Visualisasi Interaktif Algoritma BFS untuk Pencarian Rute Terpendek Berbasis Canvas VCL

Ade Hastuty[1], Zulkifli[2], Syamsiani S[3], Syamsul Bahri[4]
[1,2,3]Universitas Muhammadiyah Parepare, Teknik Informatika, Fakultas Teknik, Jl. Ahmad Yani No. 1, Parepare, Sulawesi Selatan

Email : [1]adehastuty@iainpare.ac.id, [2]zulkifli32883@gmail.com, [3]syamsianisyam875@gmail.com, [4]syamsulbahrinonci@gmail.com,

Abstrak

Persoalan pencarian jalur terpendek pada graf tak berbobot lazim diselesaikan menggunakan algoritma *Breadth-First Search* (BFS) karena kesederhanaan dan jaminan optimalitasnya dalam hal jumlah langkah. Kendala yang sering muncul bukan pada algoritmanya, melainkan pada cara menyampaikan proses pencariannya: keluaran berupa angka atau teks membuat pengguna sulit memahami bagaimana simpul-simpul dikunjungi hingga solusi ditemukan. Penelitian ini merancang dan membangun aplikasi desktop berbasis *Visual Component Library* (*VCL*) Delphi yang memanfaatkan komponen *TCanvas* untuk menampilkan tahapan traversal BFS secara visual dan interaktif, sekaligus mengukur efisiensi eksekusinya menggunakan kelas *TStopwatch*. Pendekatan yang digunakan adalah eksperimen kuantitatif dengan dua skenario ukuran graf, yaitu 100 simpul dan 500 simpul, masing-masing diuji sebanyak tiga kali untuk memperoleh rata-rata waktu eksekusi yang stabil. Hasil pengujian menunjukkan aplikasi mampu menggambarkan proses kunjungan simpul secara bertahap dan memberikan pemahaman visual yang lebih jelas dibandingkan keluaran tekstual. Rata-rata waktu eksekusi tercatat 10.153 ms pada 100 simpul dan meningkat menjadi 56.309 ms pada 500 simpul, dengan pola kenaikan yang linier. Pola tersebut konsisten dengan kompleksitas waktu teoretis BFS, yaitu *O(V+E)*, sehingga implementasi struktur antrean pada sistem terbukti sesuai dengan karakteristik algoritmanya.

Kata Kunci: *Breadth-First Search*, *Delphi VCL*, visualisasi graf, waktu eksekusi, *TStopwatch*

ABSTRACT

Shortest-path determination on unweighted graphs is commonly solved using the Breadth-First Search (BFS) algorithm due to its simplicity and guaranteed optimality in terms of the number of steps taken. The recurring difficulty lies not in the algorithm itself but in how its search process is communicated: numeric or textual output makes it hard for users to grasp how nodes are visited before a solution is reached. This study designs and builds a desktop application based on Delphi's Visual Component Library (VCL), employing the TCanvas component to render BFS traversal stages visually and interactively, while also measuring execution efficiency using the TStopwatch class. A quantitative experimental approach was adopted with two graph-size scenarios, 100 nodes and 500 nodes, each tested three times to obtain a stable average execution time. Testing results show that the application successfully depicts the stage-by-stage node-visiting process and offers clearer visual understanding compared to textual output. The average execution time was recorded at 10.153 ms for 100 nodes and rose to 56.309 ms for 500 nodes, following a linear growth pattern. This pattern is consistent with BFS's theoretical time complexity of O(V+E), confirming that the queue implementation within the system aligns with the algorithm's expected characteristics.

Keywords: Breadth-First Search, Delphi VCL, graph visualization, execution time, TStopwatch

PENDAHULUAN

Persoalan menemukan jalur tersingkat antara dua titik telah lama menjadi bagian penting dari berbagai sistem berbasis teknologi informasi, mulai dari navigasi kendaraan, perutean paket data pada jaringan komputer, hingga sistem informasi geografis yang memetakan lokasi dan destinasi. Pada hampir seluruh konteks tersebut, hubungan antarobjek direpresentasikan sebagai graf yang tersusun atas simpul dan sisi, dan kecepatan maupun ketepatan dalam menelusuri graf tersebut sangat menentukan kualitas keputusan yang diambil oleh sistem (Sularno, Mulya, Astri, & Mulya, 2021).

