Penerapan *Floyd-Warshall Algorithm* dalam Menentukan Rute Pengiriman Terpendek Kurir Sepeda

Gregorius Jovan Kresnadi - 13518135[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13518135@std.stei.itb.ac.id

***Abstrak*—Dalam melakukan pekerjaannya, seorang kurir sepeda harus memetakan terlebih dahulu rute pengirimannya agar semua barang dapat diantarkan secepat mungkin. Untuk itu, dapat disusun sebuah graf berarah berbobot berdasarkan peta daerah, dengan simpul sebagai destinasi pengiriman atau *drop point*, sisi sebagai jalan yang menghubungkan antarsimpul, dan bobotnya adalah jarak antarsimpul. Kurir tersebut harus menyusun sebuah lintasan terpendek yang melewati seluruh *drop point*. Makalah ini ditujukan untuk mencari rute paling pendek yang dapat ditempuh oleh seorang kurir sepeda dalam menghantarkan barang-barangnya ke semua *drop point* menggunakan algoritma *Floyd-Warshall*.**

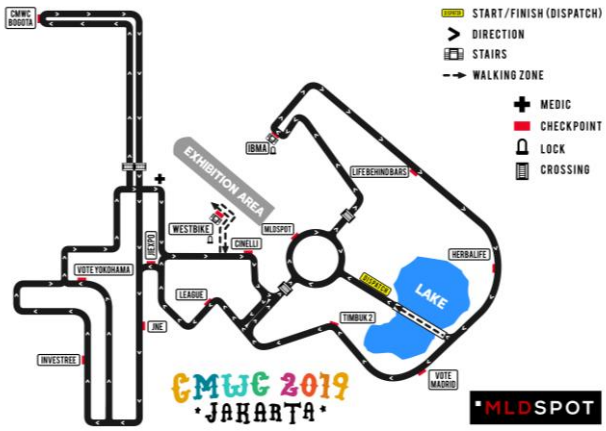
***Kata Kunci*—Graf Berarah, Bobot, *Drop Point*, Kurir.**

I. PENDAHULUAN

Jasa kurir adalah sebuah jasa menghantarkan barang dari seorang pengirim kepada seorang penerima, atau dari satu tempat ke tempat lain, dengan secepat dan seaman mungkin. Seorang kurir bukanlah hanya tukang pos yang biasa dikenal masyarakat umum, namun dalam era modern ini, paket-paket seperti dokumen penting dan pesanan belanja *online* pun dihantarkan oleh seorang kurir. Pekerjaan ini memiliki tingkat kesulitan yang cukup tinggi dengan banyaknya permintaan dan waktu yang ditekan menjadi sesingkat mungkin, dan terkadang pekerjaannya direndahkan oleh masyarakat. Maka dari itu seorang kurir harus memiliki komitmen yang tinggi dan kecekatan dalam melakukan pekerjaannya, bahkan memerlukan sebuah strategi khusus untuk memenuhi semua pesanannya.

Salah satu mode transportasi yang digunakan seorang kurir adalah sepeda. Menurut Jati, seorang kurir sepeda di Bandung, selain menggunakan mode transportasi yang ramah lingkungan, jasa pengiriman menggunakan sepeda juga terbukti lebih efisien, terutama di daerah pusat kota dan metropolitan. Selain itu kurir sepeda terbukti lebih cepat daripada kurir dengan kendaraan bermotor karena rute pengiriman yang lebih efisien (bisa memotong jalan dan parkir yang bebas), tanpa menambah resiko kerusakan pada barang. Namun barang yang diutamakan memiliki berat barang maksimal 1kg, dengan volume terbesar 30cm x 20cm x 15cm (satu kotak sepatu).

Jati menambahkan bahwa setiap tahun, terdapat kejuaraan internasional kurir sepeda, bernama *Cycle Messenger World Championship (CMWC)*, yang mensimulasikan pekerjaan sehari-hari seorang kurir sepeda. Dalam perlombaan tersebut terdapat sebuah peta yang menunjukkan dengan jelas rute dan titik antar beserta sebuah *manifest* yang menunjukkan barang yang harus diantar ke titik tertentu. Perlombaan ini bertujuan untuk mengantar semua barang ke titik pengantaran, dan kembali ke tempat semula dengan waktu tercepat. (Jati, wawancara, 3 Desember 2019)



Gambar 1. Contoh peta rute pada perlombaan CMWC tahun 2019 (Sumber: https://cmwc2019.com/wp-content/uploads/2019/08/CMWC-2019-Race-Map.jpg Diakses pada 4 Desember 2019)

