

BAB II

TINJAUAN PUSTAKA

2.1 Konsep Teoritis

Penelitian ini berlandaskan teori-teori yang dimulai dari algoritma Dijkstra, graf dan jaringan, problematika jarak terpendek, bahasa pemrograman *python* serta *software-software* pendukungnya.

2.1.1 Algoritma Dijkstra

Algoritma Dijkstra termasuk salah satu jenis algoritma rakus bukan dalam arti negatif yang dapat memecahkan permasalahan yang berhubungan dengan masalah optimasi. Sesuai dengan jenisnya yaitu rakus, maka algoritma Dijkstra akan mencari solusi yang sesuai pada langkah yang dilaluinya dengan tujuan untuk mendapatkan solusi optimum yang akan berakhir dengan solusi terbaik.

Algoritma Dijkstra bekerja dengan cara prinsip antrian yaitu antrian berprioritas, jadi hanya simpul yang memiliki prioritas tertinggi yang akan dicari dan membandingkan setiap nilai dari antara simpul pada satu level yang sama dengan nilai bersifat positif (Primadasa, 2015).

Pencarian yang dilakukan algoritma Dijkstra dengan cara mencari semua jalur optimum yang nilainya minimum dari jalur yang tersedia dimulai dari titik awal ke titik yang akan kita tentukan. Karena mencari semua jalur optimum yang ada, maka algoritma Dijkstra mempunyai kekurangan dalam hal efisiensi pencarian yang kurang dan waktu pencarian yang lama terutama jarak yang dicari jauh pada titik tujuan yang ditentukan (Sharma, Saini, & Bhandhari, 2012).

Ditemukan pada tahun 1959 oleh Edsger Dijkstra, algoritma yang memecahkan masalah secara tahap demi tahap yang disebut dengan prinsip kerja *greedy*, maka pada setiap langkahnya (Ratnasari, Ardiani, & A, 2013) :

1. Mengambil pilihan terbaik yang dapat diperoleh pada saat itu.
2. Berharap di setiap langkah dengan memilih optimum lokal akan mencapai optimum global. Algoritma *greedy* mengasumsikan bahwa optimum lokal adalah bagian dari optimum global.

Algoritma Dijkstra dapat didefinisikan ke dalam beberapa langkah sebagai berikut (Sharma et al., 2012) :

1. Mulai menentukan titik sumbernya yang akan menjadi awal permulaan pada jaringannya.
2. Tetapkan titik pertama dengan memberi nilai 0 (nol) dan beri label permanen yang pertama.
3. Memeriksa setiap titik sekitar dari titik permanen yang sudah ditentukan pertama kali.
4. Inisiasi jarak antar satu titik dengan titik lainnya dan titik yang sudah terhubung di beri label sementara.
5. Jika di antara label sementara terdapat nilai keseluruhan totalnya minimum, jadikan sebagai label permanen. Sementara itu jika sebuah titik bisa dilalui lebih dari satu arah, maka pilihlah arah yang mendapatkan angka total yang lebih minimum.
6. Ulangi langkah ketiga sampai dengan kelima hingga setiap titik yang dilalui mendapatkan label permanen.

Perhitungan algoritma Dijkstra dapat dilakukan beberapa cara seperti dengan menggunakan tabel, iterasi, memakai sistem eliminasi dengan gambaran kita seperti lingkaran ataupun melihat secara langsung jika jumlah titik-titik yang menghubungkannya tidak dalam jumlah yang besar. Penggunaannya membutuhkan pemahaman mendalam serta tidak dapat menghitung angka yang bernilai negatif.

2.1.2 Graf dan Jaringan

Jaringan-jaringan pada bidang sains dan teknologi dibuat berdasarkan dari teori tentang graf. Graf merupakan suatu hal yang direpresentasikan secara abstrak yang terdiri dari titik (*node*) dan yang menghubungkannya yaitu sisi (*edge*). Dua buah titik dikatakan saling terhubung jika ada sisi yang menghubungkan mereka, dua buah sisi dikatakan saling terhubung jika mereka berbagi titik yang sama dan sebuah titik dengan sisi terbentuk jika sisi saling menghubungkan antara satu titik dengan titik lainnya.

Berdasarkan arah pada sisi graf, maka graf dapat dibedakan menjadi dua jenis yaitu graf tak berarah dan graf berarah. Graf tak berarah merupakan graf yang tidak memiliki arah pada sisinya dan urutan pasangan simpul yang dihubungkan oleh sisi diabaikan. Graf berarah merupakan graf yang memiliki arahan yang jelas pada sisinya dan mempunyai busur (*arc*) untuk penggambarannya (Harahap & Khairina, 2017).

Graf dalam kehidupan sehari-hari yang kita ketahui berupa jaringan. Hubungan kita dengan yang lainnya dapat digambarkan dengan jaringan, dimisalkan titik merupakan orang dan sisi yang menghubungkannya berupa status

hubungan sosial kita seperti keluarga dan lainnya. Hampir semua jaringan yang nyata merupakan jaringan yang kompleks, jaringan yang bisa dilihat pada peta merupakan salah satu contoh jaringan kompleks yang menghubungkan antar satu *node* dengan *node* lainnya karena terdapat jalur jalan raya, kereta api, selokan, dan sebagainya (Boeing, 2017).

2.1.3 Problematika Jarak Terpendek

Berpergian ke suatu tempat perlu memperhatikan berapa kecepatan yang kita perlukan dengan menimbang waktu yang sudah ditetapkan jika ingin tiba di tempat secara cepat ataupun tepat waktu. Mulai dari awal hari memulai aktivitas, kita dihadapkan dengan masalah dalam memilih jalur mana yang cocok bagi kita untuk dimisalkan pergi kerja ataupun ke pasar. Penentuan jarak terdekat atau terpendek (*shortest path*) merupakan suatu sistematika untuk menghitung jarak terpendek ataupun biaya yang minimum antara satu *node* dengan *node* lainnya yang menghubungkan antara jalur yang dilaluinya.

Proses perhitungannya dibagi menjadi dua macam yaitu proses pemberian berupa label serta pemeriksaan *node* (Retnani et al., 2015). Penanganan masalah jarak terpendek yang dihadapi biasanya seperti lintasan terpendek antara dua simpul, lintasan terpendek antara semua pasangan simpul, ataupun lintasan terpendek dari simpul tertentu ke semua simpul lainnya (Harahap & Khairina, 2017).

2.1.4 Bahasa Pemrograman *Python*

Peralatan yang kita gunakan dalam kegiatan sehari-hari seperti gawai tidak akan terlepas dari peran bahasa pemrograman yang mendukungnya untuk beroperasi. Bahasa pemrograman tidak hanya mengendalikan apa saja di dalamnya, tetapi juga untuk sistem dalam video *game* bahkan GPS dalam mobil kita yang dikendalikan oleh program.

