JIPETIK : Jurnal Ilmiah Penelitian Teknologi Informasi & Komputer
Vol. 5 No. 1, Juni 2024, Hal. 11-18
Available Online at journal.upgris.ac.id/index.php/jipetik
ISSN 2722-953X

IMPLEMENTASI ALGORITMA DIJKSTRA DALAM MENENTUKAN RUTE TERPENDEK MENUJU MUSEUM DI JAKARTA

Jonathan Adriano Pane[1)], Ida Fitriani[)] , Mei Lestari[)]
[1]Universitas Indraprasta PGRI
email: jonathanpanee@gmail.com
[2]Universitas Indraprasta PGRI
email: jonathanpanee@gmail.com
[3]Universitas Indraprasta PGRI
email: jonathanpanee@gmail.com

Abstrak

Sekarang ini, permasalahan menemukan jalur terpendek serta menghemat waktu menjadi sangat penting dalam dinamika masyarakat perkotaan. Banyaknya rute yang ditempuh juga menjadi masalah untuk mencapai tujuan. Sistem ini akan menentukan titik mana yang harus dilalui untuk mencapai tujuan dengan jarak terpendek dan waktu paling optimal menggunakan algoritma Dijkstra. Menemukan jalur terpendek merupakan masalah optimisasi. Nilai pada sisi graph dapat diwakili oleh jarak antar kota. Lintasan terpendek dapat dipahami sebagai proses meminimalkan bobot lintasan. Untuk mengatasi masalah tersebut, diperlukan suatu simulasi yang dapat membantu menentukan jalur terpendek. Menggunakan algoritma Dijkstra, dihitung jarak terpendek dari suatu titik ke museum yang dipilih sebagai tujuan. Aplikasi ini bertujuan untuk mengoptimalkan rute menuju museum berdasarkan jarak terpendek di kota Jakarta. Algoritma Dijkstra juga dapat dianggap sebagai algoritma Greedy yang dalam pembahasan ini dapat memungkinkan kita untuk menemukan jalur terpendek dengan lebih mudah dan efisien.

Kata kunci: Rute Terpendek, Algoritma Dijkstra, Aplikasi Pencarian, *Shortest Path*.

PENDAHULUAN

Museum adalah lembaga atau pusat penelitian ilmiah yang harus selalu mengkomunikasikan hasil penelitiannya kepada masyarakat. Tujuan dari museum ini adalah sebagai pengingat akan sejarah penting yang ada dan sebagai pengenalan budaya ke suatu negara. Museum juga mengingatkan masyarakat untuk melestarikan dan merawat warisan budaya masa lalu (Soejatmi, 2015). Pada tahun 1971, Departemen Umum museum membagi museum menjadi tiga kategori: museum umum, museum khusus, dan museum lokal. Rangkaian tersebut kemudian dimodifikasi pada tahun 1975 menjadi museum umum, museum khusus, dan museum pendidikan. Pada tahun 1980, pengelompokan museum disederhanakan menjadi museum umum dan museum khusus. Museum umum dan museum khusus menurut tingkatan dan letaknya diubah menjadi museum tingkat nasional, museum tingkat daerah (provinsi) dan daerah (kota). (Asmara, 2019) Masyarakat yang ingin

mengunjungi museum di Jakarta harus mempertimbangkan museum mana yang terdekat atau rute mana yang terpendek untuk ditempuh untuk sampai ke museum tujuan kita. Pada proses pencarian rute yang dilakukan pada sebuah peta, terdapat sebuah algoritma pemrograman di dalamnya yang berfungsi untuk menjalankan proses pencariannya.

Algoritma merupakan urutan logis langkah-langkah penyelesaian masalah yang disusun secara sistematis. Algoritma sangat identik pada persoalan rute terpendek, karena di dalam penentuan rute terpendek terdapat tahap-tahap yang disusun secara sistematis. Dalam proses penentuan rute terpendek dilakukan dari seluruh jalan yang mempunyai hubungan antara satu jalan dengan jalan yang lainnya dan membentuk graph. Penelitian terkait penentuan rute terpendek sudah banyak dilakukan dengan berbagai Algoritma diantaranya adalah algoritma A*, Floyd-Warshall, Bellman-Ford, Greedy. Untuk penentuan rute terpendek pada permasalahan ini menggunakan algoritma Dijkstra.

