HASIL CEK_66 Analisis

by 66 Analisis Jurnal Dosen

Submission date: 28-May-2022 09:55AM (UTC+0700)
Submission ID: 1845720795
File name: 66_Analisis.pdf (1.6M)
Word count: 3329
Character count: 18308

Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK) DOI: 10.25126/jtiik.202182866
Vol. 8, No. 2, April 2021, hlm. 227-234 p-ISSN: 2355-7699
Akreditasi KEMENRISTEKDIKTI, No. 36/E/KPT/2019 e-ISSN: 2528-6579

ANALISIS PERBANDINGAN ALGORITMA DJIKSTRA, A-STAR, DAN FLOYD WARSHALL DALAM PENCARIAN RUTE TERDEKAT PADA OBJEK WISATA KABUPATEN DOMPU

Rusydi Umar[1], Anton Yudhana[2], Andi Prayudi*[3]

[13]Teknik Informatika, Universitas Ahmad Dahlan
[2]Teknik Elektro, Universitas Ahmad Dahlan
e-mail : [1]rusydi@mti.uad.ac.id, [2]eyudhana@ee.uad.ac.id , [3]andi1807048016@webmail.uad.ac.id,
*Penulis Korespondensi

(Naskah masuk: 11 Desember 2019, diterima untuk diterbitkan: 18 Maret 2021)

Abstrak

Diera industri 4.0, penggunaan peta tidak lagi berbentuk lembaran ataupun buku. Kini terdapat sebuah layananan peta digital yaitu *platform* Leafleat.js, yang memudahkan penggunanya untuk mendapatkan informasi rute dari objek ke objek lainnya dan mencari lokasi hampir diseluruh dunia. Pada penelitian ini menggunakan objek yang *real* yaitu menampilkan lokasi sebenarnya menggunakan *platform* Leaflet.js dan parameter yang berbeda, dari hal tersebut penelitian ini akan membandingkan kinerja dari Algoritma *Dijkstra*, A* dan *Floyd Warshall* untuk menentukan waktu proses pencarian rute terdekat dari objek wisata ke objek wisata lain menggunakan bahasa pemograman PHP. Hasil pengujian program didapatkan jarak dan rute yang sama serta rata-rata waktu proses program yang berbeda. Waktu proses algoritma *Dijkstra* sebesar 0,0060 detik, algoritma A* sebesar 0,0067 dan algoritma *Floyd Warshall* sebesar 0,0433 detik. Berdasarkan hasil tersebut bahwa algoritma *Dijkstra* lebih unggul dalam proses pencarian rute.

Kata kunci: *Algoritma, Djikstra, A-Star, Floyd Warshall*

COMPARATIVE ANALYSIS OF DJIKSTRA, A-STAR, AND FLOYD WARSHALL ALGORITHM IN SEARCHING THE NEAREST ROUTE SEARCH TO TOURISM OBJECT IN DOMPU REGENCY

Abstract

*In the industrial era 4.0, the use of maps is no longer made of book sheets. Now a digital map service is available, the Leafleat.js platform, which provides users to get route information from other attractions and find locations that have been saved by the world. In this study using real objects that display the actual location using the Leaflet.js platform and different parameters, from this study will compare the performance of the Dijkstra, A * and Floyd Warshall Algorithms for the process of finding other tourist information using the PHP programming language. The results of testing the program obtained the same distance and route with different program processing time. Dijkstra algorithm processing time is 0.0060 seconds, A * algorithm is 0.0067 and Floyd Warshall algorithm is 0.0433 seconds. Based on these results, Dijkstra is superior in the route search process.*

Keyword: *Algorithms, Djikstra, A-Star, Floyd Warshall*

1. PENDAHULUAN

Diera industri 4.0, penggunaan peta tidak lagi berbentuk lembaran ataupun buku. Kini terdapat sebuah layananan peta digital yaitu *platform* Leafleat.js, yang memudahkan penggunanya untuk mendapatkan informasi rute dari objek ke objek lainnya dan mencari lokasi hampir diseluruh dunia. Leaflet.js merupakan *library javascript open source* yang berguna untuk membangun aplikasi peta interaktif berbasis *web* (Manurung, 2018). Terdapat banyak Algortima yang menyelesaikan permasalahan pencarian rute terdekat diantaranya adalah algoritma A* (A *Star*), *Floyd Warshall*, *Dijkstra*, dan *Bellman-ford.* Banyaknya penelitian yang telah melakukan hal yang serupa, diantaranya penggunaan algoritma yang sama untuk mencari rute terdekat dari objek wisata ke objek wisata lainnya dan penerapan ke sebuah *game*.