II. TEORI DASAR

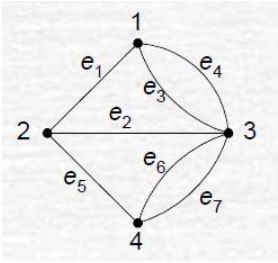
A. *Graf*

a. Definisi Graf

Secara matematis, suatu graf G dapat didefinisikan sebagai sebuah pasangan himpunan *(V, E)* yang dapat ditulis dalam notasi G = (V, E). Simbol V adalah himpunan dari simpul (*vertices* atau *nodes*) yang tidak boleh kosong, sedangkan simbol E adalah himpunan dari sisi (*edges*) yang menghubungkan simpul dan himpunan bisa kosong.

Simpul pada graf dapat diberi nama dengan huruf (a, b, c, d, …), angka (1, 2, 3, …), gabungan keduanya, atau sebuah kata. Sedangkan sisi pada graf biasa dilambangkan

dengan e_1, e_2, e_3, dst. Sebuah sisi yang menghubungkan simpul *a* dan *b* dapat dinyatakan sebagai sisi (*a*, *b*).



Gambar 2. Ilustrasi graf (Sumber:http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2015-2016/Graf%20(2015).pdf Diakses pada 4 Desember 2019)

Graf diatas memiliki notasi sebagai berikut:
G = (V, E)
V = {1, 2, 3, 4}
E = {e_1, e_2, e_3, e_4, e_5, e_6, e_7}
= {(1,2), (2,3), (1,3), (3,1), (2,4), (4,3), (3,4)}

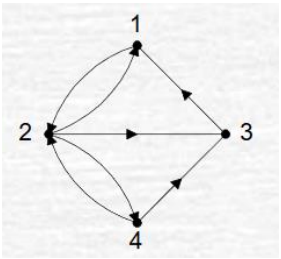
b. Jenis Graf

Berdasarkan keberadaan sisi ganda atau sisi gelang, graf dapat dibedakan menjadi dua jenis, yaitu:

1. Graf Sederhana (*simple graph*)
   Graf sederhana adalah graf yang tidak memiliki sisi ganda ataupun sisi gelang.
2. Graf Tak-Sederhana (*unsimple graph*)
   Graf tidak sederhana adalah graf yang memiliki sisi ganda atau sisi gelang. Gambar 1 merupakan contoh dari graf tak-sederhana.

Berdasarkan orientasi arah pada sisi, graf dapat dibedakan menjadi dua jenis, yaitu:

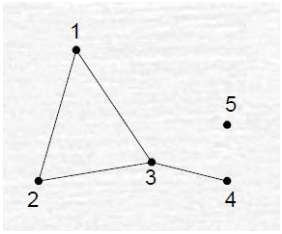
1. Graf Tak-Berarah (*undirected graph*)
   Graf tak-berarah adalah graf yang sisinya tidak memiliki orientasi arah. Gambar 1 merupakan contoh dari graf tak-berarah.
2. Graf Berarah (*directed graph*)
   Graf berarah adalah graf yang setiap sisinya memiliki orientasi arah.



Gambar 3. Ilustrasi graf berarah (Sumber:http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2015-2016/Graf%20(2015).pdf Diakses pada 4 Desember 2019)

c. Terminologi Dasar Graf

1. Ketetanggaan (*Adjacent*)
   Dua buah simpul dalam graf tak-berarah dikatakan bertetangga jika dihubungkan oleh sebuah sisi. Pada sebuah sisi (*a*, *b*), simpul *a* dan simpul *b* dikatakan bertetangga.
2. Bersisian (*Incidency*)
   Sebuah sisi e dikatakan bersisian dengan simpul *a* dan simpul *b* jika pada graf terdapat sisi (*a*, *b*).
3. Simpul Terpencil (*Isolated Vertex*)
   Sebuah simpul dikatakan simpul terkecil jika tidak terdapat sisi yang bersisian dengannya.



