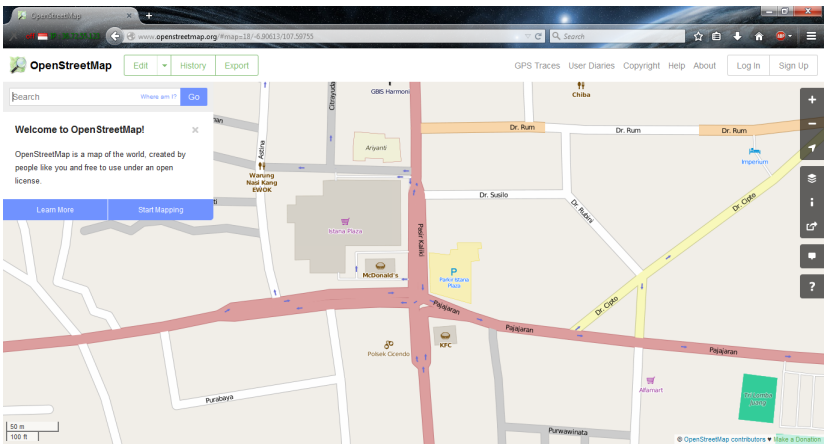
BAB 1

PENDAHULUAN

1.1 Latar Belakang

Mengemudi merupakan salah satu pilihan bagi masyarakat untuk bepergian dari suatu tempat ke tempat lain yang dituju. Contohnya adalah seorang wanita karir yang mengemudikan kendaraan pribadi dari rumah menuju kantor atau tempat kerjanya. Contoh lainnya adalah seorang sopir taksi yang mengemudikan kendaraannya untuk mengantar penumpang hingga sampai ke tujuan. Untuk dapat sampai ke titik tujuan, banyak rute yang dapat dilalui oleh seorang pengemudi. Seorang pengemudi, tentu saja akan mencari rute terdekat yang dapat dilalui, hal tersebut bertujuan untuk menghemat penggunaan bahan bakar dan juga waktu. Pemilihan rute terdekat untuk dapat sampai ke tujuan menjadi cukup penting, karena saat ini mobilitas masyarakat yang semakin tinggi. Aplikasi pencarian rute terdekat dapat membantu seorang pengemudi untuk menemukan rute terdekat untuk sampai ke tempat tujuan lebih cepat. Dengan cara menunjukkan rute menyetir terdekat dari satu tempat ke tempat lain.

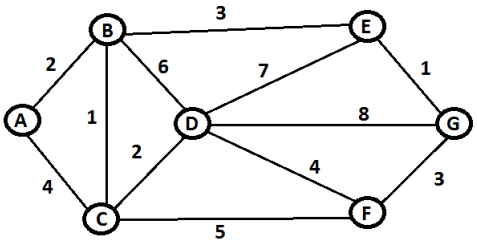
Aplikasi yang dibuat akan berbasis OpenStreetMap dan menggunakan algoritma Dijkstra. OpenStreetMap adalah portal peta terbuka yang menyediakan data dalam bentuk peta maupun XML, pengguna dapat mencari lokasi dan memilih area yang diinginkan. Setelah pengguna memilih area yang diinginkan, pengguna dapat menggunakan fitur export untuk mengunduh data XML pada area tersebut. Tampilan website OpenStreetMap dapat dilihat pada Gambar 1.1. Data yang disediakan oleh OpenStreetMap dalam bentuk XML biasa



Gambar 1.1: Tampilan website OpenStreetMap [1]

[1] http://www.openstreetmap.org

disebut dengan OpenStreetMap XML dan disingkat menjadi OSMXML. OSMXML adalah dokumen XML yang berisi data-data peta OSM. Pada dasarnya, OSMXML berisi data primitif (node, way, dan relation) yang merupakan arsitektur dari model OSM. Node dapat diartikan sebagai titik pada peta dijital, way merupakan informasi garis pada peta yang melambangkan jalan atau elemen lain seperti rel kereta, dan relation memberikan informasi node-node yang bersinggungan, elemen relation dapat menggambarkan suatu area seperti lapangan, taman bermain, rute bus, dan lain-lain. Sedangkan algoritma Dijkstra adalah algoritma untuk mencari jarak terpendek pada sebuah graf berarah dengan bobot yang bernilai tidak negatif pada setiap sisinya [1]. Graf adalah himpunan objek yang terdiri dari simpul(node) dan sisi (edge), graf digambarkan sebagai kumpulan titik yang dihubungkan oleh garis. Contoh graf dapat dilihat pada Gambar 1.2.



Gambar 1.2: Contoh Graf

Aplikasi yang dibuat akan mengolah data yang disediakan oleh OpenStreetMap dalam bentuk XML dan memodelkannya ke dalam bentuk graf. Selanjutnya akan digunakan algoritma Dijkstra untuk mencari rute terdekat pada graf tersebut dan menunjukkan hasilnya secara visual.

1.2 Rumusan Masalah

Berdasarkan latar belakang, maka rumusan masalah berikut:

- Bagaimana cara memodelkan data OSMXML menjadi sebuah graf?
- Bagaimana cara menggunakan atau mengimplementasikan algoritma Dijkstra pada sebuah graf untuk mencari rute terdekat?
- Bagaimana cara membuat visualisasi graf dan rute terdekat pada peta dijital?

1.3 Tujuan

Berdasarkan rumusan masalah yang telah diuraikan di atas, maka tujuan dari penelitian yang dilakukan adalah:

- Mengetahui cara memodelkan data OSMXML menjadi sebuah graf.
- Mempelajari cara kerja algoritma Dijkstra dan mengimplementasikannya pada sebuah graf.

- Mempelajari cara membuat visualisasi graf dan rute terdekat pada peta dijital.

1.4 Batasan Masalah

Batasan permasalahan dari pembuatan aplikasi ini adalah :

- Aplikasi tidak mencari rute terdekat kedua dan seterusnya.

1.5 Metodologi Penelitian

Langkah-langkah yang akan dilakukan dalam melakukan penelitian adalah :

1. Melakukan studi pustaka untuk mengetahui teori-teori yang dapat mendukung proses pembuatan aplikasi pencarian rute terdekat.
2. Melakukan analisis teori-teori yang mendukung proses pembuatan aplikasi.
3. Membuat rancangan aplikasi.
4. Melakukan implementasi berdasarkan rancangan yang telah dibuat.
5. Melakukan pengujian aplikasi.
6. Melakukan pengambilan kesimpulan berdasarkan pengujian yang telah dilakukan.