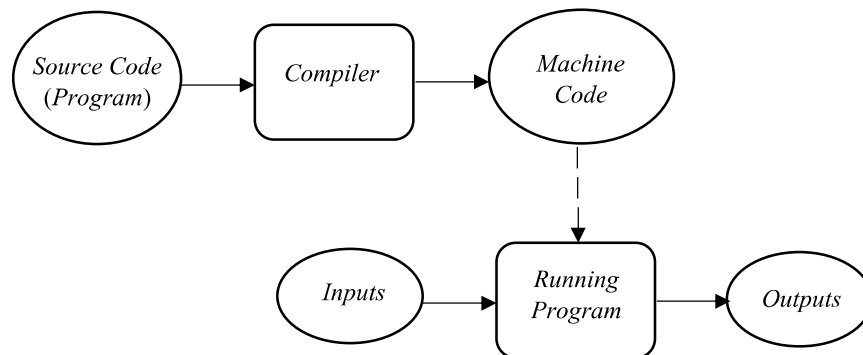
Program seperti pikiran, jika kita tidak memiliki pikiran maka yang kita lakukan hanyalah seperti polisi tidur. Pemikiran kita seperti “bangun” adalah sebuah instruksi atau perintah yang memberi tahu tubuh kita untuk berdiri supaya tidak rebahan di lantai, sama seperti bahasa pemrograman yang memberi tahu kepada komputer apa yang akan dilakukan.

Python merupakan salah satu bahasa pemrograman yang dinamai secara unik karena memiliki kesamaan dengan nama ular, akan tetapi pemberian namanya bukan berasal dari nama hewan, melainkan dari acara komedi televisi di Inggris (*British*) bernama *Monty Python's Flying Circus* yang diciptakan oleh Guido Van Rossum. Hal yang paling penting untuk menggunakan *python* adalah kita dapat menulis program (*coding*) secara ringkas dan efisien dengan cepat. *Python* tidak memiliki simbol-simbol yang rumit seperti *braces* (`{}`), *hashes* (`#`) ataupun *dollar signs* (`$`) yang membuat bahasa pemrograman selain *python* kurang tepat untuk pemula (Briggs, 2013).

Bahasa pemrograman *python* termasuk bahasa tingkat tinggi (*high level language*) karena proses komputerisasinya yang dapat dipahami oleh manusia, perangkat keras seperti komputer hanya dapat memahami bahasa tingkat rendah

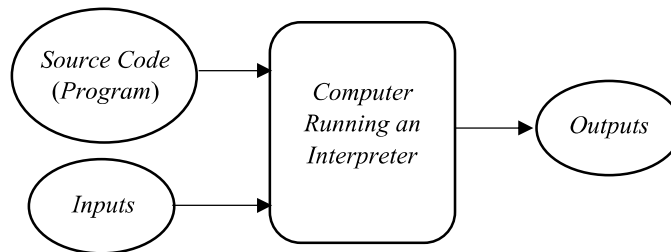
yaitu bahasa mesin (*machine language*). Jika dimisalkan sebuah persamaan $c = a + b$ maka kita dapat langsung memahaminya, hal yang berbeda dengan komputer. Ada dua cara supaya bahasa tingkat tinggi dapat dieksekusi oleh komputer yaitu dengan cara di-*compile* atau diinterpretasikan (Zelle, 2010).

Sebuah *compiler* merupakan suatu program kompleks yang membawa program yang ditulis pada bahasa tingkat tinggi dan diterjemahkan pada beberapa komputer yang berbahasa mesin. Pada gambar 2.3 menunjukkan proses meng-*compile*, *compiler* mengkombinasikan *source code* dan menghasilkan *machine code* yaitu sebuah program yang dapat dieksekusi langsung oleh komputer.



Gambar 2.1 Meng-*compile* Bahasa Tingkat Tinggi
Sumber : (Zelle, 2010)

Interpreter merupakan sebuah program yang dapat mensimulasikan komputer untuk memahami bahasa tingkat tinggi, *interpreter* tidak menerjemahkan *source code* ke bahasa mesin secara langsung akan tetapi menganalisis dan mengeksekusi *source code* sesuai instruksi yang diberikan.



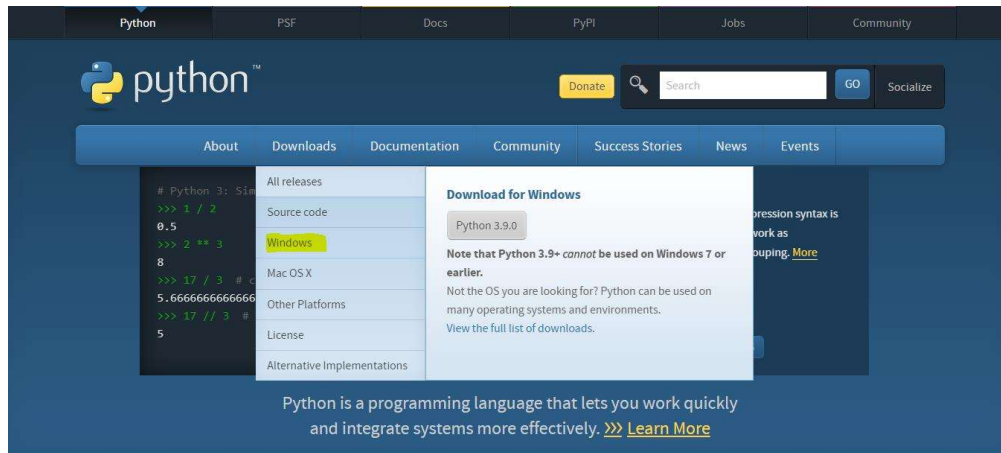
Gambar 2.2 Menginterpretasikan Bahasa Tingkat Tinggi
Sumber : (Zelle, 2010)

Perbedaan dari kedua proses di atas adalah jika meng-*compile* hasilnya sudah langsung satu kesatuan program yang bisa digunakan berulang tanpa perlu diproses *compiler* atau dari *source code* kembali, sedangkan pada proses interpretasi, *interpreter* dan *source* dibutuhkan pada saat menjalankan program. Dilihat dari segi waktu, maka program yang diproses dengan *compiler* lebih cepat karena proses menerjemahkannya tanpa perlu mengulang, akan tetapi bahasa pemrograman yang diinterpretasikan dapat dijalankan pada *environment* komputer yang lebih fleksibel selama program bisa dirancang dan dieksekusi secara interaktif.

