p-ISSN : 2640-7363
e-ISSN : 2614-6606

Perbandingan Algoritma *Greedy* dan Algoritma *Dijkstra* dalam Pencarian Rute Terpendek dari Kabupaten Tuban ke Kota Surabaya

Jonathan Steven Iskandar[#1], Yosefina Finsensia Riti[#2]
[#]*Program Studi S1 Informatika, Fakultas Teknik, Universitas Katolik Darma Cendika Surabaya*
Jl. Dr. Ir. H. Soekarno No.201, Klampis Ngasem, Kec. Sukolilo, Kota SBY, Jawa Timur, Indonesia
[1]jonathan.iskandar@student.ukdc.ac.id
[2]yosefina.riti@ukdc.ac.id

Abstract — **Finding distance a common problem for transport users from the start. This is because transportation users need a short distance and a short time in order to reach the trip quickly. Transportation problems often arise from remote locations, especially if there are transport users trying to go to a big city. Many algorithms have been used to help find the shortest route from one city to another, including the Greedy algorithm and Dijkstra's algorithm. In this study, the two algorithms were implemented to find the shortest path with a coverage area from Tuban Regency (as vertex T) to Surabaya City (as vertex S). The two algorithms used are compared using 4 (four) parameters, namely (1) algorithm time, (2) algorithm complexity, (3) algorithm programming sequence, and (4) distance traveled when completing the algorithm. From the comparison results, it can be concluded that the Greedy algorithm is superior in terms of execution time, complexity, and algorithm sequence, but for the shortest route, Dijkstra's algorithm is superior to the Greedy algorithm with the shortest route being 105 km, while the Greedy algorithm produces the shortest route 129 km.**

Keywords— **Shortest Path, Graph, Greedy Algorithm, Dijkstra's Algorithm**

Abstrak — **Menemukan jarak terpendek adalah masalah umum bagi pengguna transportasi dari awal. Hal ini dikarenakan para pengguna transportasi membutuhkan jarak yang pendek dan waktu yang singkat agar dapat mencapai perjalanan dengan cepat. Masalah transportasi sering muncul dari lokasi terpencil, terutama jika ada pengguna transportasi mencoba untuk pergi ke kota besar. Telah banyak algoritma yang digunakan untuk membantu menemukan rute terpendek dari satu kota ke kota lain diantaranya algoritma *Greedy* dan algoritma *Dijkstra*. Pada penelitian ini diimplementasi kedua algoritma tersebut untuk mencari jalur terpendek dengan cakupan wilayah dari Kabupaten Tuban (sebagai *vertex* T) hingga ke Kota Surabaya (sebagai *vertex* S). Kedua algoritma yang digunakan dibandingkan dengan menggunakan 4 (empat) parameter yaitu (1)waktu algoritma, (2) kompleksitas algoritma, (3) urutan pemrograman algoritma, dan (4) hasil jarak yang ditempuh saat menyelesaikan algoritma. Dari hasil perbandingan dapat disimpulkan bahwa algoritma *Greedy* lebih unggul terkait waktu eksekusi, kompleksitas, dan urutan algoritma, namun untuk hasil rute terpendek algoritma *Dijkstra* lebih unggul dibandingkan dengan algoritma *Greedy* dengan rute terpendek yang dihasilkan 105 Km, sedangkan algoritma *Greedy* menghasilkan rute terpendek 129 Km.**

Kata Kunci— **Rute Terpendek, Graf, Algoritma *Greedy*, Algoritma *Dijkstra***

I. PENDAHULUAN

Dalam ilmu komputer, teori graf adalah topik utama penelitian suatu jalur node ke node. Teori graf adalah graf yang mengandung informasi tertentu jika diinterpretasikan dengan benar. Dalam kehidupan sehari-hari, graf digunakan untuk menggambarkan berbagai jenis struktur yang ada. Tujuannya adalah untuk memvisualisasikan sesuatu dan membuatnya lebih mudah dipahami. Ada banyak struktur yang dapat diwakili oleh graf, dan ada banyak masalah yang dapat diselesaikan dengan graf, bahkan sering digunakan untuk mewakili bentuk jaringan-jaringan. Misalkan dengan kota atau wilayah sebagai simpul *(vertex)*, dan setiap kota memiliki rute atau jalan sebagai sisi *(edge)*, dan bobotnya adalah berapa nilai kilometer disetiap panjang jalan yang dilalui. Secara matematis, graf didefinisikan sebagai pasangan himpunan yang ditulis dengan simbol $G = (V, E)$, bahwa V adalah himpunan dari simpul tak kosong (simpul atau node) dan E adalah himpunan yang menghubungkan sisi ke beberapa simpul.

Angkutan umum sebagai sarana transportasi telah lama digunakan di masyarakat, namun menemukan rute terpendek merupakan masalah yang banyak dibahas dan dipelajari sejak akhir tahun 1950-an. Dalam permasalahan graf tersebut, maka memungkinkan untuk melakukan proses analitik, yang merupakan suatu model pengambilan keputusan yang komprehensif dengan mempertimbangkan hal-hal yang bersifat kualitatif dan kuantitatif. Hal ini berguna untuk menentukan hasil akhir dari beberapa hasil yang ditentukan, dengan memperhatikan simpul (*vertex*) dan sisi (*edge*) pada graf. Dengan adanya proses analitik dalam penentuan rute transportasi, maka seolah-olah dapat mencapai tempat itu dengan mudah, cepat dan akurat [1]. Dalam proses analitik,

dibutuhkan penentuan bobot kriteria yang lebih prioritas, dengan melalui beberapa survei, maka didapatkan bahwa rute dengan jumlah bobot lebih sedikit.

Dalam permasalahan pencarian rute terpendek dapat menggunakan dengan dua algoritma, yaitu algoritma *Greedy* dan algoritma *Dijkstra* seperti pada peneliti-peneliti terdahulu, berpendapat bahwa algoritma *Greedy* dapat diterapkan pada banyak jenis masalah keputusan [2]. Dalam urutan untuk menerapkan *Carousel Greedy algorithm (CG)* ke masalah keputusan tertentu, kami berasumsi bahwa sudah ada keserakahan konstruktif algoritma untuk masalah itu; jika tidak, mungkin lebih sulit untuk diterapkan. Fokus penelitian lainnya seperti pada penerapan algoritma *Greedy* dengan membahas rute terpendek dari kecamatan Ngaliyan ke kecamatan Sampangan, dari jarak yang akan ditempuh menggunakan algoritma *Greedy* dengan terdapat beberapa alternatif rute yang bisa dilalui dari dari kecamatan Ngaliyan ke kecamatan Sampangan [3]. Seperti juga halnya dalam penerapan algoritma *Greedy* dalam penentuan jalur tercepat rumah sakit di kota Palu agar masyarakat tahu bahwa rumah sakit apa saja yang ada di kota Palu serta memberi tahu lokasi dan jarak yang ditempuh untuk sampai di rumah sakit tersebut [4], serta perbandingan algoritma Genetika dengan algoritma *Greedy* untuk pencarian rute terpendek [5], dan dapat digunakan dalam sistem pencarian rute terpendek pencarian hotel dengan mempertimbangkan daya tarik wisata menggunakan algoritma *Greedy* [6]. Dalam makalah penelitian terdahulu mengusulkan algoritma *Greedy* untuk merutekan daya dari sumber ke beban dalam *Digital Microgrid (DMG)* terhadap pemilihan jalur tingkat biaya terkecil dari beban ke *Distributed Energy Resources (DERs)* pemasok nya [7]. Selebih itu, algoritma *Greedy* digunakan untuk pengoptimalan pembangunan jalan di kota Lhokseumawe, dengan harapan dapat meningkatkan perekonomian penduduk sekitar dalam sektor pembangunan jalan [8].