Terdapat beberapa penelitian sebelumnya yang menjadi dasar dari penelitian ini salah satunya adalah yang dilakukan oleh (Suyitno, 2016) menggunakan algoritma A* untuk mencari rute terdekat dari rumah menuju objek wisata Candi Jiwa Batujaya yang

ditampilkan melalui *platform* Google Maps. berikutnya dilakukan oleh (Pugas, Somantri & Satoto, 2014) membandingkan algoritma A* dengan *Dijkstra* untuk mencari rute terdekat antara objek wisata ke objek wisata lain di Kota Sawahlunto yang ditampilkan melalui *platform Mapserver* dengan parameter jarak tempuh antara objek wisata. Penelitian selanjutnya dilakukan oleh (M. Azan Cahyadi, Bambang, Widhiarso & Yohannes, 2016) membandingkan algoritma *Dijkstra*, A*, dan Floyd Warshall untuk menentukan jalur terpendek pada Bacteria Defense Game dengan parameter *Weight, Path Count, dan Checked Node.* dan terakhir dilakukan oleh (Sazaki, Satria, Primanita & Syahroyni., 2018) membandingkan algoritma A* dan Dinamic Pathfinding dalam pencarian rute terpendek yang ada pada *Simulation Game* khususnya *Car Racing Game*.

Pada penelitian ini terdapat perbedaan dengan penelitian sebelumnya yakni, menggunakan objek yang *real* yaitu menampilkan lokasi sebenarnya menggunakan *platform* Leaflet.js dan parameter yang berbeda. Dengan dasar tersebut kami akan membandingkan kinerja dari Algoritma *Dijkstra*, A* dan *Floyd Warshall* untuk menentukan waktu proses pencarian rute terdekat dari objek wisata ke objek wisata lain menggunakan Bahasa pemograman PHP. PHP yaitu skrip bahasa pemograman yang membuat dokumen HTML secara langsung eksekusi oleh *web server*.(Sidik, 2017).

Algoritma *Dijkstra* adalah algoritma *greedy* (serakah) yang digunakan untuk menyelesaikan masalah dalam jalur terpendek menggunakan grafik berarah dengan bobot tepi dihargai non-negatif (Agusta & Ferdinand, 2017). Algoritma A* merupakan bentuk pencarian terbaik yang paling populer, Algoritma ini digunakan untuk menemukan jalur dari node awal yang diberikan ke *node* tujuan. Algoritma ini memperkiraan nilai heuristik h(x) yang memberikan perkiraan rute terbaik yang akan dilewati oleh simpul. (Shabina Banu Mansuri, 2018). Algoritma *Floyd-Warshall* adalah metode yang memberikan solusi dengan melihat jawaban yang akan diperoleh sebagai keputusan yang saling terkait. Ini berarti bahwa solusi terbentuk dari solusi yang berasal dari tahap sebelumnya, dan ada kemungkinan lebih dari satu solusi. Algoritma ini ditemukan oleh Warshall. (Ningrum & Andrasto ,2016).

2. METODE PENELITIAN

2.1 Pengumpulan Data

Pada penelitian ini data objek wisata didapatkan secara langsung pada Dinas Kebudayaan dan Pariwisata (DISBUDPAR) Kabupaten Dompu dan menggunakan *platform Google Maps* untuk mengambil titik koodinat lokasi objek wisata. *Google Maps* adalah sebuah aplikasi peta yang diciptakan oleh perusahaan *Google* yang dapat berjalan didalam browser (Ariyanti, Khairil & Kanedi, 2015). *Google Maps* memiliki beberapa fasilitas yang dapat digunakan, yakni menemukan sebuah lokasi dengan memasukkan kata kunci, seperti kota, jalan, nama tempat. (Umar and Hari Prabowo, 2016). Titik koordinat digunakan sebagai dasar penentuan rute terdekat. beberapa objek wisata alam beserta koorditannya dapat dilihat pada table 1.

