

# **TUGAS P06 P07 PENGEMBANGAN APLIKASI WEB**

## **“Review 1”**

*Disusun untuk memenuhi tugas kuliah*

*Mata kuliah: Pengembangan Aplikasi Web*



**Disusun Oleh:**

Abdul Rahem Faqih 220411100029

**Dosen pengampu:**

Moch. Kautsar Sophan S.Kom., M.MT.

**PROGRAM STUDI TEKNIK INFORMATIKA**

**FAKULTAS TEKNIK**

**UNIVERSITAS TRUNOJOYO MADURA**

**BANGKALAN**

**TAHUN 2023**

## 1. Source Code rute.php

```
}

<?php

function hitung_haversine($lat1, $lon1, $lat2, $lon2)
{
    $lat1 = deg2rad($lat1);
    $lon1 = deg2rad($lon1);
    $lat2 = deg2rad($lat2);
    $lon2 = deg2rad($lon2);

    $dlat = $lat2 - $lat1;
    $dlon = $lon2 - $lon1;

    $a = sin($dlat / 2) * sin($dlat / 2) + cos($lat1) * cos($lat2) * sin($dlon /
2) * sin($dlon / 2);
    $c = 2 * atan2(sqrt($a), sqrt(1 - $a));

    $radius = 6371;

    return $c * $radius;
}

$data_koordinat = [
    "Surabaya" => ["latitude" => -7.255859181105917, "longitude" =>
112.75799779607146],
    "Banten" => ["latitude" => -6.403731703056451, "longitude" =>
106.05933104389545],
    "Bogor" => ["latitude" => -6.597574089116659, "longitude" =>
106.80836692644529],
    "Bandung" => ["latitude" => -6.916053058158814, "longitude" =>
107.60607412705717],
    "Yogyakarta" => ["latitude" => -7.787361111089316, "longitude"
=> 110.3622707339422],
    "Semarang" => ["latitude" => -6.969002755840674, "longitude" =>
110.43024352928138],
    "Bekasi" => ["latitude" => -6.365639709304437, "longitude" =>
107.19013475107063],
```

```

    "Cirebon" => ["latitude" => -6.69536307085594, "longitude" =>
108.5587586437141],
    "Tuban" => ["latitude" => -6.8860692117137825, "longitude" =>
112.0390017630515],
    "Pekalongan" => ["latitude" => -6.897477005152818, "longitude" =>
109.67660019023775],
    "Tasikmalaya" => ["latitude" => -7.324976342663581, "longitude" =>
108.22411983517829],
    "Jakarta" => ["latitude" => -6.207877571749103, "longitude" =>
106.83293282757978],
    "Klaten" => ["latitude" => -7.726256860406675, "longitude" =>
110.64806589487056],
    "Madiun" => ["latitude" => -7.641841047711919, "longitude" =>
111.52623845626613],
    "Magelang" => ["latitude" => -7.476981274010894, "longitude" =>
110.19782502001567],
];

$data_jarak = [];

function hitung_jarak($kota1, $kota2)
{
    global $data_koordinat;
    return hitung_haversine($data_koordinat[$kota1]['latitude'],
$data_koordinat[$kota1]['longitude'], $data_koordinat[$kota2]['latitude'],
$data_koordinat[$kota2]['longitude']);
}

$data_jarak = [
    "Banten" => [
        "Jakarta" => hitung_jarak("Banten", "Jakarta"),
        "Bogor" => hitung_jarak("Banten", "Bogor"),
    ],
    "Jakarta" => [
        "Bogor" => hitung_jarak("Jakarta", "Bogor"),
        "Bekasi" => hitung_jarak("Jakarta", "Bekasi"),
        "Banten" => hitung_jarak("Jakarta", "Banten")
    ],
    "Bogor" => [