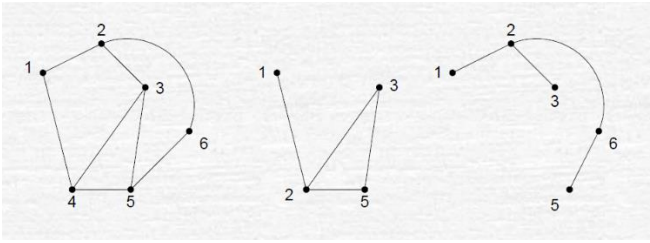
Gambar 4. Ilustrasi graf dengan simpul 5 sebagai simpul terpencil (Sumber :http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2015-2016/Graf%20(2015).pdf Diakses pada 4 Desember 2019)

4. Graf Kosong (*null graph* atau *empty graph*)
   Sebuah graf dikatakan graf kosong jika himpunan sisinya merupakan himpunan kosong (E = {}) Semua simpul pada graf kosong adalah simpul terpencil.
5. Derajat (*Degree*)
   Jumlah sisi yang bersisian dengan suatu simpul disebut derajat simpul. Derajat dinyatakan dengan notasi d(v). Pada graf berarah, suatu simpul memiliki dua jenis derajat, yaitu derajat masuk ke simpul yang dinyatakan dengan $d_{in}(v)$, dan derajat keluar dari simpul yang dinyatakan dengan $d_{out}(v)$.
6. Lintasan (*Path*)
   Lintasan didefinisikan jika sebuah lintasan memiliki panjang *n* dari simpul awal v_0 menuju simpul akhir v_n, di dalam graf G, maka lintasan yang terbentuk adalah barisan berselang-seling simpul-simpul dan sisi-sisi dengan bentuk v_0, e_1, v_1, e_2, v_2, …, e_n, v_n sedemikian sehingga $e_1 = (v_0, v_1)$, $e_2 = (v_1, v_2)$, …, $e_n = (v_{n-1}, v_n)$ adalah sisi-sisi dari graf G.
   Pada graf sederhana, maka sebuah lintasan cukup dinyatakan dengan barisan simpul-simpul yang dilaluinya. Namun pada graf ganda, penulisan lintasan berupa simpul-simpul dan sisi-sisi yang berselang-seling.
7. Siklus (*Cycle*) atau Sirkuit (*Circuit*)
   Sebuah lintasan yang berawal dan berakhir di simpul yang sama dapat disebut sebagai sebuah siklus atau sirkuit. Dalam sebuah graf berarah, terdapat kemungkinan sebuah siklus tidak dapat terbentuk.
8. Terhubung (*Connected*)
   Dua buah simpul *a* dan simpul *b* dapat dikatakan terhubung jika terdapat sebuah lintasan dari *a* ke *b*. Suatu graf G disebut sebagai graf terhubung jika terdapat lintasan yang menghubungkan setiap simpulnya. Jika tidak setiap simpul saling terhubung, maka graf disebut sebagai graf tak-terhubung.

Dua simpul, *a* dan *b*, dalam graf terhubung kuat (*strongly connected*) jika terdapat lintasan berarah dari *a* ke *b*, demikian juga dari *b* ke *a*. Jika hanya terdapat satu buah jalur, maka kedua simpul tersebut dikatakan terhubung lemah (*weakly connected*).

9. Upagraf (*Subgraph*)

   Sebuah graf A dikatakan upagraf dari graf G jika terdapat simpul *Va* yang merupakan himpunan bagian dari *V*, dan sisi *Ea* yang merupakan himpunan bagian dari *E*.



(a) (b) (c)

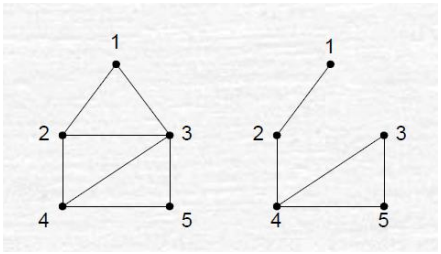
Gambar 5. Ilustrasi graf (a) dengan kedua upagraf (b, c) (Sumber :http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2015-2016/Graf%20(2015).pdf Diakses pada 4 Desember 2019)

10. Komplemen Upagraf

    Komplemen dari sebuah upagraf A terhadap graf G adalah sebuah graf B = (Vb, Eb), sedemikian sehingga Eb = E – Ea dan Vb adalah himpunan simpul yang bersisian dengan anggota-anggota Eb.