Tabel 1. Koordinat Objek Wisata

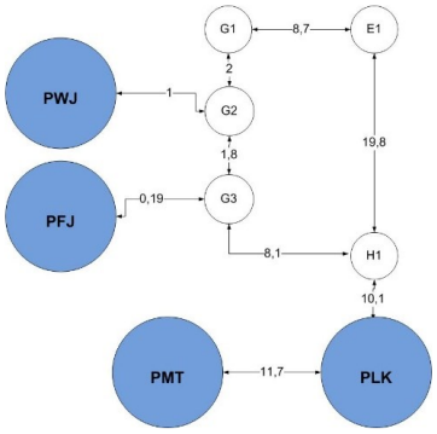
No	Kode	Nama Objek	Kooditat	
			Latitude	Longitude
1	PLK	Pantai Lakey	-8.7958347	118.3798741
2	PFJ	Pantai Felo Janga	-8.680857	118.430676
3	PWJ	Pantai Wadu jao	-8.6661146	118.4247945
4	PMT	Puncak Matiti	-8.8636108	118.431523

Koordinat pada table 1 didapatkan dari *platform Google Maps*, dari koodinat tersebut peneliti memasukan ke dalam aplikasi. *Marker* dipasang setiap titik koodinat objek wisata alam, dapat dilihat pada gambar 1.

Gambar 1. *Marker* objek wisata

Gambar 2 merupakan tampilan maps menggunakan *platform* Leafleat.js dan terlihat 4 (empat) buah marker yakni titik koordinat objek wisata.

Gambar 2 merupakan rute perjalanan detinasi wisata. Jarak antara titik diambil menggunakan fitur rute pada *platform Google Maps*. Dari gambar 2 dapat dilihat bahwa jarak untuk G1 ke E1 adalah 8,7 Km. begitu juga dengan jarak yang lain.



Gambar 2. Rute perjalanan objek wisata

Ketenrangan Simbol :

: Objek Wisata

: Persimpangan Jalan

: Jalur 2 arah

2.2 Algoritma

2.2.1 Algoritma Dijkstra

Algoritma *Dijkstra* adalah algoritma greedy (serakah) yang digunakan untuk menyelesaikan masalah dalam jalur terpendek menggunakan grafik berarah dengan bobot tepi dihargai non-negatif (Agusta & Ferdinand, 2017). Algoritma *Dijkstra* merupakan algoritma untuk mencari panjang rute terpendek dari sebuah titik s ke sebuah titik t di graf bobot G, dengan bobot setiap sisi G adalah bilangan positif (Ardyan, Mulyono & Suyitno, 2017). Algoritma *Dijkstra* merupakan algoritma pencari rute terbaik. Prinsip greedy (serakah) digunakan dalam algoritma ini. Bahwa setiap jalur akan dipilih sisi yang berbobot kecil (Wahyuningsih & Syahreza, 2018). Algoritma *Dijkstra* bekerja dengan membuat jalur ke satu simpul optimal pada setiap langkah. Jadi pada langkah ke n, setidaknya ada n node yang sudah diketahui jalur terpendek (Sunardi, Yudhana and Kadim, 2019). Berikut *pseudocode* algoritma Djikstra.

2.2.2 Algoritma A*

Algoritma A*r merupakan bentuk pencarian terbaik yang paling populer, Algoritma ini digunakan untuk menemukan jalur dari node awal yang diberikan ke node tujuan. Algoritma ini memperkiraan nilai heuristik h(x) yang memberikan perkiraan rute terbaik yang akan dilewati oleh simpul. (Shabina Banu Mansuri, 2018). Algoritma A * adalah algoritma pencarian heuristik, yang menggunakan fungsi heuristik untuk menilai biaya transit *node* ke *node* tujuan.(Wang, Wang, Qin, Wu, Duan, Cao & Ou, 2015). Algoritma dapat dinyatakan sebagai berikut

$$f(n)=g(n)+h(n) \quad (1)$$

Dimana:

f(n) = Hasil perhitungan jarak
g(n) = Jarak sesungguhnya dari titik awal ke titik n
h(n) = Jarak *heuristic* dari titik awal ke titik n

```
1.  procedure dijkstra
2.     bobot ← 0
3.     edge ← []
4.     Foreach edge AS val
5.        Bobot[edge] ← ∞
6.     endforeach
7.     bobotMinimal ← min [bobot]
8.  while bobot > 0 do
9.    If bobotMinimal = titik tujuan then
10.          break
11.       Endif
12.       Foreach edge[bobotMinimal] as key in val
13.      If bobot[key]>0 AND
         bobot[bobotMinimal]+val< bobot[key]
14.           Then
15.      Bobot[key] ← bobot[bobotMinimal] + val
16.      S[key] ← bobot[key] AND bobotMinimal
17.          Endforeach
18.    endwhile
19.    remove bobotMinimal
20.    end dijkstra
```

Gambar 3. *Pseudocode* Algoritma Dijkstra

Untuk menghitung jarak *heuristic* menggunakan metode *Euclidean distance* pada persamaan (2). *Euclidean distance* adalah perhitungan jarak dari 2 buah titik dalam *Euclidean space*. *Euclidean* space diperkenalkan oleh Euclid, seorang matematikawan dari Yunani sekitar tahun 300 B.C.E. untuk mempelajari hubungan antara sudut dan jarak. *Euclidean* ini berkaitan dengan *Teorema Phytagoras* dan biasanya diterapkan pada 1, 2 dan 3 dimensi (Setiawan, Andriyanto, Putro, Pradana, Permana, 2018). Berikut formula *Euclidean*.

$$d = \sqrt{(x_1 - x_2) + (y_1 - y_2)^2} \quad (2)$$

Sehingga dari Formula diatas kita dapat implementasi menjadi :

$$d = \sqrt{(lat_1 - lat_2) + (long_1 - long_2)^2} \quad (3)$$

Dimana :

d = Jarak dengan satuan kilometer (Km)

Lat_1 = *Latitude* titik awal
Lat_2 = *Latitude* titik tujuan
$Long_1$ = *Longitude* titik awal
$Long_2$ = *Longitude* titik tujuan

Dari hasil formula tersebut dikalikan dengan 111.319 km (1 derajat bumi = 111.319 km). Berikut *pseudocode* algoritma A*.

```
1.  procedure A-Star
2.    node ← data jalur
3.    Heuristic ← []
4.    For (i←0 to for i<jumlah node)
5.      Heuristic(node[i]) ← data node
6.    Endfor
7.    bobot ← []
8.    Foreach node AS v
9.      Bobot[v] ← ∞
10.     Endforeach
11.
12.   Bobot [titikAwal] ← 0
13.   openList ← []
14.
15.     while bobot > 0 do
16.   bobotMin ← min [bobot]
17.     If bobotMinimal = titik tujuan then
18.       break
19.     else
20.     Foreach bobotMin as r in v
21.       If bobot[bobotMin] + heuristic[] + v <
    bobot[r]
22.         Then
23.         Bobot[r] ← bobot[bobotMin] + v
24.         Openlist[r] ← bobotMIn + bobot[r]
25.       endif
26.     Endforeach
27.     endwhile
28.   remove bobotMinimal
29.   end A-Star
```

