Implementasi Algoritma Dijkstra untuk Mencari Rumah Kost Terdekat di Kodya Yogyakarta Berbasis Android

Bhernardin Erryco Gagah Pattriskak[1], R.Gunawan Santosa[2], Antonius Rachmat Chrismanto[3]
Program Studi Informatika, Universitas Kristen Duta Wacana Yogyakarta
Jl. Dr. Wahidin Sudirohusodo No.5-25, Yogyakarta
[1]bhernardin.erryco@ti.ukdw.ac.id
[2]gunawan@staff.ukdw.ac.id
[3]anton@ti.ukdw.ac.id

Abstract*— The problem that is often faced by newcomer university students is the difficulty in finding a place to live, one of these places is a boarding house. However, some students have considerations in choosing their boarding, one of those is the distance of boarding from some universities. This research will try to implement Dijkstra for searching the nearest boarding on Android, this application will have the shortest path search, maximum price and distance filter. Dijkstra is the algorithms for finding the smallest distance and shortest path. The results of the closest boarding distance and the shortest path will be compared with the results from Google Maps, the process of searching time also recorded. From this research, Dijkstra algorithm was successfully implemented to search the nearest boarding, search based on filter, and the shortest path search on Android. In testing the nearest boarding distance, results obtained that the Dijkstra in finding the smallest distance is close to the data from Google Maps with an average distance difference of 6.1 m. This research also found that the farther the distance of the boarding being searched using the Dijkstra, the longer the search time due to the more vertices being worked on.

Intisari— Permasalahan yang sering dihadapi oleh para mahasiswa terutama mahasiswa pendatang adalah sulitnya untuk memperoleh tempat tinggal, salah satunya adalah rumah kost. Namun beberapa mahasiswa memiliki pertimbangan tertentu dalam memilih kos-kosannya salah satunya adalah jarak tempat kost dari universitas tertentu. Penelitian ini akan mencoba mengimplementasikan algoritma Dijkstra untuk melakukan pencarian kost terdekat di Android, pada aplikasi ini juga akan terdapat fitur pencarian jalur terpendek, filter harga maksimal, dan filter jarak maksimal. Algoritma Dijkstra merupakan salah satu algoritma untuk melakukan pencarian jarak terkecil dan jalur terpendek. Hasil pencarian jarak kost terdekat dan jalur terpendek akan dibandingkan dengan hasil dari Google Maps, proses lamanya waktu pencarian pada sistem juga akan dicatat. Dari penelitian ini algoritma Dijkstra berhasil diimplementasikan untuk melakukan pencarian kost terdekat, pencarian kost berdasarkan filter, dan pencarian jalur terpendek di Android. Pada pengujian pencarian jarak kost terdekat didapatkan hasil bahwa algoritma Dijkstra dalam mencari jarak terkecil hampir mendekati data dari Google Maps dengan rata-rata selisih jarak sebesar 6,1 m. Pengujian ini juga menemukan bahwa semakin jauh jarak dan jalur rumah kost yang dicari menggunakan algoritma Dijkstra maka akan semakin lama waktu pencariannya dikarenakan semakin banyaknya vertek yang dikerjakan.

***Kata Kunci*— Dijkstra, Android, pencarian kost terdekat, pencarian jalur terpendek**

I. PENDAHULUAN

Permasalahan yang sering dihadapi oleh para mahasiswa terutama mahasiswa pendatang adalah sulitnya untuk memperoleh tempat tinggal, salah satunya adalah rumah kost. Kost merupakan jasa penyewaan yang menawarkan tempat tinggal berupa sebuah kamar dengan sejumlah pembayaran tertentu untuk setiap periode tertentu. Kost merupakan pilihan tempat tinggal yang umum dan paling diminati dikalangan mahasiswa.

Beberapa mahasiswa memilih untuk tinggal di rumah kost mungkin dengan alasan mahasiswa tersebut ingin hidup mandiri atau mungkin mereka hanya sekedar ingin mencari tempat tinggal yang dekat dengan area kampus. Disamping itu harga sewa kost lebih terjangkau untuk kalangan mahasiswa daripada menyewa atau mengkontrak rumah. Beberapa mahasiswa juga memiliki pertimbangan tertentu dalam memilih kost-kostannya salah satunya adalah jarak tempat kost dari universitas tertentu, misalkan seorang mahasiswa yang ingin mencari tempat kost yang dekat dengan UKDW.

Dari masalah-masalah tersebut peneliti ingin membuat sistem aplikasi pencarian rumah kost dalam bentuk *maps* yang mampu mencari dan merangking rumah kost terdekat dari posisi saat ini serta dapat menampilkan jalur terpendek menuju rumah kost tersebut dengan menggunakan algoritma Dijkstra. Pencarian rumah kost hanya pada lingkup Kota Madya Yogyakarta dan aplikasi ini akan berbasis Android. Salah satu alasan mengapa peneliti menggunakan basis *smartphone* Android untuk pembuatan aplikasi ini adalah karena pada smartphone Android terdapat fitur GPS. Diharapkan nantinya mahasiswa dapat menemukan rumah kost terdekat dari posisinya saat ini dan dapat menampilkan jalur terpendek menuju rumah kost tersebut.

Pada penelitian [1] Christian membuat jurnal tentang studi literatur perbandingan algoritma Dijkstra dan Bellman-Ford dalam pencarian jarak terdekat. Pada jurnal ini terdapat penjalasan mengenai cara kerja algoritma Dijkstra, disimpulkan bahwa algoritma Dijkstra lebih cepat dalam proses pencarian jarak terdekat dibandingkan dengan Bellman-Ford. Begitu pula dengan penelitian [2] yang membuat studi literatur perbandingan algoritma Dijkstra dengan Floyd Warshall untuk menentukan jalur terpendek,

disimpulkan pula bahwa algoritma Dijkstra lebih cepat daripada algoritma Floyd dalam pencarian jalur terpendek.

Penelitian [3] merancang aplikasi untuk menentukan jalur terpendek rumah sakit di purbalingga dengan algoritma Dijkstra. Didalam jurnal ini peneliti mengimplementasikan algoritma Dijkstra untuk melakukan pencarian jalur terpendek dari *node* awal menuju *node* target yang telah ditentukan.

Penelitian [4] mengimplementasikan algoritma Dijkstra untuk melakukan pencarian hotel di kota Semarang. Pada aplikasinya mereka menetapkan GPS sebagai *node source* dan hotel yang dipilih sebagai *node* target dengan demikian jalur menuju hotel akan terbentuk. Hampir sama dengan penelitian [5] yang membuat aplikasi Android untuk mencari sekolah pesantren terdekat dari posisi GPS. Namun pada penelitian [5] aplikasi sudah dilengkapi dengan info rute jalan.

Penelitian [6] membuat aplikasi Android untuk mencari jalur terpendek menuju lokasi tempat futsal menggunakan algoritma Dijkstra. Dalam pengaplikasiannya mereka menggunakan '*onMapClick*' sebagai *node* awal dan lokasi futsal dalam bentuk menu dropdown yang dapat dipilih sebagai *node* akhir kemudian sistem akan mencari jalur terpendeknya.