1.6 Sistematika Pembahasan

Pada setiap bab akan dibahas beberapa hal sebagai berikut :

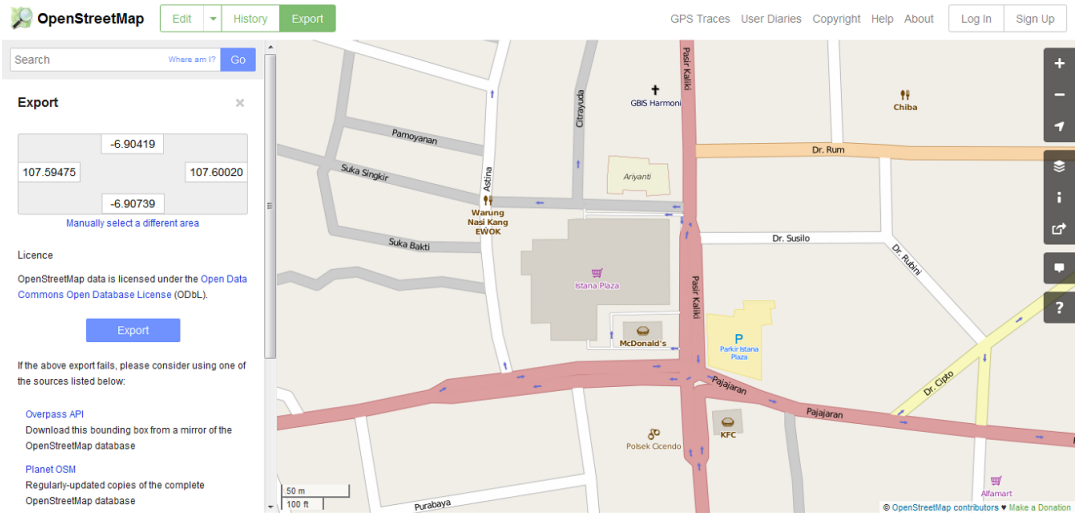
1. Bab Pendahuluan
   Bab 1 berisi latar belakang, rumusan masalah, tujuan, batasan masalah, metodologi penelitian, dan sistematika pembahasan.
2. Bab Dasar Teori
   Bab 2 berisi teori-teori dasar mengenai OpenStreetMap, algoritma Dijkstra, Google Map Api, Graf, XML, dan beberapa teori lain yang mendukung pembuatan aplikasi.
3. Bab Analisis
   Bab 3 berisi deskripsi sistem yang akan dibuat, analisis dasar teori, dan analisis cara kerja algoritma Dijkstra pada graf.
4. Bab Perancangan
   Bab 4 berisi perancangan antarmuka aplikasi disertai beberapa gambar.
5. Bab Implementasi dan Pengujian
   Bab 5 berisi hasil implementasi yang dilakukan disertai dokumentasi mengenai penjelasan aplikasi tersebut dan hasil pengujian yang dilakukan berupa *screenshot*
6. Bab Kesimpulan dan Saran
   Bab 6 berisi kesimpulan dari seluruh hasil penelitian dan saran untuk pengembangan aplikasi yang akan datang.

BAB 2

DASAR TEORI

2.1 OpenStreetMap

OpenStreetMap (OSM) adalah portal peta terbuka yang menyediakan data dalam bentuk peta atau XML [2]. OSM menyediakan peta dijital dan dapat diedit dari seluruh dunia, juga memungkinkan pengguna untuk mengakses gambar peta yang terdapat pada situs www.openstreetmap.org secara gratis. OSM terbentuk dan mendapatkan datanya dari berbagai sukarelawan yang bersedia untuk berkontribusi, misalnya para pengguna OSM yang menggunakan aplikasi untuk mengedit peta dan mengunggah data yang telah diedit ke situs OSM. Selain itu, OSM menyediakan beberapa aplikasi bagi para pengguna untuk mengedit peta, seperti iD online editor dan JOSM. Untuk mendapatkan gambar peta ataupun data peta dalam bentuk lain, pengguna dapat menggunakan fitur export pada situs OSM. Fitur export pada situs OSM dapat dilihat pada Gambar 2.1.



Gambar 2.1: Ekspor data pada situs OpenStreetMap

Berikut ini adalah beberapa data yang dapat diambil menggunakan fitur export [?]:

1. OpenStreetMap XML Data

   OSM XML data dapat diperoleh dengan cara menggunakan tombol Export di bagian atas untuk membuka sidebar. Tombol Export mengarahkan langsung browser kepada OpenStreetMap API yang menyediakan data mentah OSM dalam bentuk XML.

2. Mapnik *Image*

Memungkinkan ekspor data OSM dalam bentuk PNG, JPEG, SVG, PDF dan peta PostScript.

3. *Embeddable* HTML

   Fitur ini memungkinkan pengguna untuk mendapatkan kode HTML yang dapat disalin dan digunakan pada halaman web lain. Kode HTML tersebut akan menyisipkan peta dalam sebuah iframe lengkap dengan javascript.

2.2 XML

XML adalah singkatan dari eXtensible Markup Language, XML adalah bahasa markup yang dikembangkan oleh W3C (World Wide Web Consortium) [3]. Berikut ini adalah contoh dokumen XML:

```
<?xml version="1.0" encoding="utf-8"?>
  <catalog>
   <book id="bk101">
       <author>Gambardella, Matthew</author>
       <title>XML Developer's Guide</title>
       <genre>Computer</genre>
       <price>44.95</price>
       <publish_date>2000-10-01</publish_date>
       <description>An in-depth look at creating applications
       with XML.</description>
   </book>
   <book id="bk102">
       <author>Ralls, Kim</author>
       <title>Midnight Rain</title>
       <genre>Fantasy</genre>
       <price>5.95</price>
       <publish_date>2000-12-16</publish_date>
       <description>A former architect battles corporate zombies,
       an evil sorceress, and her own childhood to become queen
       of the world.</description>
   </book>
  <catalog>
```

Contoh di atas memberikan informasi mengenai katalog buku yang disimpan pada dokumen XML. Pada awal dokumen tertera versi XML dan *encoding* yang digunakan. Setelah itu, terdapat tag catalog yang memiliki *child* yaitu tag buku beserta informasinya. Terdapat informasi id buku yang tertera pada atribut tag buku, seperti <book id="bk101">. Dan juga informasi lain seperti judul buku, penulis, genre, harga, tanggal terbit, dan deskripsi.

XML dikembangkan terutama untuk mengatasi keterbatasan pada HTML (Hypertext Markup Language). HTML adalah salah satu bahasa markup yang paling populer dan terus dikembangkan, banyak tag baru yang diperkenalkan. Pada versi pertama, HTML memiliki satu lusin tag dan pada HTML pada versi 4.0 sudah hampir mencapai seratus

tag. Namun, pada aplikasi seperti *electronic commerce* dibutuhkan tag lebih untuk produk, harga, nama, alamat, dan banyak lagi atau situs *streaming* memerlukan tag lebih untuk mengontrol gambar dan suara.