11. Upagraf Merentang (*Spanning Subgraph*)

    Upagraf merentang dapat didefinisikan untuk sebuah upagraf A = (Va, Ea) terhadap graf G = (V, E) di mana Va mengandung semua V. (Semua simpul graf G terdapat pada graf A)



(a) (b)

Gambar 6. Ilustrasi graf (a) dengan upagraf merentangnya (b) (Sumber :http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2015-2016/Graf%20(2015).pdf Diakses pada 4 Desember 2019)

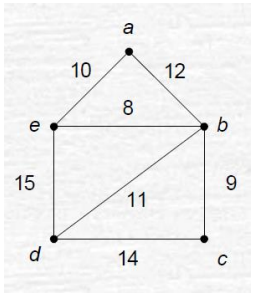
12. *Cut-Set*

    Cut-Set dari sebuah graf terhubung G adalah himpunan sisi dari graf G yang jika dibuang maka akan membuat graf G menjadi tak-terhubung.

13. Graf Berbobot (*Weighted Graph*)

    Graf berbobot adalah sebuah graf yang setiap sisinya memiliki sebuah harga (bobot). Bobot yang diberikan pada suatu graf memiliki makna yang berbeda-beda, tergantung bergantung pada masalah yang diilustrasikan oleh graf tersebut. Contoh-contoh bobot adalah jarak antarkota, biaya yang perjalanan antarkota, dan waktu perjalanan pesawat antarnegara.

    Dalam permasalahan rute pengiriman kurir sepeda, graf yang digunakan adalah graf berbobot. Graf ini akan memodelkan rute yang dapat ditempuh oleh seorang kurir, dengan bobot yang dimaksud adalah jarak antar titik pengiriman. Bobot-bobot ini sangat penting dalam perhitungan penyusunan rute terpendek.



Gambar 7. Ilustrasi graf berbobot (Sumber:http://informatika.stei.itb.ac.id/~rinaldi.munir /Matdis/2015-2016/Graf%20(2015).pdf Diakses pada 4 Desember 2019)

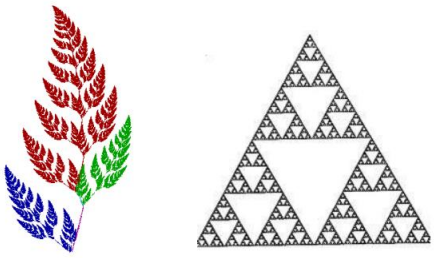
B. *Permasalahan Lintasan Terpendek Melewati Seluruh Simpul (All Pairs Shortest Path Problem)*

Dalam memecahkan permasalah lintasan terpendek yang melewati seluruh simpul ini, akan digunakan sebuah graf berbobot. Sebuah lintasan harus dibentuk sedemikian rupa sehingga seluruh simpul bisa dilewati dan lintasan membuahkan bobot terkecil. Contoh penerapan permasalahan ini adalah dalam sebuah perlombaan kurir sepeda, di mana peserta harus mencari sirkuit dengan lintasan terpendek yang melalui semua *drop point.*

Algoritma yang dapat digunakan adalah algoritma Floyd-Warshall. Algoritma ini cocok untuk menyelesaikan contoh kasus yang sedang dibahas, karena algoritma tidak hanya dapat digunakan dalam graf tak-berarah, namun juga dalam graf berarah. Selain itu, algoritma juga menghasilkan lintasan yang akan melewati seluruh simpul.

C. *Rekursif*

Rekursif adalah sesuatu yang didefinisikan secara berulang oleh dirinya sendiri. Dalam pemrograman, sifat rekursif biasa digunakan untuk memanggil fungsi rekursi, yaitu fungsi yang di dalamnya terdapat fungsi itu sendiri.



Gambar 8. Contoh objek rekursif (Sumber: http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2015-2016/Rekursi%20dan%20Relasi%20Rekurens%20(2015).pdf Diakses pada 4 Desember 2019)

Fungsi rekursif dibagi menjadi dua bagian, yaitu:

1. Basis
   Basis adalah bagian yang berisi nilai fungsi yang terdefinisi secara eksplisit. Bagian basis juga akan menghentikan pemanggilan fungsi rekursif yang diulang-ulang, dan pada akhirnya akan mengembalikan sebuah nilai.
2. Rekurens
   Bagian rekurens adalah bagian yang mendefinisikan dirinya sendiri di dalam fungsi. Bagian rekurens biasanya berisi operasi yang akan mengevaluasi suatu input atau melakukan perhitungan, yang nilainya akan dikembalikan ketika pemanggilan diberhentikan oleh basis fungsi.