2.1.4.1 Memasang *Python* di Komputer

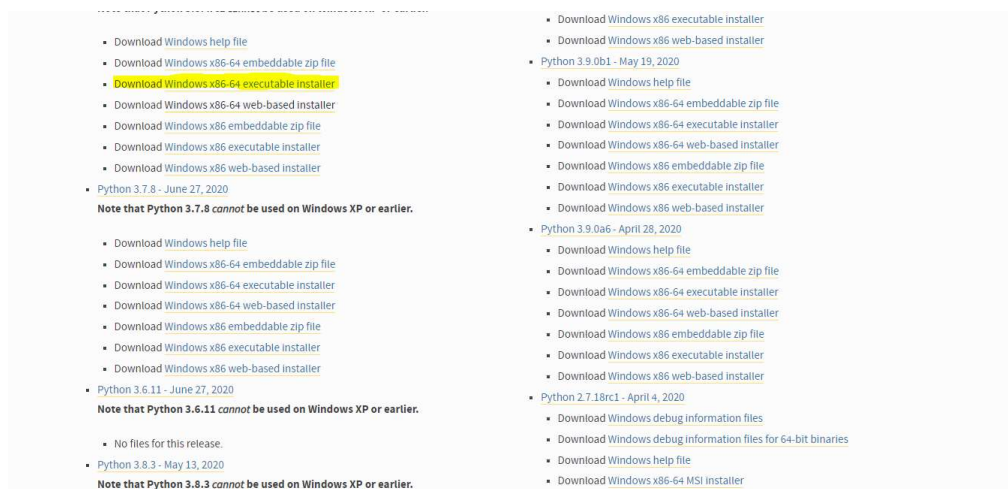
Untuk menginstalasi *python* di komputer kita, pertama kita menyetikkan <http://www.python.org/> di pencarian internet dan pilihlah jenis *Python* 3 sesuai yang diinginkan dan langkah-langkahnya sebagai berikut :

1. Setelah mengunjungi website resminya, pilih tab *download* dan cari versi sistem operasi komputer yang kita miliki. Pada penelitian ini peneliti memilih versi *windows* sistem operasinya.



Gambar 2.3 Tampilan Awal *Python*
Sumber : Data Peneliti (2020)

2. Memasuki laman baru, maka kita cari pada bagian tulisan *Stable Releases* (pada sisi kiri) dan carilah versi *python* yang kita inginkan selama *python* yang dipilih merupakan versi 3 yang ditandai dengan angka 3 di depannya. Pada saat melihat banyak pilihan, pilihlah yang bertuliskan *executable installer* dan sesuaikan versinya. Jika komputernya versi 32 bit maka pilih yang x86.



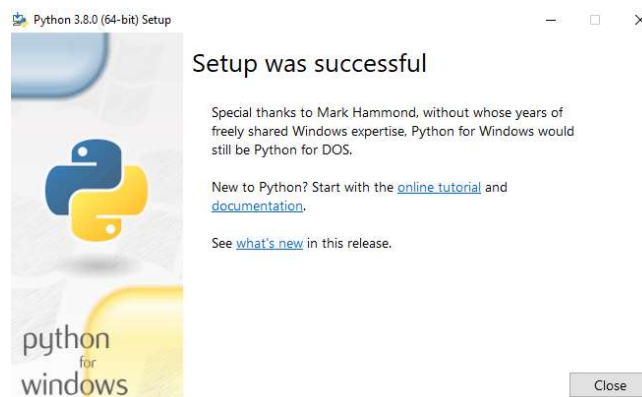
Gambar 2.4 Pemilihan *Installer Python*
Sumber : Data Peneliti (2020)

3. Setelah mengunduh *installer*-nya, maka carilah file nya di tempat yang sudah kita simpan dan klik file *installer*-nya, maka nanti akan muncul tampilan seperti di bawah berikut. Jangan lupa untuk memberi centang pada “*Add Python 3.8 to PATH*” dan pilih yang “*Install Now*”



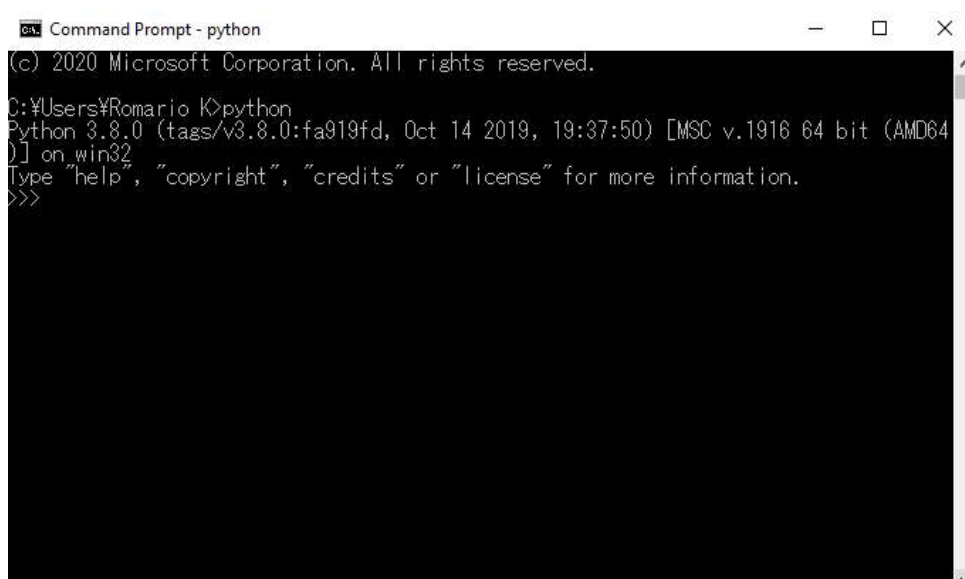
Gambar 2.5 Tampilan Awal Saat Membuka *Installer*
Sumber : Data Peneliti (2020)

4. Tunggu proses instalasi selesai yang ditandai dengan jalannya kotak panjang berwarna hijau. Jika proses selesai maka akan muncul seperti gambar di bawah.



Gambar 2.6 Tampilan Selesai Memasang *Python*
Sumber : Data Peneliti (2020)

5. Untuk memastikan apakah sudah terpasang *python* kita pada komputer maka buka *command prompt* atau ketikkan “cmd” tanpa tanda petik pada tombol *windows start* dipojok kiri bawah kemudian ketikkan kata “python” tanpa tanda petik, maka akan muncul tampilan *python* kita sesuai dengan versi yang kita pasang, klik tanda silang untuk keluar dari *command prompt*.