Gambar 4. *Pseudocode* Algoritma A*

2.2.3 Algoritma Floyd Warshall

Algoritma *Floyd Warshall* mempunyai jalur yang berbobot dan keluaran dari algoritma ini adalah dengan menghitung bobot minimum pada jalur yang saling berkaitan pada pasangan *node*, dan mengeksekusi pada semua pasangan *node* (Nawagusti, 2018). *Floyd Warshall* adalah teknik yang mengambil keuntungan dari *subproblem* yang tumpang tindih, substruktur optimal, dan memperdagangkan ruang untuk waktu guna meningkatkan kompleksitas runtime algoritma. Eksekusi algoritma tunggal akan menemukan jalur terpendek antara semua pasangan simpul (Cruz, Magwili, Mundo, Gregorio, Lamoca & Villasenor, 2017). Berikut *pseudocode* algoritma *Floyd Warshall*.

```
1.  procedure Floyd Warshall
2.  graph ← data jalur
3.  dist ← []
4.  pred ← []
5.  for (i←0 to i<jumalah data)
6.      for (j←0 to j<jumalah data)
7.              if i=j then
8.                      dist[i,j]←0
9.              elseif graph[i,j] then
10.                     dist[i,j]←graph[i,j]
11.             else
12.                     dist[i,j]←∞
13.             endif
14.     endfor
15.     pred[i,j]←[i]
16. endfor
17. for (k←0 to k<jumalah data)
18.     for (i←0 to i<jumalah data)
19.       for (j←0 to j<jumalah data)
20.       If dist[i,j] > dist[i,k] + dist[k,j] then
21.       dist[i,j]=dist[i,k] + dist[k,j]
22.       pred[i,j] = pred[k,j]
23.      endfor
24.     endfor
25. endfor
26. end Floyd Warshall
```

Gambar 5. *Pseudocode* Algoritma Floyd Warshall

3. IMPLEMENTASI DAN HASIL

3.1 Pengujian Waktu Proses Algoritma

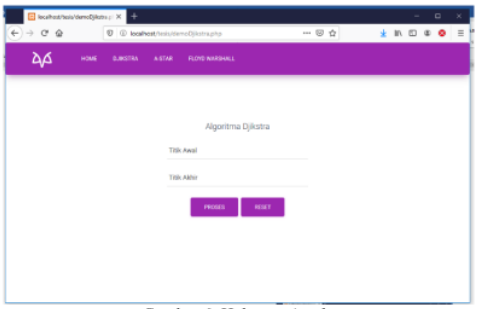
Program dikembangkan menggunakan Bahasa pemograman PHP dan *platform* Leaflet.js sebagai tampilan *maps*. Kemudian program tersebut diujicoba dengan beberapa data tes. Adapun alat-alat pendukung dalam pengujian program sebagai berikut:

1. *Hardware*
   a. *Laptop Asus X441U*
   b. *Prosessor Intel Core i3 2.0GHz.*
   c. *RAM 4Gb*
2. *Software*
   a. *Sistem Operasi Windows 10 64Bit*
   b. *Xampp 3.3.2.*
   c. *Mozilla Firefox 71.0 64bit*

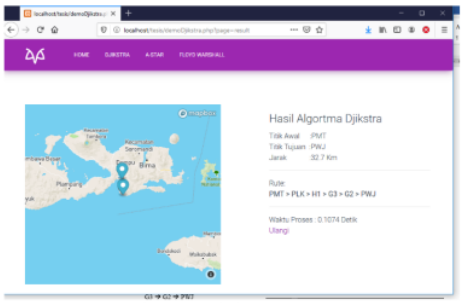
Terdapat 4 objek wisata dan 5 persimpangan. Setiap data akan dimasukan kedalam program kemudian diproses untuk mencari jalur terdekat.

Seperti terlihat pada gambar 6 terdapat 2 *form input* yang berfungsi untuk menentukan titik awal dan akhir dan 1 tombol untuk menjalankan program. Kemudian program dijalankan dengan cara menekan tombol cari.

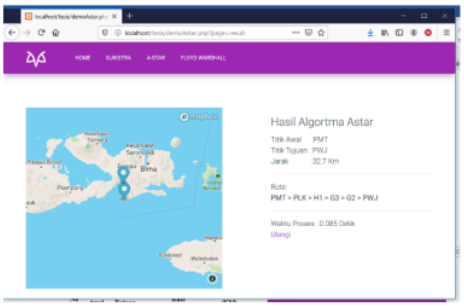
Gambar 5 merupakan hasil komputasi algoritma *Dijkstra* yang telah diproses oleh program dengan menampilkan rute terdekat melalui *maps*, jarak tempuh, dan waktu proses.