Algoritma Dijkstra adalah salah satu algoritma yang digunakan untuk menyelesaikan masalah jarak terpendek *(shortest path problem)* pada sebuah graf yang terarah *(directed graph*). Algoritma Dijkstra banyak diteliti untuk diaplikasikan pada proses pencarian rute terpendek. Algoritma ini ditemukan oleh Edsger Dijkstra, seorang ilmuwan komputer asal Belanda.

Pada proses implementasinya penelitian ini diterapkan pada museum-museum yang ada di Jakarta. Sekarang ini banyak masyarakat yang bergantung dengan *Google Maps* dalam menemukan letak suatu objek serta mencari rute menuju suatu tempat. Namun tidak seluruhnya proses ini sesuai dengan yang diharapkan dalam mencari rute terpendek melalui *maps*, keadaan bisa saja berubah drastis saat dicek waktu tempuh nya sebentar, rute tidak berwarna orange atau merah yang menandakan kepadatan atau kemacetan, tetapi dilokasi ternyata situasinya berbeda.

Penerapan jalur terpendek sering kali ditemukan dalam aplikasi kehidupan nyata, misalnya pada perangkat lunak pemetaan atau sistem navigasi. Ini melibatkan masalah menemukan jalur dari satu titik ke titik lain, dan jalur dengan biaya paling sedikit akan menjadi

jalur terpendek. Kasus manajemen transportasi dan desain teknik juga terkait erat dengan masalah jalur terpendek dalam banyak hal, seperti mengelola rute pemrosesan yang berbeda, memasang saluran listrik atau pipa, mengumpulkan rute dalam peta listrik, dll.(Sugianti et al., 2020)

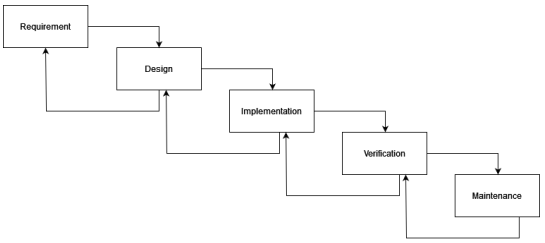
Dengan adanya jalur terpendek yang bisa dilalui maka perjalanan menuju museum di Jakarta akan optimal dengan tidak akan memakan banyak waktu. Oleh karena kondisi diatas Algoritma Dijkstra ini cocok untuk menentukan rute terpendek sehingga peneliti menggunakan “Implementasi Algoritma Dijkstra Dalam Menentukan Rute Terpendek Menuju Museum di Jakarta”.

METODE

Penelitian ini akan dilakukan di daerah Jakarta. Melalui penelitian ini, penulis rute terpendek kepada para pengunjung yang ingin melakukan perjalanan menuju museum di Jakarta.

1. Diagram Alir

   Berikut ini merupakan sebuah diagram alir yang menggambarkan urutan langkah-langkah dalam penelitian studi mengenai Implementasi Algoritma Dijkstra Dalam Menentukan Rute Terpendek Menuju Museum di Jakarta dapat dilihat pada Gambar 1(Fitria & Triansyah, 2013)



Gambar 1 Diagram Alir Penelitian

1. Requirement
   Pada tahap ini akan direncanakan apa saja kebutuhan perangkat yang akan digunakan oleh pengguna. Hal ini sangat penting karena *software* harus dapat berinteraksi dengan elemen lain seperti *hardware*, *database*, dll.
2. Design
   Pada tahap ini direncanakan bagaimana sistem bekerja pada proses sebelumnya menjadi representasi dalam bentuk "*blue print*" *software*. Sebelum coding dimulai, desain harus dapat memenuhi persyaratan yang telah direncanakan pada langkah sebelumnya.
3. Implementation
   Pada tahap ini, penulis mulai direncanakan sebelumnya. Agar suatu desain dapat dipahami oleh mesin, dalam hal ini komputer, maka desain tersebut harus diubah melalui proses coding menjadi bentuk yang dapat dipahami oleh mesin, yaitu bahasa pemrograman. Fase ini merupakan implementasi dari fase desain yang akan dilakukan oleh programmer.
4. Verification
   Pada tahap ini, sistem yang dibuat akan diuji dan dicari kesalahannya.