Selanjutnya, adapun peneliti terdahulu yang membahas implementasi penerapan algoritma *Dijkstra* dalam menentukan jalur tercepat menuju salah satu titik tempat wisata air yang ada di Klaten dengan titik awal Terminal Delanggu dengan penerapan algoritma *Dijkstra* [9] dan penelitian pencarian lokasi hotel dengan simpul tempat wisata di daerah Medan berdasarkan rute terpendek dengan metode algoritma *Dijkstra* [10]. Algoritma *Dijkstra* juga dapat digunakan untuk menemukan jalur terpendek antara dua area/stasiun Mumbai berdasarkan rute sebenarnya dari bus BEST [11] dan pencarian rute terpendek ke museum di Jakarta [12]. Penelitian lainnya juga mengenai dalam penerapan algoritma *Dijkstra* dalam sistem perencanaan angkutan umum wilayah Metropolitan Bangkok yang membantu pengguna angkutan umum terhadap informasi dan data, serta menyediakan perencanaan sistem untuk merencanakan rute dan moda transportasi dengan tarif [13]. algoritma *Dijkstra* juga pernah dilakukan dalam mencari jalur terpendek antara dua kota di Republik Kosovo, dan kota-kota tersebut berada di kedua ujung negara, di mana titik awalnya adalah kota Gjakova dan tujuannya adalah kota Gjilan [14]. Selain itu, algoritma *Greedy* atau *best first search* dan algoritma *Dijkstra* juga dilakukan secara analisis serta perbandingan terhadap aplikasi pencarian jalur pendonor darah terdekat [15]. Tidak hanya itu, penelitian algoritma *Dijkstra* dapat menentukan rute terpendek untuk mengunjungi destinasi tempat bersejarah dari pusat kota Surabaya, sehingga dapat mengurangi waktu berkendara, dengan demikian dapat menghasilkan lima rute terpendek yang bisa dilalui melalui titik awal di Stasiun Gubeng [16].

Seperti yang terjadi saat ini, banyak masyarakat kabupaten Tuban yang ingin menuju kota Surabaya dengan mencari jalur terpendek. Dalam mengambil rute terpendek dapat memudahkan masyarakat kabupaten Tuban dan sekitar wilayah desa, sehingga sangat diperlukan untuk mencari rute tercepat ke kota Surabaya dengan menghabiskan biaya yang minimal dan waktu yang efektif. Jalur pendek adalah perpindahan dari satu baris ke baris lainnya, dengan ukuran atau nilai akhir yang lebih kecil dari baris pertama ke baris terakhir. Oleh karena itu, dalam jurnal penelitian ini akan dibahas penerapan algoritma *Greedy* dan algoritma *Dijkstra* yang keduanya digunakan untuk menentukan efisiensi waktu dan cakupan area

II. METODE PENELITIAN

Metode penelitian ini menggunakan uji kuantitatif atau metode komparatif untuk membandingkan pengolahan dua variabel atau lebih. Tujuan metode penelitian kuantitatif untuk mengidentifikasi perbedaan antara dua atau lebih situasi, peristiwa, kegiatan, atau program. Perbandingan yang dilihat dari kedua algoritma, yaitu algoritma *Greedy* dan algoritma *Dijkstra* yang bagaimana seluruh unsur dalam komponen penelitian terkait antara satu sama lain. Perhitungan yang digunakan metode penelitian kuantitatif diungkapkan dalam bentuk persamaan dan perbedaan antara faktor-faktor yang mendukung perencanaan, pelaksanaan dan hasil pendukung.

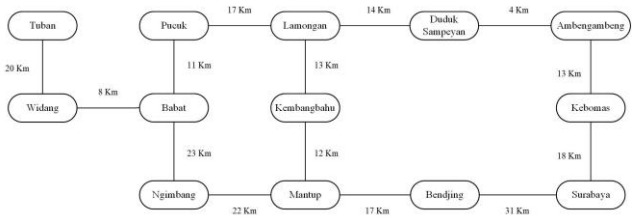
A. Pemilihan Data

Pada penelitian ini, data yang digunakan adalah data yang diperoleh dari *Google Maps* yaitu data jarak antar daerah Tuban, Widang, Babat, Pucuk, Ngimbang, Mantup, Kembangbahu, Lamongan, Duduk Sampeyan, Bendjing, Ambengambeng, Kebomas, dan Surabaya. Kemudian, dari pemilihan data-data tersebut, disusun pada sebuah tabel di *Excel*, seperti pada Gambar 1, yang terdapat 13 simpul *(vertex)*, dan 14 memiliki rute atau jalur sebagai sisi *(edge)*, serta memiliki bobot *(weight)* yang berbeda-beda. Selain itu, untuk bentuk gambar implementasi graf sederhananya seperti pada Gambar 2.

p-ISSN : 2640-7363
e-ISSN : 2614-6606

No.	Sisi (edge)	Bobot (weight)	No.	Sisi (edge)	Bobot (weight)		
1.)	T-W	20 Km	8.)	M-KU	12 Km	*#inisialisasi simpul (vertex)*	
2.)	W-B	8 Km	9.)	L-D	14 Km	Tuban = T	Mantup = M
3.)	BA-P	11 Km	10.)	M-BE	17 Km	Widang = W	Duduk Sampeyan = D
4.)	BA-N	23 Km	11.)	D-A	4 Km	Babat = BA	Bendjing = BE
5.)	P-L	17 Km	12.)	A-KS	13 Km	Pucuk = P	Ambengambeng = A
6.)	N-M	22 Km	13.)	BE-S	31 Km	Ngimbang = N	Kebomas = KS
7.)	L-KU	13 Km	14.)	KS-S	18 Km	Lamongan = L	Surabaya = S
						Kembangbahu = KU	

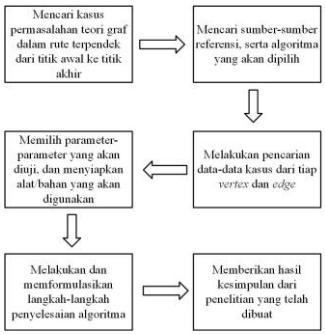
Gambar 1. Data graf



Gambar 2. Graf sederhana dari kabupaten Tuban ke kota Surabaya

B. Pra-Pemprosesan

Bentuk penggambaran awal dalam membuat langkah-langkah tahapan dalam pra-pemrosesan jurnal peneltian ini, seperti pada gambar 3.



Gambar 3. Penggambaran awal dalam pembuatan jurnal penelitian

Kemudian dari tahapan tersebut, dijelaskan secara terperinci dalam beberapa cara langkah pengerjaan, diantaranya sebagai berikut :

(1) Memperhatikan permasalahan dalam kasus persoalan yang terjadi di kabupaten Tuban mengenai jalur terpendek ke kota Surabaya, serta memperhatikan bobot pada setiap rute;

(2) Melakukan penggambaran bentuk graf sederhana;

(3) Memformulasikan dan membentuk model matematis dari persoalan jarak terpendek yang harus dilalui dengan algoritma *Greedy* dan algoritma *Dijkstra*;

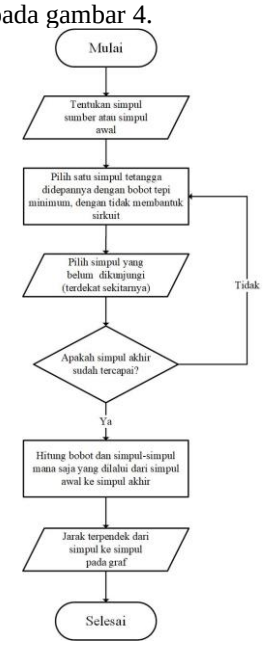
(4) Memperoleh hasil akhir dalam penyelesaian algoritma *Greedy* dan algoritma *Dijkstra* dengan jarak atau rute yang terpendek dari titik awal kabupaten Tuban ke titik akhir kota Surabaya;

(5) Memberikan kesimpulan akhir dari perbandingan algoritma *Greedy* dan algoritma *Dijkstra* dan dapat mengimplementasikan hasil studi jurnal penelitian ini dalam kehidupan sehari-hari.

C. Sistematika Algoritma Greedy dan Algoritma Dijkstra

1. Sistematika Algoritma Greedy

Penyelesaian kasus teori graf sederhana untuk menentukan lintasan terpendek dapat diperoleh dengan menggunakan algoritma *Greedy*, yang merupakan salah satu algoritma yang langsung melakukan pemecahan masalah tanpa mempertimbangkan konsekuensi kebelakangnya. Algoritma *Greedy* berisi pendekatan yang mengkonstruksi solusi melalui tahapan urutan yang terus berkembang hingga solusi masalah tercapai. Bentuk gambaran flowchart algoritma *Greedy*, seperti pada gambar 4.