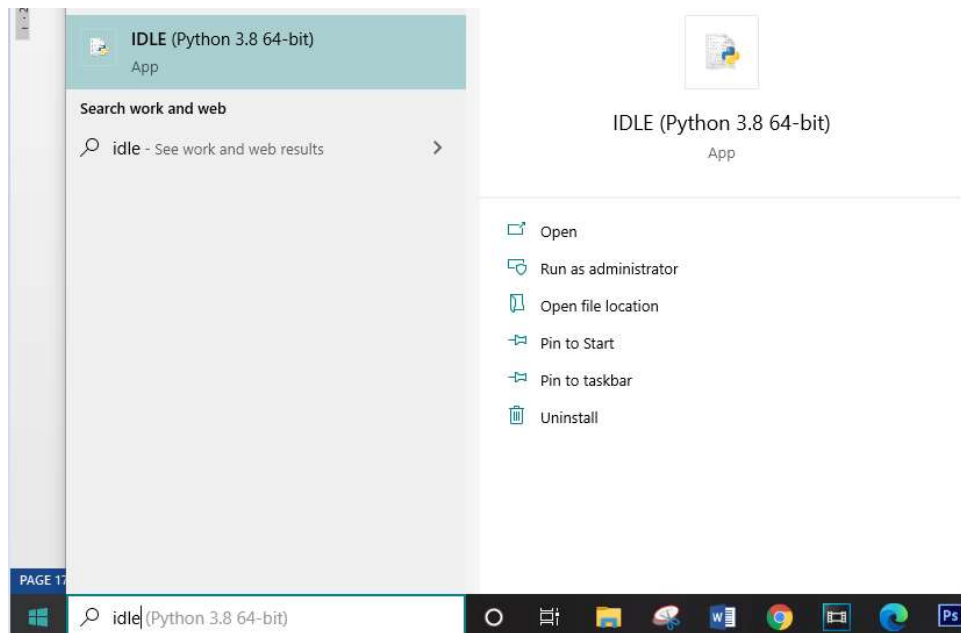


```
Command Prompt - python
(c) 2020 Microsoft Corporation. All rights reserved.
C:\Users\Romario K>python
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:37:50) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Gambar 2.7 Tampilan *Python* sudah Terpasang dengan *Command Prompt*
Sumber : Data Peneliti (2020)

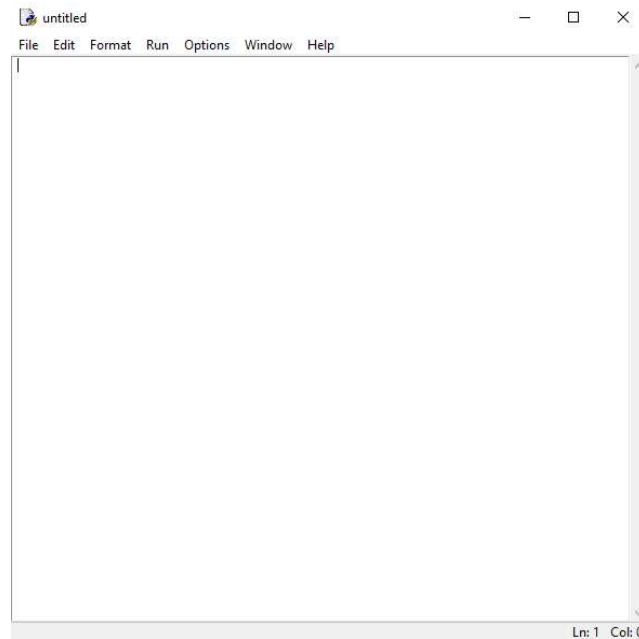
Langkah selanjutnya setelah memasang adalah menulis kode programnya, maka kita dapat melakukannya sebagai berikut :

1. Pilih tombol *windows start* pada pojok kiri bawah dan ketikkan kata “idle” tanpa tanda petik, maka akan muncul hasil pencariannya dan klik hasil pencarian tersebut.



Gambar 2.8 Membuka IDLE
Sumber : Data Peneliti (2020)

2. Setelah memilih, maka akan muncul tampilan *shell*. Klik menu *file* dan pilih *new file* untuk membuka IDLE bawaan. Kode program akan diketikkan pada layar *untitled* dan untuk mengeksekusinya pilih menu *run* kemudian klik *run module*. Sebelum dieksekusi kita akan dimintai untuk menyimpan *file* dari kode program yang akan dieksekusi.



Gambar 2.9 Tempat Untuk Mengetik Kode Program
Sumber : Data Peneliti (2020)

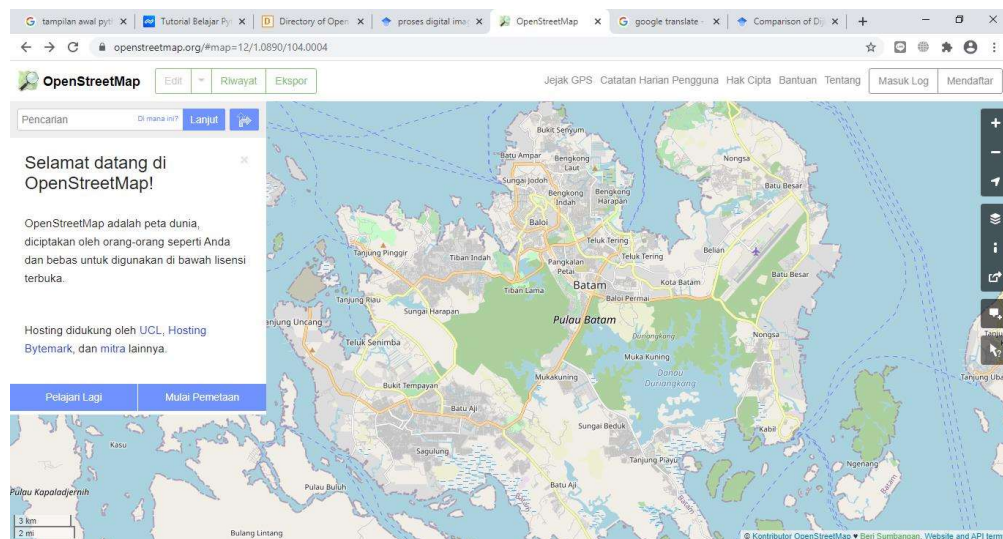
2.1.5 *Open Street Map*

Perkembangan teknologi berupa peta digital dibuat secara tidak mudah karena melalui serangkaian proses yang pada umumnya membutuhkan bantuan perangkat lunak Sistem Informasi Grafis (SIG) dan dilakukan oleh orang yang ahli dalam pemetaan. Cara ini tentu saja membuat tidak bisa sembarangan orang yang memetakan dan terbatas, hal ini tidak sama dengan pemetaan yang dilakukan dengan *Open Street Map (OSM)*.

Open Street Map merupakan suatu layanan atau *platform* pemetaan berbasis internet yang bersifat partisipatif atau terbuka bagi siapa saja dan dikenal dengan istilah *crowdmapping* yaitu peta dibuat oleh banyak orang dilakukan secara bersama-sama tetapi tetap memperhatikan dalam hal manajemen kontrol.

Menggunakan *Open Street Map* tidak dikenai biaya dan bisa kapan saja untuk mengaksesnya (Nurrohmah & Sulistioningrum, 2018).

Penggunaan *OSM* yang tidak berkepentingan ataupun tujuan yang dianggap tidak benar yang dimisalkan mengambil data-data dalam jumlah yang besar, maka *OSM* akan menghalang atau *block* akses ke server-nya tanpa adanya peringatan, terdapat beberapa operasi untuk mengolah data seperti *create*, *read*, *update* ataupun *delete*.



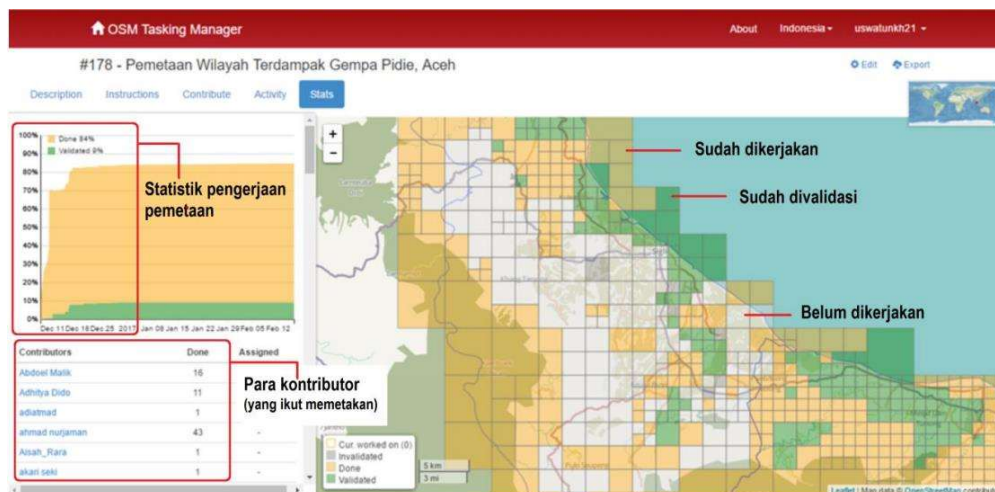
Gambar 2.10 Tampilan Halaman Awal *Open Street Map*
Sumber : Data Peneliti (2020)

OSM dapat diakses melalui <https://openstreetmap.org/> dan diperlukan sebuah autentikasi berupa nama dan kata sandi ataupun bisa dengan bantuan API OAuth yang bekerja dengan cara mendeteksi dan mencegah dua orang pengelola data (*mapper*) mengubah pada fitur yang sama dalam waktu bersamaan (Putra, Ernawati, & Coastera, 2016).

Sumber utama *OSM* mendapatkan data berasal dari citra satelit dengan resolusi yang tinggi (CSRT) dengan bantuan beberapa *provider* atau penyedia

seperti *bing aerial imagery*, atau *digital globe premium imagery* yang cakupan penggambarannya cukup luas.

Selain fitur CSRT, *OSM* pengerjaannya juga dapat diselesaikan dengan cepat berkat bantuan fitur *tasking manager* yaitu membagi beberapa area pemetaan menjadi ruang lingkup yang lebih kecil sehingga bisa dikerjakan lebih dari satu *mapper* untuk mengolah data secara bersamaan tanpa terjadinya tumpang tindih. Pemetaan yang dilakukan dengan *OSM* datanya dapat dibagikan atau dipublikasikan dengan mudah dan mengambil data dari *OSM* dapat diunduh dengan bentuk *shapefile* dan beberapa *format* peta digital lainnya yang dapat diolah sesuai kebutuhan pengguna data.



Gambar 2.11 Tampilan *Tasking Manager* Pada *OSM*
Sumber : (Nurrohmah & Sulistioningrum, 2018)

Tersedianya CSRT, layanan yang dapat memetakan secara partisipatif, dan kemudahan dalam menggunakan atau berbagi data membuat *OSM* menjadi salah satu peluang yang dapat dimanfaatkan penerapan teknologi dalam peta digital serta penentuan tempat yang terdekat.

2.1.6 OSMnx