Di antara berbagai metode penelusuran graf, *Breadth-First Search* (BFS) menjadi pilihan yang populer khususnya untuk graf tak berbobot karena cara kerjanya yang menjelajah simpul lapis demi lapis berdasarkan kedalaman yang sama sebelum berpindah ke lapisan berikutnya. Karakteristik ini membuat BFS selalu menemukan jalur dengan jumlah langkah paling sedikit dari titik asal ke titik tujuan, sehingga banyak dijadikan dasar pada aplikasi pencarian jalur maupun studi traversal graf secara umum (Aditiya & Herdiana, 2021) .

Persoalannya, kemudahan memahami BFS secara konseptual tidak selalu berbanding lurus dengan kemudahan memahami prosesnya ketika dijalankan. Kebanyakan implementasi hanya menyajikan hasil akhir berupa daftar simpul yang dilalui, tanpa memperlihatkan bagaimana algoritma bergerak dari satu simpul ke simpul lain hingga sampai pada solusi. Akibatnya, pengguna terutama mahasiswa yang baru mempelajari struktur data graf kesulitan membangun intuisi tentang mekanisme kerja algoritma hanya dari keluaran tekstual atau numerik semata (Burch, van de Wetering, Wallner, Rooks, & Morra, 2023).

Kondisi ini semakin terasa dalam konteks pembelajaran, di mana pemahaman terhadap tahapan proses jauh lebih bernilai daripada sekadar mengetahui jawaban akhirnya (Kolomeets, Desnitsky, Kotenko, & Chechulin, 2023) .

Kajian-kajian terdahulu memang telah banyak membahas persoalan rute terpendek, namun sebagian besar berfokus pada perbandingan performa antaralgoritma. Sularno et al. (2021) menunjukkan bahwa BFS efektif digunakan untuk menentukan rute menuju lokasi evakuasi bencana pada graf tak berbobot, sementara Aditiya & Herdiana (2021) membandingkan performa BFS dengan algoritma A* pada beragam konfigurasi jalur. Di sisi lain, algoritma berbobot seperti Dijkstra dan Floyd-Warshall juga telah diuji pada berbagai kasus praktis, mulai dari navigasi kampus oleh Saputrama & Hartatiana (2021) hingga pemetaan destinasi wisata oleh Krisnamurti & Geong (2021), dan terus dikembangkan pada ranah sistem informasi geografis maupun pemodelan graf untuk kebutuhan edukasi, sebagaimana terlihat pada optimasi jalur oleh Taneo, Ndun, Fallo, & Do'o (2025) dan analisis pemahaman mahasiswa informatika oleh Jaiman, Fallo, & Banung (2025)

Meski literatur tersebut memberikan fondasi teoretis yang memadai mengenai efisiensi komputasi, aspek visualisasi terhadap proses traversal itu sendiri masih jarang menjadi fokus utama. Padahal, sejumlah studi menegaskan bahwa representasi visual berperan penting dalam menjembatani kesenjangan pemahaman pengguna terhadap perilaku algoritma secara real-time baik dalam konteks analisis rute di lingkungan urban (Alvito & Ikhsan, 2025) maupun dalam pemodelan alternatif berbasis bioinformatika (Kolomeets et al., 2023), sebagaimana juga ditegaskan oleh Burch et al. (2023) bahwa

visualisasi dinamis membantu mengungkap pola kerja algoritma graf yang sebelumnya tersembunyi di balik angka.