Semua fungsi perangkat lunak harus diuji untuk memastikan bahwa perangkat lunak bebas dari *error*. Hasilnya harus benar-benar sesuai dengan kebutuhan yang telah ditentukan sebelumnya.

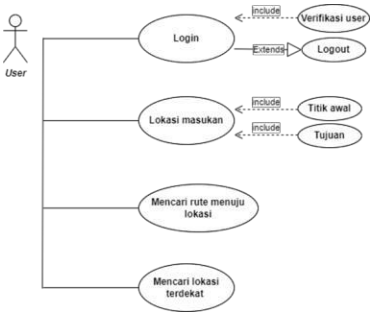
5. Maintenance
   Pada tahap ini sistem dirilis serta dipelihara dan jika memungkinkan bisa dilakukan pembaharuan/peningkatan yang sesuai secara berkala. (Setiawan et al., 2019)

Algoritma Dijkstra digunakan untuk menentukan rute atau jalur terpendek berdasarkan kriteria tertentu yang digunakan sebagai batasan. Algoritma ini menemukan jalur terpendek dari satu titik ke titik lainnya berdasarkan bobot terendah. Misalnya, sebuah titik menggambarkan sebuah bangunan dan sebuah garis menggambarkan sebuah jalan, sehingga Algoritma Dijkstra menghitung semua kemungkinan bobot terkecil untuk setiap titik.(Cantona et al., 2020) Berikut *flowchart* alur Algoritma Dijkstra.

2. Desain Sistem

   a. Use Case Diagram

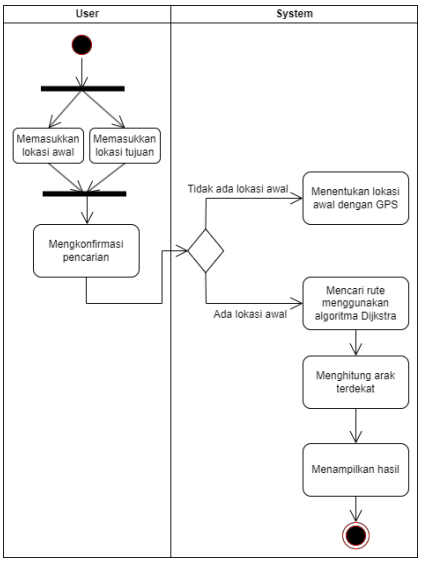
      Use case diagram digunakan untuk menggambarkan hubungan antara fungsi dalam sistem yang tersedia serta dapat digunakan oleh aktor atau user. (Abdillah, 2021)



Gambar 2. *Use Case Diagram*

b. Activity Diagram

Activity Diagram adalah diagram yang digunakan dalam *UML (Unified Modeling Language)* untuk menggambarkan alur kerja atau urutan aktivitas dalam suatu proses atau sistem. Berikut merupakan *Activity Diagram* Pencarian Rute

