IMPLEMENTASI ALGORITMA DIJKSTRA UNTUK MENCARI JARAK TERPENDEK

Muhammad Iqbal Fattah
Informatika
2021071002
Universitas Pembangunan Jaya

Iqbal Ramadhan Saputra
Informatika
2021071010
Universitas Pembangunan Jaya

Khairullah Nurfitri Ramadhan
Informatika
2021071009
Universitas Pembangunan Jaya

Habibi Ar-daffenzy
Informatika
2021071031
Universitas Pemabangunan Jaya

Abstrak -- Kemacetan merupakan salah satu masalah yang dihadapi oleh masyarakat, khususnya di kota besar. Salah satu penyebabnya adalah tidak sebandingnya jumlah kendaraan dengan ruas jalan raya yang dilalui. Salah satu upaya penanganan kemacetan yang terjadi tersebut adalah dengan menggunakan pengalihan kendaraan ke jalur alternatif yang jumlah kendaraannya lebih sedikit. Untuk itu diperlukan metode pemilihan jalur alternatif yang tepat untuk mengurangi masalah kemacetan tersebut. Penelitian ini bertujuan untuk menentukan jalur-jalur alternatif yang lebih efektif dan efisien sehingga dapat mengurangi kemacetan di suatu ruas-ruas jalan tertentu dengan menentukan bobot terkecil dari masing-masing ruas jalan menggunakan Algoritma Djikstra. Dari penelitian ini akan dihasilkan jalur-jalur alternatif yang dapat dilalui pengendara untuk menghindari terjadinya kemacetan di ruas-ruas jalan tertentu.

Kata kunci: Algoritma Djikstra, Bobot terkecil, Jalur alternatif, Kemacetan lalu lintas.

I. PENGENALAN

Kemacetan lalu lintas merupakan salah satu permasalahan yang dihadapi banyak kota besar dan menjadi agenda yang harus diselesaikan oleh pihak berwenang karena semakin hari permasalahan kemacetan lalu lintas ini semakin mengkhawatirkan. Salah satu faktor utama penyebab kemacetan disuatu ruas jalan adalah tidak sebandingnya ruas jalan dengan volume kendaraan yang ingin melintasi jalan tersebut. Hal ini diperparah dengan semakin bertambahnya jumlah pengguna kedaaraan seiring berjalannya waktu yang tidak dibarengi dengan pembangunan, penambahan, atau pelebaran ruas jalan. Akibatnya kemacetan semakin hari semakin sering terjadi terutama di waktu ramai kendaraan seperti saat berangkat atau pulang sekolah/kantor, juga pada hari libur atau akhir pekan. Penanganan permasalahan kemacetan memang bisa dibilang cukup sulit dan membutuhkan waktu yang tidak singkat, namun saat kemacetan terjadi tentu diperlukan suatu penanganan agar kemacetan tidak terjadi berlarut-larut sebab banyak kerugian yang diakibatkan dari kemacetan ini, antara lain waktu yang terbuang percuma untuk menghadapi kemacetan, juga pemborosan bahan bakar kendaraan sebab meskipun kendaraan berhenti tapi mesin kendaraan tetap menyala, selain itu juga menimbulkan polusi yang lebih banyak.

Oleh karena itu diperlukan suatu cara untuk mengurai kemacetan yang terjadi. Meskipun hanya jangka pendek, cara penguraian kemacetan dengan efektif dan efisien dapat mengurangi kerugian yang disebabkan oleh kemacetan. Salah satu cara penguraian kemacetan adalah mengalihkan kendaraan ke jalur alternatif agar volume kendaraan tidak menumpuk disalah satu ruas jalan tertentu.