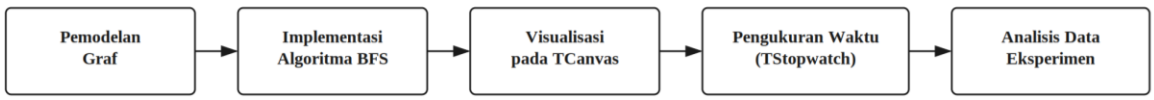
Kebaruan penelitian ini terletak pada integrasi visualisasi traversal BFS secara real-time dengan pengukuran performa presisi dalam satu aplikasi desktop, sebuah kombinasi yang belum banyak dieksplorasi pada penelitian rute terpendek sebelumnya yang cenderung memisahkan aspek visual dan aspek performa.

Bertolak dari kesenjangan tersebut, penelitian ini difokuskan pada perancangan aplikasi simulasi yang tidak hanya mengeksekusi BFS, tetapi juga menampilkan mekanisme traversalnya secara visual dan interaktif melalui komponen grafis pada platform desktop. Pendekatan ini diharapkan dapat menjadi alat bantu yang lebih intuitif bagi mahasiswa maupun praktisi yang ingin memahami cara kerja BFS secara mendalam, sekaligus melengkapi sisi visual tersebut dengan pengukuran efisiensi komputasi agar karakteristik performanya dapat diverifikasi secara empiris.

Secara spesifik, penelitian ini bertujuan untuk: (1) mengimplementasikan algoritma BFS pada aplikasi desktop berbasis Delphi yang mampu menampilkan jalur terpendek pada graf tak berbobot; (2) memvisualisasikan dan mengevaluasi efisiensi algoritma pada dua skenario graf dengan skala berbeda, yaitu 100 simpul dan 500 simpul, guna menguji konsistensi performa pada tingkat kompleksitas yang bervariasi; (3) mengukur waktu eksekusi menggunakan *TStopwatch* sebagai indikator performa sistem; serta (4) mengevaluasi kesesuaian hasil implementasi dengan karakteristik kompleksitas waktu teoretis BFS sebesar *O(V+E)* (Cormen, Leiserson, Rivest, & Stein, 2022; Levitin, 2024)

METODE

Penelitian ini menggunakan pendekatan eksperimen kuantitatif berbasis komputasi laboratorium, dengan tahapan yang dimulai dari pemodelan graf tak berbobot secara visual, dilanjutkan dengan implementasi algoritma BFS untuk penelusuran dari simpul awal ke simpul tujuan, dan diakhiri dengan perekaman waktu komputasi guna mengevaluasi kinerja algoritma tersebut.



Gambar 1. Diagram Alur Tahapan Penelitian

Aplikasi dikembangkan menggunakan IDE Delphi modern berbasis *Visual Component Library* (*VCL*). Representasi graf disusun dalam bentuk *adjacency list*, sementara struktur antrean pada proses BFS diimplementasikan menggunakan *TQueue<Integer>* dari pustaka System.Generics.Collections agar manajemen memori antrean berlangsung secara dinamis (Cantú, 2021).

Visualisasi proses pencarian ditampilkan melalui komponen *TCanvas*, dengan simpul digambarkan menggunakan prosedur Ellipse dan sisi penghubung digambarkan melalui kombinasi MoveTo dan LineTo. Setiap kali suatu simpul dievaluasi oleh algoritma, warna simpul tersebut diperbarui secara dinamis pada kanvas sehingga pengguna dapat mengikuti urutan kunjungan simpul secara langsung.

Pengukuran waktu eksekusi dilakukan menggunakan kelas *TStopwatch* dari pustaka System.Diagnostics, dengan pencatatan waktu dimulai tepat sebelum simpul awal dimasukkan ke dalam antrean dan dihentikan segera setelah simpul tujuan ditemukan, sehingga durasi yang tercatat murni mencerminkan proses pencarian BFS tanpa tercampur waktu proses lain seperti rendering antarmuka.

Skenario pengujian dirancang dengan dua variasi ukuran graf (N), yaitu 100

simpul dan 500 simpul, dengan pembatasan tersebut disesuaikan terhadap kapasitas rendering komponen canvas pada sistem. Setiap skenario dijalankan sebanyak tiga kali pengulangan untuk mengurangi pengaruh proses latar belakang sistem operasi terhadap hasil pengukuran, dan nilai yang dilaporkan pada bagian hasil merupakan rata-rata dari ketiga pengulangan tersebut.