Gambar 4. Flowchart algoritma *Greedy*

Masalah yang dihadapi, menurut [5] algoritma *Greedy* adalah algoritma yang memecahkan masalah langkah demi langkah, misalnya pada setiap langkah; (1) Memilih opsi terbaik yang tersedia pada waktu tertentu saat itu, dan (2) Berharap bahwa dengan memilih opsi optimum lokal pada setiap langkah akan meraih optimum global, dengan algoritma *Greedy* mengasumsikan bahwa optimum lokal merupakan bagian dari optimum global.

Seiring dengan algoritma lain, algoritma *Greedy* juga memiliki kelebihan dan kekurangan. Kelebihannya antara lain; (1) Pengambilan keputusan yang cepat, (2) Efektif dan tidak memakan waktu, dan (3) Mudah diterapkan dalam banyak masalah yang ada. Sedangkan kekurangannya antara lain; (1) Hasil akhirnya buruk, karena tidak memikirkan konsekuensinya, (2) Tidak dapat memantau parameter, dan (3) Tidak ada pilihan lain jika masalah tidak dapat diselesaikan. Selain itu, algoritma *Greedy* didasarkan pada pemindahan *edge-to-edge* dan tanpa mempertimbangkan konsekuensi depannya pada setiap langkah, algoritma *Greedy* tidak sepenuhnya bekerja pada semua solusi alternatif dan tidak selalu berhasil memberikan solusi yang benar-benar optimal, tetapi menyediakan sebuah solusi yang mendekati nilai optimal [4].

Dengan demikian, sistematika pada algoritma *Greedy* membangun solusi sepotong demi sepotong, selalu memilih

bagian berikutnya yang menawarkan bobot paling minim dan langsung. Jadi masalah di mana memilih optimal lokal juga mengarah ke solusi global yang paling cocok.

2. Sistematika Algoritma Dijkstra

Algoritma *Dijkstra* digunakan untuk menyelesaikan masalah jarak terpendek dan masalah ketidakpastian pencarian rute, yang dapat diselesaikan dengan menggunakan graf alternatif yang disebut algoritma *Dijkstra* [12]. Kelebihan dari algoritma ini adalah dapat menghitung secara akurat dari *edge-to-edge* terdekat yang dipilih dengan tujuan hasil titik yang diberikan sebagai target, sedangkan kelemahan dari akurasi algoritma sangat besar dan panjang. Secara umum, algoritma *Dijkstra* lebih cenderung diterapkan untuk mencari lintasan paling pendek pada graf berarah, tetapi pada algoritma juga dapat diterapkan pada graf tak berarah. Bentuk gambaran flowchart algoritma *Dijkstra*, seperti pada gambar 5.

Gambar 5. Flowchart algoritma *Dijkstra*

Pada [17], ketika menentukan jalur terpendek menggunakan algoritma *Dijkstra*, inputnya adalah graf berbobot $G(e, v)$, sedangkan outputnya adalah jalur terpendek dari simpul awal ke semua simpul graf. Dengan demikian algoritma *Dijkstra* dapat menemukan solusi terbaik. Algoritma *Dijkstra* bekerja dengan cara yang sama seperti algoritma *Greedy*, yaitu menggunakan prinsip antrian "pengurutan", tetapi antrian yang digunakan oleh algoritma *Dijkstra* adalah "antrian prioritas" [18]. Oleh karena itu, sistem hanya mencari node dengan prioritas tertinggi.

Persoalan mencari lintasan terpendek di dalam graf merupakan salah satu persoalan optimasi, misalnya dalam bentuk permasalahan mencari suatu lokasi. Cara menentukan jalur terpendek menggunakan metode algoritma *Dijkstra* sebagai berikut; (a) algoritma *Dijkstra* melakukan komputasi pada node tetangga yang terkait dengan node utama, dan hasil yang didapat lebih kecil adalah node kedua, kemudian pergi ke simpul terdekat lainnya, dan (b) hitung jarak antara setiap titik dan lokasi pada rute. Hasil yang diperoleh pada satu node terkadang menunjukkan jarak yang berbeda, dikarenakan sisi jarak pada rute ke rute sama sekali berbeda-beda.

Dengan demikian, sistematika pada algoritma *Dijkstra* dapat menemukan sebuah rute perjalanan terpendek dengan lebih akurat, dan langkah-langkahnya sebagai berikut :

(1) Buat mengatur *sptSet (shortest path tree set)* yang melacak simpul yang termasuk dalam pohon jalur terpendek, yaitu jarak minimum dari sumber dihitung dan diselesaikan, dengan mulainya set kosong;

(2) Tetapkan nilai jarak ke semua simpul dalam graf input. Inisialisasi semua nilai jarak sebagai *infinite*. Tetapkan nilai jarak sebagai 0 untuk simpul sumber sehingga dipilih terlebih dahulu;

(3) Sementara mengatur *sptSet* tidak mencakup semua simpul. Pertama, pilih simpul u yang tidak ada di *sptSet* dan memiliki nilai jarak minimum. Setelah itu, Sertakan *u* ke *sptSet*. Dan terakhir, perbarui nilai jarak dari semua simpul yang berdekatan dari *u*. Untuk memperbarui nilai jarak, iterasi melalui semua simpul yang berdekatan. Untuk setiap simpul bertetangga *v*, jika jumlah nilai jarak u (dari sumber asal) dan bobot sisi *uv* lebih kecil dari nilai jarak v, maka perbarui nilai jarak *v*.

D. Parameter Pengujian Algoritma

Dalam penelitian ini, menggunakan bahasa *Python* pada tiap-tiap algoritma *Greedy* dan algoritma *Dijkstra*, agar memastikan hasil yang diperoleh dengan benar atau tidak ada kesalahan. Selain itu, adapun perangkat pendukung lainnya yang digunakan, misalnya sebagai berikut:

1. *Perangkat Keras (Hardware)*

(a) Laptop Lenovo Ideapad Slim 3;

(b) Prosesor Intel Core i5-10210U berkecepatan 2.1 GHz;

(c) *RAM* 8 *GB.*

2. *Perangkat Lunak (Software)*

(a) *Spyder (Python 3.7)*;

(b) *Microsoft Excel*.

Parameter-parameter yang digunakan dalam perbandingan algoritma *Greedy* dan algoritma *Dijkstra* adalah sebagai berikut :

1. Waktu Algoritma

Dalam menyelesaikan suatu permasalahan, waktu pengeksekusian algoritma merupakan hal yang terpenting dalam memanajemen waktu agar lebih cepat. Saat menjalankan sebuah algoritma, jika semakin sedikit waktu yang dijalankan, maka semakin sedikit tahapan-tahapan dalam penyelesaian masalah yang ditinjau secara sistematis.

2. Kompleksitas Algoritma

p-ISSN : 2640-7363
e-ISSN : 2614-6606

Sebuah algoritma diukur dari seberapa jauh kompleksitas itu dijalankan, agar lebih efisien dalam meminimumkan kebutuhan waktu dan ruang suatu langkah pengeksekusian algoritma.

3. Sequence Algoritma

Hal yang mendasari suatu algoritma terkadang dilihat dari urutan *(sequence)* dari pengerjaan suatu perintah atau statement pertama sampai dengan perintah atau statement terakhir. Setiap instruksi dikerjakan satu per satu tidak akan mempengaruhi hasil akhir, walaupun urutannya berbeda.

4. Hasil Akhir Jarak yang ditempuh

Secara matematis, terkadang dalam menggunakan algoritma yang berbeda, belum tentu menghasilkan hasil yang sama, tetapi terkadang ada yang lebih efisiensi dalam menyelesaikan suatu permasalahan yang terjadi. Hasil akhir suatu jarak dalam rute terpendek sangatlah berguna bagi banyak masyarakat, walaupun ada perbedaan kecil dari hasil jarak terpendek dalam sebuah algoritma yang berbeda.

III. PEMBAHASAN

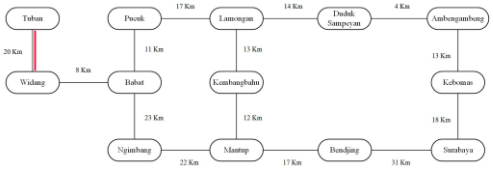
Dalam menyelesaikan permasalahan jalur terpendek, dari simpul awal kabupaten Tuban (T) ke simpul akhir kota Surabaya (S) yang diperlukan beberapa perhitungan cara langkah jarak terpendek dan pengujian empat parameter seperti yang telah dibahas pada bagian metode penelitian, terutama metode algoritma *Greedy* dan metode algoritma *Dijkstra*.