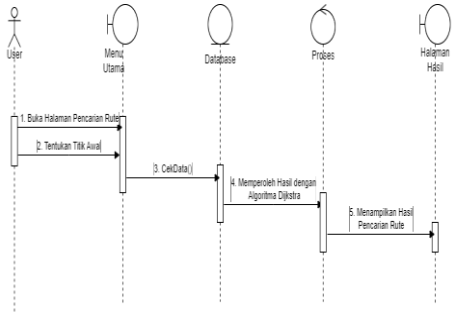


Gambar 3. *Activity Diagram*

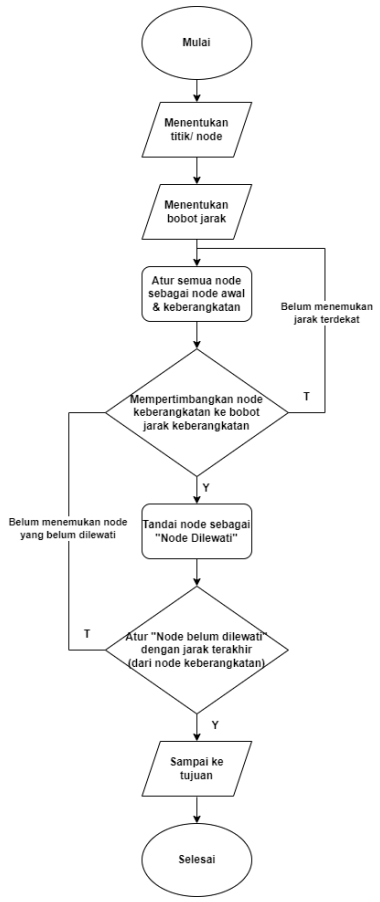
c. Sequence Diagram

Sequence Diagram adalah jenis diagram interaksi dalam Unified Modeling Language (UML) yang digunakan untuk menggambarkan

bagaimana objek berinteraksi dalam skenario atau proses. Berikut contoh *Sequence Diagram* dari pencarian rute terpendek.



Gambar 4. *Sequence Diagram*



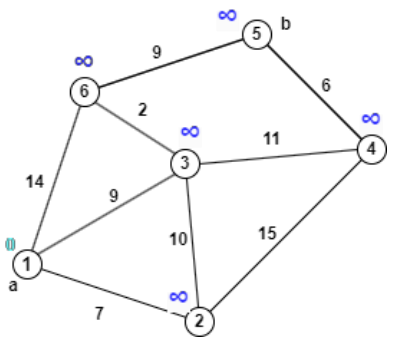
Gambar 5. *Flowchart* Algoritma Dijkstra

Berikut ini merupakan cara kerja dari Algoritma Dijkstra.

1. Tetapkan titik awal di mana itu akan menjadi simpul pertama atau node awal, lalu tentukan bobot jarak dari simpul asal ke simpul terdekat secara bergantian, Algoritma Dijkstra akan melakukan developing untuk menemukan bobot terkecil dari satu node ke node lainnya.
2. Tentukan bobot (jarak) setiap node ke node lainnya, kemudian beri nilai 0 untuk node pertama dan nilai tak terhingga untuk node lainnya
3. Atur semua node yang belum dilalui dan tetapkan node awal sebagai “node keberangkatan”.
4. Pada node keberangkatan, hitung node lain yang terdekat dengan titik keberangkatan yang belum dilalui dan hitung jarak dari awal keberangkatan. Apabila jarak kurang dari jarak sebelumnya (dihitung sebelumnya) hapus data jarak yang lama sebelumnya dan simpan dengan data jarak yang lebih pendek.
5. Setelah melakukan perhitungan dan mempertimbangkan setiap jarak ke node lain berdekatan, tandai simpul yang dilewati sebagai "Node yang dilewati". Node yang dilewati tidak akan diperiksa kembali, jarak akan disimpan sebagai jarak dengan bobot terkecil.
6. Tetapkan "Node belum dilewati", dengan bobot jarak terkecil (dihitung dari node keberangkatan) sebagai "Node Keberangkatan" untuk keberangkatan berikutnya dan ulangi langkah-langkahnya (Harahap & Khairina, 2017)

HASIL DAN PEMBAHASAN

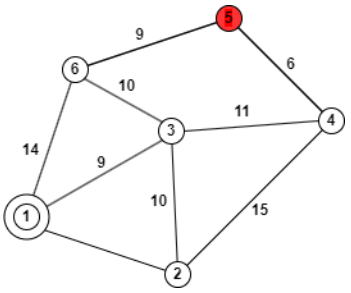
Algoritma Dijkstra bertujuan mencari jalur terpendek dengan bobot terkecil dari satu titik ke titik lainnya. Sebagai contoh, jika sebuah titik menggambarkan sebuah bangunan dan sebuah garis menggambarkan sebuah jalan, algoritma Dijkstra menghitung semua bobot minimum yang mungkin untuk setiap titik. Pada Gambar dibawah ini menjelaskan hubungan antara titik dalam algoritma Dijkstra.