HASIL DAN PEMBAHASAN

Hasil

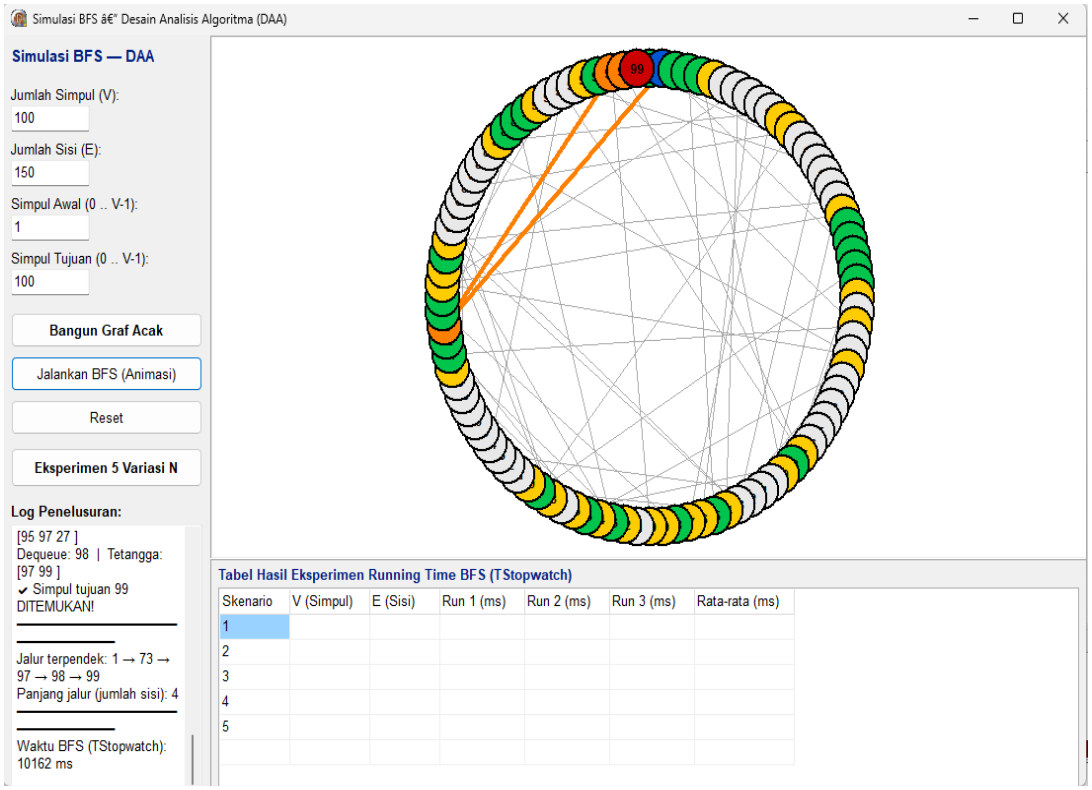
Implementasi inti algoritma BFS beserta instrumentasi *TStopwatch* pada aplikasi ditunjukkan pada cuplikan kode berikut.

```
QTemp := TQueue<Integer>.Create;
// <--- BUKTI PENGGUNAAN QUEUE

SW := TStopwatch.StartNew;
// <--- BUKTI PENGGUNAAN TSTOPWATCH

VisTemp[StartN] := True;
QTemp.Enqueue(StartN);
while (QTemp.Count > 0) and not Found
do
begin
  U := QTemp.Dequeue;
  for W in AdjTemp[U] do
  begin
    if not VisTemp[W] then
    begin
      VisTemp[W] := True;
      if W = GoalN then
      begin
        SW.Stop;
        Ms                                    :=
SW.Elapsed.TotalMilliseconds;
        Found := True;
        Break;
      end;
      QTemp.Enqueue(W);
    end;
  end;
end;
```

Antarmuka visual dibangun di atas *TCanvas*, dengan simpul digambar sebagai elips dan sisi sebagai garis penghubung. Simpul yang telah dikunjungi ditandai dengan perubahan warna secara dinamis selama proses traversal berjalan, sebagaimana ditunjukkan pada Gambar 1.