Penelitian [7] membuat aplikasi pencarian jalur terpendek menuju masjid yang dipilih menggunakan algoritma Dijkstra dan berbasis Android. Dari penelitiannya mereka menyimpulkan bahwa pencarian jalur terpendek menggunakan algoritma Dijkstra dapat di implementasikan pada perangkat Android.

Dari hasil pemaparan tinjauan pustaka diatas dapat dinyatakan bahwa penelitian implementasi algoritma Dijkstra untuk melakukan pencarian rumah kost terdekat di Kodya Yogyakarta dalam bentuk maps dan berbasis Android belum pernah dilakukan sebelumnya walau memiliki beberapa kedekatan dan kesamaan dengan penelitian yang telah ada sebelumnya.

Peneliti ingin membuat aplikasi pencarian rumah kost terdekat berbasis Android dengan menggunakan algoritma Dijkstra karena berdasarkan tinjauan pustaka diatas algoritma Dijkstra dapat di implementasikan untuk mencari jalur terpendek dalam sebuah peta serta algoritma Dijkstra memiliki kompleksitas waktu yang lebih kecil dibandingkan dengan algoritma Bellman-Ford atau Floyd. Graf yang digunakan peneliti nantinya tidak memiliki *loop* dan sisi negatif sehingga algoritma Dijkstra lebih cocok di implementasikan untuk kasus pencarian rumah kost terdekat.

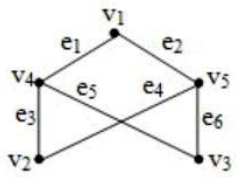
II. LANDASAN TEORI

Ilmu dan metode yang diterapkan dalam penelitian ini antara lain Teori Graf, jenis-jenis graf, Android, dan Dijkstra.

A. *Teori Graf*

Teori graf adalah cabang ilmu matematika yang mempelajari mengenai terminologi dari graf, jenis dan sifatnya. Dalam teori graf, sebuah graf G(V,E) direpresentasikan dalam bentuk *node* atau simpul atau verteks V={v1,v2,…} dimana terdapat garis-garis atau *edge* E={e1,e2,…} yang menghubungkan verteks tertentu. Garis tersebut menggambarkan keterhubungan antara dua buah verteks yang dihubungkannya, garis dalam graf disebut sebagai sisi atau *edge* [8], kemudian adapula istilah yaitu derajat (*degree*), derajat d(v) adalah jumlah sisi yang bersisian dengan simpul v.

Gambar 1 merupakan contoh bentuk dari sebuah graf yang memiliki verteks V = {v1,v2,v3,v4,v5} dan memiliki *edge* E = {e1,e2,e3,e4,e5,e6}, kemudian verteks v4 memiliki derajat d(v4) = 3.

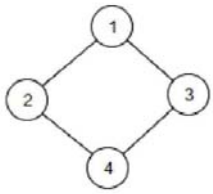


Gambar 1 Contoh Bentuk Graf

B. *Jenis-Jenis Graf*

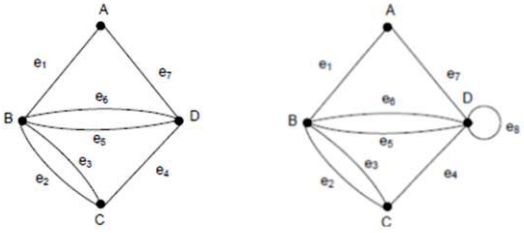
Berdasarkan sudut pandang pengelompokannya graf dibagi menjadi beberapa jenis. Pengelompokan ini dilihat berdasarkan ada tidaknya *edge* ganda, bobot pada *edge*, atau berdasarkan panah pada *edge* [9]. Berdasarkan *edge* pada graf, maka graf dikelompokkan menjadi:

- Graf sederhana (*simple graph*)
  Graf yang tidak memiliki *edge* ganda dan atau *loop*. Pada *simple graph*, *edge* tidak terurut. Jadi *edge* berhubungan (u,v)=(v,u). Contoh *simple graph* ada pada Gambar 2.



Gambar 2 Bentuk Simple Graph

- Graf tidak sederhana (*unsimple graph*)
  Graf yang memiliki *edge* ganda atau memiliki *loops*. Graf ini ada dua jenis yaitu graf ganda (*multigraph*) dan graf semu (*pseudograph*). Graf ganda mengandung *edge* ganda. *Edge* yang terhubung antar verteks bisa lebih dari dua. Graf semu adalah graf yang mengandung *loop* (gelang), bisa juga memiliki *loop* dan *edge* ganda. Graf tidak sederhana dapat dilihat pada Gambar 3.

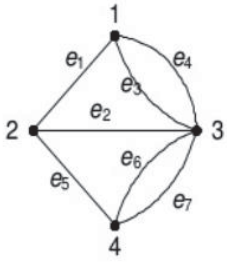


Gambar 3 Graf Ganda (kiri) dan Pseudograph (kanan)

Selain berdasarkan *edge* yang ganda atau *loop*, graf dapat memiliki panah atau arah. Berdasarkan panah atau arah pada *edge* maka graf dibedakan menjadi dua yaitu:

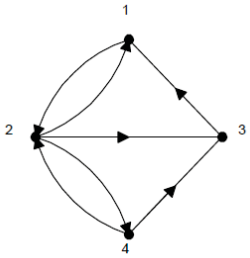
JUTEI Edisi Volume.4 No.1 April 2020
ISSN 2579-3675, e-ISSN 2579-5538
DOI 10.21460/jutei.2020.14.193

- Graf tidak berarah (*undirected graph*)
  Graf yang *edge* tidak mempunyai panah atau arah. Pada graf ini, *edge* berhubungan (u,v)=(v,u). Gambar 4 merupakan gambar dari graf tidak berarah, graf tersebut juga merupakan graf ganda.



Gambar 4 Bentuk Graf Tidak Berarah

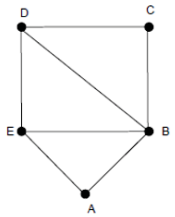
- Graf berarah (*directed graph*)
  Graf yang setiap *edge* memiliki panah atau arah. *Edge* berarah pada graf ini biasanya disebut *arc* (busur). Pada graf ini, (u,v) dan (v,u) merupakan dua *edge* berhubungan yang berbeda. Gambar 5 merupakan bentuk dari graf berarah, *edge* yang berhubungan memiliki panah atau arah.



Gambar 5 Bentuk Graf Berarah

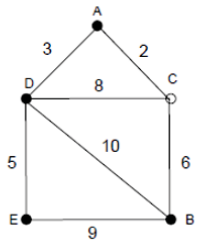
Pengelompokkan graf juga berdasarkan bobot yang dimilikinya. Berdasarkan bobot, graf dibagi menjadi dua yaitu:

- Graf tidak berbobot (*unweighted graph*)
  Graf yang *edge*-nya tidak memiliki bobot. Gambar 6 merupakan bentuk graf tidak berbobot, pada setiap *edge* yang berhubungan tidak memiliki nilai bobot.



Gambar 6 Bentuk Graf Tidak Berbobot

- Graf berbobot (*weighted graph*)
  Graf yang masing-masing *edge*-nya memiliki bobot. Bobot pada setiap *edge* dapat berbeda-beda. Gambar 7 merupakan bentuk graf berbobot.