Gambar 6. Hubungan antara titik dalam algoritma Dijkstra

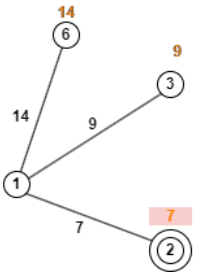
Pertama, menentukan titik mana yang akan menjadi node awal, kemudian memberi bobot jarak dari node pertama ke node terdekat secara bergantian, Dijkstra akan mengembangkan pencarian dari satu titik ke titik lainnya dan langkah demi langkah ke titik berikutnya. Gambar 4 di bawah ini menjelaskan secara detail cara mencari jalur terpendek mulai dari node awal hingga node tujuan dengan nilai jarak terkecil.

1. Node awal diberi nilai 1, dan node tujuan 5. Setiap edge yang terhubung antar node diberi nilai.



Gambar 7. Langkah 1

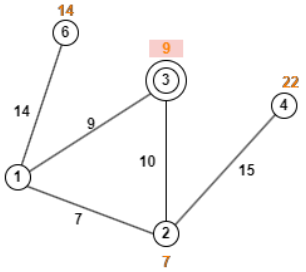
2. Algoritma Dijkstra melakukan perhitungan pada node tetangga yang terhubung langsung dengan node awal (node 1) dan hasil yang didapatkan adalah node 2 karena bobot nilai pada node 2 paling kecil dibandingkan dengan nilai pada node lainnya yaitu nilai 7 (0+7).



Gambar 8. Langkah 2

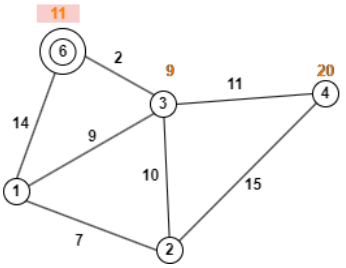
3. Node 2 ditetapkan sebagai node keberangkatan dan ditandai sebagai

node yang telah dilalui. Algoritma Dijkstra menghitung ulang node tetangga yang terhubung langsung ke node yang dilalui. Dan perhitungan Algoritma Dijkstra menunjukkan bahwa node 3 merupakan node keberangkatan berikutnya karena memiliki bobot yang paling kecil dibandingkan hasil perhitungan terakhir yaitu bernilai 9 (0+9).



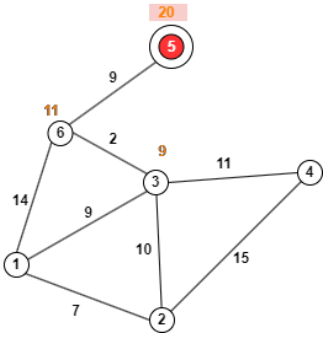
Gambar 9. Langkah 3

4. Perhitungan dilanjutkan dengan node 3 yang ditandai sebagai node yang telah dilalui. Dari semua node tetangga yang tidak dilalui yang terhubung langsung ke node yang dilalui, node yang ditandai berikutnya akan menjadi node yang telah dilalui adalah node 6 karena nilai bobotnya paling kecil yaitu 11 (9+2).



Gambar10. Langkah 4

5. Node 6 menjadi yang telah dilalui, algoritma Dijkstra menghitung lagi dan menemukan bahwa node 5 (node tujuan) dicapai melalui node 6 dengan nilai bobot 20 (11+9).