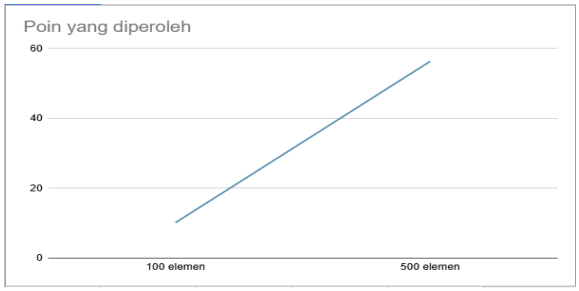
Gambar 2. Visual canvas VCL

Hasil pengujian waktu eksekusi pada dua skenario ukuran graf disajikan pada Tabel 1.

Tabel 1. Data waktu eksekusi

Skala Data (*N*)	Variasi Simpul (*V*) & Sisi (*E*)	Waktu Eksekusi Rata-rata(ms)
Skenario 1	100 elemen	10.153 ms
Skenario 2	500 elemen	56.309 ms

Perbandingan kedua nilai tersebut divisualisasikan dalam bentuk grafik pertumbuhan waktu pada Gambar 2.



Gambar 3. Visual grafik pertumbuhan waktu

Pembahasan

Data pada Tabel 1 memperlihatkan kenaikan waktu eksekusi dari 10.153 ms menjadi 56.309 ms ketika jumlah simpul dinaikkan dari 100 menjadi 500. Jika dibandingkan proporsional, kenaikan skala input sebesar lima kali lipat diikuti oleh kenaikan waktu eksekusi kurang lebih pada rentang yang sepadan, bukan meningkat secara berlipat ganda seperti pola kuadratik. Pola inilah yang menjadi indikator awal bahwa implementasi algoritma pada sistem bekerja sesuai harapan teoretisnya.

Karakteristik tersebut sejalan dengan cara kerja BFS yang bertumpu pada struktur antrean bertipe *First-In-First-Out* (*FIFO*): setiap simpul yang bertetangga dievaluasi kemudian dimasukkan ke antrean melalui operasi *enqueue*, lalu dikeluarkan secara berurutan melalui operasi *dequeue* berdasarkan kedalamannya (Cormen et al., 2022; Levitin, 2024). Karena representasi graf pada sistem ini disusun sebagai *adjacency list*, operasi *enqueue* maupun *dequeue* berlangsung dalam waktu konstan, sehingga total waktu komputasi f(N) pada dasarnya ditentukan oleh perulangan while yang memeriksa setiap simpul (V) beserta perulangan for di dalamnya yang menelusuri sisi (E) yang terhubung pada simpul tersebut. Hasil pengujian empiris pada penelitian ini menguatkan hubungan tersebut secara praktis.

Ditinjau dari batas pertumbuhan asimtotiknya, algoritma ini memiliki tiga kondisi karakteristik: kasus terbaik $\Omega(1)$ terjadi apabila simpul tujuan bertepatan langsung dengan simpul awal; kasus terburuk $O(V+E)$ terjadi ketika algoritma harus menelusuri seluruh lapisan graf sebelum menemukan simpul target; sementara batas ketat rata-rata $\Theta(V+E)$ merepresentasikan performa yang linier terhadap total elemen pada graf terhubung. Pola kenaikan waktu eksekusi yang teramati pada eksperimen dari 10.153 ms ke 56.309 ms tanpa lonjakan eksponensial maupun kuadratik secara konsisten mendukung karakteristik batas ketat tersebut, sekaligus menegaskan bahwa struktur antrean yang diimplementasikan pada sistem telah bekerja selaras dengan model kompleksitas teoretisnya.