Sifat rekursif ini dimiliki oleh algoritma Floyd-Warshall untuk menghitung lintasan terpendek atau total bobot terkecil. [4]

D. Algoritma Floyd-Warshall

Algoritma Floyd-Warshall adalah algoritma yang mirip dengan *transitive closure algorithm*, pertama ditemukan oleh Roy dan Warshall. Algoritma ini disebut juga *Floyd's Algorithm* dan *Roy-Floyd Algorithm.* Algoritma ini digunakan untuk mencari lintasan terpendek dalam graf berbobot yang melewati seluruh simpul, baik dalam graf berarah maupun tak-berarah. [5]

Algoritma Floyd-Warshall adalah sebuah contoh *dynamic programmming*, artinya algoritma ini akan memecah-mecah sebuah masalah menjadi masalah-masalah yang lebih kecil, kemudian jawaban dari masalah yang lebih kecil akan digabung untuk menyelesaikan masalah utamanya. Algoritma ini sangat berguna untuk *networking*,

Inti dari algoritma Floyd-Warshall adalah fungsi berikut ini:

```
ShortestPath(i,j,k)
```

Fungsi ini akan mengambalikan sebuah nilai berupa lintasan terpendek dari A ke C menggunakan simpul dari 1 sampai *k* pada graf. Setiap simpul diberi angka 1, 2, sampai *k*.

Algoritma ini bekerja secara rekursif, dengan basisnya adalah bobot sisi antara 2 buah simpul yang paling berdekatan.

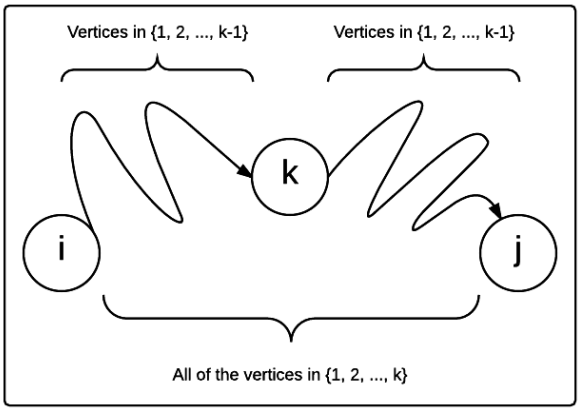
```
ShortestPath(i,j,k) = weight(i,j)
```

Kemudian, bagian rekurens dari fungsi ini akan memanfaatkan sifat *dynamic programming* dari algoritma. Rekurens akan mencari lintasan terdekat antara simpul *i* dan *j*. Hasil rekurens akan memunculkan 2 kemungkinan jawaban. Jawaban pertama, lintasan antara *i* dan *j* adalah lintasan terdekat. Jawaban kedua, lintasan terdekat adalah lintasan yang terbentuk oleh jumlah dari lintasan terdekat antara simpul *i* dan sebuah simpul *z*, ditambah lintasan terdekat antara simpul *z* dan simpul *j*. Fungsi rekurensnya adalah sebagai berikut.

```
ShortestPath(i,j,k)=min(ShortestPath(i,j,
k-1),ShortestPath(i,k,k-1)+ShortestPath(k,j,
k-1))
```

Fungsi di atas mengevaluasi apakah simpul *k* merupakan sebuah simpul antara dalam lintasan terpendek (simpul yang terdapat di antara simpul awal dan akhir)?

Jika simpul *k* bukan merupakan simpul antar, maka lintasan terdekat dari *i* ke *j* menggunakan simpul-simpul {1, 2, …, *k*-1} merupakan lintasan terdekat menggunakan simpul-simpul {1, 2, …, *k*}. Sedangkan jika simpul *k* adalah simpul antara, maka lintasan tersebut dapat dipecah menjadi dua bagian yang masing-masing dapat menggunakan simpul-simpul {1, 2, …, *k*-1} untuk membuat sebuah lintasan yang terdiri dari semua simpul {1, 2, …, *k*}. Hal ini disebabkan simpul *k* yang terletak di antara simpul awal dan akhir. Gambar di bawah ini dapat mengilustrasikan penjelasan di atas. [1]



Gambar 9. Penjelasan fungsi rekurens dari algoritma Floyd-Warsall (Sumber: https://brilliant.org/wiki/floyd-warshall-algorithm/ Diakses pada 4 Desember 2019)