Pada pembahasan awal yaitu mengenai *python* serta cara memasangnya, maka di sini akan membahas *tool* yang mendukung dalam bahasa pemrograman *python* yaitu *OSMnx*. *OSMnx* merupakan sebuah *tool* yang baru diciptakan oleh Geoff Boeing untuk memudahkan dalam pengambilan data berupa data visual jaringan jalan yang kompleks sesuai dengan singkatan namanya untuk pengambilan data pada *Open Street Map*.

Ada beberapa alasan yang diungkapkan oleh Geoff Boeing mengapa memilih *python* sebagai bahasa pemrogramannya yaitu (Boeing, 2019) :

1. *Python* merupakan bahasa pemrograman yang populer di dunia.
2. Sintaks dan kode programnya tidak sesulit bahasa pemrograman lainnya sehingga cocok untuk orang yang pertama kali ingin belajar bahasa pemrograman.
3. Merupakan bahasa standar dalam penelitian *data science* dan ekosistem serta paketnya yang cocok untuk saintifik, jaringan, dan analisis geospasial.
4. Adanya keberadaan *tool* sebelumnya seperti *dodgr*, *shp2graph*, dan *GISF2E* yang terintegrasi dengan *python* sehingga *OSMnx* merupakan pengembangan dari *tool-tool* sebelumnya.

Permasalahan yang dihadapi pada *tool – tool* sebelumnya pertama yaitu pengambilan data map yang terlalu besar akan memakan waktu yang lama serta adanya kesulitan dalam penyimpanan jenis filenya. Jika mengambil data visual

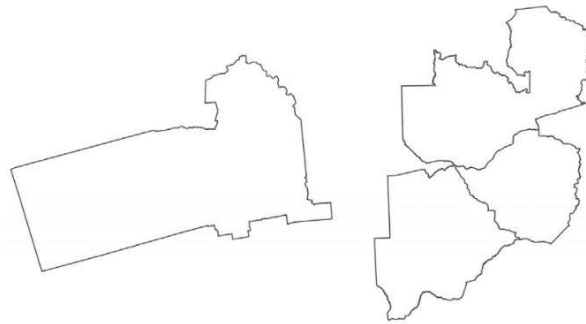
yang terlalu kecil dikhawatirkan tidak dapat mewakili suatu daerah serta belum terjamin reliabilitasnya.

Kedua yaitu penelitian tentang pemetaan yang biasanya merepresentasikan jalur-jalur jalan menjadi planar serta graf yang tidak berarah untuk memudahkan pengontrolan, biasanya para peneliti menyusun pemetaan menjadi dalam bentuk graf dari data Sistem Informasi Grafis yang diperoleh dengan menjadikan jalur sebagai sisi serta persimpangan sebagai titik, yang menjadikan masalah adalah jembatan ataupun jalan tol dianggap sebagai titik pisah yang direpresntasikan oleh titik (*node*). Hal ini sangat mempengaruhi kebenaran jalurnya.

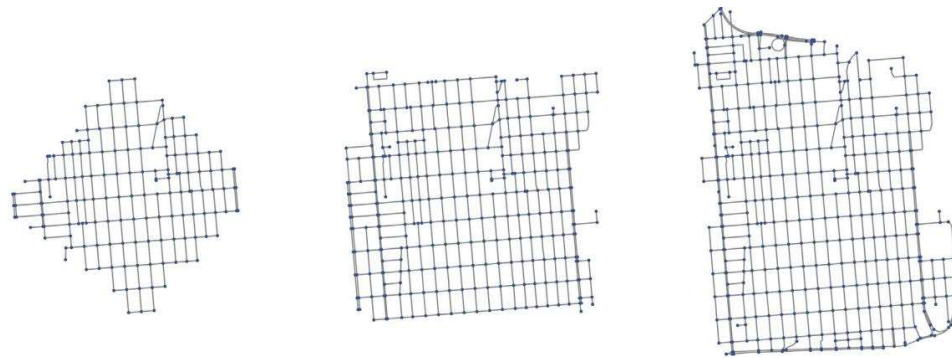
Ketiga dalam hal replikabilitas, beberapa penelitian dalam hal jalur tidak mementingkan arahnya. Arah mungkin tidak terlalu berdampak jika jalurnya untuk pejalan kaki, akan tetapi berdampak besar jika jalur yang dilalui kendaraan tidak memiliki arah yang tentu saja dapat mempengaruhi pada penelitian yang memfokuskan pada penentuan jalur.

Keempat yaitu sebelum *OSMnx* ditemukan, *tool-tool* yang ada tidak menawarkan teknik yang ideal dalam hal keseimbangan penggunaan, pengkreasian (*customizing*), penghasilan data yang mudah, dan skala dalam mengambil, membuat, serta menganalisis jaringan data yang ada. Adanya keterbatasan-keterbatasan tersebut membuat para peneliti dalam hal pemetaan bekerja tidak efisien serta memerlukan kerja yang lebih ekstra (Boeing, 2017).

OSMnx bersifat *open-source* bagi siapa saja yang ingin membantu mengembangkan ciptaannya dan dapat digunakan tanpa perlu mengeluarkan biaya serta dapat menampilkan data ke berbagai macam bentuk.



Gambar 2.12 Batas Wilayah Secara Geometri kota Berkeley, California (kiri) dan Negara Zambia, Zimbabwe, serta Botswana (kanan)
Sumber : (Boeing, 2017)



Gambar 2.13 Gambar graf dari jalur dan skala yang sama berdasarkan *network distance* (kiri), *bounding box* (tengah), dan *neighborhood polygon* (kanan)
Sumber : (Boeing, 2017)

OSMnx dapat diunduh dan tersedia secara *online* di <https://github.com/gboeing/osmnx>.

2.1.7 *NetworkX* dan *Matplotlib*

Python memiliki berbagai macam paket yang bisa diimplementasikan ke dalamnya, selain *OSMnx* terdapat satu paket yang digunakan untuk membantu dalam pembuatan, mengolah, dan mempelajari struktur dinamika fungsi jaringan yang kompleks yaitu *NetworkX*. Paket ini diciptakan pada tahun 2002 dan dirilis untuk publik pada tahun 2005 oleh Aric Hagberg, Dan Schult, dan Pieter Swart.

Pada dasarnya *NetworkX* bekerja dalam hal graf sebagai objeknya yang dapat memanipulasi serta membuat laporan yang berkaitan dengan jaringan. Beberapa tipe dasar graf yang terdapat pada *python* sebagai berikut :

1. *Graph*, kelas ini digunakan untuk mengimplementasikan graf yang tidak berarah, selain itu juga mengabaikan sisi pada antara kedua titik.
2. *DiGraph* (*Directed graph*), kelas ini digunakan untuk mengimplementasikan graf berarah.