HTML telah berkembang menjadi bahasa yang cukup kompleks, W3C memperkirakan penggunaan komputer akan terus berkurang dan penggunaan gadget seperti smartphone akan bertambah. Mesin tersebut tidak sekuat PC dan tidak bisa memproses bahasa yang kompleks seperti HTML . Meskipun HTML adalah bahasa yang populer dan cukup sukses, HTML memiliki beberapa kelemahan utama dan XML dikembangkan untuk mengatasi kelemahan tersebut. XML adalah bahasa yang digunakan untuk menggambarkan dan memanipulasi dokumen terstruktur. Perubahan utama pada XML adalah tidak adanya tag yang ditetapkan pada XML. Karena tidak ada tag yang ditetapkan, penulis dapat membuat tag yang dibutuhkan. Beberapa ketentuan pada XML dapat dilihat pada uraian berikut:

1. Tag pada XML
   Setiap elemen pada XML terdiri dari nama dan nilai, selain itu harus memiliki tag pembuka dan tag penutup. Contoh:

   ```
   <tel> 513-555-7098 </ tel>
   ```

   Elemen untuk menyimpan nomor telepon memiliki nama tag tel, ditulis dengan <tel> dan ditutup dengan </tel>.

2. Nama pada XML
   Pemberian nama pada XML harus dimulai dengan huruf atau underscore (_) dan sisanya diikuti huruf, angka, atau titik. Spasi tidak diperbolehkan pada pemberian nama.

3. Atribut
   Atribut memungkinkan untuk menyisipkan informasi tambahan, atribut juga memiliki nama dan nilai. Contoh:

   ```
   <tel preferred="true">513-555-8889</tel>
   <tel>513-555-7098</tel>
   ```

   Elemen tel dapat memiliki atribut *preferred*, memberikan informasi nomor telepon yang lebih sering digunakan.

4. Elemen Kosong
   Elemen yang tidak memiliki nilai atau isi disebut sebagai elemen kosong. Elemen kosong biasanya memiliki atribut. Contoh:

   ```
   <email href="mailto:jdoe@emailaholic.com"></email>
   ```

   Elemen email tidak memiliki nilai atau isi.

5. *Nesting of Elements*
   Sebuah elemen dapat memiliki elemen lain di dalamnya. Elemen yang berada di dalam elemen lain disebut *child*, sedangkan elemen yang memiliki elemen lain disebut *parent*. Contoh

```
<name>
  <fname>Jack</fname>
  <lname>Smith</lname>
</name>
```

Pada contoh berikut elemen name memiliki dua *child* yaitu fname dan lname dan elemen name merupakan *parent* dari kedua elemen tersebut.

6. *Root*
   Root merupakan elemen level tertinggi, pada dokumen XML harus ada satu elemen pada level tertinggi. Dengan kata lain, elemen lain harus menjadi *child* dari *root*.

7. Deklarasi XML
   Deklarasi XML dituliskan pada baris pertama dokumen. Pada deklarasi tersebut juga dituliskan versi XML yang digunakan. Contoh:

```
<?xml version="1.0"?>
```

2.2.1 OSMXML

OpenStreetMap XML atau biasa disingkat dengan OSMXML merupakan dokumen XML yang berisi data-data peta OSM. Pada dasarnya, OSMXML berisi data primitif (node, way, dan relation) yang merupakan arsitektur dari model OSM [**?**]. Berikut ini adalah contoh dokumen OSMXML:

```
<?xml version="1.0" encoding="UTF-8"?>
<osm version="0.6" generator="CGImap 0.0.2">
 <bounds minlat="54.0889580" minlon="12.2487570" maxlat="
    54.0913900" maxlon="12.2524800"/>
 <node id="298884269" lat="54.0901746" lon="12.2482632" user="
    SvenHRO" uid="46882" visible="true" version="1" changeset="
    676636" timestamp="2008-09-21T21:37:45Z"/>
 <node id="261728686" lat="54.0906309" lon="12.2441924" user="
    PikoWinter" uid="36744" visible="true" version="1" changeset="
    323878" timestamp="2008-05-03T13:39:23Z"/>
 <node id="1831881213" version="1" changeset="12370172" lat="
    54.0900666" lon="12.2539381" user="lafkor" uid="75625" visible
    ="true" timestamp="2012-07-20T09:43:19Z">
  <tag k="name" v="Neu Broderstorf"/>
  <tag k="traffic_sign" v="city_limit"/>
 </node>
 ...
 <node id="298884272" lat="54.0901447" lon="12.2516513" user="
    SvenHRO" uid="46882" visible="true" version="1" changeset="
    676636" timestamp="2008-09-21T21:37:45Z"/>
 <way id="26659127" user="Masch" uid="55988" visible="true"
    version="5" changeset="4142606" timestamp="2010-03-16
    T11:47:08Z">
```

```
  <nd ref="292403538"/>
  <nd ref="298884289"/>
  ...
  <nd ref="261728686"/>
  <tag k="highway" v="unclassified"/>
  <tag k="name" v="Pastower StraÃ§e"/>
 </way>
 <relation id="56688" user="kmvar" uid="56190" visible="true"
    version="28" changeset="6947637" timestamp="2011-01-12
    T14:23:49Z">
  <member type="node" ref="294942404" role=""/>
  ...
  <member type="node" ref="364933006" role=""/>
  <member type="way" ref="4579143" role=""/>
  ...
  <member type="node" ref="249673494" role=""/>
  <tag k="name" v="KÃijstenbus Linie 123"/>
  <tag k="network" v="VVW"/>
  <tag k="operator" v="Regionalverkehr KÃijste"/>
  <tag k="ref" v="123"/>
  <tag k="route" v="bus"/>
  <tag k="type" v="route"/>
 </relation>
 ...
</osm>
```

Struktur OSMXML:

- Dokumen OSMXML diawali dengan tag xml yang menjelaskan versi xml dan encoding yang digunakan, pada contoh di atas digunakan xml versi 1.0 dan encoding UTF-8.
- Elemen osm memberikan informasi mengenai versi API dan generator yang digunakan. Generator adalah alat untuk membuat dokumen XML pada saat fitur export digunakan.
- Elemen bound memberikan informasi mengenai cakupan area pada dokumen XML tersebut. Dilengkapi dengan atribut koordinat yaitu latitude dan longitude. Data primitif pada OSM dibagi menjadi 3 bagian, yaitu node, way, dan relation.
    1. Elemen Node merupakan informasi titik pada sebuah peta. Node memiliki beberapa atribut yaitu:
        - id
          Merupakan id dari node tersebut.
        - user
          Merupakan user yang melakukan editing pada node.
        - uid
          Id dari user.

- lat
  berisi informasi koordinat pada garis lintang.
- lon
  berisi informasi koordinat pada garis bujur.
- timestamp
  Berisi informasi waktu saat node tersebut diperbaharui.