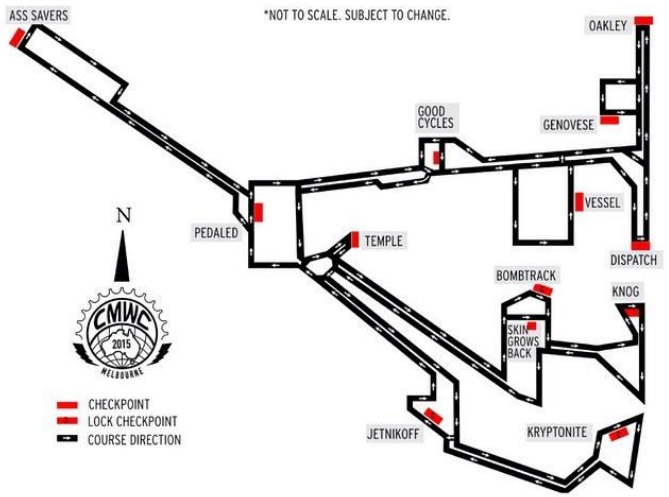
E. Perlombaan Balap Kurir Sepeda

Perlombaan balap kurir sepeda adalah sebuah perlombaan yang mensimulasikan keseharian seorang kurir sepeda dalam sebuah trek tertutup. Sebuah trek akan diatur sehingga memiliki titik-titik *checkpoint* yang akan menjadi tempat pengiriman barang, atau disebut juga *drop point*. Setiap peserta akan diberikan sebuah daftar barang, atau *manifest*, yang sama dan tidak berurutan. *Manifest* akan berisi daftar barang dan tujuan barang tersebut. Tujuan utama dari perlombaan ini adalah untuk menghantarkan semua barang ke masing-masing *drop point* dengan secepat mungkin. Maka dari itu, peserta harus menyusun sebuah sirkuit terpendek yang melewati semua *drop point*, dimulai dari titik awal yang ditentukan.

Karena perlombaan dapat dibilang sangat mirip dengan pekerjaan seorang kurir dalam kesehariannya, maka dari itu peta rutenya bisa dijadikan contoh yang baik, dengan simpul, sisi, dan bobot yang dapat dikonversi menjadi graf yang mudah dibaca dan dianalisis.

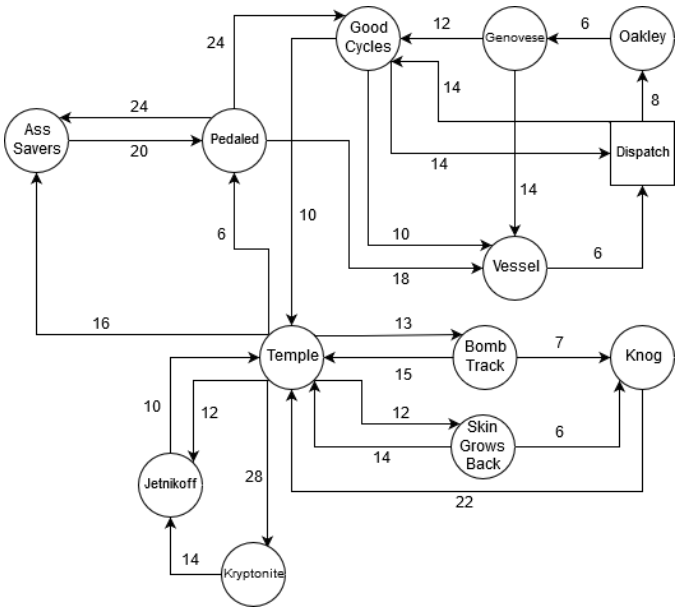
III. PEMBAHASAN

Untuk menerapkan algoritma Floyd-Warshall dalam mencari rute terpendek yang melalui semua node, akan diambil sebuah studi kasus di mana peta rute adalah sebagai berikut:



Gambar 10. Contoh peta rute pada perlombaan CMWC tahun 2015 (Sumber: https://melb-cmwc2015.tumblr.com/ Diakses pada 4 Desember 2019)

Peta rute di atas disederhanakan menjadi graf berbobot berarah yang lebih mudah dibaca menjadi sebagai berikut:



Gambar 11. Representasi Graf berarah berbobot yang disederhanakan dari peta rute pada Gambar 10

Dari graf di atas dapat dibuat sebuah tabel yang berisi data sisi yang terdiri dari simpul asal, simpul tujuan, dan bobot sisinya. Data ini akan kemudian dimasukkan ke dalam algoritma Floyd-Warshall dan ditemukan lintasan terpendek yang melewati tiap simpul.