3. *MultiGraph*, kelas ini merupakan graf yang fleksibel dan dapat menghubungkan beberapa graf yang tidak berarah.
4. *MultiDiGraph*, merupakan versi graf yang berarah pada *MultiGraph*

NetworkX tidak hanya digunakan untuk membantu pembuatan graf, tetapi juga dapat mengimplementasikan algoritma-algoritma seperti salah satunya Dijkstra, menggambar jaringan yang sederhana serta bertindak sebagai struktur data yang menggunakan prinsip “*dictionary of dictionaries of dictionaries*” sebagai jaringan dasar struktur datanya yang dapat mempercepat proses pemanggilan dalam pembuatan jaringan (Hagberg, Schult, & Swart, 2020).

Paket ini juga dapat digunakan sebagai pelengkap untuk membantu pengelompokan seperti *data mining* yang digunakan sebagai pustakanya pada pengelompokan graf yang dinamai *spectral clustering* yaitu pengelompokan data didasarkan atas kesamaan ataupun keterkaitan antara setiap data (Bauckhage, 2016).

Pelengkap lainnya yang tidak kalah penting yaitu *matplotlib*. *Matplotlib* merupakan paket yang disediakan untuk *python* sehingga data yang ditampilkan

terlihat lebih interaktif dan berwarna. *Matplotlib* dapat membantu pada bidang kecerdasan buatan, diagram, data statistik serta peta. Pada penelitian ini, peneliti hanya memanfaatkan sebagian dari kegunaan *matplotlib* yang ada hanya sebatas untuk penggambaran garis jalur yang menghubungkan dari titik awal ke titik akhir.

Memadukan *NetworkX* pada *python* maka akan membantu perhitungan masalah saintifik dalam menganalisis jaringan seperti *breadth first search*, *clustering*, dan jalur terpendek.

2.1.8 Conda Package Manager

Kemudahan-kemudahan paket yang ada *python* dapat membantu dalam proses pengelolaan data baik untuk visualisasi ataupun perhitungan pada penelitian yang akan dilakukan. Akan tetapi, diperlukan suatu media atau alat untuk menambahkan paket-paket tersebut kepada *python* agar dapat digunakan secara maksimal.

Conda package manager merupakan sebuah *tool* yang dikembangkan oleh Anaconda Inc. yang mempunyai beberapa kegunaan yang dapat membantu dalam kemudahan menambahkan paket pada *python* seperti instalasi atau menghapus paket, membuat atau menghapus *environment* ataupun melakukan pembaruan paket yang sudah terpasang (Adriyan, 2019).

Menggunakan bantuan *conda package manager*, maka *OSMnx*, *networkX* serta *matplotlib* dapat terpasang dengan mudah untuk membantu proses pengolahan data pada *python*.

2.2 Penelitian Terdahulu

Penelitian ini tidak akan berjalan tanpa adanya jurnal-jurnal atau sumber yang mendukungnya, beberapa topik yang berhubungan dengan judul penelitian yang selanjutnya peneliti gunakan sebagai bahan referensi dalam penulisan skripsi ini sebagai berikut :

1. (Ratnasari et al., 2013) **Penentuan Jarak Terpendek dan Jarak Terpendek Alternatif Menggunakan Algoritma Dijkstra Serta Estimasi Waktu Tempuh.**

ISBN: 979-26-0266-6. Permasalahan dalam hal manajemen waktu merupakan hal yang diperhatikan dalam berwisata, banyaknya jalur alternatif menuju suatu tempat memunculkan kesulitan menentukan jalur mana yang tepat dan juga memungkinkan jalur yang dipilih terdapat suatu hambatan sehingga dapat menghabiskan waktu hanya untuk menentukan jalur mana yang akan dilewati.

Penelitian yang dilakukan oleh Asti Ratnasari, Farida Ardiani, dan Fery Nurvita A. menerapkan Pemrograman Berorientasi Objek (PBO) dalam pembuatan sistemnya dengan menggunakan bahasa pemrograman Delphi dan MySQL sebagai penyimpanan basis datanya, penggunaan Delphi diprioritaskan karena dinilai dapat membuat aplikasi efisien dengan minimal *coding* manual.

Uji sistem dilakukan dengan pembuatan graf tidak berarah dengan studi kasus *non-real* tempat wisata di Yogyakarta yang diwakili sebanyak lima titik. Sistem yang dirancang dapat menentukan jarak terdekat mana yang akan dilalui serta jalur alternatif lainnya jika jalan utama terjadi halangan seperti kemacetan. Menerapkan algoritma Dijkstra dapat menentukan titik-titik mana yang akan

dilalui sehingga dapat sampai ke tempat tujuan wisata dengan jarak terpendek dan membutuhkan waktu yang tidak lama dengan menentukan berapa kecepatannya.