Node juga memiliki elemen tag sebagai *child* yang memberikan informasi tambahan pada node tersebut, contoh:

```
<tag k="name" v="Neu Broderstorf"/>
```

nama dari node tersebut adalah Neu Broderstorf.

2. Elemen Way merupakan informasi garis yang dapat diartikan sebagai jalan ataupun elemen lain seperti rel kereta pada peta OSM. Way menyimpan informasi node-node yang dilalui oleh garis dan juga sama seperti node dilengkapi atribut seperti id, uid, user, changeset, timestamp. Elemen way memiliki *child* elemen nd, contoh:

   ```
   <nd ref="292403538"/>
   ```

   atribut ref pada elemen nd mengacu pada node yang memiliki id 292403538, dan elemen tag yang memberikan informasi tambahan pada elemen way,
3. Elemen relation menyimpan informasi node-node dan way yang bersinggungan. Elemen relation dapat menggambarkan suatu area seperti lapangan, taman bermain, atau pada contoh di atas menggambarkan rute bus.

2.3 Javascript

Javascript adalah bahasa pemrograman web yang mulai dikembangkan di perusahaan yang bernama Netscape. Javascript memiliki lisensi dari Sun Microsystems yang sekarang sudah berganti nama menjadi Oracle. Saat ini, mayoritas situs web sudah menggunakan javascript. Berikut ini adalah contoh penggunaan javascript pada dokumen HTML:

```
<!DOCTYPE html>
<html>
<head>
<script>
function myFunction() {
    document.getElementById("demo").innerHTML = "Paragraph changed
        .";
}
</script>
</head>
<body>
<h1>JavaScript in Head</h1>
<p id="demo">A Paragraph.</p>
<button type="button" onclick="myFunction()">Try it</button>
```

```
</body>
</html>
```

Pada contoh di atas terdapat fungsi yang ditulis menggunakan javascript, fungsi tersebut akan mengubah string "A Paragraph" pada tag <p> menjadi "Paragraph changed" jika *button* atau tombol "Try it" di klik.

Seluruh browser yang terdapat pada komputer, konsol game, tablet, dan smartphone sudah disertai dengan javascript interpreter. Interpreter adalah suatu program yang berfungsi untuk menerjemahkan kode program ke dalam bahasa mesin. Javascript adalah bagian yang cukup penting pada sebuah halaman web, jika HTML berfungsi untuk menentukan isi dari halaman dan CSS untuk menentukan tampilan pada halaman, javascript berfungsi untuk menentukan "behavior" dari halaman web tersebut [4]. Berikut ini adalah uraian dari struktur javascript dan beberapa contoh sintaks:

1. Struktur

   - *Character Set*
     Javascript ditulis menggunakan karakter Unicode. Unicode adalah superset ASCII dan Latin-1 yang mendukung hampir seluruh bahasa di dunia.
   - *Comments*
     Javascript mendukung 2 jenis komentar yaitu komentar yang diletakkan setelah garis miring ganda // dan komentar yang diletakkan antara karakter /* dan */.

     ```
     // This is a single-line comment.
     /* This is also a comment */ // and here is another comment.
     /*
     * This is yet another comment.
     * It has multiple lines.
     */
     ```

   - Literal
     Literal adalah notasi untuk merepresentasikan nilai dan nilai yang dituliskan akan muncul secara langsung dalam program. Literal dapat berupa karakter, bilangan bulat, bilangan real, boolean. Berikut ini adalah contoh literal:

     ```
     12 // The number twelve
     1.2 // The number one point two
     "hello world" // A string of text
     'Hi' // Another string
     true // A Boolean value
     false // The other Boolean value
     /javascript/gi // A "regular expression" literal (for pattern matching)
     null // Absence of an object
     ```

   - *Identifier*
     Identifier pada javascript hanyalah nama yang digunakan untuk memberi nama pada variabel atau fungsi. Digit tidak diperbolehkan sebagai karakter pertama pada *identifier*.

- *Reserved words*
  Reserved words adalah kata-kata yang tidak dapat digunakan sebagai identifier, karena digunakan oleh javascript sebagai keyword. Beberapa contoh keyword seperti break, delete, if, null, true, false, try, dan lain-lain.
- *Optional Semicolons*
  Seperti banyak bahasa pemrograman lain, javascript menggunakan titik koma (;) untuk memisahkan perintah yang ditulis. Hal ini penting untuk membuat kode program menjadi jelas mengenai awal dan akhir. Pada javascript, titik koma dapat dihilangkan jika perintah ditulis pada baris yang berbeda, berikut adalah contoh penggunaan titik koma pada javascript:

```
a = 3;
b = 4;
```

  titik koma pertama dapat dihilangkan, namun jika ditulis pada baris yang sama, titik koma tetap diperlukan

```
a = 3; b = 4;
```

2. Sintaks

- Deklarasi Variabel
  Pembuatan variabel pada javascript menggunakan keyword var. Contoh deklarasi atau pembuatan variabel pada javascript:

```
var i;
var i, sum;
var message = "hello";
var i = 0, j = 0, k = 0;
```

- Fungsi
  Fungsi adalah blok kode program yang hanya didefinisikan sekali, tapi dapat dipanggil atau dijalankan berulang kali. Pada javascript, fungsi dapat dibuat menggunakan keyword function. Sebuah fungsi harus memiliki nama, sepasang tanda kurung untuk parameter, dan sepasang kurung kurawal. Berikut ini adalah beberapa contoh fungsi:

```
// Print the name and value of each property of o. Return undefined.
function printprops(o) {
  for(var p in o)
    console.log(p + ": " + o[p] + "\n");
}

// Compute the distance between Cartesian points (x1,y1) and (x2,y2).
function distance(x1, y1, x2, y2) {
  var dx = x2 - x1;
  var dy = y2 - y1;
  return Math.sqrt(dx*dx + dy*dy);
}
```

2.3.1 XMLHttpRequest

XMLHttpRequest adalah salah satu objek pada javascript yang dapat digunakan untuk mendapatkan *file* XML dari *server* secara *asynchronous* atau *synchronous* [5]. *Asynchronous* berarti bahwa pertukaran data dilakukan tanpa harus memuat ulang seluruh halaman *web*, sedangkan pertukaran data *synchronous* harus memuat ulang seluruh halaman *web*. Berikut ini adalah contoh penggunaan XMLHttpRequest:

```
var objXMLHTTP = new XMLHttpRequest();

objXMLHTTP.open('GET', 'books.xml', false);
objXMLHTTP.send(null);

var objXML = objXMLHTTP.responseXML;
```

Langkah pertama adalah dengan membuat objek XMLHttpRequest. Selanjutnya, dengan memanggil fungsi open("method", "url", asynchronous). Parameter *method* menentukan metode yang digunakan, contoh "GET" untuk menerima data dan "POST" untuk mengirim data, parameter url adalah alamat *file*, dan parameter boolean "false" menunjukkan bahwa permintaan tersebut dilakukan secara *synchronous*. Langkah terakhir adalah mendapatkan respon dari *server*. Berikut ini penjelasan dari setiap *method* yang digunakan:

1. open("method","url", asynchronous,"username","password")
   Melakukan inisialisasi permintaan
   Parameter:
   - *method*
     Method pada HTTP yang digunakan seperti "GET" dan "POST".
   - *url*
     Alamat url tujuan
   - *asynchronous*
     boolean Opsional, secara default bernilai *true*. *True* menyatakan bahwa operasi yang dijalankan secara *asynchronous*. Nilai *false* menyatakan sebaliknya.
   - *username*
     Opsional, berisikan *username* yang digunakan untuk keperluan otentikasi. Secara default, berisi string kosong.
   - *password*
     Opsional, berisikan *password* yang digunakan untuk keperluan otentikasi. Secara default, berisi string kosong.
2. send(content)
   Mengirimkan permintaan
   Parameter:
   - *content*
     Opsional, *content* dapat berisi string atau data lainnya seperti Array, dokumen, dan lain-lain.

3. responseXML
   Respon dari permintaan
   Return:
   DOM Object

2.3.2 XML DOM

DOM adalah singkatan dari *Document Object Model*, XML DOM adalah API umum untuk menangani dokumen XML [5]. API adalah singkatan dari *Application Programming Interface* merupakan fungsi atau perintah yang dapat digunakan untuk menangani masalah pemrograman tertentu. XML DOM menyediakan fungsi standar untuk mengakses, memodifikasi, dan menciptakan berbagai bagian dari sebuah dokumen XML. Contoh:

```
var myNodeset = objXML.getElementsByTagName('plant');
var name = myNodeset[0].getAttribute('name');
```

Pemanggilan fungsi getElementsByTagName('plant') akan mengembalikan satu set node yang memiliki nama tag 'plant'. Contoh lain, pemanggilan fungsi .getAttribute() akan mengembalikan nilai atribut. Berikut ini penjelasan dari setiap *method* yang digunakan:

1. getElementsByTagName('tagName')
   Mengembalikan elemen-elemen yang memiliki kesesuaian nama.
   Parameter:
   - tagName
     String yang menentukan nama elemen yang dicari.

   Return:
   objek berisi elemen yang memiliki nama sesuai dengan yang dicari.

2. getAttribute('name')
   Mengembalikan nilai atribut
   Parameter:
   - name
     String yang menentukan nama atribut yang dicari.

   Return:
   Mengembalikan string jika atribut memiliki nilai, jika tidak mengembalikan *null*.

2.3.3 Google Maps Javascript API

Google Maps Javascript API memungkinkan untuk sebuah halaman web menampilkan peta dunia yang datanya didapat dari server google [6]. Google menyediakan fungsi atau perintah untuk menampilkan dan menyesuaikan peta sesuai dengan kebutuhan. Berikut ini adalah contoh halaman web yang menampilkan peta di lokasi Sydney, Australia:

```
<!DOCTYPE html>
<html>
  <head>
```

```
<style type="text/css">
  html, body, #map-canvas { height: 100%; margin: 0; padding:
     0;}
</style>
<script type="text/javascript"
  src="https://maps.googleapis.com/maps/api/js?key=API_KEY">
</script>
<script type="text/javascript">
  function initialize() {
    var mapOptions = {
      center: { lat: -34.397, lng: 150.644},
      zoom: 8
    };
    var map = new google.maps.Map(document.getElementById('map
       -canvas'),
        mapOptions);
  }
  google.maps.event.addDomListener(window, 'load', initialize)
     ;
</script>
</head>
<body>
<div id="map-canvas"></div>
</body>
</html>
```

- Declaring
  Google menyarankan untuk membuat deklarasi tipe dokumen pada awal dokumen yaitu dengan menulis <!DOCTYPE html>. Setelah itu diperlukan CSS yang bekerja untuk mengatur tampilan peta pada halaman web.

```
<style type="text/css">
  html { height: 100% }
  body { height: 100%; margin: 0; padding: 0 }
  #map-canvas { height: 100% }
</style>
```

  Kode CSS pada contoh menunjukkan tag yang memiliki id map-canvas akan memiliki tinggi 100% pada saat ditampilkan dan juga menunjukkan persentase yang sama pada <html> dan <body>.

- Loading Google Maps API
  Untuk dapat menampilkan peta diperlukan juga melakukan *load* javascript. URL yang terdapat pada tag script adalah lokasi file javascript yang akan memuat seluruh simbol dan definisi yang dibutuhkan untuk menggunakan Google Maps API ini. Paramater key berisi API key yang dimiliki oleh pengguna.

```
<html>
  <head>
    <script type="text/javascript"
      src="https://maps.googleapis.com/maps/api/js?key=API_KEY">
    </script>
```

- Initialize
  Setelah melakukan load javascript, diperlukan pemanggilan fungsi initialize. Di dalam fungsi tersebut dapat ditambahkan beberapa variabel yang dibutuhkan.

  ```
  function initialize() {}
  ```

  Untuk inisialisasi peta, diperlukan variabel map options

  ```
  var mapOptions = {};
  ```

  Selanjutnya diperlukan koordinat pusat peta yang akan ditampilkan, sedangkan zoom menunjukkan level zoom yang ingin ditampilkan

  ```
  center: new google.maps.LatLng(-34.397, 150.644),
  zoom: 8
  ```

- Map Object
  objek peta perlu dibuat dengan cara melakukan inisialisasi kelas google.maps.Map. Pada contoh, peta diletakkan pada yang memiliki id map-canvas.

  ```
  var map = new google.maps.Map(document.getElementById("map-canvas"),
      mapOptions);
  ```

- Loading the Map
  Google Maps API menyediakan fungsi untuk memuat peta

  ```
  google.maps.event.addDomListener(window, 'load', initialize);
  ```

Berikut ini adalah penjelasan kelas dan fungsi yang digunakan:

1. google.maps.Map class
   Membuat peta baru pada halaman html.
   Parameter:
   - mapDiv:Node
     node yang digunakan untuk membuat peta.
   - opts?:MapOptions
     Opsi dari *map* yang akan dibuat.
2. google.maps.LatLng class
   Membuat objek Latlng yang merepresentasikan titik geografis.
   Parameter:

- lat:number
  Latitude dalam derajat.
- lng:number
  Longitude dalam derajat.
- noWrap?:boolean
  Latitude ditentukan dalam rentang derajat -90 hingga 90 dan *longitude* ditentukan dalam rentang derajat -180 hingga 180. Isikan nilai *true* pada boolean noWrap untuk mengaktifkan nilai di luar batas tersebut.