Gambar 7 Bentuk Graf Berbobot

Pada penelitian pencarian rumah kost terdekat ini peta akan dimodelkan dalam bentuk graf ganda yang berarah dan mempunyai bobot. Menggunakan graf ganda berarah karena pada pencarian jarak rumah kost terdekat akan menggunakan rute pejalan kaki yang tidak memperhatikan kondisi jalan satu arah. Pada graf ini juga akan memiliki bobot yang merupakan jarak dari setiap jalannya.

C. *Android*

Android merupakan sistem operasi berbasis Linux yang dikembangkan untuk perangkat *mobile*. Menurut Supardi [10], Android merupakan sebuah sistem operasi perangkat *mobile* dengan platform terbuka (*open source*) bagi para pengembang untuk membuat aplikasi. Android pada awalnya adalah milik Android Inc., namun pada tahun 2005 dibeli oleh Google. Versi-versi Android yang telah dirilis memiliki keunikan dalam penamaan *codename*nya, yaitu pasti pada setiap versinya diberikan *codename* nama makanan. Pada tahun 2018, versi terendah Android yang masih memiliki tingkat distribusi atau pengguna yang cukup banyak adalah Android versi 4.4 (*kitkat*) yang memiliki persentase 7,6%.

D. *Algoritma Dijkstra*

Algoritma Dijkstra merupakan salah satu algoritma yang digunakan untuk mencari lintasan terpendek dari suatu tempat ke tempat lainnya dalam sebuah graf berarah atau tidak berarah yang memiliki bobot khususnya dengan bobot positif [11]. Prinsip dari algoritma Dijkstra adalah mendapatkan bobot terkecil dari setiap titik berhubungan yang dilaluinya sampai pada titik tujuan dan kemudian berhenti, dengan begitu didapatkan jarak dan lintasan terpendek dari titik awal sampai ke titik tujuan. Syarat algoritma ini adalah bobot sisinya yang harus non-negatif. Pseudocode algoritma Dijkstra dapat dilihat pada Gambar 8 yang dikutip dari buku *Introduction to Algorithms* [12].

```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  S = ∅
3  Q = G.V
4  while Q ≠ ∅
5      u = EXTRACT-MIN(Q)
6      S = S ∪ {u}
7      for each vertex v ∈ G.Adj[u]
8          RELAX(u, v, w)
```

Gambar 8 Pseudocode Algoritma Dijkstra

Pseudocode algoritma Dijkstra pada Gambar 8 adalah langkah-langkah algoritma Dijkstra dalam mencari bobot terkecil dan jalur terpendek dari vertek *source* ke semua

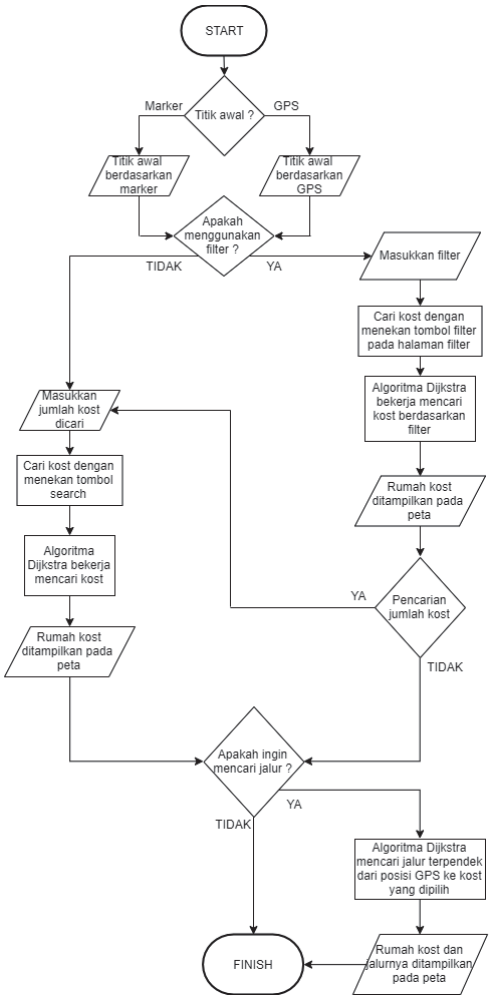
vertek. Baris 1 merupakan inisialisasi vertek *source* (s) pada graf G. Variabel S pada baris 2 merupakan kumpulan (*set*) dari vertek yang telah dikunjungi (*visited*), pada awal jalannya algoritma variabel S di-*set* masih kosong. Pada baris 3, variabel Q merupakan kumpulan vertek (V) pada graf G yang belum dikerjakan. *While* pada baris 4-8 akan terus berjalan sampai semua vertek sudah dikerjakan. Pada baris 5, vertek saat ini (*u*) adalah *element* dari Q dengan bobot (*distance*) yang paling minimal. Kemudian pada baris 6 vertek saat ini (*u*) akan dimasukan kedalam *list* vertek yang telah dikunjungi (S). Baris 7-8 merupakan perulangan yang dilakukan algoritma Dijkstra saat mengecek tetangga (*adjacent*) dari vertek saat ini (*u*) dalam graf G lalu pada baris 8 akan dilakukan *update* (*relax*) bobot jika bobot baru lebih kecil.

III. METODOLOGI PENELITIAN

Penelitian dilakukan dengan cara pembuatan aplikasi berbasis Android untuk mencari rumah kost terdekat. Program diimplementasikan menggunakan Bahasa Pemrograman Java dan menggunakan IDE Android Studio. Pada pemrograman pemilihan titik awal, konversi graf, dan algoritma Dijkstra penulis banyak belajar dari *github* milik mokox [13]. Sistem yang dibuat penulis pencarian rumah kost terdekat mampu berdasarkan filter jarak maksimal, harga maksimal, dan jumlah kost yang dicari. Aplikasi juga dapat mencari jalur terpendek dari posisi saat ini menuju rumah kost yang dipilih. Hasil rumah kost dapat ditampilkan di peta dan di halaman *list* kost terdekat.

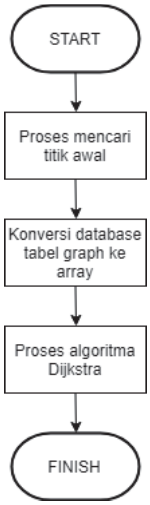
Ketika *user* ingin mencari rumah kost yang pertama dilakukan adalah menentukan titik awal. Jika *user* membuat *marker* pada peta berarti titik awal berdasarkan *marker* namun jika *marker* tidak dibuat berarti titik awal berdasarkan posisi GPS. *User* dapat menggunakan filter untuk proses pencarian. Filternya yaitu berdasarkan harga maksimal atau jarak maksimal dalam satuan meter. Jika pencarian kost menggunakan filter maka pencarian kost akan berdasarkan filter yang ada. Filter dapat mencari kost dengan batasan maksimal harga atau jarak maksimal atau keduanya.