A. Hasil Pengujian Algoritma Greedy

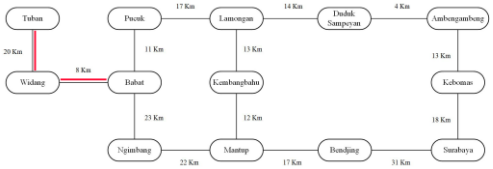
1. Jarak Terpendek

Pencarian jalur terpendek dengan algoritma *Greedy* untuk tahap pertama dilakan perhitungan cara manual, sedangkan untuk tahap kedua dilakukan dengan menggunakan bahasa pemrograman *Python* pada aplikasi *Spyder (Python 3.7)*, dengan memasukkan jumlah *vertex* dan *edges* yang merepresentasikan titik dan jalur dari kabupaten Tuban ke kota Surabaya. Tahapan algoritma *Greedy* menggunakan pendekatan penyelesaian masalah dengan mencari nilai sedikit atau minima pada setiap langkah di depannya. Berikut ini adalah cara manual menentukan jalur terpendek menggunakan metode algoritma *Greedy* adalah sebagai berikut :

(a) Dimulai dari simpul T, lalu lanjut ke simpul W, dengan bobot total adalah 20 (km), seperti pada gambar 6. Lalu dari simpul W ke BA, dengan total bobot sekarang adalah 28 (km), seperti pada gambar 7.

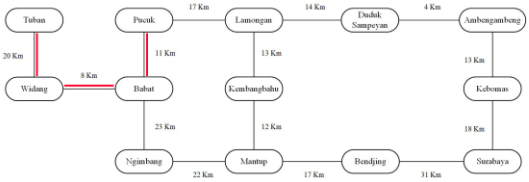


Gambar 6. Penggambaran langkah pertama dalam penyelesaian jarak terpendek metode algoritma *Greedy*



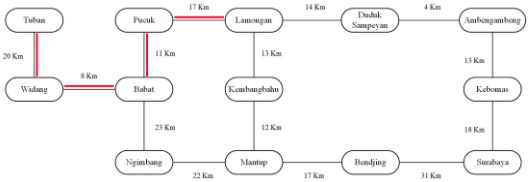
Gambar 7. Penggambaran langkah kedua dalam penyelesaian jarak terpendek metode algoritma *Greedy*

(b) Selanjutnya pada simpul BA, terdapat 2 sisi yang terhubung selanjutnya, yaitu sisi BA-P dengan bobot 11 (km) dan BA-N dengan bobot 23 (km). Dikarenakan algoritma *Greedy* memilih langkah terpendek hanya di depannya, jadi memilih sisi BA-P dan total bobot sekarang adalah adalah 39 (km), seperti pada gambar 8.



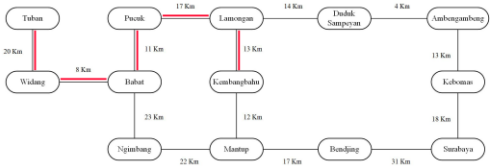
Gambar 8. Penggambaran langkah ketiga dalam penyelesaian jarak terpendek metode algoritma *Greedy*

(c) Dari simpul P, akan lanjut ke simpul L, dengan total bobot sekarang ini sebesar 56 (km), seperti pada gambar 9.



Gambar 9. Penggambaran langkah keempat dalam penyelesaian jarak terpendek metode algoritma *Greedy*

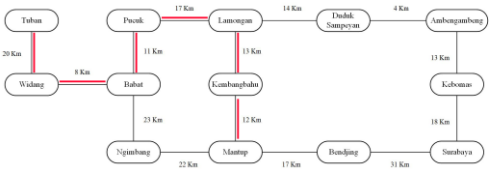
(d) Pada simpul L, ada 2 sisi yang terhubung selanjutnya, yaitu sisi L-D dengan bobot 14 (km) dan L-KU dengan bobot 13 (km). Kemudian dipilihlah sisi L-KU, dikarenakan sisi tersebut lebih kecil daripada sisi L-D. Sehingga diperoleh total bobot sekarang ini sebesar 69 (km), seperti pada gambar 10.



Gambar 10. Penggambaran langkah kelima dalam penyelesaian jarak terpendek metode algoritma *Greedy*

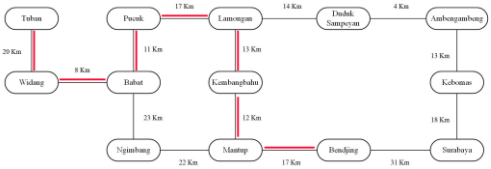
(e) Lalu pada simpul KU berlanjut langsung pada simpul M, dengan total bobot saat ini sebesar 81 (km), seperti pada gambar 11.

p-ISSN : 2640-7363
e-ISSN : 2614-6606



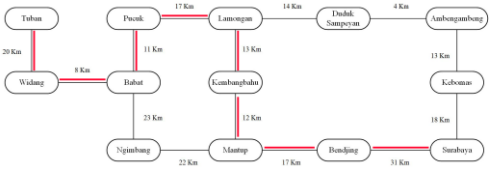
Gambar 11. Penggambaran langkah keenam dalam penyelesaian jarak terpendek metode algoritma *Greedy*

(f) Simpul M juga memiliki 2 sisi yang terhubung selanjutnya, yaitu sisi M-N dengan bobot 22 (km) dan M-BE dengan bobot 17 (km). Dikarenakan jika memilih sisi M-N itu akan melakukan sebuah *looping* dan hasilnya tidak pasti, jadi akhirnya memilih sisi M-BE, dengan total bobot saat ini sebesar 98 (km), seperti pada gambar 12.



Gambar 12. Penggambaran langkah ketujuh dalam penyelesaian jarak terpendek metode algoritma *Greedy*

(g) Akhirnya pada simpul BE, selanjutnya langsung ke simpul akhir yaitu S, maka total hasil bobot yang diperoleh adalah 129 (km), seperti pada gambar 13.



Gambar 13. Penggambaran langkah kedelapan dalam penyelesaian jarak terpendek metode algoritma *Greedy*

Dengan demikian, menurut metode algoritma *Greedy* diperoleh bobot 129 (km) dengan graf G = T-W-BA-P-L-KU-M-BE-S (Tuban - Widang - Babat - Pucuk - Lamongan - Kembangbahu - Mantup - Bending - Surabaya).

Selain itu perhitungan jalur terpendek dilakukan dengan menggunakan bahasa pemrograman *Python* pada aplikasi *Spyder (Python 3.7)*. Mengenai kode program dan hasil eksekusi -nya, dapat dilihat pada Gambar 14 dan Gambar 15, adalah sebagai berikut :

```python
import timeit

start = timeit.default_timer()

def rute_terpendek(graf, mulai, akhir):
    result = [] # node dengan jarak terpendek simpan ke dalam list
    result.append(mulai) # inisialisasi node pertama dengan nilai asal
    while akhir not in result: # telusuri graf sampai tujuan ditemukan
        node_sekarang = result[-1]
        jarak_terpendek = min(graf[node_sekarang].values())
        for node, jarak in graf[node_sekarang].items(): # iterasi mencari
                                                        # node selanjutnya
            if jarak == jarak_terpendek:
                result.append(node)
    returr result

input_graf = {
    "Tuban": {"Widang": 20},
    "Widang": {"Babat": 8},
    "Babat": {"Pucuk": 11, "Ngimbang": 23},
    "Pucuk": {"Lamongan(1)": 17},
    "Ngimbang": {"Mantup(1)": 22},
    "Lamongan(1)": {"Kembangbahu(1)": 13, "Duduk Sampeyan": 14},
    "Lamongan(2)": {"Duduk Sampeyan": 14},
    "Kembangbahu(1)": {"Mantup(1)": 13},
    "Kembangbahu(2)": {"Lamongan(2)": 13},
    "Mantup(1)": {"Bendjing": 17},
    "Mantup(2)": {"Kembangbahu(2)": 13, "Bendjing": 17},
    "Duduk Sampeyan": {"Ambengambeng": 4},
    "Bendjing": {"Surabaya": 31},
    "Ambengambeng": {"Kebomas": 13},
    "Kebomas": {"Surabaya": 18},
}

mulai_vertex = "Tuban"
akhir_vertex = "Surabaya"
Greedy = rute_terpendek(input_graf, mulai_vertex, akhir_vertex)
print ("Jalur Dari Tuban ke Surabaya adalah, ", Greedy)

stop = timeit.default_timer()
lama_eksekusi = stop - start

print("Lama eksekusi: ", lama_eksekusi, "detik")
```