Pemilihan jalur alternatif untuk membantu menguraikan kemacetan perlu dipertimbangkan dengan tepat agar pengalihan jalur dapat mengurai kemacetan di jalur yang dialihkan bukan malah menjadikan jalur alternatif menjadi titik kemacetan baru, oleh karena itu diperlukan cara pemilihan jalur alternatif yang tepat untuk menentukan jalur alternatif ini berdasarkan banyaknya volume kendaraan yang melintas di ruas-ruas jalan yang bersangkutan. Dalam makalah ini akan dibahas cara pemilihan jalur alternatif guna menangani masalah kemacetan dengan menggunakan salah satu konsep dalam matematikan diskrit, yaitu graf, juga algoritma Dijkstra yang berhubungan dengan kajian ini dalam teori graf untuk menyelesaikan permasalah pemilihan jalur alternatif dengan tepat.

II. PEMBAHASAN

- Definisi Algoritma Dijkstra

Algoritma Dijkstra merupakan Algoritma yang ditemukan dan dinamai menurut penemunya yaitu seorang ilmuwan komputer yang bernama Edsger Djikstra, adalah sebuah algoritma rakus(greedy algorithm) yang dipakai dalam memecahkan permasalahan jarak terpendek (shortest path problem) untuk sebuah graf berarah (directed graph) dengan bobot-bobot garis (edge weights) yang bernilai non negatif. Djikstra merupakan salah satu varian bentuk popular dalam pemecahan persoalan terkait masalah optimasi pencarian lintasan terpendek, sebuah lintasan yang mempunyai panjang minimum dari vertaks A ke Z dalam graph berbobot, bobot tersebut adalah bilangan positif jadi tidak dapat dilalui oleh node negatif. Namun jika terjadi, maka penyelesaian yang diberikan adalah infiniti atau bisa disebut (Tak Hingga). Pada Algoritma Djikstra, node digunakan karena Algoritma Djikstra menggunakan graph berarah untuk penentuan rute lintasan terpendek. Dalam mencari solusi, Algoritma Djikstra menggunakan prinsip Greedy, yaitu mencari solusi optimum pada setiap langkah yang dilalui dengan tujuan untuk mendapatkan solusi optimum pada langkah selanjutnya yang akan mengarah pada solusi terbaik. Algoritma Djikstra ini mencari Panjang lintasan terpendek dari vertex A ke Z dalam sebuah graf berbobot.

- Rumusan Masalah

Dalam hal ini penulis telah merumuskan beberapa masalah yang terjadi saat pengendara ataupun hendak berpergian sebagai berikut:

1) Pengguna aplikasi peta tersesat sehingga mengalami insiden kecelakaan karena aplikasi tidak akurat saat menganalisa rute dan tujuan yang dimaskudkan pengendara.
2) Ketidaksesuaian antara besar jalan dengan volume kendaraan dan rute yang kurang efektif.
3) Pengendara sering diarahkan ke jalan yang sebelumnya telah dilalui secara berulang-ulang.

Rumusan masalah bukanlah hal yang direkayasa, namun beberapa pengguna mengalami hal yang serupa dengan rumusan masalah di atas di antaranya sebagai berikut:

1) Waze dikritik oleh pengguna karena mengarahkan pengguna ke jalan curam.
2) Pengemudi roda 4 tersesat dan masuk ke dalam hutan karena mengikuti arahan google map.

- Tujuan implementasi algortima untuk analisis dan menentukan rute tercepat sebagai berikut :
    1) Memantau jalan macet Untuk memantau jalan yang macet, pengguna bisa mengetahuinya dengan memperhatikan beberapa kode warna yang muncul. Kode warna itu di antaranya oranye, biru, hijau, dan merah.
    2) Mengetahui jalan ganjil genap Rekayasa lalu lintas ganjil genap beberapa kali dilakukan di perkotaan besar seperti DKI Jakarta.
    3) Memantau situasi jalan penerapan satu arah Selain ganjil genap, rekayasa lalu lintas one way juga bisa diketahui melalui aplikasi Google Maps.

