *Newnes*.

LAMPIRAN

1. Data posisi node

| Nama | Kode | Latitude | Longitude |
|--|-------------|-----------------|------------------|
| Dr. Abdul | D1 | -7.33416651 | 112.81106319 |
| Dr. Fitri Yuda Mayasari | D2 | -7.34177390 | 112.80172376 |
| Dr. Ani Budiwati | D3 | -7.33597052 | 112.80002715 |
| Dr. Eko Iswahyudi | D4 | -7.33652257 | 112.79632167 |
| Dr. Jimmy S. | D5 | -7.33331395 | 112.79686486 |
| Dr. Luksadi | D6 | -7.33149401 | 112.77200001 |
| Dr. Brana Pradata | D7 | -7.33603313 | 112.76843121 |
| Mushola & TPQ Jl. Wisma Tirta Agung Asri | T1 | -7.33716645 | 112.81448358 |
| Perumahan Wisma Indah | T2 | -7.33607377 | 112.81059970 |
| Musholla AlFattah Puri Gununganyar Regency | T3 | -7.33533380 | 112.81152717 |
| Taman Gunung Anyar | T4 | -7.33464206 | 112.80900963 |
| Rusunawa Gunung Anyar | T5 | -7.33095389 | 112.81475987 |
| Musholla Hubbul Wathon | T6 | -7.33199479 | 112.80540012 |
| Masjid Al Muttaqin, Gunung Anyar Emas | T7 | -7.33440798 | 112.80317085 |
| Bengkel cat mobil Grand Star | T8 | -7.33271733 | 112.80077577 |
| DAMKAR GUNUNG ANYAR | T9 | -7.33805311 | 112.80194598 |
| Masjid Nur Madrikah | T10 | -7.33838505 | 112.80006815 |
| Panti Asuhan Ulul Azmi | T11 | -7.34039890 | 112.80190324 |
| SD Budi Mulia Islamic Shool | T12 | -7.34269705 | 112.80575107 |
| Gunung Anyar Town House | T13 | -7.34425333 | 112.80543159 |
| Masjid Roudlotul Mukminin | T14 | -7.34414147 | 112.80155038 |
| Gunung Anyar Asri | T15 | -7.34375594 | 112.80123834 |
| PERUM BUMI PRATAMA ASRI | T16 | -7.34337962 | 112.80019323 |

| | | | |
|--|-----|-------------|--------------|
| Politeknik Pelayaran Surabaya | T17 | -7.34302591 | 112.79479166 |
| Perumahan Gunung Anyar Permai | T18 | -7.33860742 | 112.79596451 |
| Masjid al Mubarak | T19 | -7.33669157 | 112.79508927 |
| PT. Berhasil Sinar Gemilang Abadi | T20 | -7.33390413 | 112.79720483 |
| Apotek Doa Sehat | T21 | -7.33470896 | 112.79576748 |
| SPBU Pertamina - Gunung Anyar | T22 | -7.33344254 | 112.79394668 |
| Fotocopy Alfin | T23 | -7.33294780 | 112.79221827 |
| UPN "Veteran" Jawa Timur | T24 | -7.33225209 | 112.78849596 |
| Mr. Suprek Rungkut | T25 | -7.33157163 | 112.78298514 |
| Masjid Assalam Purimas | T26 | -7.33903455 | 112.78466655 |
| UINSA Kampus 2 | T27 | -7.34422439 | 112.78665880 |
| McDonald's - Rungkut Madya | T28 | -7.33132323 | 112.77560415 |
| SPBU Pertamina 54.601.99 Rungkut Mapan | T29 | -7.33477780 | 112.77505521 |
| SDN Rungkut Mananggal I | T30 | -7.33605799 | 112.77330609 |
| Masjid Al-Muslimun | T31 | -7.33668695 | 112.77288917 |
| Kantor Kelurahan Rungkut Tengah | T32 | -7.33312868 | 112.77110175 |
| Masjid At-Taibin | T33 | -7.33302809 | 112.76853020 |
| Kantor Kelurahan Rungkut Mananggal | T34 | -7.33746547 | 112.76860250 |
| Pondok Pesantren Manbaul Falah | T35 | -7.33895626 | 112.76793831 |
| Taman Rungkut Mananggal | T36 | -7.33850203 | 112.76706837 |
| Masjid Baiturrozaq SIER Surabaya | T37 | -7.33222082 | 112.75701217 |