Gambar 14. Program Python metode algoritma *Greedy*

```
In [1]: runfile('C:/Users/LENOVO/.spyder-py3/Algoritma Greedy -
Short Path.py', wdir='C:/Users/LENOVO/.spyder-py3')
Jalur Dari Tuban ke Surabaya adalah,  ['Tuban', 'Widang',
'Babat', 'Pucuk', 'Lamongan(1)', 'Kembangbahu(1)', 'Mantup(1)',
'Bendjing', 'Surabaya']
Lama eksekusi:  0.00016449999999945675 detik
```

Gambar 15. Hasil eksekusi program Python metode algoritma *Greedy*

2. *Waktu Algoritma*

Waktu menjalankan program bahasa *Python* algoritma *Greedy* dilakukan secara 3 kali, agar lebih cepat dalam mengambil rata-ratanya. Pada pengujian eksekusi program bahasa *Python* pertama sebesar 0.00016449999999945675 detik, lalu pada pengujian eksekusi program kedua bahasa *Python* sebesar 0.00036090000003241585 detik, dan terakhir pada pengujian eksekusi program bahasa *Python* ketiga sebesar 0.00018580000005385955 detik. Sehingga, diperoleh rata-rata hasil waktu pengujian eksekusi program bahasa *Python* sebesar 0.00023706666669524405 detik.

3. *Kompleksitas Algoritma*

Kompleksitas saat menjalankan program algoritma *Greedy* bahasa *Python* sebesar 43 baris.

4. *Sequence Algoritma*

Urutan atau sequence saat menjalankan program algoritma *Greedy* bahasa *Python*, yaitu diselesaikan hanya memperhatikan langkah panjang rute terpendek hanya di

p-ISSN : 2640-7363
e-ISSN : 2614-6606

depannya, yang terkadang tidak memperhatikan resiko-resiko yang terjadi.

5. Hasil Akhir Jarak yang ditempuh

Hasil jarak saat menjalankan program algoritma *Greedy* bahasa *Python*, yaitu diperoleh total bobot 129 (km) dengan graf G = T-W-BA-P-L-KU-M-BE-S (Tuban - Widang - Babat - Pucuk - Lamongan - Kembangbahu - Mantup - Bendjing - Surabaya).

B. Hasil Pengujian Algoritma Dijkstra

1. Jarak Terpendek

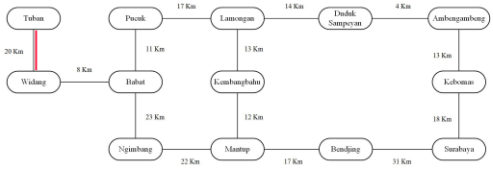
Pencarian jalur terpendek dengan algoritma *Dijkstra* untuk tahap pertama dilakan perhitungan cara manual, sedangkan untuk tahap kedua dilakukan dengan menggunakan bahasa pemrograman *Python* pada aplikasi *Spyder (Python 3.7)*, dengan memasukkan jumlah vertex dan edges yang sama seperti pada algoritma *Greedy* sebelumnya, yang menggambarkan titik dan jalur dari kabupaten Tuban ke kota Surabaya. Tahapan algoritma *Dijkstra* menggunakan pendekatan dalam penentuan lintasan terpendek dari suatu titik tertentu ke setiap titik lain pada suatu graf, dan membentuk pohon *Dijkstra* pada suatu graf dengan pengulangan sebanyak $|V(G)| - 1$, sehingga membentuk pohon pembangun *(spanning tree) Dijkstra* setelah pengulangan terakhir dari graf G. Berikut ini adalah data tabel yang dilakukan pada *Microsoft Excel* pada cara manual dalam menentukan jalur terpendek menggunakan metode algoritma *Dijkstra*, seperti pada Gambar 16 adalah sebagai berikut :

vertex	T	W	BA	P	N	L	KU	M	D	BE	A	KS	S
T	0	20 t	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
W		20 t	28 w	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
BA			28 w	39 ba	51 ba	∞	∞	∞	∞	∞	∞	∞	∞
P				39 ba	51 ba	56 p	∞	∞	∞	∞	∞	∞	∞
L					51 ba	56 p	69 l	∞	70 l	∞	∞	∞	∞
N					51 ba		69 l	73 n	70 l	∞	∞	∞	∞
KU							69 l	73 n	70 l	∞	∞	∞	∞
D								73 n	70 l	∞	74 d	∞	∞
M								73 n		90 m	74 d	∞	∞
A										90 m	74 d	87 a	∞
KS										90 m		87 a	105 ks
BE										90 m			105 ks
S													105 ks

Gambar 16. Data tabel metode algoritma *Dijkstra*

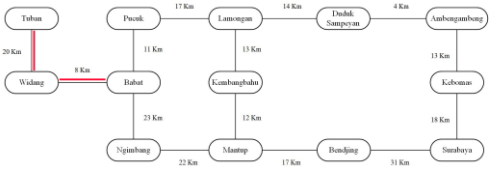
Dari data *Microsoft Excel* dengan metode algoritma *Dijkstra* telah ditemukan jalur terpendek sebesar 105 (km). Lalu, penyelesaian data tersebut dibuat penjelasan langkah-langkah hasil penggambaran graf jalur terpendek, seperti berikut :

(a) Pada iterasi pertama, simpul awal dimulai dari T berbobot awal 0 dan berikutnya menuju simpul W, dengan penulisan bobot 20t (km) dan kolom dengan simbol ∞ berarti simpul tersebut belum bisa dijangkau. Karena simpul T hanya ada satu simpul yang dikunjungi maka simpul berikutnya menuju simpul W, seperti pada gambar 17.



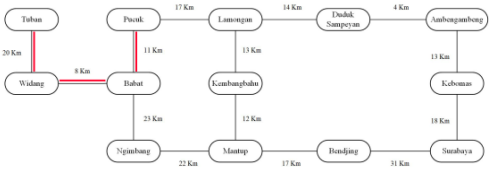
Gambar 17. Penggambaran langkah pertama dalam penyelesaian jarak terpendek metode algoritma *Dijkstra*

(b) Pada iterasi kedua, simpul W akan menuju simpul BA dengan total bobot 28w (km). Karena simpul W hanya ada satu simpul yang dikunjungi maka simpul berikutnya menuju simpul BA, seperti pada gambar 18.



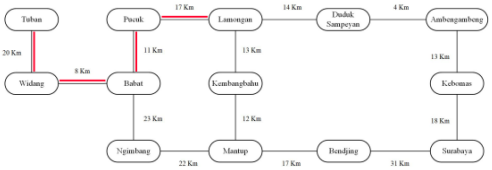
Gambar 18. Penggambaran langkah kedua dalam penyelesaian jarak terpendek metode algoritma *Dijkstra*

(c) Pada iterasi ketiga, simpul BA dapat mengunjungi dua simpul, yaitu simpul P dengan total bobot 39ba (km) dan simpul N dengan total bobot 51ba (km). Karena algoritma ini mencari jarak terdekat, jadi memilih simpul P sebagai kota dengan bobot minimal, seperti pada gambar 19.



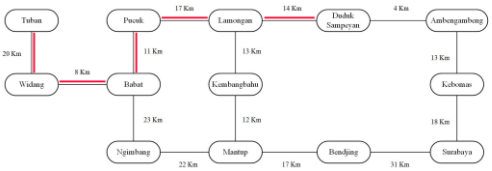
Gambar 19. Penggambaran langkah ketiga dalam penyelesaian jarak terpendek metode algoritma *Dijkstra*

(d) Pada iterasi keempat, simpul P selanjutnya menuju ke simpul L total bobot 56p (km). Karena simpul P hanya ada satu simpul yang dikunjungi maka simpul berikutnya menuju simpul L, seperti pada gambar 20.