Dibandingkan dengan penelitian Sularno et al. (2021) yang menekankan efektivitas BFS dalam menentukan rute evakuasi tanpa menyoroti sisi visual prosesnya, serta studi Aditiya & Herdiana (2021) yang berfokus pada perbandingan performa BFS dan A* semata, kontribusi utama penelitian ini terletak pada penyajian proses traversal secara visual sekaligus terukur. Aspek ini turut menjawab kebutuhan yang disoroti oleh Burch et al. (2023) dan Kolomeets et al. (2023), yaitu bahwa visualisasi dinamis dapat membantu pengguna memahami perilaku algoritma graf secara real-time, alih-alih hanya menerima keluaran akhir berupa jalur atau angka.

Dari sisi kekuatan sistem, pendekatan berbasis *TCanvas* terbukti mampu menggambarkan proses kunjungan simpul secara bertahap tanpa memerlukan pustaka visualisasi eksternal, sehingga aplikasi

tetap ringan dan mudah dijalankan pada lingkungan desktop standar. Penggunaan *TQueue<Integer>* juga memastikan urutan traversal *FIFO* terjaga secara konsisten sepanjang pengujian. Meski demikian, penelitian ini juga memiliki keterbatasan: proses rendering pada *TCanvas* dijalankan pada thread yang sama dengan komputasi inti BFS, sehingga pada skala data yang jauh lebih besar berpotensi menimbulkan jeda tampilan (freeze) yang dapat memengaruhi pengalaman pengguna meskipun tidak memengaruhi keakuratan pengukuran waktu komputasi itu sendiri, karena pencatatan *TStopwatch* hanya membungkus proses BFS murni tanpa menyertakan waktu rendering.

Dibandingkan dengan algoritma pencarian rute lain seperti Dijkstra atau A*, BFS tetap menjadi pilihan yang paling tepat untuk graf tak berbobot karena kedua algoritma tersebut pada dasarnya dirancang untuk menangani bobot sisi yang bervariasi, sehingga penerapannya pada graf tak berbobot justru menambah kompleksitas komputasi yang tidak diperlukan. BFS unggul dalam kesederhanaan implementasi dan kepastian menemukan jalur dengan jumlah langkah minimum tanpa memerlukan struktur data tambahan seperti priority queue yang digunakan Dijkstra maupun fungsi heuristik yang menjadi syarat A*.

Dari sisi keterbatasan penelitian, pengujian pada penelitian ini masih dibatasi pada dua skala graf (100 dan 500 simpul), sehingga generalisasi terhadap graf berskala sangat besar (di atas 1000 simpul) maupun graf dengan topologi berbeda (misalnya graf tidak terhubung penuh) belum sepenuhnya terverifikasi. Selain aspek performa, visualisasi yang dikembangkan juga berperan penting dalam konteks pembelajaran, karena representasi bertahap dari proses kunjungan simpul memungkinkan mahasiswa mengaitkan langsung antara konsep teoretis algoritma dengan perilaku aktualnya saat dieksekusi, sesuatu yang sulit dicapai hanya melalui pembelajaran berbasis pseudocode atau penjelasan tekstual semata.

SIMPULAN (PENUTUP)

Penelitian ini berhasil mencapai tujuan yang ditetapkan, yaitu mengembangkan aplikasi simulasi berbasis Delphi *VCL* yang mampu mengimplementasikan sekaligus memvisualisasikan proses pencarian rute terpendek menggunakan algoritma *Breadth-First Search*. Kontribusi utama penelitian ini terletak pada penggabungan tiga elemen sekaligus dalam satu sistem, yaitu implementasi algoritma yang tervalidasi, visualisasi traversal interaktif melalui *TCanvas*, dan pengukuran performa presisi menggunakan *TStopwatch*, yang bersama-sama membuktikan bahwa perilaku empiris algoritma pada sistem ini konsisten dengan kompleksitas waktu teoretisnya, $O(V+E)$. Secara praktis, penelitian ini memberikan manfaat sebagai alat bantu pembelajaran yang dapat digunakan mahasiswa maupun praktisi untuk memahami mekanisme kerja BFS secara lebih intuitif, tidak sekadar melalui hasil akhir berupa jalur, melainkan juga melalui proses pencapaiannya.