2. Data jarak antar node

| Asal | Tujuan | Jarak (km) | Asal | Tujuan | Jarak (km) |
|------|--------|------------|------|--------|------------|
| T1 | T2 | 0.51 | D3 | D4 | 0.54 |
| T1 | T3 | 0.54 | T8 | D5 | 0.43 |
| T2 | T4 | 0.35 | T20 | D5 | 0.13 |


| | | | | | |
|-----|-----|------|-----|-----|------|
| T3 | T4 | 0.36 | T19 | T21 | 0.26 |
| T4 | D1 | 0.28 | T21 | T22 | 0.35 |
| T5 | D1 | 0.91 | T22 | D5 | 0.32 |
| T4 | T6 | 0.66 | T22 | T23 | 0.24 |
| T5 | T6 | 1.04 | T23 | T24 | 0.47 |
| T6 | T7 | 0.42 | T24 | T25 | 0.61 |
| T6 | T8 | 0.53 | T24 | T26 | 1.08 |
| T8 | D3 | 0.44 | T25 | T26 | 1.15 |
| T9 | D3 | 0.59 | T25 | T28 | 0.81 |
| T10 | D3 | 0.27 | T26 | T27 | 0.83 |
| T9 | T11 | 0.26 | T26 | T28 | 1.56 |
| T11 | D2 | 0.15 | T28 | T29 | 0.42 |
| T12 | D2 | 0.54 | T29 | T30 | 0.43 |
| T12 | T13 | 0.18 | T30 | T31 | 0.2 |
| T13 | T14 | 0.56 | T28 | D6 | 0.42 |
| T14 | D2 | 0.77 | T32 | D6 | 0.26 |
| T14 | T15 | 0.64 | T33 | D6 | 0.53 |
| T15 | T16 | 0.14 | T33 | D7 | 0.33 |
| T16 | T17 | 0.64 | T34 | D7 | 0.17 |
| T17 | T18 | 0.56 | T34 | T35 | 0.22 |
| T18 | T11 | 0.7 | T35 | T36 | 0.1 |
| T14 | T27 | 2.27 | T37 | D6 | 1.88 |
| T18 | T19 | 0.36 | T3 | D1 | 0.15 |
| T19 | D4 | 0.14 | | | |

3. Mockup desain aplikasi GUI “GoingDoc”

a. Tampilan awal aplikasi

GoingDoc

File Help



FIND YOUR DOCTOR

in Gunung Anyar

Masukkan titik awal:

cari rute


Dokter terdekat:
Jalur terpendek:
Jarak:

| Code | Name |
|------|---|
| D1 | Dr. Abdul |
| D2 | Dr. Fitri Yuda Mayasari |
| D3 | Dr. Ani Budiwati |
| D4 | Dr. Eko Iswahyudi |
| D5 | Dr. Jimmy S. |
| D6 | Dr. Luksodi |
| D7 | Dr. Brana Pradota |
| T1 | Mushola & TPQ Jl. Wisma Tirta Agung Aari |
| T2 | Perumahan Wisma Indah |
| T3 | Mushola AlFattah Puri Gununganyar Regency |

b. Tampilan menu help

GoingDoc

File Help



FIND YOUR DOCTOR

in Gunung Anyar

Masukkan titik awal:

cari rute

Dokter terdekat:
Jalur terpendek:
Jarak:

| Code | Name |
|------|---|
| D1 | Dr. Abdul |
| D2 | Dr. Fitri Yuda Mayasari |
| D3 | Dr. Ani Budiwati |
| D4 | Dr. Eko Iswahyudi |
| D5 | Dr. Jimmy S. |
| D6 | Dr. Luksodi |
| D7 | Dr. Brana Pradota |
| T1 | Mushola & TPQ Jl. Wisma Tirta Agung Aari |
| T2 | Perumahan Wisma Indah |
| T3 | Mushola AlFattah Puri Gununganyar Regency |

c. Tampilan setelah menu help di klik

GoingDoc

File Help

User Guide

FIND YOUR DOCTOR

In Gunung Anyar

Masukkan titik awal:

cari rute

Dokter terdekat:
Jalur terpendek:
Jarak:

| Code | Name |
|------|--|
| D1 | Dr. Abdul |
| D2 | Dr. Fitri Yuda Mayasari |
| D3 | Dr. Ani Budiwati |
| D4 | Dr. Eko Iswahyudi |
| D5 | Dr. Jimmy S. |
| D6 | Dr. Luksodi |
| D7 | Dr. Brana Pradota |
| T1 | Mushola & TPQ Jl. Wisma Tirta Agung Asri |
| T2 | Perumahan Wisma Indah |
| T3 | Musholla AlFattah Puri Gununganyar Regency |

d. Tampilan fitur User Guide

GoingDoc

File Help

User Guide

FIND YOUR DOCTOR

In Gunung Anyar

Masukkan titik awal:

cari rute

Dokter terdekat:
Jalur terpendek:
Jarak:

Panduan Penggunaan Mencari Rute Dokter Terdekat:

OK

| Code | Name |
|------|--|
| D1 | Dr. Abdul |
| D2 | Dr. Fitri Yuda Mayasari |
| D3 | Dr. Ani Budiwati |
| D4 | Dr. Eko Iswahyudi |
| D5 | Dr. Jimmy S. |
| D6 | Dr. Luksodi |
| D7 | Dr. Brana Pradota |
| T1 | Mushola & TPQ Jl. Wisma Tirta Agung Asri |
| T2 | Perumahan Wisma Indah |
| T3 | Musholla AlFattah Puri Gununganyar Regency |

e. Tampilan untuk menutup User Guide

GoingDoc

File Help

FIND YOUR DOCTOR

in Gunung Anyar

Masukkan titik awal:

cari rute

Dokter terdekat:
Jalur terpendek:
Jarak:

Panduan Penggunaan Mencari Rute Dokter Terdekat:

OK

| Code | Name |
|------|--|
| D1 | Dr. Abdul |
| D2 | Dr. Fitri Yuda Mayasari |
| D3 | Dr. Ani Budiwati |
| D4 | Dr. Eko Iswahyudi |
| D5 | Dr. Jimmy S. |
| D6 | Dr. Luksadi |
| D7 | Dr. Brana Pradota |
| T1 | Mushola & TPQ Jl. Wisma Tirta Agung Aari |
| T2 | Perumahan Wisma Indah |
| T3 | Musholla AlFattah Puri Gununganyar Regency |

f. Tampilan sebelum mengeklik drop box kode node

GoingDoc

File Help

FIND YOUR DOCTOR

in Gunung Anyar

Masukkan titik awal:

cari rute

Dokter terdekat:
Jalur terpendek:
Jarak:

| Code | Name |
|------|--|
| D1 | Dr. Abdul |
| D2 | Dr. Fitri Yuda Mayasari |
| D3 | Dr. Ani Budiwati |
| D4 | Dr. Eko Iswahyudi |
| D5 | Dr. Jimmy S. |
| D6 | Dr. Luksadi |
| D7 | Dr. Brana Pradota |
| T1 | Mushola & TPQ Jl. Wisma Tirta Agung Aari |
| T2 | Perumahan Wisma Indah |
| T3 | Musholla AlFattah Puri Gununganyar Regency |

g. Tampilan ketika drop box di klik

GoingDoc

File Help



Masukkan titik awal:


D1
D2
D3
D4
D5
D6
D7
T1
T2
T3
T4
T5
T6

Code:
Name
D1 Dr. Abdul
D2 Dr. Fitri Yuda Mayasari
D3 Dr. Ani Budiwati
D4 Dr. Eko Iswahyudi
D5 Dr. Jimmy S.
D6 Dr. Luksodi
D7 Dr. Brana Pradata
T1 Mushola & TPQ Jl. Wisma Tirta Agung Aari
T2 Perumahan Wisma Indah
T3 Musholla AlFattah Puri Gununganyar Regency

h. Tampilan setelah memilih titik awal

GoingDoc

File Help



Masukkan titik awal:

T5

cari rute

Dokter terdekat:
Jalur terpendek:
Jarak:

Code:
Name
D1 Dr. Abdul
D2 Dr. Fitri Yuda Mayasari
D3 Dr. Ani Budiwati
D4 Dr. Eko Iswahyudi
D5 Dr. Jimmy S.
D6 Dr. Luksodi
D7 Dr. Brana Pradata
T1 Mushola & TPQ Jl. Wisma Tirta Agung Aari
T2 Perumahan Wisma Indah
T3 Musholla AlFattah Puri Gununganyar Regency

i. Tampilan tombol cari rute

GoingDoc

File Help

FIND YOUR DOCTOR

in Gunung Anyar

Masukkan titik awal:

T5

cari rute

Dokter terdekat:

Jalur terpendek:

Jarak:

| Code | Name |
|------|---|
| D1 | Dr. Abdul |
| D2 | Dr. Fitri Yuda Mayasari |
| D3 | Dr. Ani Budiwati |
| D4 | Dr. Eko Iswahyudi |
| D5 | Dr. Jimmy S. |
| D6 | Dr. Luksadi |
| D7 | Dr. Brana Pradota |
| T1 | Mushola & TPQ Jl. Wisma Tirta Agung Aari |
| T2 | Perumahan Wisma Indah |
| T3 | Mushalla AlFatah Puri Gununganyar Regency |

j. Tampilan hasil setelah cari rute di klik dan tampilan menu file

GoingDoc

File Help

FIND YOUR DOCTOR

in Gunung Anyar

Masukkan titik awal:

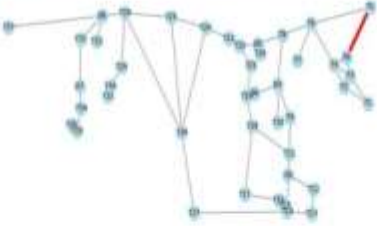
T5

cari rute

Dokter terdekat: D1

Jalur terpendek: T5 -> D1

Jarak: 0.91 km




| Code | Name |
|------|---|
| D1 | Dr. Abdul |
| D2 | Dr. Fitri Yuda Mayasari |
| D3 | Dr. Ani Budiwati |
| D4 | Dr. Eko Iswahyudi |
| D5 | Dr. Jimmy S. |
| D6 | Dr. Luksadi |
| D7 | Dr. Brana Pradota |
| T1 | Mushola & TPQ Jl. Wisma Tirta Agung Aari |
| T2 | Perumahan Wisma Indah |
| T3 | Mushalla AlFatah Puri Gununganyar Regency |

k. Tampilan fitur reset setelah menu file di klik

GoingDoc

File Help

Reset



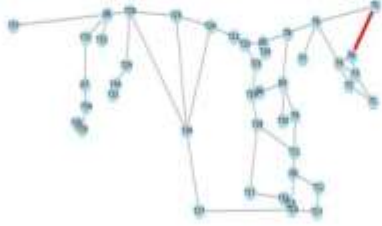
FIND YOUR DOCTOR

in Gunung Anyar

Masukkan titik awal:

T5

cari rute



Dokter terdekat: D1

Jalur terpendek: T5 -> D1

Jarak: 0.91 km

| Code | Name |
|------|---|
| D1 | Dr. Abdul |
| D2 | Dr. Fitri Yuda Mayasari |
| D3 | Dr. Ani Budiwati |
| D4 | Dr. Eko Iswahyudi |
| D5 | Dr. Jimmy S. |
| D6 | Dr. Luksadi |
| D7 | Dr. Brana Pradota |
| T1 | Mushola & TPQ Jl. Wisma Tirta Agung Aari |
| T2 | Perumahan Wisma Indah |
| T3 | Musholla AlFatah Puri Gununganyar Regency |

1. Tampilan setelah fitur reset di klik

GoingDoc

File Help



FIND YOUR DOCTOR

in Gunung Anyar

Masukkan titik awal:

cari rute

Dokter terdekat:

Jalur terpendek:

Jarak:

| Code | Name |
|------|---|
| D1 | Dr. Abdul |
| D2 | Dr. Fitri Yuda Mayasari |
| D3 | Dr. Ani Budiwati |
| D4 | Dr. Eko Iswahyudi |
| D5 | Dr. Jimmy S. |
| D6 | Dr. Luksadi |
| D7 | Dr. Brana Pradota |
| T1 | Mushola & TPQ Jl. Wisma Tirta Agung Aari |
| T2 | Perumahan Wisma Indah |
| T3 | Musholla AlFatah Puri Gununganyar Regency |