Gambar 20. Penggambaran langkah keempat dalam penyelesaian jarak terpendek metode algoritma *Dijkstra*

(e) Pada iterasi kelima, simpul L akan menuju dua simpul yang dikunjungi, yaitu simpul D dengan total bobot 70l (km) dan simpul KU dengan total bobot 69l (km). Dikarenakan pada tabel *Microsoft Excel* tidak ada keterhubungan simpul KU yang dipilih berikutnya sebagai algoritma penyelesaian jarak terpendek, maka selanjutnya menuju ke simpul D, seperti pada gambar 21.

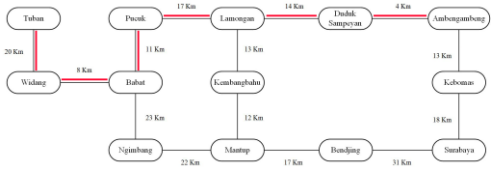


Gambar 21. Penggambaran langkah kelima dalam penyelesaian jarak terpendek metode algoritma *Dijkstra*

(f) Pada iterasi keenam, simpul D akan menuju simpul A dengan total bobot 74d (km). Karena simpul D hanya ada satu

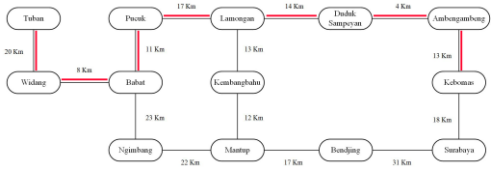
p-ISSN : 2640-7363
e-ISSN : 2614-6606

simpul yang dikunjungi maka simpul berikutnya menuju simpul A, seperti pada gambar 22.



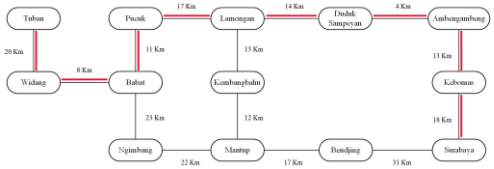
Gambar 22. Penggambaran langkah keenam dalam penyelesaian jarak terpendek metode algoritma *Dijkstra*

(f) Pada iterasi ketujuh, simpul A menuju simpul KS dengan total bobot 87a (km). Karena simpul A hanya ada satu simpul yang dikunjungi maka simpul berikutnya menuju simpul KS, seperti pada gambar 23.



Gambar 23. Penggambaran langkah keenam dalam penyelesaian jarak terpendek metode algoritma *Dijkstra*

(f) Pada iterasi kedelapan, simpul KS menuju simpul akhir S dengan total bobot 105ks (km). Sehingga dapat ditemukan jarak terpendek dari penyelesaian algoritma yang digunakan, seperti pada gambar 24.



Gambar 24. Penggambaran langkah keenam dalam penyelesaian jarak terpendek metode algoritma *Dijkstra*

Dengan demikian, menurut metode algoritma *Dijkstra* diperoleh bobot 105 (km) dengan graf G = T-W-BA-P-L-D-A-KS-S (Tuban - Widang - Babat - Pucuk - Lamongan - Duduk Sampeyan - Ambengambeng - Kebomas – Surabaya).

Selain itu perhitungan jalur terpendek dilakukan dengan menggunakan bahasa pemrogaman *Python* pada aplikasi Spyder (*Python* 3.7). Mengenai kode program dan hasil eksekusi -nya, dapat dilihat pada Gambar 25 dan Gambar 26, adalah sebagai berikut :

```python
import timeit

start = timeit.default_timer()

class Dijkstra:
    def __init__(self, vertex, graf):
        self.vertex = vertex
        self.graf = graf
    def rute_terpendek(self, mulai, akhir):
        unvisited = {n: float("inf") for n in self.vertex}
        unvisited[mulai] = 0  # setel titik awal ke 0
        visited = {}  # daftar semua node yang dikunjungi
        parents = {}  # pendahulu
        while unvisited:
            min_vertex = min(unvisited, key=unvisited.get)  # dapatkan jarak
                                                            # terkecil
            for tetangga, _ in self.graf.get(min_vertex, {}).items():
                if tetangga in visited:
                    continue
                new_distance = unvisited[min_vertex]
                + self.graf[min_vertex].get(tetangga, float("inf"))
                if new_distance < unvisited[tetangga]:
                    unvisited[tetangga] = new_distance
                    parents[tetangga] = min_vertex
            visited[min_vertex] = unvisited[min_vertex]
            unvisited.pop(min_vertex)
            if min_vertex == akhir:
                break
        return parents, visited
    @staticmethod
    def generate_path(parents, mulai, akhir):
        jalur = [akhir]
        while True:
            key = parents[jalur[0]]
            jalur.insert(0, key)
            if key == mulai:
                break
        return jalur

input_vertex = ("Tuban", "Widang", "Babat", "Pucuk", "Ngimbang", "Lamongan",
                "Kembangbahu", "Mantup", "Duduk Sampeyan", "Bendjing",
                "Ambengambeng", "Kebomas", "Surabaya")
input_graf = {
    "Tuban": {"Widang": 20},
    "Widang": {"Tuban": 20, "Babat": 8},
    "Babat": {"Widang": 8, "Pucuk": 11, "Ngimbang": 23},
    "Pucuk": {"Babat": 11, "Lamongan": 17},
    "Ngimbang": {"Babat": 23, "Mantup": 22},
    "Lamongan": {"Pucuk": 17, "Kembangbahu": 13, "Duduk Sampeyan": 14},
    "Kembangbahu": {"Mantup": 12, "Lamongan": 13},
    "Mantup": {"Ngimbang": 22, "Kembangbahu": 12, "Bendjing": 17},
    "Duduk Sampeyan": {"Lamongan": 14, "Ambengambeng": 4},
    "Bendjing": {"Mantup": 17, "Surabaya": 31},
    "Ambengambeng": {"Duduk Sampeyan": 4, "Kebomas": 13},
    "Kebomas": {"Ambengambeng": 13, "Surabaya": 18},
    "Surabaya": {"Kebomas": 18, "Bendjing": 31}
}

mulai_vertex = "Tuban"
akhir_vertex = "Surabaya"
dijkstra = Dijkstra(input_vertex, input_graf)
p, v = dijkstra.rute_terpendek(mulai_vertex, akhir_vertex)
se = dijkstra.generate_path(p, mulai_vertex, akhir_vertex)
print("Jalur terpendek dari %s ke %s adalah, ['%s']" % (mulai_vertex,
akhir_vertex, "', '".join(se)))

stop = timeit.default_timer()
lama_eksekusi = stop - start
print("Lama eksekusi: ", lama_eksekusi, "detik")
```

Gambar 25. Program *Python* metode algoritma *Dijkstra*

```
In [1]: runfile('C:/Users/LENOVO/.spyder-py3/Algoritma Dijkstra
- Short Path.py', wdir='C:/Users/LENOVO/.spyder-py3')
Jalur terpendek dari Tuban ke Surabaya adalah, ['Tuban',
'Widang', 'Babat', 'Pucuk', 'Lamongan', 'Duduk Sampeyan',
'Ambengambeng', 'Kebomas', 'Surabaya']
Lama eksekusi:  0.0009605999999999781 detik
```

Gambar 26. Hasil eksekusi program *Python* metode algoritma *Dijkstra*

2. *Waktu Algoritma*

Waktu menjalankan program bahasa *Python* algoritma *Dijkstra* dilakukan secara 3 kali, agar lebih cepat dalam mengambil rata-ratanya. Pada pengujian eksekusi program bahasa *Python* pertama sebesar 0.0009605999999999781 detik, lalu pada pengujian eksekusi program kedua bahasa *Python* sebesar 0.0005941999997958192 detik, dan terakhir pada pengujian eksekusi program bahasa *Python* ketiga sebesar 0.0002778000007310766 detik. Sehingga, diperoleh

p-ISSN : 2640-7363
e-ISSN : 2614-6606

rata-rata hasil waktu pengujian eksekusi program bahasa *Python* sebesar 0.0006108666668422913 detik.