Gambar 6. Halaman Awal

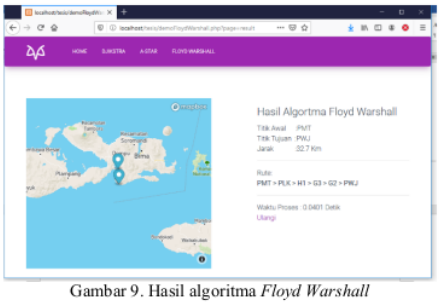


Gambar 7. Hasil algoritma *Dijkstra*



Gambar 8. Hasil algoritma A*

Gambar 8 merupakan hasil komputasi algoritma A* yang telah diproses oleh program dengan menampilkan rute terdekat melalui *maps*, jarak tempuh, dan waktu proses.



Gambar 9. Hasil algoritma *Floyd Warshall*

Gambar 9 merupakan hasil komputasi algoritma *Floyd Warshall* yang telah diproses oleh program dengan menampilkan rute terdekat melalui maps, jarak tempuh, dan waktu proses. Setelah program dijalankan mendapatkan hasil yang sama pada rute dan jarak akan tetapi rata-rata waktu proses program berbeda. Hasil jarak dan rute dari program dapat dilihat pada tabel 2,3 dan 4.

Tabel 2 Hasil uji program algoritma *Dijkstra*

No	Titik Awal	Titik Tujuan	Rute	Jarak (KM)
1	PWJ	PFJ	PWJ → G2 → G3 → PFJ	2,99
		PLK	PWJ → G2 → G3 → H1 → PLK	21
		PMT	PWJ → G2 → G3 → H1 → PLK → PMT	32,7
2	PFJ	PWJ	PFJ → G3 → G2 → PWJ	2,99
		PLK	PFJ → G3 → H1 → PLK	18,39
		PMT	PFJ → G3 → H1 → PLK → PMT	30,09
3	PLK	PWJ	PLK → H1 → G3 → G2 → PWJ	21
		PFJ	PLK → H1 → G3 → PFJ	18,39
		PMT	PLK → PMT	11,7
4	PMT	PWJ	PMT → PLK → H1 → G3 → G2 → PWJ	32,7
		PFJ	PLK → H1 → G3 → PFJ	30,09
		PLK	PMT → PLK	11,7

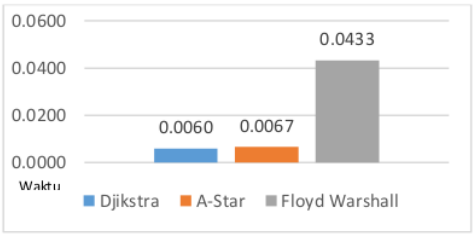
Tabel 3 Hasil uji program algoritma A*

No	Titik Awal	Titik Tujuan	Rute	Jarak (KM)
1	PWJ	PFJ	PWJ → G2 → G3 → PFJ	2,99
		PLK	PWJ → G2 → G3 → H1 → PLK	21
		PMT	PWJ → G2 → G3 → H1 → PLK → PMT	32,7
2	PFJ	PWJ	PFJ → G3 → G2 → PWJ	2,99
		PLK	PFJ → G3 → H1 → PLK	18,39
		PMT	PFJ → G3 → H1 → PLK → PMT	30,09
3	PLK	PWJ	PLK → H1 → G3 → G2 → PWJ	21
		PFJ	PLK → H1 → G3 → PFJ	18,39
		PMT	PLK → PMT	11,7
4	PMT	PWJ	PMT → PLK → H1 → G3 → G2 → PWJ	32,7
		PFJ	PLK → H1 → G3 → PFJ	30,09
		PLK	PMT → PLK	11,7

Tabel 4 Hasil uji program algoritma *Floyd Warshall*

No	Titik Awal	Titik Tujuan	Rute	Jarak (KM)
1	PWJ	PFJ	PWJ → G2 → G3 → PFJ	2,99
		PLK	PWJ → G2 → G3 → H1 → PLK	21
		PMT	PWJ → G2 → G3 → H1 → PLK → PMT	32,7

No	Titik Awal	Titik Tujuan	Rute	Jarak (KM)
2	PFJ	PWJ	PFJ → G3 → G2 → PWJ	2,99
		PLK	PFJ → G3 → H1 → PLK	18,39
		PMT	PFJ → G3 → H1 → PLK → PMT	30,09
3	PLK	PWJ	PLK → H1 → G3 → G2 → PWJ	21
		PFJ	PLK → H1 → G3 → PFJ	18,39
		PMT	PLK → PMT	11,7
4	PMT	PWJ	PMT → PLK → H1 → G3 → G2 → PWJ	32,7
		PFJ	PLK → H1 → G3 → PFJ	30,09
		PLK	PMT → PLK	11,7