```

```

        "Bekasi" => hitung_jarak("Bogor", "Bekasi"),
        "Bandung" => hitung_jarak("Bogor", "Bandung"),
        "Jakarta" => hitung_jarak("Bogor", "Jakarta"),
        "Banten" => hitung_jarak("Bogor", "Banten")
    ],
    "Bekasi" => [
        "Jakarta" => hitung_jarak("Bekasi", "Jakarta"),
        "Bogor" => hitung_jarak("Bekasi", "Bogor"),
        "Bandung" => hitung_jarak("Bekasi", "Bandung"),
        "Cirebon" => hitung_jarak("Bekasi", "Cirebon"),
    ],
    "Bandung" => [
        "Bogor" => hitung_jarak("Bandung", "Bogor"),
        "Bekasi" => hitung_jarak("Bandung", "Bekasi"),
        "Cirebon" => hitung_jarak("Bandung", "Cirebon"),
        "Tasikmalaya" => hitung_jarak("Bandung", "Tasikmalaya"),
    ],
    "Cirebon" => [
        "Bekasi" => hitung_jarak("Cirebon", "Bekasi"),
        "Bandung" => hitung_jarak("Cirebon", "Bandung"),
        "Tasikmalaya" => hitung_jarak("Cirebon", "Tasikmalaya"),
        "Pekalongan" => hitung_jarak("Cirebon", "Pekalongan")
    ],
    "Tasikmalaya" => [
        "Bandung" => hitung_jarak("Tasikmalaya", "Bandung"),
        "Cirebon" => hitung_jarak("Tasikmalaya", "Cirebon"),
        "Pekalongan" => hitung_jarak("Tasikmalaya", "Pekalongan"),
        "Magelang" => hitung_jarak("Tasikmalaya", "Magelang"),
        "Yogyakarta" => hitung_jarak("Tasikmalaya", "Yogyakarta")
    ],
    "Pekalongan" => [
        "Cirebon" => hitung_jarak("Pekalongan", "Cirebon"),
        "Magelang" => hitung_jarak("Pekalongan", "Magelang"),
        "Semarang" => hitung_jarak("Pekalongan", "Semarang"),
        "Tasikmalaya" => hitung_jarak("Pekalongan", "Tasikmalaya")
    ],
    "Magelang" => [
        "Tasikmalaya" => hitung_jarak("Magelang", "Tasikmalaya"),
        "Pekalongan" => hitung_jarak("Magelang", "Pekalongan"),
    ]
}

```

```

    "Semarang" => hitung_jarak("Magelang", "Semarang"),
    "Madiun" => hitung_jarak("Magelang", "Madiun"),
    "Klaten" => hitung_jarak("Magelang", "Klaten"),
    "Yogyakarta" => hitung_jarak("Magelang", "Yogyakarta")
],
"Yogyakarta" => [
    "Tasikmalaya" => hitung_jarak("Yogyakarta", "Tasikmalaya"),
    "Magelang" => hitung_jarak("Yogyakarta", "Magelang"),
    "Klaten" => hitung_jarak("Yogyakarta", "Klaten")
],
"Klaten" => [
    "Yogyakarta" => hitung_jarak("Klaten", "Yogyakarta"),
    "Magelang" => hitung_jarak("Klaten", "Magelang"),
    "Madiun" => hitung_jarak("Klaten", "Madiun"),
    "Surabaya" => hitung_jarak("Klaten", "Surabaya")
],
"Madiun" => [
    "Klaten" => hitung_jarak("Madiun", "Klaten"),
    "Magelang" => hitung_jarak("Madiun", "Magelang"),
    "Semarang" => hitung_jarak("Madiun", "Semarang"),
    "Tuban" => hitung_jarak("Madiun", "Tuban"),
    "Surabaya" => hitung_jarak("Madiun", "Surabaya")
],
"Semarang" => [
    "Pekalongan" => hitung_jarak("Semarang", "Pekalongan"),
    "Magelang" => hitung_jarak("Semarang", "Magelang"),
    "Madiun" => hitung_jarak("Semarang", "Madiun"),
    "Tuban" => hitung_jarak("Semarang", "Tuban")
],
"Tuban" => [
    "Semarang" => hitung_jarak("Tuban", "Semarang"),
    "Madiun" => hitung_jarak("Tuban", "Madiun"),
    "Surabaya" => hitung_jarak("Tuban", "Surabaya")
],
"Surabaya" => [
    "Tuban" => hitung_jarak("Surabaya", "Tuban"),
    "Madiun" => hitung_jarak("Surabaya", "Madiun"),
    "Klaten" => hitung_jarak("Surabaya", "Klaten")
]