3. Kompleksitas Algoritma

Kompleksitas saat menjalankan program algoritma *Dijkstra* bahasa *Python* sebesar 69 baris.

4. Sequence Algoritma

Urutan atau sequence saat menjalankan program algoritma *Dijkstra* bahasa *Python*, yaitu diselesaikan selalu dalam memperhatikan keseluruhan rute yang dilewati atau dilalui, sehingga mendapatkan bobot paling kecil dari total keseluruhan.

5. Hasil Akhir Jarak yang ditempuh

Hasil jarak saat menjalankan program algoritma *Dijkstra* bahasa *Python*, yaitu diperoleh total bobot 105 (km) dengan graf G = T-W-BA-P-L-D-A-KS-S (Tuban - Widang - Babat - Pucuk - Lamongan - Duduk Sampeyan - Ambengambeng - Kebomas - Surabaya).

C. Hasil Perbandingan Metode Algoritma Dijkstra Metode Algoritma dan Greedy

Setelah memperoleh hasil pengujian dari 4 parameter itu, maka selanjutkan membuat data tabel dari hasil perbandingan dari pengujian output metode algoritma *Greedy* dan metode algoritma *Dijkstra* yang dijalan untuk menerapkan rute terpendek dari kabupaten Tuban ke kota Surabaya, dapat dilihat seperti terlihat pada Tabel 1.

TABEL I
HASIL PERBANDINGAN METODE ALGORITMA *GREEDY* DAN ALGORITMA *DIJKSTRA* DALAM MENYELESAIKAN RUTE TERPENDEK DARI KABUPATEN TUBAN KE KOTA SURABAYA

No.	**Hasil Perolehan**		
	Parameter Pengujian	**Algoritma *Greedy***	**Algoritma *Dijkstra***
1	1. Waktu eksekusi program (detik) a. Pengujian pertama b. Pengujian kedua c. Pengujian ketiga d. Rata-rata	 0.000164…75 0.000360…85 0.000185…55 0.000237…05	 0.000960…81 0.000594…92 0.000277…66 0.000610…13
2	Kompleksitas program	43 baris	69 baris
3	*Sequence* program	Pada urutan algoritma *Greedy*, diselesaikan hanya memperhatikan langkah panjang rute terpendek hanya di depannya.	Pada urutan algoritma *Dijkstra*, diselesaikan selalu dalam memperhatikan keseluruhan rute yang dilewati atau dilalui, sehingga mendapatkan bobot paling kecil dari total keseluruhan.
4	Hasil akhir Jarak yang ditempuh	Tuban-Widang-Babat-Pucuk-Lamongan-Kembangbahu-Mantup-Bendjing-Surabaya (dengan bobot 129 km)	Tuban-Widang-Babat-Pucuk-Lamongan-Duduk Sampeyan-Ambengambeng-Kebomas-Surabaya (dengan bobot 105 km)

Berdasarkan pada Hasil Pengujian Program *Python* Metode Algoritma *Greedy* Dan Algoritma *Dijkstra* Dalam Menyelesaikan Rute Terpendek Dari Kabupaten Tuban Ke Kota Surabaya Tabel 1, Waktu eksekusi program bahasa *Python* dari kedua algoritma tersebut sama-sama dilakukan pengujian selama tiga kali, agar lebih cepat dalam mengambil rata-ratanya dan seimbang. Pengujian tiga kali dari yang pertama algoritma *Greedy* (detik) adalah (1) 0.0001644999999945675, (2) 0.0003609000003241585, dan (3) 0.0001858000005385955. Sehingga, diperoleh rata-rata hasil waktu pengujian eksekusi program algoritma *Greedy* bahasa Python sebesar 0.00023706666669524405 detik. Setelah itu, dilakukan juga pengujian tiga kali sama seperti pada algoritma *Greedy* pada algoritma *Dijkstra*, pengujian tiga kali dari yang pertama algoritma *Dijkstra* (detik) adalah (1) 0.0009605999999999781, (2) 0.0005941999997958192, dan (3) 0.0002778000007310766. Sehingga, diperoleh rata-rata hasil waktu pengujian eksekusi program algoritma *Dijkstra* bahasa Python sebesar 0.0006108666668422913 detik. Dengan demikian, diperoleh bahwa waktu eksekusi program lebih cepat algoritma *Greedy* daripada algoritma *Dijkstra*.

Pada parameter kompleksitas dan *sequence* program pada algoritma *Greedy* sebesar 43 baris dan mengambil urutan dengan cepat atau instan, sedangkan kompleksitas dan sequence program pada algoritma *Dijkstra* sebesar 69 baris dan mengambil urutan dengan langkah lebih panjang dan dicek secara berkala agar mendapatkan hasil yang benar-benar minimum, oleh karena itu kompleksitas dan sequence program pada algoritma *Greedy* cepat lebih pendek dan singkat daripada algoritma *Dijkstra*.

Parameter terakhir, yaitu hasil yang diperoleh pada algoritma *Greedy* mendapatkan hasil rute terpendek sebesar 129 (km), dengan rute-rute yang dilalui : Tuban - Widang - Babat - Pucuk - Lamongan - Kembangbahu - Mantup-Bendjing – Surabaya. Sedangkan, pada algoritma *Dijkstra* mendapatkan hasil rute terpendek sebesar 105 (km), dengan rute-rute yang dilalui : Tuban - Widang - Babat- Pucuk - Lamongan - Duduk Sampeyan - Ambengambeng - Kebomas-Surabaya. Sebagai itu, diperoleh selisih perbandingan jarak 24 (km) dari kedua algoritma tersebut. Maka dari itu, hasil jarak yang ditempuh oleh algoritma *Dijkstra* ternyata lebih unggul dari algoritma *Greedy*.

Dengan demikian, dalam hasil pencarian rute terpendek lebih bagus menggunakan algoritma *Dijkstra* daripada

algoritma *Greedy*, walaupun waktu, kompleksitas, dan sequence program lebih lama, dengan jalur-jalur terpendek yang dilalui sebagai berikut: Tuban - Widang - Babat - Pucuk - Lamongan - Duduk Sampeyan - Ambengambeng - Kebomas - Surabaya (dengan bobot 105 km).

IV. KESIMPULAN DAN SARAN

Dalam permasalahan penentuan rute terpendek dari kabupaten Tuban ke kota Surabaya, dapat diselesaikan dengan kedua algoritma, yaitu: algoritma *Greedy* dan algoritma *Dijkstra*. Algoritma *Greedy* dan algoritma *Dijkstra*, sama-sama merupakan suatu algoritma yang dapat menyelesaikan jalur terpendek dengan cara dan langkah masing-masing.

Pada peneliti terdahulu mengenai pencarian rute terpendek dari pendonor darah terdekat, menyimpulkan hasil pengujian pertumbuhan waktu kode program didapatkan bahwa nilai fungsi *n* dari algoritma *Dijkstra* lebih besar daripada algoritma *Greedy*, sehingga waktu eksekusi dari algoritma *Dijkstra* lebih cepat dibandingkan dengan algoritma *Greedy*. [15]. Namun pada penelitian ini, pengujian parameter pertama menghitung waktu eksekusi kode program yang dijalankan melalui bahasa *Python* ternyata membuktikan bahwa algoritma *Greedy* lebih cepat daripada algoritma *Dijkstra* dengan pengujian 3 kali dimana sebesar 0.00023706666669524405 detik, dan pada algoritma *Dijkstra* 0.0006108666668422913 detik.

Parameter kedua mengenai kompleksitas kode program bahasa *Python* yang didapatkan algoritma *Greedy* sebanyak 43 baris, dan algoritma *Dijkstra* sebanyak 69 baris.

Parameter ketiga melakukan pengujian urutan atau *sequence* program yang dijalankan melalui program bahasa *Python.* Hasil algoritma *Greedy* pada urutannya hanya memperhatikan langkah panjang rute terpendek di depannya, akan tetapi algoritma *Dijkstra* pada urutannya selalu memperhatikan keseluruhan rute yang dilewati atau dilalui, sehingga mendapatkan bobot paling kecil dari total keseluruhan. Sehingga urutan pengujian perbandingan ini lebih cepat algoritma *Greedy* daripada algoritma *Dijkstra*.