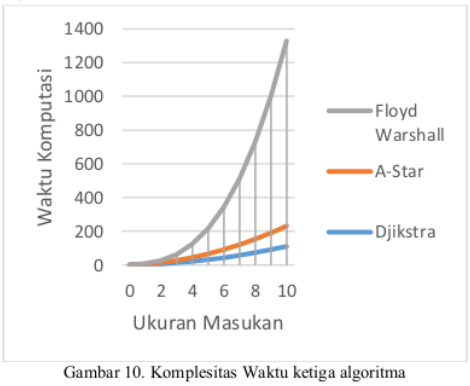


Gambar 9. Rata-rata waktu proses ketiga algoritma

Gambar 9 merupakan hasul rata-rata waktu proses dari ketiga algoritma yang didapat dari proses eksekusi program.

3.2 Komplesitas Waktu (*Big-O*)

Dari analisis *pseudocede* ketiga algoritma dapat diketahui bahwa kompleksitas waktu algoritma *Dijkstra* adalah $f(n)=O(n+n^2)$, algoritma A* adalah $f(n)=O(2n+n^2)$ dan algoritma *Floyd Warshall* adalah $f(n)=O(n^2+n^3)$. Ketiga kompleksitas waktu tersebut dapat dilihat dalam bentuk grafik seperti pada gambar 10



Gambar 10. Komplesitas Waktu ketiga algoritma

Dari gambar 10 dilihat bahwa penggunaan algoritma *Dijkstra* memiliki nilai komputasi waktu terrendah dari pada algoritma A* dan *Floyd Warshall*

4. KESIMPULAN

Dari pengujian program didapatkan jarak dan rute yang sama serta rata-rata waktu proses program yang berbeda. Waktu proses algoritma *Dijkstra* sebesar 0,0060 detik, algoritma A* sebesar 0,0067 dan algoritma *Floyd Warshall* sebesar 0,0433 detik. Berdasarkan hasil tersebut bahwa algoritma *Dijkstra* lebih unggul dalam proses pencarian rute.

5. DAFTAR PUSTAKA

AGUSTA, D. & FERDINAND, F. N. (2017) 'DJIKSTRA Algorithm Based Approach to Shortest Path Model in Public Bus Transportation', International Journal of Computer Science Engineering and Information Technology Research, 7(6), pp. 1–8. doi: 10.24247/ijcseitrdec20171.

ARDYAN, S., MULYONO & SUYITNO, A. (2017) 'Implementasi Algoritma Dijkstra Dalam Pencarian Rute Terpendek Tempat Wisata Di Kabupaten', UNNES Journal of Mathematics, 6(2), pp. 108–116.

ARIYANTI, R., KHAIRIL AND KANEDI, I. (2015) 'Pemanfaatan Google Maps Api Pada Sistem Informasi Geografis Direktori Perguruan Tinggi Di Kota Bengkulu', Media Infotama, 11(2), pp. 119–129.

CAHYADI, M. A., BAMBANG, M. A. P., WIDHIARSO, W., YOHANNES. (2016) 'Perbandingan Algoritma A*, Dijkstra dan Floyd Warshall Untuk Menentukan Jalur Terpendek Pada Permainan "Bacteria Defense"', Teknik Tnformatika.

CRUZ, J. C. D., MAGWILI, G. V., MUNDO, J. P. E., GREGORIO, G. P. B., LAMOCA, M. L. L., VILLASENOR, J. A (2017) 'Items-mapping and route optimization in a grocery store using Dijkstra's, Bellman-Ford and Floyd-Warshall Algorithms', IEEE Region 10 Annual International Conference, Proceedings/TENCON, pp. 243–246. doi: 10.1109/TENCON.2016.7847998.

MANURUNG, S. (2018) Mengenal Leaflet Js, Alternatif Membangun Peta Interaktif Berbasis Web Tanpa Google Maps Api. Available at: https://www.sgtgeomedia.com/detailpost/mengenal-leaflet-js-alternatif-membangun-peta-interaktif-berbasis-web-tanpa-google-maps-api.