Untuk pengembangan selanjutnya, disarankan agar proses rendering visual pada canvas dipisahkan dari proses komputasi inti melalui pendekatan *multithreading*, sehingga pengujian performa dapat dilakukan pada skala data yang jauh lebih besar misalnya di atas 5000 simpul tanpa menimbulkan jeda pada antarmuka sistem.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada Program Studi Teknik Informatika, Fakultas Teknik, Universitas Muhammadiyah Parepare atas dukungan fasilitas dan lingkungan akademik yang diberikan selama pelaksanaan penelitian ini.

DAFTAR PUSTAKA

Aditiya, V., & Herdiana, B. (2021). Comparative analysis of Breadth First Search (BFS) algorithm and A*

algorithm. *Telekontran: Jurnal Ilmiah Telekomunikasi, Kendali dan Elektronika Terapan*, *9*(2), 139–153. https://doi.org/10.34010/telekontran.v9i2.5635

Alvito, P., & Ikhsan, M. (2025). Visualisasi dan perbandingan efisiensi algoritma A*, Greedy, dan Dijkstra dalam mencari rute terpendek di Kota Medan menggunakan OpenStreetMap. *JURIKOM (Jurnal Riset Komputer)*, *12*(4), 539–554. https://doi.org/10.30865/jurikom.v12i4.8869

Burch, M., van de Wetering, H., Wallner, G., Rooks, F., & Morra, O. (2023). Exploring the dynamics of graph algorithms. *Journal of Visualization*, *26*(2), 477–492. https://doi.org/10.1007/s12650-022-00885-0

Cantú, M. (2021). *Object Pascal handbook: Delphi 11 Alexandria edition*. Wintech Italia Srl.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). *Introduction to algorithms* (4th ed.). MIT Press.

Jaiman, F., Fallo, D. Y. A., & Banung, F. L. (2025). Analisis pemahaman mahasiswa informatika dalam menyelesaikan rute terpendek menggunakan algoritma Dijkstra dengan graf berbobot. *Jurnal Kridatama Sains dan Teknologi*, *7*(1), 508–518. https://doi.org/10.53863/kst.v7i01.1678

Kolomeets, M., Desnitsky, V., Kotenko, I., & Chechulin, A. (2023). Graph visualization: Alternative models inspired by bioinformatics. *Sensors*, *23*(7), Article 3747. https://doi.org/10.3390/s23073747

Krisnamurti, C. N., & Geong, E. A. P. (2021). Implementasi algoritma Floyd-Warshall untuk menentukan rute terpendek destinasi wisata populer di Labuan Bajo. *Unnes Journal of Mathematics*, *10*(1), 75–84. https://doi.org/10.15294/ujm.v10i1.45717

Levitin, A. (2024). *Introduction to the design and analysis of algorithms* (4th ed.). Pearson.

Saputrama, R., & Hartatiana. (2021). Aplikasi algoritma Dijkstra untuk menentukan rute terpendek dari Kampus A ke B UIN Raden Fatah. *E-Jurnal Matematika*, *10*(3), 173–178. https://doi.org/10.24843/MTK.2021.v10.i03.p339

Sularno, D. P. Mulya, R. Astri, & D. Mulya. (2021). Determination of the shortest route based on BFS algorithm for purpose to disaster evacuation shelter. *Scientific Journal of Informatics*, *8*(1), 33–42. https://doi.org/10.15294/sji.v8i1.27863

Taneo, R. E., Ndun, R., Fallo, D. Y. A., & Do'o, F. (2025). Optimasi jalur terpendek menggunakan algoritma Dijkstra dan Greedy pada sistem informasi geografis. *Jurnal Kridatama Sains dan Teknologi*, *7*(1), 572–582. https://doi.org/10.53863/kst.v7i01.1664