```

```

];
function dijkstra($graf, $asal, $tujuan)
{
    $jarak = [];
    $sebelumnya = [];
    $antrian = new SplPriorityQueue();

    foreach ($graf as $kota => $tetangga) {
        $jarak[$kota] = INF;
        $sebelumnya[$kota] = null;
    }

    $jarak[$asal] = 0;

    $antrian->insert($asal, 0);

    while (!$antrian->isEmpty()) {
        $saatIni = $antrian->extract();

        if ($saatIni === $tujuan) {
            break;
        }

        foreach ($graf[$saatIni] as $tetangga => $jarakTetangga) {
            $alternatif = $jarak[$saatIni] + $jarakTetangga;
            if ($alternatif < $jarak[$tetangga]) {
                $jarak[$tetangga] = $alternatif;
                $sebelumnya[$tetangga] = $saatIni;
                $antrian->insert($tetangga, -$alternatif);
            }
        }
    }

    $rute = [];
    $saatIni = $tujuan;
    $totalJarak = 0;
    while ($saatIni !== null) {
        $rute[] = $saatIni;
        $kotaSebelumnya = $sebelumnya[$saatIni];
    }
}

```

```

        if ($kotaSebelumnya !== null) {
            $jarakKeSebelumnya = $graf[$saatIni][$kotaSebelumnya];
            $totalJarak += $jarakKeSebelumnya;
        }

        $saatIni = $kotaSebelumnya;
    }

    $rute = array_reverse($rute);
    return ["rute" => $rute, "totalJarak" => $totalJarak];
}

```

### Penjelasan rute.php

Kode di atas ialah program untuk menghitung jarak antara beberapa kota dengan rumus haversine lalu melakukan shortpath atau pencarian rute terpendek antara dua kota yaitu kota asal dan kota tujuan menggunakan algoritma dijkstra, berikut penjelasan lengkapnya:

1. Fungsi `hitung_haversine` : fungsi ini digunakan untuk menghitung jarak antara dua titik koordinat (latitude dan longitude). Fungsi ini menerima 4 parameter: `lat1` dan `lon1` ini adalah koordinat dari titik pertama dan `lat2` dan `lon2` adalah koordinat titik kedua. Pada fungsi ini akan melakukan konversi koordinat dari derajat radian, menghitung perbedaan latitude dan longitude lalu menghitung jarak haversine dengan rumus yang ada.
2. Array `data_koordinat` : ini adalah array yang menyimpan koordinat kota (latitude dan longitude).
3. Fungsi `hitung_jarak` : ini adalah fungsi untuk menghitung jarak haversine antara dua kota berdasarkan titik koordinatnya yang ada pada array `data_koordinat`, pada fungsi ini akan memanggil fungsi `hitung_haversine` dengan koordinat dari dua kota yang akan dihitung jaraknya.
4. Array `data_jarak` : ini adalah array untuk menyimpan data jarak antara kota-kota yang terhubung langsung (sesuai pada graph), key yang pertama itu adalah nama katanya dan valuenya adalah kota-kota yang terhubung dengannya, lalu didalam value tersebut terdapat jarak antara kota kunci luar dan kota pada kunci dalamnya (array 2 dimensi), jarak antara dua kota tersebut dihitung menggunakan fungsi `hitung_jarak`.
5. Fungsi `dijkstra` : ini adalah fungsi yang digunakan untuk mencari rute terpendek antara dua kota dalam `data_jarak` yang sudah dihitung tadi. Fungsi ini terdapat 3 parameter yaitu “graf” atau jarak antara kota-kota pada array `data_jarak`, “asal” ialah kota asalnya dan “tujuan” ialah kota tujuannya, algoritma ini mencari rute terpendek dari kota asal ke kota tujuan dengan melihat jarak terpendeknya, berikut penjelasannya:

1. Fungsi ini menggunakan priority queue untuk mengelola kota-kota yang akan di explore berdasarkan jarak terpendeknya
2. Algoritma ini berjalan hingga tujuan katanya ditemukan atau semua kota telah di jelajahi
3. Setiap langkahnya nanti akan memperbarui jarak terpendek ke setiap kota yang terhubung dan mengidentifikasi jalur terpendek.
4. Hasil akhir atau returnnya adalah rute terpendek dari kota asal ke kota tujuan dan total jaraknya.
5. Total jarak dihitung dengan menambahkan jarak antar kota-kota yan terhubung dalam rute.