Tabel 1. Tabel Data Sisi

No	Simpul Asal	No	Simpul Tujuan	Bobot
1	Dispatch	2	Oakley	8
1	Dispatch	5	Good Cycles	14
2	Oakley	3	Genovese	6
3	Genovese	4	Vessel	14
3	Genovese	5	Good Cycles	12
4	Vessel	1	Dispatch	6
5	Good Cycles	1	Dispatch	14
5	Good Cycles	4	Vessel	10
5	Good Cycles	6	Temple	10
6	Temple	7	Bomb Track	13
6	Temple	8	Skin Grows Back	12
6	Temple	10	Kryptonite	28
6	Temple	11	Jetnikoff	12
6	Temple	12	Ass Savers	16
6	Temple	13	Pedaled	6
7	Bomb Track	6	Temple	15
7	Bomb Track	9	Knog	7
8	Skin Grows Back	6	Temple	14
8	Skin Grows Back	9	Knog	6
9	Knog	6	Temple	22
10	Kryptonite	11	Jetnikoff	14
11	Jetnikoff	6	Temple	10
12	Ass Savers	13	Pedaled	20
13	Pedaled	4	Vessel	18
13	Pedaled	5	Good Cycles	24
13	Pedaled	12	Ass Savers	24

Tabel di atas dikonversi ke dalam bentuk matriks yang menampilkan jarak antar setiap simpul, dengan nilai INF menunjukkan bahwa simpul baris *i* dengan kolom *j* tidak berhubungan.

```
[[0 ,  8,INF,INF, 14,INF,INF,INF,INF,INF,INF,INF,INF],
[INF,  0,  6,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF],
[INF,INF,  0, 14, 12,INF,INF,INF,INF,INF,INF,INF,INF],
[  6,INF,INF,  0,INF,INF,INF,INF,INF,INF,INF,INF,INF],
[ 14,INF,INF, 10,  0, 10,INF,INF,INF,INF,INF,INF,INF],
[INF,INF,INF,INF,INF,  0, 13, 12,INF, 28, 12, 16,  6],
[INF,INF,INF,INF,INF, 15,  0,INF,  7,INF,INF,INF,INF],
[INF,INF,INF,INF,INF, 14,INF,  0,  6,INF,INF,INF,INF],
[INF,INF,INF,INF,INF, 22,INF,INF,  0,INF,INF,INF,INF],
[INF,INF,INF,INF,INF,INF,INF,INF,INF,  0, 14,INF,INF],
[INF,INF,INF,INF,INF, 10,INF,INF,INF,INF,  0,INF,INF],
[INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,  0, 20],
[INF,INF,INF, 18, 24,INF,INF,INF,INF,INF,INF, 24,  0]]
```

Gambar 12. Matriks jarak antarsimpul

Agar dapat dibentuk sebuah sirkuit dengan jarak terdekat, data matriks di atas harus diubah untuk mendapat lintasan terdekat yang dapat dicapai tiap simpul ke simpul lainnya Berikut adalah algoritma Floyd-Warshall dalam bahasa Python untuk menentukannya:

```python
def floydWarshall(graph):
    dist = graph
    for k in range(V):
        for i in range(V):
            for j in range(V):
                dist[i][j] = min(dist[i][j],(dist[i][k]+ dist[k][j]))
    return dist
```

Gambar 13. Algoritma Floyd-Warshall dalam bahasa Python

Hasil dari penerapan algoritma tersebut terhadap matriks pada gambar 11 adalah sebagai berikut:

```
Following matrix shows the shortest distances between every pair of vertices
  0   8  14  24  14  24  37  36  42  52  36  40  30
 26   0   6  20  18  28  41  40  46  56  40  44  34
 20  28   0  14  12  22  35  34  40  50  34  38  28
  6  14  20   0  20  30  43  42  48  58  42  46  36
 14  22  28  10   0  10  23  22  28  38  22  26  16
 30  38  44  24  30   0  13  12  18  28  12  16   6
 45  53  59  39  45  15   0  27   7  43  27  31  21
 44  52  58  38  44  14  27   0   6  42  26  30  20
 52  60  66  46  52  22  35  34   0  50  34  38  28
 54  62  68  48  54  24  37  36  42   0  14  40  30
 40  48  54  34  40  10  23  22  28  38   0  26  16
 44  52  58  38  44  54  67  66  72  82  66   0  20
 24  32  38  18  24  34  47  46  52  62  46  24   0
```

Gambar 14. Hasil penerapan algoritma Floyd-Warshall terhadap matriks pada gambar 11