2. (Primadasa, 2015) **Pencarian Rute Terpendek Menggunakan Algoritma Dijkstra Pada SIG Berbasis Web Untuk Distribusi Minuman (Studi Kasus PT. Coca-Cola Kota Padang)**. ISSN: 2356-0010. Pendistribusian barang merupakan kegiatan logistik yang perlu memperhatikan jalur mana yang akan dilalui serta waktunya karena tidak hanya menuju satu tempat melainkan banyak tempat yang perlu dilalui.

Penelitian yang dilakukan Yogi Primadasa mengambil data peta pada *Google Map* dengan *software* pendukungnya berupa *MapInfo Professional 10.0* yang sudah disediakan juga untuk pembuatan basis datanya dan memetakan delapan tempat yang digambarkan dengan titik untuk mendistribusikan cola.

Aplikasi Sistem Informasi Grafis (SIG) dibuat dengan berbasis *web* untuk menentukan jalur terdekat yang akan dilalui oleh distributor dan dilengkapi dengan gambar lokasi tujuannya tampak depan sehingga tidak akan kebingungan untuk menentukan ketepatan tempatnya. Informasi merupakan hal yang penting untuk menyusun strategi apa dan bagaimana menyelesaikan suatu permasalahan dengan memperhatikan hal seperti waktu sehingga mendapatkan jalur rute terpendek yang dapat mempercepat proses pendistribusian.

3. (Retnani et al., 2015) **Pencarian SPBU Terdekat dan Penentuan Jarak Terpendek Menggunakan Algoritma Dijkstra (Studi Kasus di Kabupaten Jember)**. ISSN: 2302-2949. Memiliki objek-objek wisata yang beragam dapat menarik minat para wisatawan untuk berkunjung ke tempat secara langsung.

Kabupaten Jember merupakan salah satu destinasi tempat yang dikunjungi banyak orang karena memiliki banyak objek wisata, selain itu Jember juga merupakan jalur alternatif untuk dilewati saat mudik. Sebelum berpergian, maka kita harus mempersiapkan kebutuhan yang diperlukan selama perjalanan seperti bekal, tempat akomodasi bahkan hal yang kecil seperti bahan bakar perlu mendapat perhatian.

Penelitian yang dilakukan Windi Eka Yulia R., Dwiretno Istiadi, serta Abdul Roqib menggunakan *OpenStreetMap* sebagai tampilan pemetaan datanya dan merancang aplikasi berbasis *web* yang dinamakan *SPBU Finder* dengan SPBU yang dipilih sebanyak 33 tempat. Perhitungan jarak dilakukan oleh sistem dengan memilih posisi *user* pada peta dan menggunakan 2 buah parameter yaitu parameter *geometry* awal berupa *user* dan *geometry* akhir yang merupakan posisi SPBU terdekat dengan menggunakan bantuan rumus *Haversine* untuk mendapatkan perbandingan jaraknya.

Untuk menguji ketepatan perhitungan serta implementasi algoritma Dijkstra, penentuan jarak terpendek menggunakan bantuan *software* berupa *QuantumGIS* dan didapatkan hasilnya sudah benar. Mengetahui titik tempat pengisian bahan bakar akan membantu kelancaran kita selama berwisata sehingga kita tidak akan kekurangan informasi pada daerah yang baru kita kunjungi.

4. (Putra et al., 2016) **Penerapan *Open Street Map* Untuk Mencari Lokasi ATM Terdekat Dengan Algoritma Kruskal Berbasis *Smartphone Android* (Studi Kasus: Lokasi ATM di Kota Bengkulu)**. ISSN: 2303-0755. Tempat untuk menyettor atau menarik uang (ATM) merupakan fasilitas publik yang akan

selalui dicari oleh banyak orang untuk mendukung kegiatan sehari-hari. Kedatangan orang-orang dari luar kota menyebabkan jumlah penduduk di tempat asal bertambah sehingga yang bukan penduduk asli membutuhkan informasi tempat-tempat publik yang penting seperti ATM dengan bantuan peta.

Penelitian yang dilakukan Edwin Dwi Anggara Putra, Ernawati, dan Funny Farady Coastera menggunakan bantuan aplikasi *Eclipse* dan sumber data yang didapatkan berasal dari *OpenStreetMap* dan melakukan survei objek pengamatan dengan cara melakukan *tracking* bantuan GPS untuk mendapatkan data titik koordinat ATM. Untuk menguji keakuratan jarak antara yang dihasilkan oleh sistem dengan sesungguhnya, maka perhitungan manual dilakukan dengan menggunakan bantuan *spidometer* pada motor. Pemanfaatan peta digital dapat membantu orang-orang mencari informasi keberadaan tempat yang akan dikunjungi sehingga dapat menentukan jalan mana yang akan dilalui.