## 2. Source Code index.php

```
<?php
require "rute.php";
?>
<!doctype html>
<html lang="en">

<head>

    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <link
href="https://cdn.jsdelivrivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
T3c6CoIi6uLrA9TneNEoa7RxnatzjcDSCmG1MXxSR1GAsXEV/Dwwykc2MPK8M2HN"
crossorigin="anonymous">

    <title>Rute Terpendek</title>
</head>

<body>

    <div class="container my-4">
```



```

<h2 class="text-center">Mencari Rute Terpendek 15 Kota di Pulau Jawa</h2>

<div class="container my-4 w-50 card">

    <div class="card-header">

        Perutean Jalur Terpendek

    </div>

    <form action="" method="post">

        <div class="card-body">

            <div class="mb-3">

                <label for="kota_awal" class="py-2">Pilih Kota
Awal</label>

                <select class="form-select" aria-label="Default select
example" name="kota_asal">

                    <option selected>Pilih Kota Awal</option>

                    <?php foreach ($data_koordinat as $index => $value) :
?>

                        <option value="<?=$index ?"><?=$index
?></option>

                    <?php endforeach; ?>

                </select>

            </div>

            <div class="mb-3">

                <label for="kota_tujuan" class="py-2">Pilih Kota
Tujuan</label>

                <select class="form-select" aria-label="Default select
example" id="kota_tujuan" name="kota_tujuan">

                    <option selected>Pilih Kota Tujuan</option>

                    <?php foreach ($data_koordinat as $index => $value) :
?>

                        <option value="<?=$index ?"><?=$index
?></option>

```

```

        <?php endforeach; ?>

    </select>

</div>

<button class="btn btn-primary" name="submit">Submit</button>

</div>

</form>

</div>

<?php
if (isset($_POST["submit"])) :

    $kota_asal = $_POST["kota_asal"];

    $kota_tujuan = $_POST["kota_tujuan"];

    $hasil = dijkstra($data_jarak, $kota_asal, $kota_tujuan);; ?>

    <h4 class="text-center">Rute terpendek dari <?= $kota_asal ?> ke <?=
    $kota_tujuan ?></h4>

    <p class="text-center">⬇️</p>

    <p class="text-center"><?= implode(" -> ", $hasil["rute"]) . ' (' .
    number_format($hasil['totalJarak'], 2) . ' Km)' ?></p>

    <?php endif; ?>

</div>

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js
" integrity="sha384-
C6RzsynM9kWDrMNeT87bh95OGNyZPhcTNXj1NW7RuBCsyN/o0jlpcV8Qyq46cDfL"
crossorigin="anonymous"></script>

</body>

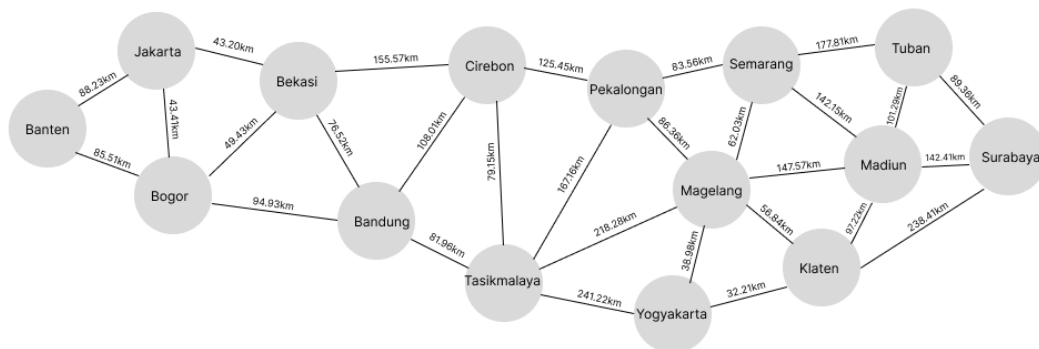
</html>

```

## Penjelasan Index.php

File ini digunakan untuk membuat tampilan form input kota awal dan kota tujuan, data inputan tersebut menggunakan select agar user hanya mengklik opsi yang ada, value dari datanya saya ambil dari \$data\_koordinat lalu di foreach unruk mengambil setiap datanya yang nantinya data tersebut akan diproses dengan memanggil fungsi djikstra pada file rute.php yang sudah di require pada file ini, ketika user mengklik submit nanti akan muncul hasil ouputnya dengan memanggil fungsi djikstra dengan parameter \$data\_jarak, \$kota\_asal dari inputan user dan \$kota\_tujuan yang juga dari inputan user yang disimpan pada \$hasil, fungsi implode disitu digunakan untuk menggabungkan elemen elemen pada array rute di fungsi djikstranya menjadi string tinggal.