NAWAGUSTI, V. A. (2018) 'Penerapan Algoritma Floyd Warshall Dalam Aplikasi Penentuan Rute Terpendek Mencari Lokasi BTS (Base Tower Station) Pada PT.GCI Palembang', Jurnal Nasional Teknologi dan Sistem Informasi, 4(2), pp. 81–88. doi: 10.25077/teknosi.v4i2.2018.81-88.

NINGRUM, F. W. & ANDRASTO, T. (2016) 'Penerapan Algoritma Floyd-Warshall dalam Menentukan Rute Terpendek pada Pemodelan Jaringan Pariwisata di Kota

Semarang', Jurnal Teknik Elektro, 8(1), pp. 21–24.

PUGAS, D. OK., SOMANTRI, M. & SATOTO, K. I. (2014) 'Pencarian Rute Terpendek Menggunakan Algoritma Dijkstra dan Astar (A*) pada SIG Berbasis Web untuk Pemetaan Pariwisata Kota Sawahlunto', Transmisi.

SAZAKI, Y., SATRIA, H., PRIMANITA, A., SYAHROYNI, M (2018) 'ANALISA PERBANDINGAN ALGORITMA A * DAN DYNAMIC PATHFINDING ALGORITHM DENGAN DYNAMIC PATHFINDING ALGORITHM UNTUK NPC', 5(1), pp. 95–103. doi: 10.25126/jtiik.

SETIAWAN, A., ANDRIYANTO, F., PUTRO, L. S., PRADANA, N. T. P., PERMANA, U (2018) 'Menghitung Rute Terpendek Menggunakan Algoritma A* Dengan Fungsi Euclidean Distance', SENTIKA 2018, 2018(Sentika), pp. 23–24.

SHABINA BANU MANSURI, S. K. (2018) 'Comparative Analysis of Path Finding Algorithms', IOSR Journal of Computer Engineering, 20(5), pp. 38–45.

SIDIK, B. (2017) Pemograman Web Dengan PHP 7. Bandung: Informatika.

SUNARDI, S., YUDHANA, A. & KADIM, A. A. (2019) 'Implementasi Algoritma Dijkstra dan Algoritma Semut Untuk Analisis Rute Transjogja Berbasis Android', It Journal Research and Development, 01, pp. 32–38. doi: 10.25299/itjrd.2019.vol4(1).2483.

SUYITNO, P. P. W. (2016) 'Pencarian Jalur Terpendek dari Rumah Menuju Candi Jiwa Batujaya Menggunakan Algoritma A-Star', II STMIK Nusa Mandiri. ISBN: 978-602-72850-1-9, pp. 169–174.

UMAR, R. & HARI PRABOWO, P. (2016) 'Pencarian Dan Pemesanan Travel Berbasis Mobile dengan Google Maps API', Annual Research Seminar 2016, 2(ISBN : 979-587-626-0), pp. 369–372.

WAHYUNINGSIH, D. & SYAHREZA, E. (2018) 'Shortest Path Search Futsal Field Location With Dijkstra Algorithm', IJCCS (Indonesian Journal of Computing and Cybernetics Systems), 12(2), p. 161. doi: 10.22146/ijccs.34513.

WANG, C., WANG, L., QIN, J., WU, Z., PUAN, L., CAO, Z. L. M, OU, X (2015) 'Path planning of automated guided vehicles based on improved A-Star algorithm', 2015 IEEE International Conference on Information and Automation, ICIA 2015 - In conjunction with 2015 IEEE International Conference on Automation and Logistics, (August), pp. 2071–2076. doi:10.1109/ICInfA.2015.7279630.

Halaman ini sengaja dikosongkan

HASIL CEK_66 Analisis

ORIGINALITY REPORT

6%	7%	4%	2%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	stmikpontianak.ac.id Internet Source	2%
2	blogs.itb.ac.id Internet Source	2%
3	Submitted to Institut Teknologi Kalimantan Student Paper	2%

Exclude quotes	On	Exclude matches	< 2%
Exclude bibliography	On		