Setelah kost yang didapat sesuai dengan kost yang dicari maka algoritma akan berhenti dan rumah kost akan ditampilkan pada peta dalam bentuk *marker* dan pada halaman *list* kost. Jika *user* ingin mencari jalur menuju suatu kost maka *user* perlu memilih *marker* rumah kost tersebut lalu pilih tombol *direction*. Kemudian algoritma Dijkstra akan bekerja lagi dari titik awal GPS menuju titik lokasi rumah kost yang dipilih setelah itu jalur terpendeknya akan digambar dan ditampilkan di peta. Gambar 9 merupakan flowchart cara kerja aplikasi secara umum.



Gambar 9 Flowchart Aplikasi Secara Umum

Gambar 10 merupakan flowchart sistem. Setelah *user* menekan tombol "*search*" sistem akan melakukan pencarian titik awal, kemudian konversi data dari basis data ke array 2 dimensi, baru setelah itu algoritma Dijkstra dijalankan.



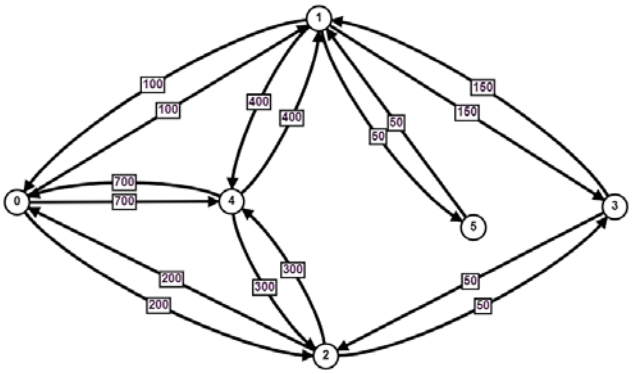
Gambar 10 Flowchart Sistem

Pada Gambar 10 setelah menekan tombol "*search*" sistem akan melakukan pencarian titik awal. Pada proses pemilihan titik awal akan dihitung jarak dari posisi yang dipilih user ke setiap verteks jalur menggunakan fungsi distanceTo setelah dapat jaraknya pilih jarak yang paling

JUTEI Edisi Volume.4 No.1 April 2020
ISSN 2579-3675, e-ISSN 2579-5538
DOI 10.21460/jutei.2020.14.193

minimum. Verteks dengan jarak paling minimum itulah yang akan menjadi verteks awal. Verteks yang dibuat menjadi verteks awal adalah verteks persimpangan pada graf yang dibuat. Pada proses pemilihan titik awal sistem tidak bisa membuat vertek baru di luar jalur atau di antara jalur. Setelah proses pencarian titik awal berhasil maka langkah selanjutnya adalah konvesi graf ke array.

Sebelum algoritma Dijkstra dijalankan data graf yang asalnya dari basis data SQL dirubah menjadi array 2 dimensi. Gambar 11 adalah gambar contoh graf yang akan dirubah jadi array 2 dimensi.



Gambar 11 Contoh Bentuk Graf

TABEL I
CONTOH BENTUK DATA DALAM ARRAY 2 DIMENSI

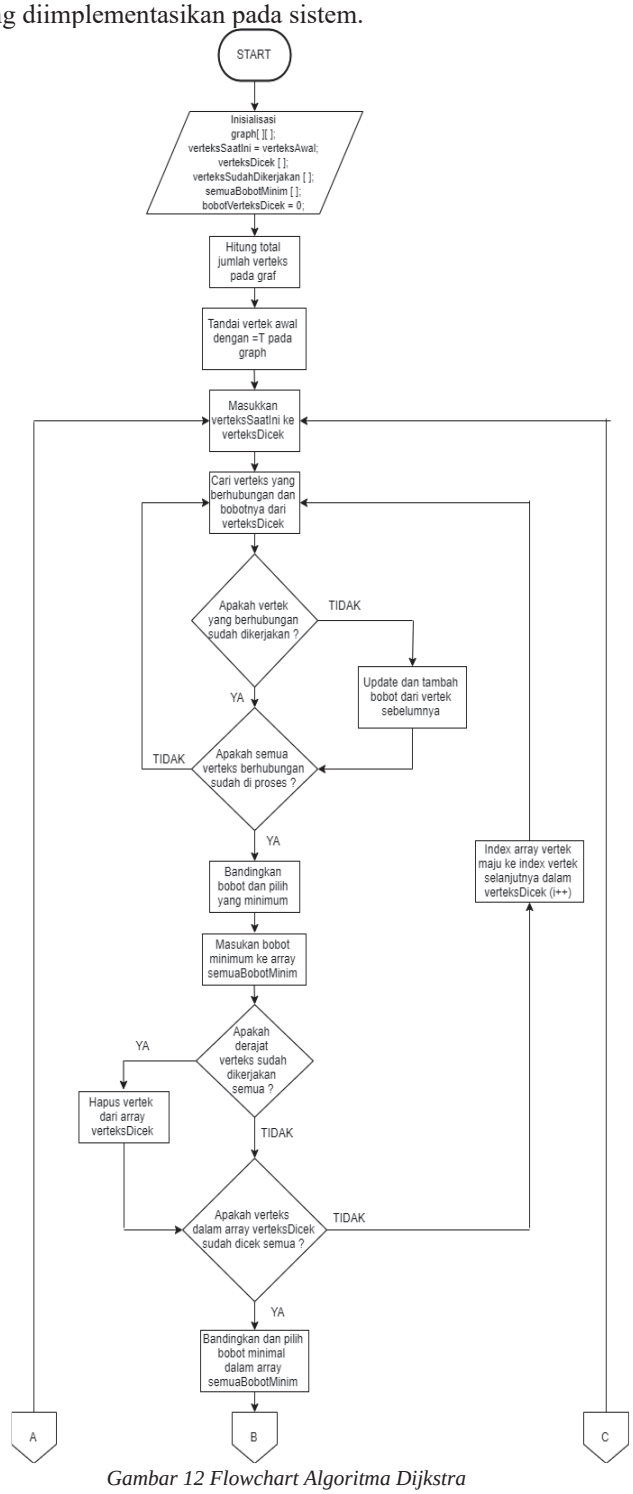
	0	1	2	3
0	1=100	2=200	4=700	null
1	0=100	3=150	4=400	5=50
2	0=2000	3=50	4=300	null
3	1=150	2=50	null	null
4	0=700	1=400	2=300	null
5	1=50	null	null	null

Tabel I adalah contoh bentuk data setelah dikonversi dari basis data SQL ke array 2 dimensi. Baris merupakan vertek sedangkan kolom adalah derajat (*degree*) *node*. Contohnya pada baris ke 0 kolom ke 0 berisi 1=100 kolom ke 1 berisi 2=200 dan kolom 2 berisi 4=700 setelah itu *null* pada kolom 3 yang artinya vertek 0 berhubungan dengan 3 vertek yaitu vertek 1 dengan bobot 100, verteks 2 dengan bobot 200 dan verteks 4 dengan bobot 700 selanjutnya pada kolom 3 *null* karena vertek 0 hanya berhubungan dengan 2 vertek yang artinya persimpangan pada verteks 0 merupakan pertigaan.