## Graph kota yang saya buat



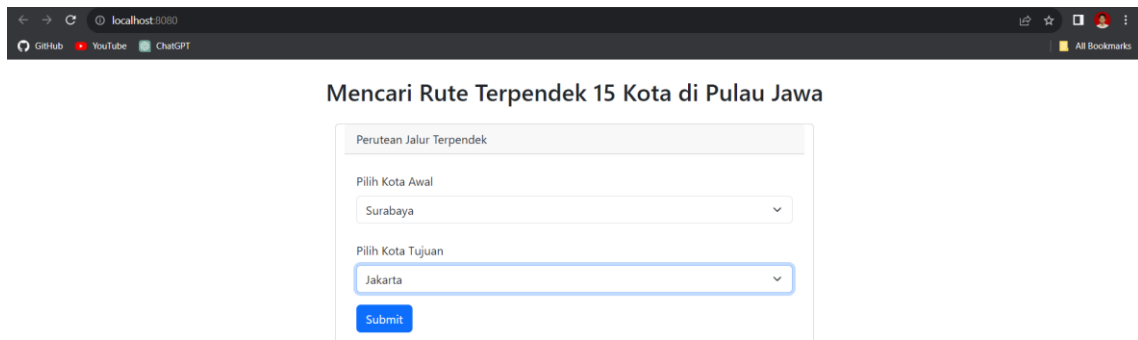
## Penjelasan Algoritmanya

1. Disini diperlukan variabel jarak, sebelumnya, dan atrian prioritas
2. Setiap kota memiliki jarak yang di atur tak hingga (infinity), kecuali kota asal yang memiliki jarak 0
3. Kota asal tersebut dimasukkan kedalam atrian prioritas dengan prioritas 0
4. Lalu lakukan perulangan selama antrain prioritas tidak kosong
5. Pada setiap perulangannya, ambil kota dengan jarak terpendek dari antrian prioritas, kota ini lah yang akan dieksplorasi selanjutnya
6. Untuk setiap tetangga yang terhubung dengan kota saat ini, hitung jarak alternatif dari kota asal ke tetangga tersebut melalui kota saat ini.

7. Jika jarak alternatifnya lebih pendek atau lebih kecil dari jarak saat ini ketetangga, perbarui jarak dan tetangga sebelumnya tetangga dengan jarak alternatif yang lebih pendek tadi
8. Lalu masukkan tetangga yang telah diperbarui ke dalam antrian prioritas dengan prioritas yang sesuai (jarak alternatifnya)
9. Ulangi Langkah Langkah pada dari poin 4 sampai seluruh kota sudah dijelajahi sepenuhnya atau jika kota tujuan sudah dicapai.
10. Setelah sampai di kota tujuan barulah dapat mencari jalur terpendeknya, mulai dari kota tujuan dapat mengikuti jejak kota-kota sebelumnya yang sudah dicata untuk mencari kota-kota dalam jalur terpendek

## Hasil Running

1. User menginputkan kota asal dan kota tujuan



← → ↻ localhost:3000

GitHub YouTube ChatGPT All Bookmarks

### Mencari Rute Terpendek 15 Kota di Pulau Jawa

Perutean Jalur Terpendek

Pilih Kota Awal

Surabaya

Pilih Kota Tujuan

Jakarta

Submit

## 2. Setelah user menekan tombol submit

### Mencari Rute Terpendek 15 Kota di Pulau Jawa

Perutean Jalur Terpendek

Pilih Kota Awal

Pilih Kota Awal

Pilih Kota Tujuan

Pilih Kota Tujuan

Submit

#### Rute terpendek dari Surabaya ke Jakarta



Surabaya -> Tuban -> Semarang -> Pekalongan -> Cirebon -> Bekasi -> Jakarta (674.98 Km)