3. google.maps.event.addDomListener(instance:Object, eventName:string, handler:Function, capture?:boolean)
   Menambahkan fungsi *listener*
   Parameter:

   - instance:Object
     Objek yang ditambahkan *listener*.
   - eventName:string
     Nama dari *listener* tersebut.
   - handler:Function
     fungsi yang menangani *listener*.
   - capture?:boolean

   Return:
   MapsEventListener

2.3.3.1 Menggambar pada Peta

Peta pada Google Maps API dapat ditambahkan objek seperti titik, garis, area, atau objek lainnya. objek tersebut dinamakan *overlay*. Terdapat beberapa jenis *overlay* yang dapat ditambahkan pada peta yaitu *marker* dan *polyline*. Berikut ini adalah penjelasan kelas yang digunakan yaitu kelas *marker* dan kelas *polyline*:

1. google.maps.Marker class
   Membuat *marker* pada peta dengan opsi tertentu.
   Parameter:

   - opts?:MarkerOptions
     Opsi dari *marker* yang dibuat.

2. google.maps.Polyline class
   Membuat *polyline* pada peta dengan opsi tertentu.
   Parameter:

   - opts?:PolylineOptions
     Opsi dari *polyline* yang dibuat.

Berikut ini adalah contoh penggunaan *marker* dan *polyline* pada peta:

1. *Marker*

   Lokasi tunggal pada peta ditunjukkan oleh *Marker*.

   - Menambahkan *Marker*

     Untuk menampilkan *marker* pada peta harus membuat objek google.maps.Marker. Berikut ini adalah atribut penting pada saat membuat objek *marker*:

     (a) *position*

     atribut *position* diperlukan untuk mengatur letak *marker* pada peta.

     (b) *map*

     atribut *map* bersifat opsional, untuk menentukan marker tersebut akan diletakkan pada peta. Jika atribut *map* tidak diatur, maka *marker* akan tetap dibuat tetapi tidak akan ditampilkan pada peta.

     Berikut ini adalah contoh kode program untuk menambahkan *marker* pada peta:

```
var myLatlng = new google.maps.LatLng(-25.363882,131.044922);
var mapOptions = {
  zoom: 4,
  center: myLatlng
}
var map = new google.maps.Map(document.getElementById
("map-canvas"), mapOptions);

// To add the marker to the map, use the 'map' property
var marker = new google.maps.Marker({
    position: myLatlng,
    map: map,
    title:"Hello World!"
});
```

     Pada contoh, objek google.maps.Marker yang dibuat disimpan pada variabel marker, terdapat atribut *position* menggunakan variabel myLatlng yang berisi koordinat (-25.363882,131.044922), atribut map menunjukkan bahwa *marker* akan ditampilkan pada objek map yang tersimpan pada variabel map, dan atribut yang menunjukkan judul *marker*.

   - Menghapus *Marker* pada peta

     Untuk menghapus *marker* pada peta, hanya diperlukan pemanggilan fungsi setMap() dan mengisi parameter fungsi tersebut dengan *null*. Contoh:

```
marker.setMap(null);
```

     Pada contoh di atas hanya menghilangkan *marker* dari peta dan tidak menghapus objek *marker*.

   - Animasi *Marker*

     Menambahkan animasi pada *marker*, hanya memerlukan pengaturan atribut pada konstruktor google.maps.Marker. Contoh:

```
  var marker = new google.maps.Marker({
```

```
    position: myLatlng,
    map: map,
    animation: google.maps.Animation.BOUNCE,
    title:"Hello World!"
});
```

Pada contoh, menambahkan animasi *bounce* pada marker sehingga *marker* bergerak melompat-lompat pada peta.

- Mengubah Ikon
  Gambar *marker* pada peta dapat diubah sesuai keinginan, hanya memerlukan pengaturan atribut pada konstruktor google.maps.Marker. Contoh:

```
var image = 'images/beachflag.png';
var myLatLng = new google.maps.LatLng(-33.890542, 151.274856);
var beachMarker = new google.maps.Marker({
    position: myLatLng,
    map: map,
    icon: image
});
```

  Pada contoh, ikon *marker* akan ditampilkan menggunakan *file* gambar beachflag.png

- *Draggable*
  Draggable memungkinkan pengguna untuk menyeret marker ke lokasi yang berbeda, hanya memerlukan pengaturan atribut pada konstruktor google.maps.Marker. Contoh:

```
var marker = new google.maps.Marker({
  position: myLatlng,
  map: map,
  draggable: true,
  title:"Hello World!"
});
```

2. *Polyline*
   Objek *polyline* adalah serangkaian garis pada peta, *polyline* berguna untuk menunjukkan dari satu titik ke titik lain. *Polyline* memiliki atribut yang dapat diubah sesuai kebutuhan seperti warna, *opacity*, dan *weight*. Berikut ini penjelasan dari beberapa atribut tersebut:

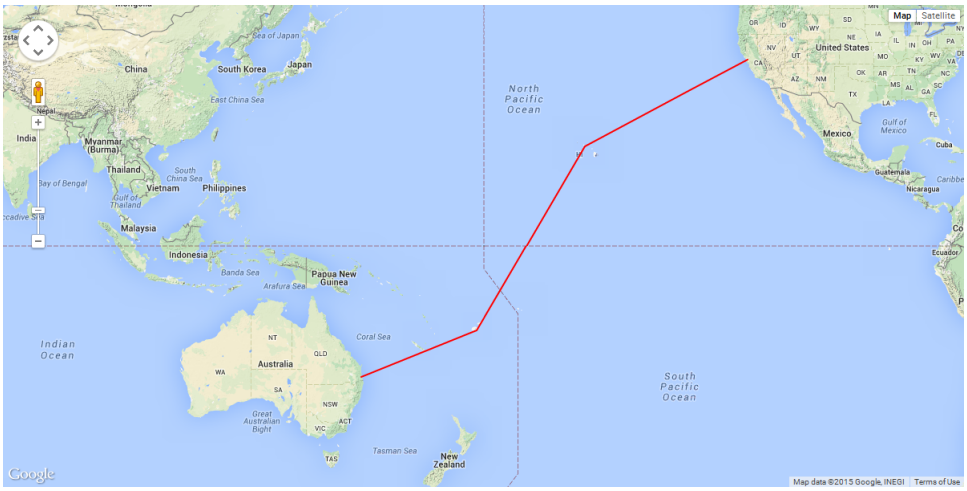
   - *strokeColor*
     Atribut *strokeColor* menentukan warna dalam format heksadesimal, contoh "#FFFFFF".
   - *strokeOpacity*
     Atribut *strokeOpacity* menentukan *opacity* dalam nilai antara 0.0 dan 1.0.
   - *strokeWeight*
     Atribut *strokeWeight* menentukan lebar garis dalam piksel.