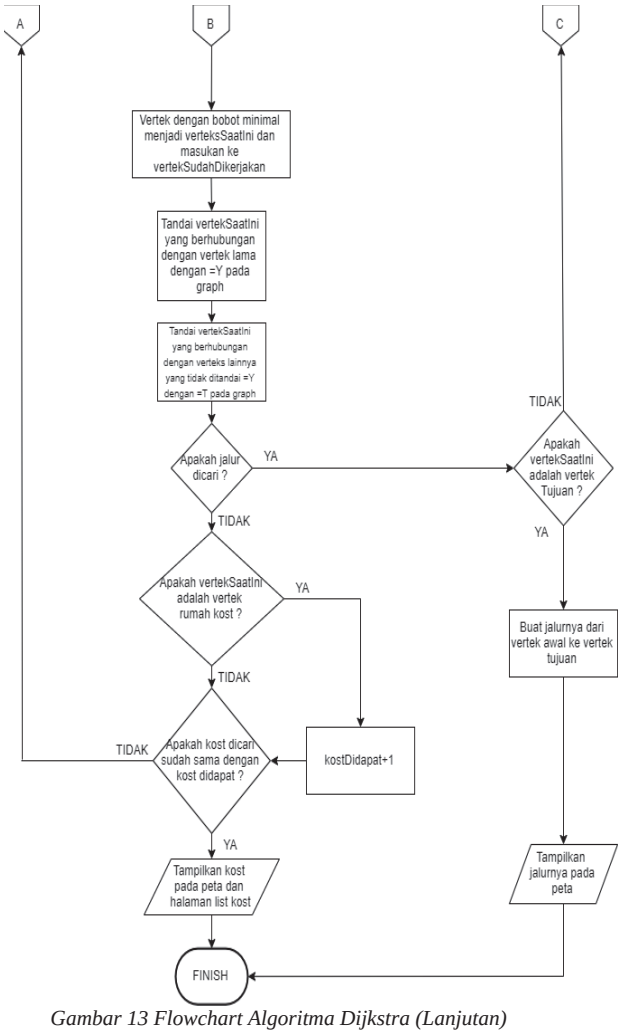
Setelah selesai proses pemilihan titik awal dan konversi graf ke array 2 dimensi selanjutnya adalah proses algoritma Dijkstra. Pertama adalah mengecek setiap verteks berhubungan, jika semua vertek berhubungan telah di kerjakan maka vertek tersebut akan dihapus dari array verteksDicek agar tidak dicek lagi.

Setelah dicek *node* yang berhubungan dan jalur bobotnya kemudian dibandingkan dan dipilih yang paling kecil. Setelah semua jalur berhubungan diperiksa bobot jalur tersebut dimasukan dalam array semuaBobotMinim, nantinya semua bobot ini akan dibandingkan lagi dan dicari yang paling minimal lalu tandai jalur tersebut karena sudah dikerjakan dan sudah didapat bobot minimalnya.

Node yang ditandai dengan =Y artinya jalur tersebut merupakan jalur yang paling minimal dan nantinya dipakai untuk buat jalur, sedangkan yang ditandai dengan =T artinya jalur *node* bukan jalur dengan bobot minimum dan *node* tersebut tidak boleh di*update* lagi bobotnya karena sudah minimal. Kemudian jika kita hanya mencari rumah kost maka kondisi berhenti algoritma bergantung pada kondisi kost yang didapat saat proses pencarian. Namun jika pencarian jalur maka kondisi berhenti adalah jika verteksSaatIni sudah sampai vertek tujuan. Gambar 12 dan Gambar 13 merupakan gambar flowchart algoritma Dijkstra yang diimplementasikan pada sistem.

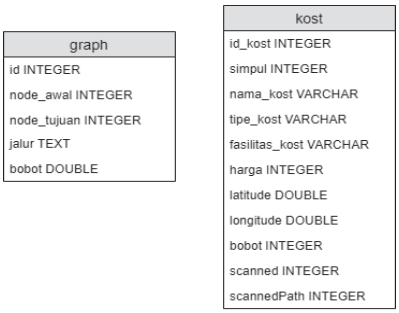


Gambar 12 Flowchart Algoritma Dijkstra



Gambar 13 Flowchart Algoritma Dijkstra (Lanjutan)

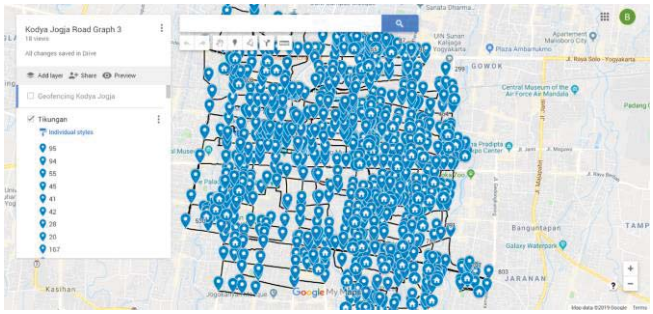
Untuk basis data yang digunakan pada aplikasi, penulis merancang sebuah basis data dengan membuat dua tabel sebagai berikut:



Gambar 14 Rancangan Basis Data

Gambar 14 merupakan diagram rancangan basis data. Basis data menggunakan 2 tabel yaitu tabel "graph" dan tabel "kost". Tabel "graph" berisi vertek, jalur, dan bobot yang terdiri dari id (PK), node_awal, node_tujuan, jalur, dan bobot. Jalur graf yang dimasukkan adalah jalur bolak balik karena graf pada sistem ini semua jalur adalah ganda bolak balik yang diumpamakan seperti jalur pejalan kaki. Kemudian untuk tabel "kost" berisi id_kost (PK), simpul, nama_kost, tipe_kost, fasilitas_kost, harga, latitude, longitude, bobot, scanned, dan scannedPath. Scanned pada tabel kost nantinya akan digunakan untuk proses pengecekkan saat pencarian rumah kost dan scannedPath digunakan untuk proses pengecekkan saat pencarian jalur terpendek.

Data graf Kodya Yogyakarta yang digunakan dibuat manual titik dan jalurnya menggunakan Google MyMaps. Titik-titik yang dibuat melambangkan persimpangan dan rumah kost lalu jalur adalah jalan. Titik-titik dan bobot jalur nantinya akan dimasukan kedalam basis data. Total terdapat 924 titik persimpangan, 72 titik rumah kost dan 1478 jalur yang dibuat pada Google MyMaps. Gambar 15 merupakan gambar peta yang dibuat penulis pada Google MyMaps.



Gambar 15 Hasil Graf Pada Google MyMaps

Proses pengujian sistem ini akan dilakukan dengan berbagai cara. Adapun pengujian sistem yang dilakukan adalah:

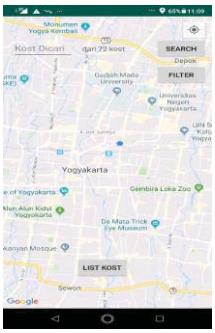
1. Sistem Akan diuji hasil jarak rumah kost terdekatnya. Jarak rumah kost terdekat yang dihasilkan oleh sistem akan dibandingkan dengan hasil Google Maps, akan dicatat hasil dari beberapa titik awal kemudian akan dicari rata-rata selisih jaraknya.
2. Durasi pencarian kost terdekat pada sistem akan dicatat dan dicari rata-ratanya.
3. Diuji pencarian kost berdasarkan filter jarak, akan dicek durasi waktu pencarian dari berbagai masukan jarak.
4. Pencarian jalur menuju rumah kost terdekat juga akan diuji. Akan dibandingkan jalurnya dengan Google Maps.