- Metode Implementasi

Penerapan Algoritma Dijkstra dapat dilakukan dengan bahasa pemograman python, java, dll. Namun penulis memutuskan untuk melakukan implementasi algoritma menggunakan dengan bahasa pemograman python untuk mencari rute terpendek/tercepat pada simulasi data. Simulasi data akan terbagi menjadi dua bagian yaitu :

1) Nodes sebagai kota-kota/tempat tertentu
2) Edegs sebagai jarak antara lokasi awal ke lokasi tujuan.

Berikut penulis telah menyertakan hasil implementasi Algoritma Dijkstra dengan bahasa pemograman python:

```python
def dijkstra(current, nodes, distances):
    unvisited = {node: None for node in nodes}
    visited = {}
    currentDistance = 0
    unvisited[current] = currentDistance
    while True:

        for neighbour, distance in distances[current].items():

            if neighbour not in unvisited: continue
            newDistance = currentDistance + distance
            if unvisited[neighbour] is None or unvisited[neighbour] > newDistance:
                unvisited[neighbour] = newDistance

        visited[current] = currentDistance
        del unvisited[current]
        if not unvisited: break
        candidates = [node for node in unvisited.items() if node[1]]
        print(sorted(candidates, key = lambda x: x[1]))
        current, currentDistance = sorted(candidates, key = lambda x: x[1])[0]
    return visited
```

Dalam hal di atas, pertama kali akan dilakukan pendefinisian fungsi dari algoritma yang menyertakan perkondisian untuk setiap keluaran. Setiap node yang dilewati akan menyiimpan nilai di setiap edges, sehingga program bisa melakukan perbandingan nilai dari posisi awal hingga akhir tujuan.

```python
nodes = ('A', 'B', 'C', 'D', 'E')
distances = {
    'A': {'B': 2, 'C': 3, 'D': 6},
    'B': {'A': 2, 'D': 4},
    'C': {'A': 3, 'D': 3},
    'D': {'B' : 4,'E': 7},
    'E': {'B':7, 'D': 2}}
current = 'E'
```

Di atas merupakan kumpulan simulasi data untuk menjalankan algoritma Dijkstra, nodes diibaratkan sebagai tempat atau lokasi tujuan. Distance akan diibaratkan sebagai garis-garis yang menghubungkan antara nodes. Sebagai contoh A : { B : 2, C: 3 , D : 6} maka, lokasi A menghubungkan lokasi B, lokasi C, dan D dengan nilai disetiap garis penghubungnya atau edges.

```python
current = 'E'

print(dijkstra(current, nodes, distances))
```

Berikutnya, fungsi current menyatakan posisi awal/lokasi awal pengguna, berapapun nilai yang ada pada nodes lokasi awal akan dianggap bernilai 0.

- Analisa Hasil implemntasi

Posisi awal di Nodes A:

```python
nodes = ('A', 'B', 'C', 'D', 'E')
distances = {
    'A': {'B': 2, 'C': 3, 'D': 6},
    'B': {'A': 2, 'D': 4},
    'C': {'A': 3, 'D': 3},
    'D': {'B' : 4,'E': 7},
    'E': {'B':7, 'D': 2}}
current = 'A'

print(dijkstra(current, nodes, distances))
```

```
[('B', 2), ('C', 3), ('D', 6)]
[('C', 3), ('D', 6)]
[('D', 6)]
[('E', 13)]
{'A': 0, 'B': 2, 'C': 3, 'D': 6, 'E': 13}
```

Posisi awal di Nodes B:

```python
nodes = ('A', 'B', 'C', 'D', 'E')
distances = {
    'A': {'B': 2, 'C': 3, 'D': 6},
    'B': {'A': 2, 'D': 4},
    'C': {'A': 3, 'D': 3},
    'D': {'B' : 4,'E': 7},
    'E': {'B':7, 'D': 2}}
current = 'B'

print(dijkstra(current, nodes, distances))
```

```
[('A', 2), ('D', 4)]
[('D', 4), ('C', 5)]
[('C', 5), ('E', 11)]
[('E', 11)]
{'B': 0, 'A': 2, 'D': 4, 'C': 5, 'E': 11}
```