Data yang diperoleh dari matriks di atas akhirnya dapat diolah untuk menyusun sebuah sirkuit dengan bobot paling rendah. Berdasarkan peta pada gambar 10, peserta harus memulai perjalanannya dari simpul "Dispatch", yaitu simpul nomor 1. Maka dari itu dibuat sebuah algoritma yang menyusun sebuah sirkuit yang dimulai dari simpul 1. Berikut adalah algoritma tersebut dalam bahasa Python.

```python
def makeShortestCircuit(graph):
    for k in range(1 , V-1):
        queueinit = [[INF,INF] for x in range(V+1)]
        # First iteration
        if k == 1 :
            queue = queueinit
        # Inisialisasi
        queueinit[0][0] = 0
        queueinit[0][1] = 0
        # Mulai dari k
        j = k
        queueinit[1][0] = j
        queueinit[1][1] = graph[0][j]
        i = 2
        # Loop
        while (1):
            if i == V :
                queueinit[i][0] = 0
                queueinit[i][1] = graph[j][0]
                j = k
                break
            else:
                urutan, jarak = findMinNotInQueue(graph[j],queueinit)
                queueinit[i][0] = urutan
                queueinit[i][1] = jarak
                j = urutan
                i += 1
        printQueue(queueinit)
        print(distQueue(queue))
        print(distQueue(queueinit))
        jaraktotal = min(distQueue(queue), distQueue(queueinit))
        if jaraktotal == distQueue(queueinit) :
            queue = queueinit
    return queue,jaraktotal
```

Gambar 15. Algoritma yang digunakan untuk membuat sirkuit terpendek, dimulai dari simpul 1

Algortima di atas akan menghasilkan sebuah sirkuit yang mencantumkan simpul-simpul yang dilewati beserta jarak tiap simpulnya. Berikut adalah hasil penerapannya:

Tabel 2. Tabel Lintasan

No	Simpul	Jarak
1	Dispatch	-
10	Kryptonite	52
11	Jetnikoff	14
6	Temple	10
13	Pedaled	6
4	Vessel	18
2	Oakley	14
3	Genovese	6
5	Good Cycles	12
8	Skin Grows Back	22
9	Knog	6
7	Bomb Track	35
12	Ass Savers	31
1	Dispatch	44

IV. KESIMPULAN

Algoritma Floyd-Warshall dapat membantu mencari lintasan rute pengiriman terpendek kurir sepeda. Dapat dilihat kompleksitas program algoritma tersebut adalah $O(n^3)$ yang disebabkan oleh tiga kali pengulangan. Namun, menggunakan algoritma Floyd-Warshall saja tidak cukup untuk membuat sebuah sirkuit, karena hasil dari algoritma adalah jarak terpendek yang dapat ditempuh dari satu simpul ke semua simpul yang lain. Perlu dibuat sebuah algoritma yang mengolah data tersebut untuk membuat sirkuit terpendek yang melalui semua simpul.

VI. UCAPAN TERIMA KASIH

Puji Syukur kepada Tuhan Yang Maha Esa karena berkat dan bimbingan-Nya, makalah ini dapat diselesaikan dengan tepat waktu. Penulis mengucapkan terima kasih kepada kedua orang tua dan teman-teman yang selalu mendukung dalam studi dan proses pembelajaran. Penulis juga tidak lupa untuk mengucapkan terima kasih kepada Ibu Fariska, selaku dosen pengampu mata kuliah IF2120 Matematika Diskrit kelas K03, atas segala bimbingan dan inspirasi dalam menjalankan kuliah, dan Bapak Rinaldi Munir atas segala referensi dan bahan bacaan yang telah diberikannya.

DAFTAR PUSTAKA

[1] Tim Brilliant. https://brilliant.org/wiki/floyd-warshall-algorithm/ , diakses tanggal 4 Desember 2019 pukul 17.05 GMT +7

[2] TIM CMWC 2019. https://cmwc2019.com, diakses tanggal 4 Desember 2019 pukul 16.40 GMT+7.

[3] Tim GeeksforGeeks. https://www.geeksforgeeks.org/floyd-warshall-algorithm-dp-16/ , diakses tanggal 4 Desember 2019 pukul 19.15 GMT +7

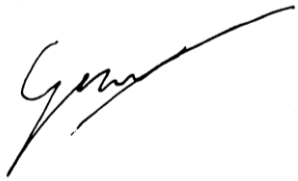
[4] Munir, Rinaldi. Matematika Diskrit. Bandung: Informatika. 2016, edisi keenam

[5] Tim Wolfram. http://mathworld.wolfram.com/Floyd-WarshallAlgorithm.html, diakses tanggal 4 Desember 2019 pukul 16.55 GMT +7

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 6 Desember 2019



Gregorius Jovan Kresnadi
13518135