Untuk uji waktu pencarian penulis akan menggunakan *smartphone* Asus Zenfone Maxpro M2 yang memiliki *processor* Snapdragon 660, memori internal 32 GB, GPU Adreno 512 dan RAM 4GB. Smartphone menggunakan sistem operasi Android 8.1 (Oreo) dan sudah dilengkapi fitur GPS.

IV. HASIL DAN ANALISIS

Pada saat aplikasi dibuka tampilan awal berupa peta Google. Gambar tampilan awal dapat dilihat pada Gambar 16. Di bagian kiri atas terdapat *input number* untuk memasukkan jumlah kost yang ingin dicari. Jumlah kost yang dicari tidak dapat melebihi data kost yang tersedia di *database*, tidak bisa 0, dan tidak bisa kosong. Terdapat pula tombol "*search*" dan tombol "*filter*" di kanan atas serta tombol "list kost" di bagian bawah. Tombol "*search*" akan mencari rumah kost, tombol "*filter*" akan menuju ke halaman filter dan tombol "list kost" akan menuju halaman list kost.

JUTEI Edisi Volume.4 No.1 April 2020
ISSN 2579-3675, e-ISSN 2579-5538
DOI 10.21460/jutei.2020.14.193



Gambar 16 Tampilan Awal Aplikasi

Titik awal pencarian kost berdasarkan GPS. Jika *user* membuat *marker* di peta dengan cara menekan peta beberapa saat, maka *marker* akan muncul seperti pada Gambar 17. Ketika *user* membuat *marker* maka akan muncul tombol "*clear marker*" pada bagian bawah di sebelah tombol "list kost". Fungsi tombol "*clear marker*" adalah untuk menghapus *marker* jika *marker* sudah tidak dikehendaki atau ingin diganti posisinya.

Gambar 17 Tampilan Awal dengan Titik Awal Marker

Setelah *user* melakukan pencarian kost dengan menekan tombol "*search*" maka akan muncul *marker* merah sebagai titik awal dan *marker* dengan gambar rumah sebagai titik rumah kost. Akan ditampilkan pula durasi waktu proses algoritma Dijkstra mencari rumah kost dalam *miliseconds*. Gambar 18 merupakan tampilan aplikasi setelah dilakukan pencarian kost terdekat.

Gambar 18 Tampilan Cari Kost pada Aplikasi

Ketika suatu *marker* kost dipilih akan muncul informasi nama kost dan jaraknya dalam meter. Jarak itu merupakan jarak terpendek atau terkecil dari titik awal. Akan muncul pula tombol "*direction*" di kanan bawah di sebelah tombol "list kost". Gambar 19 merupakan tampilan ketika suatu *marker* kost dipilih pada aplikasi.

Gambar 19 Tampilan Pilih Kost pada Aplikasi

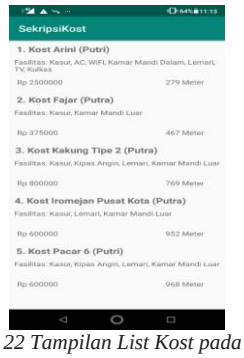
Setelah itu ketika suatu *marker* kost dipilih lalu pilih tombol "*direction*" maka algoritma Dijkstra akan berjalan mencari jalur terpendek menuju kost yang dipilih. Pada pencarian jalur, titik awal selalu berdasarkan GPS. Gambar 20 merupakan tampilan jalur terpendek pada aplikasi.

Gambar 20 Tampilan Jalur Terpendek Pada Aplikasi

Ketika *user* pada tampilan awal menekan tombol "filter" maka halaman akan menuju halaman filter. Ada dua filter pada aplikasi ini yaitu harga maksimal dan jarak maksimal. Filter harga menggunakan satuan Rupiah dan filter jarak menggunakan satuan meter. Pada kondisi awal angka filter adalah 0 jika *input* dikosongkan maka akan dianggap 0, kemudian masukan filter hanya dapat diisi angka positif. Filter yang diterapkan dapat berupa harga maksimal saja, jarak maksimal saja atau keduanya. Gambar tampilan filter dapat dilihat pada Gambar 21.

Gambar 21 Tampilan Halaman Filter pada Aplikasi

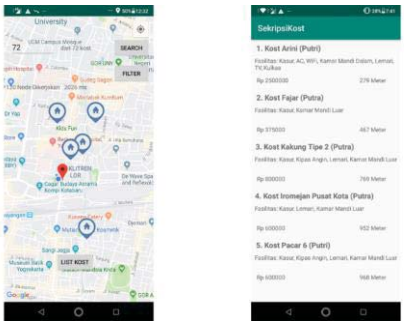
Ketika *user* menekan tombol "list kost" pada tampilan awal maka halaman akan menuju halaman *list* kost. Isi halaman ini adalah daftar rumah kost yang telah dicari. Jika aplikasi belum melakukan pencarian kost maka halaman ini akan kosong. Kost yang ditampilkan dalam bentuk *list recyclerview* dan sudah diurutkan yang terdekat dari lokasi awal. *List* rumah kost juga memiliki beberapa informasi yaitu nama kost, tipe kost, fasilitas, harga, dan jarak dari posisi awal. Tampilan list kost dapat dilihat pada Gambar 22.



Gambar 22 Tampilan List Kost pada Aplikasi

Algoritma yang digunakan pada aplikasi ini adalah algoritma Dijkstra. Algoritma Dijkstra di aplikasi ini digunakan untuk mencari kost terdekat dan untuk membuat jalur terpendek menuju rumah kost yang dipilih. Saat melakukan pencarian rumah kost, titik awal dapat berupa posisi GPS atau posisi *marker* namun untuk pencarian jalur, posisi awal selalu berdasarkan posisi GPS.

Pada pencarian kost kondisi berhenti adalah berdasarkan kost didapat pada sistem, jika sudah sesuai dengan kost dicari maka algoritma akan berhenti. Kondisi berhenti juga bisa dipengaruhi oleh filter. Ketika user menggunakan filter jarak maksimal algoritma akan berhenti saat bobot minimal vertek yang dikerjakan sudah lebih dari jarak maksimal *inputan user*, namun kondisi berhenti ini masih bisa dipengaruhi oleh jumlah kost yang ingin dicari. Ketika hasil tidak ditemukan maka akan muncul *toast* dibagian bawah. Gambar 23 adalah gambar hasil pencarian kost dengan filter jarak, akan dicoba menampilkan semua rumah kost dengan maksimal jarak 1000 m dari titik awal UKDW.



Gambar 23 Hasil Pencarian Semua Kost dengan Filter Jarak Maksimal 1000 m

Jika user menggunakan filter harga maka kost didapat akan dalam batasan harga maksimal *inputan user*, ketika vertek yang dikerjakan adalah vertek kost dan kost pada vertek tersebut memiliki harga yang kurang atau sama dengan harga masukan *user* maka kolom bobot kost dan *scanned* kost akan di*update* pada tabel kost dan kost didapat ditambah 1. Semua rumah kost pada kolom *scanned* di *database* tabel kost yang memiliki angka 1 akan ditampilkan pada peta dan *list* kost.