Gambar 11. Langkah 5

Dari contoh kasus diatas didapati bahwa jalur terpendek adalah 1-3-6-5 dan nilai bobot yang didapat adalah 20 (11+9). Ketika node tujuan tercapai, perhitungan algoritma Dijkstra dinyatakan selesai (Nuuryagandhi, 2016)

SIMPULAN

Berdasarkan pembahasan yang telah dilakukan pada bab sebelumnya, dapat disimpulkan bahwa algoritma Dijkstra merupakan solusi untuk mencari jalur

terpendek. Dengan menggunakan metode algoritma Dijkstra yang diterapkan pada aplikasi, pengguna dapat secara efektif mempersingkat waktu pencarian rute menuju museum di Jakarta.

Untuk menentukan jalur terpendek, algoritma Dijkstra bekerja dari node awal ke node tujuan. Dimana setiap node memiliki nilai jarak yang telah ditentukan. Dalam proses pencarian rute terpendek menggunakan metode Dijkstra, dimungkinkan untuk menentukan simpul awal dan simpul tujuan kemudian membandingkan nilai simpul-simpul tersebut.

Penentuan jalur terpendek dengan menggunakan metode algoritma Dijkstra ini diharapkan dapat membantu para pengguna dalam meningkatkan efektifitas waktu dalam melakukan perjalanan menuju museum sejarah yang ada di Jakarta. selain itu, sistem ini juga dapat memberikan informasi tentang museum sejarah yang ada di Jakarta.

DAFTAR PUSTAKA

Abdillah, R. (2021). Pemodelan Uml Untuk Sistem Informasi Persewaan Alat Pesta. *Jurnal Fasilkom*, *11*(2), 79–86. https://doi.org/10.37859/jf.v11i2.2673

Asmara, D. (2019). Peran Museum dalam Pembelajaran Sejarah. *Kaganga:Jurnal Pendidikan Sejarah Dan Riset Sosial-Humaniora*, *2*(1), 10–20. https://doi.org/10.31539/kaganga.v2i1.707

Cantona, A., Fauziah, F., & Winarsih, W. (2020). Implementasi Algoritma Dijkstra Pada Pencarian Rute Terpendek ke Museum di Jakarta. *Jurnal Teknologi Dan Manajemen Informatika*, *6*(1), 27–34. https://doi.org/10.26905/jtmi.v6i1.3837

Fitria, & Triansyah, A. (2013). Implementasi Algoritma Dijkstra Dalam Aplikasi Untuk Menentukan Lintasan Terpendek Jalan Darat Antar Kota Di Sumatera Bagian Selatan. *Jurnal Sistem Informasi (JIS)*, *5*(2), 611–621. http://ejournal.unsri.ac.id/index.php/jsi/article/download/840/430

Harahap, M. K., & Khairina, N. (2017). Pencarian Jalur Terpendek dengan Algoritma Dijkstra. *SinkrOn*, *2*(2), 18. https://doi.org/10.33395/sinkron.v2i2.61

Nuuryagandhi, R. F. (2016). *Implementasi Algoritma Dijkstra Untuk Objek Wisata di Kabupaten Pati*. https://lib.unnes.ac.id/27930/

Setiawan, J., Prakoso, R. S., Suryaningrum, K. M., Universitas, M., Mulia, B., Universitas, D., Mulia, B., Raya, J. L., Rw, R. T., Pademangan, K., & Utara, K. J. (2019). *Perbelanjaan Di Jakarta Menggunakan Algoritma Dijkstra*. *21*(3), 156–165.

Soejatmi, S. (2015). *PENATAAN BENDA KOLEKSI MUSEUM TERHADAP KEPUASAN PENGUNJUNG DI MUSEUM WAYANG DAN MUSEUM SEJARAH JAKARTA KAWASAN KOTA TUA JAKARTA Boby*. *4*(1).

Sugianti, N., Mardhiyah, A., & Fadilah, N. R. (2020). Komparasi Kinerja Algoritma BFS, Dijkstra, Greedy BFS, dan A* dalam Melakukan Pathfinding. *JISKA (Jurnal Informatika Sunan Kalijaga)*, *5*(3), 194–204. https://doi.org/10.14421/jiska.2020.53-07