Posisi awal di Nodes C:

```python
nodes = ('A', 'B', 'C', 'D', 'E')
distances = {
    'A': {'B': 2, 'C': 3, 'D': 6},
    'B': {'A': 2, 'D': 4},
    'C': {'A': 3, 'D': 3},
    'D': {'B' : 4,'E': 7},
    'E': {'B':7, 'D': 2}}
current = 'C'

print(dijkstra(current, nodes, distances))
```

```
[('A', 3), ('D', 3)]
[('D', 3), ('B', 5)]
[('B', 5), ('E', 10)]
[('E', 10)]
{'C': 0, 'A': 3, 'D': 3, 'B': 5, 'E': 10}
```

Posisi awal di Nodes D:

```
nodes = ('A', 'B', 'C', 'D', 'E')
distances = {
    'A': {'B': 2, 'C': 3, 'D': 6},
    'B': {'A': 2, 'D': 4},
    'C': {'A': 3, 'D': 3},
    'D': {'B' : 4,'E': 7},
    'E': {'B':7, 'D': 2}}
current = 'D'

print(dijkstra(current, nodes, distances))
```

```
[('B', 4), ('E', 7)]
[('A', 6), ('E', 7)]
[('E', 7), ('C', 9)]
[('C', 9)]
{'D': 0, 'B': 4, 'A': 6, 'E': 7, 'C': 9}
```

Posisi awal di Nodes E:

```
nodes = ('A', 'B', 'C', 'D', 'E')
distances = {
    'A': {'B': 2, 'C': 3, 'D': 6},
    'B': {'A': 2, 'D': 4},
    'C': {'A': 3, 'D': 3},
    'D': {'B' : 4,'E': 7},
    'E': {'B':7, 'D': 2}}
current = 'E'

print(dijkstra(current, nodes, distances))
```

```
[('D', 2), ('B', 7)]
[('B', 6)]
[('A', 8)]
[('C', 11)]
{'E': 0, 'D': 2, 'B': 6, 'A': 8, 'C': 11}
```

Berdasarkan percobaan posisi awal pada setiap node dapat disimpulkan bahwa hasil keluaran implementasi algoritma dapat menganalisa dan membandingkan antara node A sampai node terakhir yaitu E. Algoritma mencoba semua nilai-nilai (edges) yang menghubung antara nodes posisi awal , sehingga rute terpendek/tercepat dengan nilai seminimal mungkin dapat ditentukan berdasarkan rute tercepat. Contoh keluaran rute tercepat diposisi awal A ={ 'B': 2, 'C': 3, 'D': 6, 'E': 13}, maka rute tercepat yang dilalui node A sampe ke E telah ditetapkan.

III. PENUTUP

Berdasarkan pada pembahasan yang telah dilakukan pada bab sebelumnya dapat disimpulkan bahwa algoritma yang digunakan merupakan solusi dalam mencari rute terpendek, Dengan metode algoritma Dijkstra yang diterapkan dalam sistem peta digital pengguna dapat mempersingkat efektifitas waktu untuk mencari jalan tercepat. Rute yang dihasilkan pada algoritma ini sangat efektif bila dilalui dengan mengendarakan motor dengan mengesampingkan kemacetan dan kondisi ganjil genap di Jakarta. Berdasarkan pengujian yang dilakukan, penerapan algoritma Dijkstra untuk mendapatkan rute terpendek dinilai efektif.

IV. DAFTAR PUSTAKA

[1] gramedia.com. Agustus 2022. Cara Membuat Makalah: Pengertian, Struktur, Jenis dan Contohnya. Diakses pada 03 November 2022, dari Cara menulis makalah.

[2] pythonpool.com. 6 May 2022, Implementation Dijkstra's Algorithm in Python. Diakses pada 03 November 2022, dari Implementation Djikstra Algorithm.

[3] mti.binus.ac.id. 28 November 2017, Algoritma Dijkstra. Diakses pada 03 November 2022, dari Algoritma Dijkstra.