Pada pencarian jalur, algoritma Dijkstra akan terus berjalan sampai vertek sudah dikerjakan adalah vertek tujuan *user*. Setelah vertek jalur diperoleh akan digambar jalur *polyline* dan ditampilkan jalurnya pada peta.

Graf yang digunakan untuk proses algoritma Dijkstra berasal dari *database* tabel graph. Data pada tabel graph diisi dan dibuat secara manual berdasarkan titik koordinat persimpangan yang sudah ditandai pada Google *MyMaps*, jalur yang dibuat juga dimasukkan koordinatnya. Begitu pula dengan data rumah kost dibuat dan dimasukkan titiknya yang telah ditandai secara manual dari Google *MyMaps*. Data koordinat jalur pada basis data sistem berupa *json object* dengan *value coordinates* berupa *json array*.

Pada pengujian yang pertama adalah uji dan analisis waktu pencarian kost terdekat. Akan diambil total 20 data dari titik awal yang berbeda, namun dirata-rata waktu pencariannya per-5 data dengan jarak per-375 m. Data terjauh diberi batasan 1500 m karena sistem umumnya mendapatkan kost terdekat dengan jarak dibawah 1500 m.

TABEL II
HASIL WAKTU PENCARIAN KOST TERDEKAT

Percobaan	Titik Awal	Titik Akhir	Jarak Sistem (m)	Waktu Pencarian (ms)
0 m – 375 m				
1	-7.79614, 110.39158	-7.79624, 110.3938	291	143
2	-7.78613, 110.37877	-7.78504, 110.37938	179	103
3	-7.80746, 110.3581	-7.80551, 110.35851	374	180
4	-7.81975, 110.39324	-7.82061, 110.39214	209	94
5	-7.80845, 110.38992	-7.80889, 110.38884	200	104
Rata-Rata				**124,8**
376 m – 750 m				
6	-7.79892, 110.38345	-7.79875, 110.3812	445	218
7	-7.80158, 110.39206	-7.80414, 110.39163	481	232
8	-7.82853, 110.37976	-7.82667, 110.38027	399	101
9	-7.81477, 110.36837	-7.81601, 110.37313	612	244
10	-7.7926, 110.39313	-7.79008, 110.39509	412	136
Rata-Rata				**186,2**
751 m – 1125 m				
11	-7.78983, 110.36212	-7.78757, 110.35844	792	519
12	-7.78587, 110.37439	-7.78157, 110.37808	849	846
13	-7.79209, 110.37362	-7.79257, 110.38093	965	1405
14	-7.8014, 110.36475	-7.80551, 110.35851	1122	1366
15	-7.81344, 110.36295	-7.81428, 110.35784	834	338
Rata-Rata				**894,8**
1126 m – 1500 m				
16	-7.79908, 110.36501	-7.80551, 110.35851	1383	2144
17	-7.79696, 110.35265	-7.78757, 110.35844	1456	809
18	-7.79887, 110.36934	-7.80121, 110.37846	1376	2338
19	-7.78992, 110.36551	-7.78757, 110.35844	1170	1399
20	-7.80153, 110.36817	-7.80121, 110.37846	1193	1486
Rata-Rata				**1635,2**

JUTEI Edisi Volume.4 No.1 April 2020
ISSN 2579-3675, e-ISSN 2579-5538
DOI 10.21460/jutei.2020.14.193

Tabel II diatas adalah tabel hasil waktu pencarian kost terdekat. Data percobaan satu sampai lima adalah percobaan pencarian kost terdekat yang memiliki jarak antara 0 m sampai 375 m, dari data tersebut diperoleh rata-rata waktu pencariannya 124,8 ms. Sedangkan data percobaan enambelas sampai duapuluh adalah percobaan pencarian kost terdekat yang memiliki jarak antara 1126 m sampai 1500 m, dari data tersebut diperoleh rata-rata waktu pencariannya 1635,2 ms. Dapat dilihat bahwa semakin jauh jarak kost yang didapatkan maka akan semakin lama waktu pencariannya.

Pengujian yang selanjutnya adalah uji perbandingan jarak dan jalur yang dihasilkan oleh sistem dengan Google Maps, akan dibandingkan jarak dan jalur dari titik awal ke titik rumah kost terdekat yang didapatkan oleh sistem. Data yang diambil ada 20 dari titik awal yang berbeda dan dari 20 data tersebut akan dicari rata-rata selisih jaraknya.

TABEL III
HASIL PERBANDINGAN JARAK ANTARA SISTEM DENGAN GOOGLE MAPS

Percobaan	Titik Awal	Titik Akhir	Hasil Sistem (m)	Hasil Google Maps (m)	Selisih Jarak (m)
1	-7.79614, 110.39158	-7.79624, 110.3938	291	450	-159
2	-7.78613, 110.37877	-7.78504, 110.37938	179	170	9
3	-7.80746, 110.3581	-7.80551, 110.35851	374	400	-26
4	-7.81975, 110.39324	-7.82061, 110.39214	209	210	-1
5	-7.80845, 110.38992	-7.80889, 110.38884	200	190	10
6	-7.79892, 110.38345	-7.79875, 110.3812	445	450	-5
7	-7.80158, 110.39206	-7.80414, 110.39163	481	500	-19
8	-7.82853, 110.37976	-7.82667, 110.38027	399	350	49
9	-7.81477, 110.36837	-7.81601, 110.37313	612	600	12
10	-7.7926, 110.39313	-7.79008, 110.39509	412	400	12
11	-7.78983, 110.36212	-7.78757, 110.35844	792	800	-8
12	-7.78587, 110.37439	-7.78157, 110.37808	849	850	-1
13	-7.79209, 110.37362	-7.79257, 110.38093	965	950	15
14	-7.8014, 110.36475	-7.80551, 110.35851	1122	1100	22
15	-7.81344, 110.36295	-7.81428, 110.35784	834	800	34
16	-7.79908, 110.36501	-7.80551, 110.35851	1383	1400	-17
17	-7.79696, 110.35265	-7.78757, 110.35844	1456	1400	56
18	-7.79887, 110.36934	-7.80121, 110.37846	1376	1200	176
19	-7.78992, 110.36551	-7.78757, 110.35844	1170	1200	-30
20	-7.80153, 110.36817	-7.80121, 110.37846	1193	1200	-7
Rata-Rata			**737,1**	**731**	**6,1**

Tabel III adalah hasil perbandingan jarak yang dihasilkan oleh sistem dengan Google Maps. Dari 20 kali percobaan semuanya menghasilkan selisih jarak yang berbeda. Dapat dilihat 50 % jarak keluaran sistem lebih kecil dibandingkan dengan Google Maps. Namun dari data diatas dapat disimpulkan bahwa jarak keluaran Google Maps lebih kecil dibandingkan sistem karena rata-rata jarak yang dihasilkan Google Maps adalah 731 m sedangkan rata-rata jarak yang dihasilkan sistem adalah 737,1 m dengan selisih jarak 6,1 m.