5. (Harahap & Khairina, 2017) **Pencarian Jalur Terpendek dengan Algoritma Dijkstra**. ISSN: 2541-2019. Meminimalkan pengeluaran merupakan cara yang dilakukan untuk penghematan baik dari segi waktu ataupun sumber daya lainnya. Menentukan jalur terpendek tidak mudah karena banyaknya jumlah jalan yang menghubungkan ke tempat tujuan yang kita inginkan sehingga kita kesulitan menentukan jalur mana yang akan dilalui, permasalahan ini membutuhkan sebuah tata cara atau instruksi yang dapat menyelesaikan permasalahan dengan terurut dan logis untuk menentukan jalur yang tepat dari titik awal ke titik akhir.

Penelitian yang dilakukan M. Khoiruddin Harahap dan Nurul Khairina menggunakan bahasa pemrograman VB. Net serta data-data yang diolah berupa *non-real* atau hanya simulasi yang sudah ditentukan untuk menentukan jarak terdekat suatu titik serta perhitungannya dengan cara iterasi, menggunakan metode Dijkstra dapat menentukan jalur terpendek serta membanding nilai jarak tersebut.

6. (Boeing, 2017) ***OSMnx: New Methods for Acquiring, Constructing, Analyzing, and Visualizing Complex Street Networks***. ISSN: 0198-9715. Mendapatkan data berupa visual untuk pemetaan dapat dilakukan dengan bantuan *tool-tool* yang tersedia seperti QGIS, akan tetapi kesulitan mulai dihadapi pada saat proses pengambilan data yang tidak banyak tetapi memakan kapasitas memori yang besar.

Penelitian yang dilakukan oleh Geoff Boeing merupakan sebuah penemuan untuk mendapatkan data yang dikhususkan pada *Open Street Map* dengan bantuan bahasa pemrograman *python*. Kemudahan dan efisiensi kerja juga perlu diperhatikan mengingat data yang diambil beragam sesuai kebutuhan pengguna terutama dalam jumlah yang banyak sehingga *OSMnx* hadir untuk menjawab tantangan yang dihadapi seperti cara mendapatkan data yang lebih mudah. *OpenStreetMap* dipilih sebagai sumber datanya karena sifatnya yang *open source* serta regulasi yang tidak sulit untuk mengambil datanya.

7. (Boeing, 2019) ***Street Network Models and Measures for Every U.S. City, County, Urbanized Area, Census Tract, and Zillow-Defined Neighborhood***. ISSN: 2413-8851. Pengembangan *OSMnx* tidak berhenti hanya pada cara

mendapatkan, menampilkan serta menganalisis jalur ataupun pemetaan di suatu daerah. Geoff Boeing memaparkan secara langsung pada penelitian ini dan mengungkapkan menggunakan *python* sebagai bahasa pemrogramannya dikarenakan bahasa *python* dapat dipelajari baik untuk pemula ataupun yang ingin mengembangkan kemampuannya terutama dalam bidang geografis datar.

Penelitian ini mengambil dan memanfaatkan data sensus penduduk di Amerika, *OSMnx* dapat menampilkannya dengan visual warna yang menandakan kepadatan daerah penduduk ditandai dengan semakin tebal suatu warna berarti daerah tersebut padat penduduknya. Selain itu, dapat menampilkan gambaran sesuai dengan kategori yang diinginkan seperti pemetaan daerah antar perkotaan.

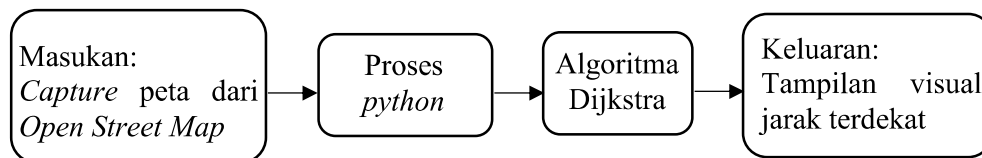
8. (Lanning, Harrell, & Wang, 2014) *Dijkstra's Algorithm and Google Maps*. ISBN: 978-14-5032-9231. Perkembangan teknologi membuat segala kegiatan kita dipermudah, seperti untuk mencari petunjuk arah jalan maka kita menggunakan peta. Peta tidak lagi dalam wujud fisik yang terbuat dari kertas, melainkan sudah bertransformasi menjadi digital dan bisa digunakan pada gawai kita. Aplikasi peta digital yang terkenal dan hampir digunakan oleh banyak orang yaitu *Google Maps*.

Penelitian yang dilakukan Daniel R. Lanning, Gregory K. Harrell, serta Jin Wang membuktikan bahwa algoritma yang digunakan pada *Google Maps* yaitu algoritma Dijkstra untuk menentukan ataupun menemukan solusi yang terbaik jalan terpendek yang dilalui. Penelitian ini mensimulasikan dimulai dari tempat mereka berkuliah yaitu Valdosta mengenai jalur penerbangan waktu tercepat yang dilalui pada bandara dengan mengambil beberapa tempat yang dilalui seperti : Atlanta, Jacksonville, New York, dan Phoenix menggunakan bahasa

pemrograman *java*. Selain itu, penelitian ini mengungkapkan dengan kemajuan teknologi maka kita dapat menyerahkan kepada komputer untuk melakukan fungsi komputasi atau menghitung untuk memproses algoritma Dijkstra.

2.3 Kerangka Pemikiran

Setelah memaparkan teori-teori di atas, penelitian ini menggunakan kerangka pemikiran yang dirancang oleh peneliti sebagai berikut :



Gambar 2.14 Kerangka Pemikiran
Sumber : Data Peneliti (2020)