Berikut ini adalah contoh potongan kode program untuk menampilkan *polyline* pada peta:

```
var flightPlanCoordinates = [
    new google.maps.LatLng(37.772323, -122.214897),
    new google.maps.LatLng(21.291982, -157.821856),
    new google.maps.LatLng(-18.142599, 178.431),
    new google.maps.LatLng(-27.46758, 153.027892)
  ];
  var flightPath = new google.maps.Polyline({
    path: flightPlanCoordinates,
    strokeColor: '#FF0000',
    strokeOpacity: 1.0,
    strokeWeight: 2
  });

  flightPath.setMap(map);
```

Pada contoh, akan menampilkan polyline pada peta yang akan menghubungkan setiap koordinat yang terdapat pada variabel flightPlanCoordinates. *Polyline* yang ditampilkan pada peta dapat dilihat pada Gambar 2.2.



Gambar 2.2: Polyline pada Peta

2.4 Graf

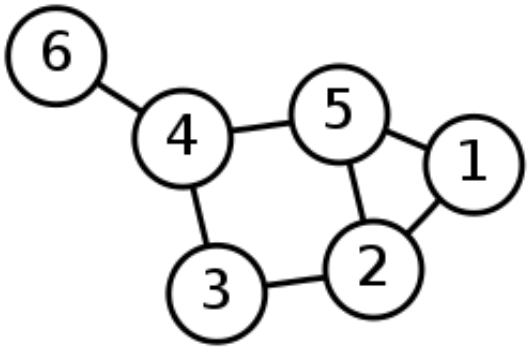
Graf adalah himpunan objek yang terdiri dari simpul (node) dan sisi (edge), graf digambarkan sebagai node yang dihubungkan oleh edge. Konsep graf telah digunakan pada banyak aplikasi komputer dan menggunakan beberapa jenis graf seperti graf sederhana, graf tidak berarah, graf berarah, graf tak terbatas, dan lain-lain. Contoh graf dapat dilihat pada Gambar 2.3. Graf mengikuti aturan berikut [2]:

1. Graf terdiri dari dua bagian yang disebut simpul dan sisi.

2. Node digambarkan berdasarkan tipenya dan nilainya mungkin terbatas atau tidak terbatas.

3. Setiap sisi menghubungkan dua buah simpul.

4. Node digambarkan sebagai kotak atau lingkaran dan edge digambarkan sebagai garis atau busur.



Gambar 2.3: Contoh Graf

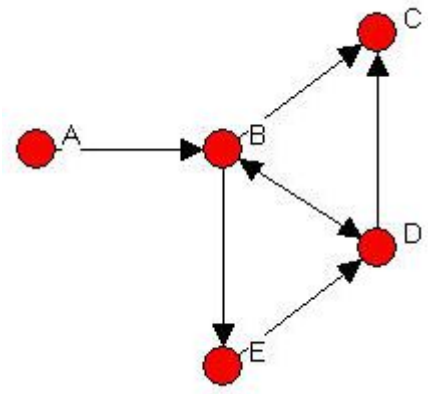
Berdasarkan contoh pada Gambar 2.3 didapatkan informasi tipe dari simpul adalah bilangan bulat
Himpunan simpul = 1,2,3,4,5,6
Himpunan edge = (6,4),(4,5),(4,3),(3,2),(5,2),(2,1),(5,1)

2.4.1 Graf Tidak Berarah

Graf tidak berarah tidak memiliki arah pada setiap edgenya, sehingga setiap simpul tidak memiliki urutan. Graf tidak berarah digambarkan dengan garis lurus antara simpul. Contoh graf berarah dapat dilihat pada Gambar 2.3.

2.4.2 Graf Berarah

Graf berarah memiliki arah pada setiap edgenya. Pada graf berarah, edge biasanya digambarkan dengan panah sesuai arahnya. Contoh graf berarah dapat dilihat pada Gambar 2.4.



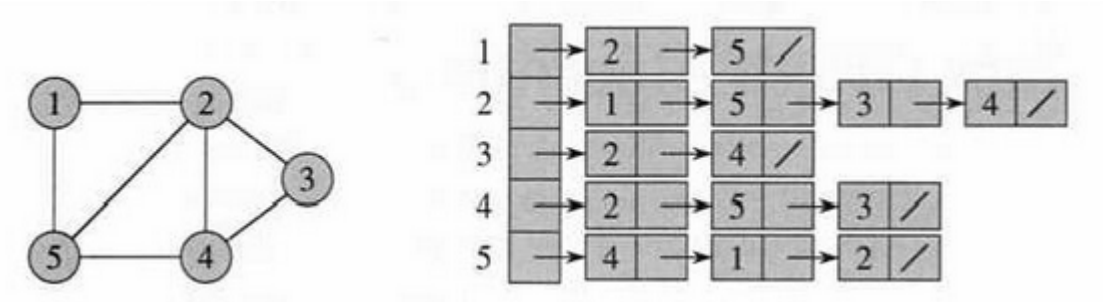
Gambar 2.4: Contoh Graf Berarah

Berdasarkan contoh pada Gambar 2.4 didapatkan informasi tipe dari simpul adalah huruf kapital
Himpunan simpul = A, B, C, D, E
Himpunan edge = (A, B), (B, C), (D, C), (B, D), (D, B), (E, D), (B, E)

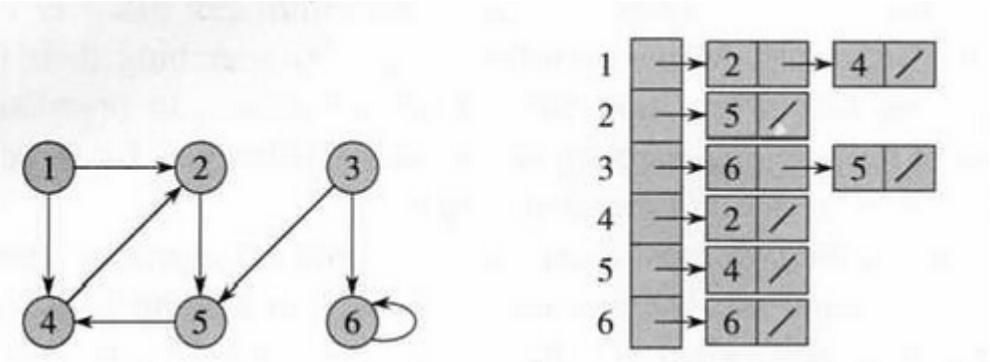
2.4.3 Representasi Graf

Terdapat dua cara untuk merepresentasikan graf yaitu dengan *adjacency list* dan *adjacency matrix* [1]. Keduanya dapat merepresentasikan graf berarah ataupun graf tidak berarah. *Adjacency list* merepresentasikan graf ke dalam bentuk array, sedangkan *adjacency matrix* merepresentasikan graf ke dalam bentuk matriks.