Parameter keempat terkait hasil jarak terpendek, baik dalam pengujian secara dasar atau manual, maupun dilakukan oleh kode program bahasa *Python.* Hasil rute terpendek yang diperoleh ternyata algoritma *Dijkstra* sebesar 105 (km) lebih pendek daripada algoritma *Greedy* sebesar 129 (km). Rute-rute yang dilalui algoritma *Dijkstra* adalah Tuban - Widang - Babat - Pucuk - Lamongan - Duduk Sampeyan - Ambengambeng - Kebomas - Surabaya .

Dengan demikian, melalui perbandingan parameter algoritma yang diuji, dapat disimpulkan bahwa parameter waktu eksekusi, kompleksitas, dan urutan pengujian program lebih baik dilakukan oleh algoritma *Greedy*. Akan tetapi, pada pengujian parameter hasil luaran jarak terpendek yang ditempuh dalam hasil rute paling, ternyata lebih baik algoritma *Dijkstra* daripada algoritma *Greedy*.

Melalui penelitian ini dapat direkomendasikan penggunaan algoritma *Greedy* dan algoritma *Dijkstra* dalam pencarian rute terpendek, selain itu dapat menggunakan algoritma-algoritma pencarian rute terpendek lain [19][20] untuk menyelesaikan kasus pencarian rute terpendek dari kabupaten Tuban ke kota Surabaya atau kasus pencarian rute lainnya.

DAFTAR PUSTAKA

[1] M. Syauqi, H. Ardani, M. A. Yaqin, and M. H. Suhartono, "Implementasi Graph Database untuk Menentukan Rute Perjalanan Transportasi Umum Clustering View project Fraud Detection View project," no. February, 2019, [Online]. Available: https://www.researchgate.net/publication/330872174.

[2] C. Cerrone, R. Cerulli, and B. Golden, "Carousel *Greedy*: A generalized *Greedy* algorithm with applications in optimization," *Comput. Oper. Res.*, vol. 85, no. May 2018, pp. 97–112, 2017, doi: 10.1016/j.cor.2017.03.016.

[3] E. N. Hayati and A. Yohanes, "Pencarian Rute Terpendek Menggunakan Algoritma *Greedy*," *Semin. Nas. IENACO*, pp. 2337–4349, 2014.

[4] D. Grace, M. S. Tanciga, and Nurdin, "Sistem Informasi Letak Geografis Penentuan Jalur Tercepat Rumah Sakit Di Kota Palu Menggunakan Algoritma *Greedy* Berbasis Web," *J. Elektron. Sist. Inf. dan Komput.*, vol. 4, no. 2, pp. 59–76, 2018.

[5] R. B. Oktaviandi, M. S. T. Hadi, A. G. Santoso, and N. El Maidah, "Perbandingan Algoritma Genetika dengan Algoritma *Greedy* Untuk Pencarian Rute Terpendek," *INFORMAL Informatics J.*, vol. 3, no. 1, p. 6, 2019, doi: 10.19184/isj.v3i1.9847.

[6] A. M. Herli, I. K. Raharjana, and P. Soeparman, "Sistem Pencarian Hotel Berdasarkan Rute Perjalanan Terpendek Dengan Mempertimbangkan Daya Tarik Wisata Menggunakan Algoritma *Greedy*," *J. Inf. Syst. Eng. Bus. Intell.*, vol. 1, no. 1, p. 9, 2015, doi: 10.20473/jisebi.1.1.9-16.

[7] Z. Jiang, V. Sahasrabudhe, A. Mohamed, H. Grebel, and R. Rojas-Cessa, "*Greedy* algorithm for minimizing the cost of routing power on a digital microgrid," *Energies*, vol. 12, no. 16, pp. 1–19, 2019, doi: 10.3390/en12163076.

[8] E. Darnila, M. Ula, and C. D. A. Soraya, "Optimasi Kelayakan Kondisi Pembangunan Jalan di Kota Lhokseumawe Menggunakan Algoritma *Greedy*," *JUKI J. Komput. dan Inform.*, vol. 1, pp. 9–14, 2019, [Online]. Available: https://www.ioinformatic.org/index.php/juki/article/view/3%0Ahttps://www.ioinformatic.org/index.php/JUKI/article/download/3/9.

[9] N. A. Sudibyo, P. E. Setyawan, and Y. P. S. R. Hidayat, "Implementasi Algoritma *Dijkstra* dalam Pencarian Rute Terpendek Tempat Wisata di Kabupaten Klaten," *Riemann Res. Math. Math. Educ.*, vol. 2, no. 1, pp. 1–9, 2020, doi: 10.38114/riemann.v2i1.49.

[10] A. R. Fadillah, P. Studi, T. Informatika, and R. Terpendek, "Sistem Pencarian Lokasi Hotel Berdasarkan Rute Terpendek Untuk Pengunjung Objek Wisata Alam Di Kota Medan Menggunakan Algoritma *Dijkstra*," vol. 6, no. 1, pp. 106–111, 2019.

[11] A. Kejriwal and P. Aqsa, "Graph Theo ry and *Dijkstra* ' s Algorithm : A solution for Mumbai ' s BEST buses," *Int. J. Eng. Sci.*, vol. 8, no. 10, pp. 40–47, 2019, doi: 10.9790/1813-0810014047www.theijes.com.

[12] A. Cantona, F. Fauziah, and W. Winarsih, "Implementasi Algoritma *Dijkstra* Pada Pencarian Rute Terpendek ke Museum di Jakarta," *J. Teknol. dan Manaj. Inform.*, vol. 6, no. 1, 2020, doi: 10.26905/jtmi.v6i1.3837.

[13] P. Tirastittam and P. Waiyawuththanapoom, "Public Transport Planning System by *Dijkstra* Algorithm: Case Study Bangkok Metropolitan Area," vol. 8, no. 1, pp. 54–59, 2014, [Online]. Available: http://waset.org/publications/9997113/public-transport-planning-system-by-*Dijkstra*-algorithm-case-study-bangkok-metropolitan-area.

[14] S. Orhani, "Finding the Shortest Route for Kosovo Cities Through

Dijkstra ' s Algorithm," *MIDDLE Eur. 247 Sci. Bull.*, no. June, 2022.

[15] A. Subagio, B. Rahayudi, and M. A. Fauzi, "Analisis Performansi Algoritma *Greedy* Best First Search dan *Dijkstra* Pada Aplikasi Pencarian Jalur Pendonor Darah Terdekat," *Pengemb. Teknol. Inf. dan Ilmu Komput.*, vol. 3, no. 1, pp. 515–520, 2019.

[16] R. Y. Bunaen Maria, Pratiwi Hanna, "Penerapan algoritma *Dijkstra* untuk menentukan rute terpendek dari pusat kota surabaya ke tempat bersejarah," vol. 4, no. 1, pp. 213–223, 2022, [Online]. Available: http://www.jurnal.unidha.ac.id/index.php/jteksis/article/view/407.

[17] S. Idwan and W. Etaiwi, "*Dijkstra* algorithm heuristic approach for large graph," *J. Appl. Sci.*, vol. 11, no. 12, pp. 2255–2259, 2011, doi: 10.3923/jas.2011.2255.2259.

[18] J. Sauwani, V. N. Putra, and H. Agung, "Implementasi Algoritma Djikstra Untuk Menentukan Lokasi Dan Jarak Tempuh Terpendek Kampus It Di Jakarta," *J. Inform.*, vol. 6, no. 1, pp. 29–36, 2019, doi: 10.31311/ji.v6i1.4723.

[19] M. A. Arsyad, D. Supriyadi, A. Veronica, L. N. Hidayah, and D. P. Pratiwi, "Penerapan Algoritma A Star Untuk Pencarian Rute Terpendek Puskesmas Rawat Inap Di Banyumas," *Conf. Electr. Eng. Telemat. Ind. Technol. Creat. Media 2019*, pp. 74–82, 2019, doi: 10.29408/geodika.v4i1.2068.

[20] R. Rizky, T. Hidayat, A. H. Nugroho, and Z. Hakim, "Implementasi Metode A*Star Pada Pencarian Rute Terdekat Menuju Tempat Kuliner di Menes Pandeglang Banten," *Geodika J. Kaji. Ilmu dan Pendidik. Geogr.*, vol. 4, no. 1, pp. 85–94, 2020, doi: 10.29408/geodika.v4i1.2068.