Kemudian pengujian selanjutnya akan dilakukan perbandingan jalur terpendek yang dibuat oleh sistem dengan Google Maps. Akan dilakukan 20 kali percobaan dan dicari apakah rute yang dilalui oleh sistem dengan Google Maps sama atau berbeda. Karena graf pada basis data sistem menggunakan graf ganda berarah maka jalur yang dipakai adalah jalur pejalan kaki yang tidak memperhatikan jalur 1 arah.

TABEL IV
HASIL PERBANDINGAN JALUR SISTEM DENGAN GOOGLE MAPS

Percobaan	Titik Awal	Titik Tujuan	Rute Yang Dilalui
1	-7.79614, 110.39158	-7.79624, 110.3938	Beda
2	-7.78613, 110.37877	-7.78504, 110.37938	Sama
3	-7.80746, 110.3581	-7.80551, 110.35851	Sama
4	-7.81975, 110.39324	-7.82061, 110.39214	Sama
5	-7.80845, 110.38992	-7.80889, 110.38884	Sama
6	-7.79892, 110.38345	-7.79875, 110.3812	Sama
7	-7.80158, 110.39206	-7.80414, 110.39163	Sama
8	-7.82853, 110.37976	-7.82667, 110.38027	Beda
9	-7.81477, 110.36837	-7.81601, 110.37313	Sama
10	-7.7926, 110.39313	-7.79008, 110.39509	Sama
11	-7.78983, 110.36212	-7.78757, 110.35844	Sama
12	-7.78587, 110.37439	-7.78157, 110.37808	Sama
13	-7.79209, 110.37362	-7.79257, 110.38093	Sama
14	-7.8014, 110.36475	-7.80551, 110.35851	Sama
15	-7.81344, 110.36295	-7.81428, 110.35784	Beda
16	-7.79908, 110.36501	-7.80551, 110.35851	Sama
17	-7.79696, 110.35265	-7.78757, 110.35844	Sama
18	-7.79887, 110.36934	-7.80121, 110.37846	Beda
19	-7.78992, 110.36551	-7.78757, 110.35844	Sama
20	-7.80153, 110.36817	-7.80121, 110.37846	Sama

Tabel IV adalah hasil perbandingan jalur yang dibuat sistem dengan Google Maps. Dari 20 kali percobaan

diperoleh hasil 20% memiliki jalur yang berbeda. Perbedaan jarak dan jalur ini dikarenakan data graf yang dimiliki sistem dengan Google Maps berbeda.

Pengujian yang selanjutnya adalah uji dan analisis waktu pencarian berdasarkan filter jarak, akan dicatat *node* yang dikerjakan dan waktu pencarian berdasarkan filter jarak per-500 m. Pada proses pencarian, titik awal akan berada di UKDW.

TABEL V
HASIL WAKTU PENCARIAN BERDASARKAN FILTER JARAK

Nomor	Jarak Pencarian (m)	Node Dikerjakan	Waktu Pencarian (ms)
1	500	37	471
2	1000	120	1826
3	1500	230	3876
4	2000	326	6096
5	2500	440	9013
6	3000	581	13401
7	3500	679	16853
8	4000	798	21326
9	4500	866	23237
10	5000	950	26560

Pada tabel V didapatkan hasil bahwa semakin besar filter jaraknya maka akan semakin lama proses pencariannya, hal ini dikarenakan semakin banyaknya *node* yang dikerjakan.

V. KESIMPULAN

Berdasarkan implementasi dan analisis sistem yang telah dilakukan, maka disimpulkan bahwa:

1. Algoritma Dijkstra dapat diimplementasikan untuk mencari rumah kost terdekat berbasis Android. Pada sistem ini algoritma Dijkstra yang diimplementasikan dapat melakukan pencarian kost berdasarkan filter yaitu, harga maksimal dan jarak maksimal. Pencarian juga bisa berdasarkan jumlah kost yang dicari. Sistem ini juga mampu mencari dan membuat jalur terpendek dari posisi GPS ke kost yang dipilih. Namun algoritma Dijkstra juga memiliki kekurangan dalam proses pencariannya diantaranya waktu pencariannya yang semakin lambat jika titik tujuan atau jarak pencarian semakin jauh.
2. Algoritma Dijkstra yang diimplementasikan dalam mencari jarak terkecil hampir mendekati data dari Google Maps karena dari 20 kali percobaan didapatkan selisih sebesar 6,1 m dari rata-rata jarak antara sistem dan Google Maps.
3. Semakin jauh jarak dan jalur rumah kost yang dicari maka akan semakin lama waktu pencariannya karena semakin banyaknya vertek yang dikerjakan.

Adapun saran yang diberikan penulis kepada pengembang selanjutnya adalah sebagai berikut:

1. Pengembang selanjutnya dapat menambahkan fungsi menambah dan mengurangi rumah kost pada sistem
2. Pengembang selanjutnya dapat menambahkan filter yang lainnya sehingga filter tidak hanya berdasarkan harga maksimal dan jarak maksimal.

DAFTAR PUSTAKA

[1] A. Christian, "Studi Literatur Perbandingan Algoritma dijkstra Dan Bellman-ford Dalam Pencarian Jarak Terdekat," 2013.

[2] G. A. Nalu, "Studi Literatur Algoritma Floyd-warshall Dan dijkstra Untuk Menentukan Jalur Terpendek," 2015.

[3] A. G. Wibowo and A. P. Wicaksono, "Rancang Bangun Aplikasi untuk Menentukan Jalur Terpendek Rumah Sakit di Purbalingga dengan Metode Algoritma Dijkstra," 2012.

[4] F. A. Pratama and D. P. Kusumaningrum, "Penggunaan Algoritma Dijkstra Pada Aplikasi Searching Hotel Di Kota Semarang," 2013.

[5] A. Budianto and E. R. Nainggolan, "Perancangan Aplikasi Islamic Boarding School Finder Berbasis Android Menggunakan Algoritma Dijkstra," 2016.

[6] D. Wahyuningsih and E. Syahreza, "Shortest Path Search Futsal Field Location With Dijkstra Algorithm," 2018.

[7] U. Hasanah, N. Safriadi and T. , "Rancang Bangun Aplikasi Location Based Service Lokasi Masjid Pontianak Menggunakan Metode Dijkstra Berbasis Android," 2015.

[8] T. Wahyuningrum and E. Usada, Matematika Diskrit: dan Penerapannya dalam Dunia Informasi, Yogyakarta: Deepublish, 2016.

[9] R. Munir, Matematika Diskrit, Bandung: Informatika, 2010.

[10] Y. Supardi, Belajar Coding Android bagi Pemula, Jakarta: Elex Media Komputindo, 2015.

[11] Marsudi, Teori Graf, UB Press, 2016.

[12] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, Introduction to Algorithms, MIT Press, 2009.

[13] mokox, "GitHub," 2015. [Online]. Available: https://github.com/mokox/dijkstra-algorithm. [Accessed 2019].

JUTEI Edisi Volume.4 No.1 April 2020
ISSN 2579-3675, e-ISSN 2579-5538
DOI 10.21460/jutei.2020.14.193