- *Adjacency List*
  Adjacency List merupakan representasi graf ke dalam bentuk array, panjang array sesuai dengan jumlah simpul pada graf. Setiap index pada array mengacu pada setiap simpul graf, setiap index array tersebut memiliki list yang merepresentasikan hubungan dengan simpul-simpul lainnya. Contoh representasi graf tidak berarah dalam bentuk *adjacency list* dapat dilihat pada Gambar 2.5 dan representasi graf berarah dalam bentuk *adjacency list* dapat dilihat pada Gambar 2.6.



Gambar 2.5: Contoh Adjacency List (Graf Tidak Berarah)

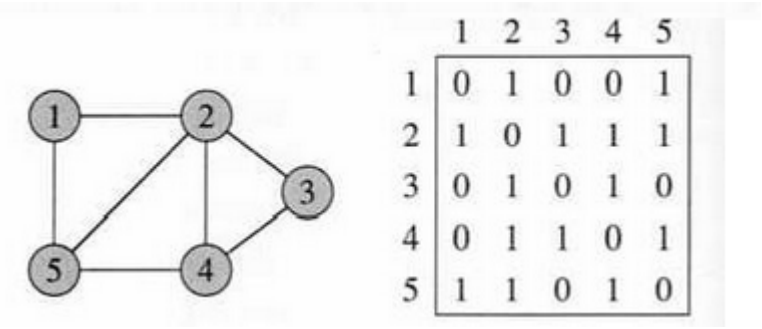


Gambar 2.6: Contoh Adjacency List (Graf Berarah)

- *Adjacency Matrix*
  Adjacency Matrix merupakan representasi graf ke dalam bentuk matriks nxn, pada matriks tersebut menyatakan hubungan antar simpul atau pada graf. Nilai n pada

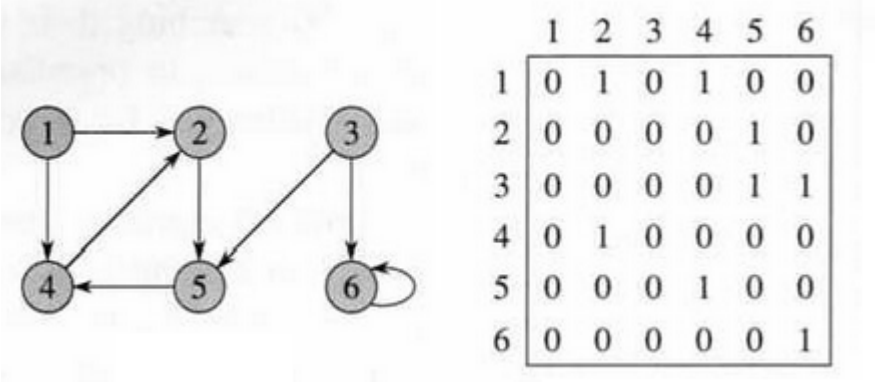
[2]http://web.cecs.pdx.edu/sheard/course/Cs163/Doc/Graphs.html

matriks nxn sesuai dengan jumlah simpul pada graf. Nilai 1 pada matriks menandakan terdapat hubungan pada simpul dan sebaliknya jika bernilai 0. Contoh representasi graf tidak berarah dalam bentuk *adjacency matrix* dapat dilihat pada Gambar 2.7 dan representasi graf berarah dalam bentuk *adjacency matrix* dapat dilihat pada Gambar 2.8.



	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

Gambar 2.7: Contoh Adjacency Matrix (Graf Tidak Berarah)



	1	2	3	4	5	6
1	0	1	0	1	0	0
2	0	0	0	0	1	0
3	0	0	0	0	1	1
4	0	1	0	0	0	0
5	0	0	0	1	0	0
6	0	0	0	0	0	1

Gambar 2.8: Contoh Adjacency Matrix (Graf Berarah)

2.5 Algoritma Dijkstra

Algoritma dijkstra adalah algoritma yang dapat mencari jalur terpendek pada graf berarah dengan persamaan G=(V,E) untuk kasus pada setiap sisinya bernilai tidak negatif [1]. Algoritma ini menggunakan prinsip greedy. Prinsip greedy pada algoritma dijkstra menyatakan bahwa pada setiap langkahnya memilih sisi yang berbobot minimum dan memasukannya dalam himpunan solusi. Berikut ini adalah langkah-langkah dari algoritma dijkstra:

1. Inisialisasi setiap node dengan nilai jarak sementara dan berikan nilai nol pada node awal.

2. Tandai node awal tersebut sebagai *current* node dan tandai juga seluruh node lainnya sebagai node yang belum dikunjungi (*unvisited* set).

3. Hitung jarak seluruh node yang bertetangga dengan *current* node dengan cara membandingkan jarak yang dimiliki *current* node dan jarak tetangganya satu persatu, jika jarak yang dimiliki *current* node lebih besar maka ganti nilai tersebut.

4. Setelah selesai menghitung jarak seluruh tetangganya, hapus *current* node dari *unvisited* set dan tandai bahwa node tersebut sudah dikunjungi.

5. Selanjutnya, pilih node yang belum dikunjungi dengan nilai jarak terkecil dan tandai node tersebut sebagai *current* node dan ulangi langkah 3.

2.6 Haversine *Formula*

Haversine *formula* adalah persamaan yang dapat memberikan jarak antara dua titik berdasarkan *latitude* atau garis lintang dan *longitude* atau garis bujur [7]. Haversine *formula* dinyatakan dalam persamaan berikut ini:

$$d = 2r \sin^{-1}(\sqrt{\sin^2(\frac{\phi_2 - \phi_1}{2}) + \cos(\phi_1)\cos(\phi_2)\sin^2(\frac{\psi_2 - \psi_1}{2})})$$

dimana:

- d : jarak antara dua buah titik
- r : radius bumi
- ϕ_1, ϕ_2 : *latitude* dari titik 1 and *latitude* dari titik 2
- ψ_1, ψ_2 : *longitude* dari titik 1 and *longitude* dari titik 2

DAFTAR REFERENSI

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Second Edition*. MIT Press and McGrawŨHill, 2001.

[2] OpenStreetMap, "OpenStreetMap Wiki." http://wiki.openstreetmap.org/, 2014. [Online; accessed 30-January-2015].

[3] B. Marchal, *XML by Example*. John Pierce, 2000.

[4] D. Flanagan, *JavaScript: The Definitive Guide, Sixth Edition*. O'Reilly Media, Inc, 2011.

[5] E. Woychowsky, *Ajax: Creating Web Pages with Asynchronous JavaScript and XML*. Prentice Hall, 2006.

[6] Google, "Google Maps JavaScript API v3." https://developers.google.com/maps/documentation/javascript/, 2015. [Online; accessed 31-January-2015].

[7] N. R.Chopde and M. K. Nichat, "Landmark based shortest path detection by using a* and haversine formula," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 1, 2013.