PEMANFAATAN ALGORITMA DIJKSTRA UNTUK PENCARIAN JALUR PENGIRIMAN INFORMASI ANTAR *DEVICE* YANG PALING OPTIMAL DI LINGKUNGAN ITB JATINANGOR

MAKALAH TUGAS BESAR

Diajukan sebagai salah satu tugas mata kuliah

Pengantar Rekayasa dan Desain

oleh

Muhamad Salman Hakim Alfarisi	**16521513**
Christophorus Dharma Winata	**16521514**
Ahmad Rivai Yahya	**16521523**
Wan Aufa Azis	**16521525**
Jason Rivalino	**16521541**
Afnan Edsa Ramadhan	**16521542**

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2022

Pemanfaatan Algoritma Dijkstra untuk Pencarian Jalur Pengiriman Informasi Antar *Device* yang Paling Optimal di Lingkungan ITB Jatinangor

Muhamad Salman Hakim Alfarisi (16521513), Christophorus Dharma Winata (16521514),
Ahmad Rivai Yahya (16521523), Wan Aufa Azis (16521525),
Jason Rivalino (16521541), Afnan Edsa Ramadhan (16521542)

Kelas Mahasiswa (K-73), Sekolah Teknik Elektro dan Informatika,
Institut Teknologi Bandung,
Jl. Let. Jend. Purn. Dr. (HC) Mashudi No.1, Sayang, Kec. Jatinangor,
Kabupaten Sumedang, Jawa Barat 45363

16521513@std.stei.itb.ac.id, 16521514@std.stei.itb.ac.id, 16521523@std.stei.itb.ac.id,
16521525@std.stei.itb.ac.id, 16521541@std.stei.itb.ac.id, 16521542@std.stei.itb.ac.id

Abstrak **— *Network Routing* merupakan proses untuk mengirimkan informasi pada suatu jaringan (*packet*) dari suatu sumber asal yaitu *device* dari *user* pertama menuju ke destinasi tujuan yaitu *device* dari *user* lainnya dengan kedua user tersebut sama-sama memakai internet. Dalam proses *Network Routing* ini, terdapat algoritma yang dipakai untuk menentukan jalur pengiriman tercepat yaitu dengan Algoritma Dijkstra dengan memetakan *user* pertama sebagai posisi *graph* awal dan *user* lainnya sebagai posisi *graph* yang dituju serta terdapat berbagai macam *router* atau sumber koneksi yang berfungsi sebagai *graph* penghubung dari kedua *user* tersebut yang kemudian proses pengiriman informasi antar *user* dalam jaringan dilakukan melalui *graph-graph* tersebut.**

Kata-kata kunci **— Dijkstra, *Graph*, Routing**

I. PENDAHULUAN

Dalam kehidupan di era modern ini, internet telah menjadi bagian penting dalam kehidupan manusia. Semua teknologi yang ada memanfaatkan internet untuk penggunaannya. Namun, dalam memanfaatkan internet, dibutuhkan juga sebuah alat yaitu *router* yang berfungsi sebagai alat penghubung dalam pengiriman informasi antar *device* dari satu *user* dengan *user* lainnya. Akan tetapi, dalam proses pengiriman informasi antar *device* yang dimiliki *user*, sering terdapat berbagai hambatan baik faktor internal seperti keadaan sumber internet pusat yang terkadang mengalami *error*, maupun faktor eksternal seperti adanya penghalang antara *router* dengan *device*, contohnya seperti dinding dan juga ruangan yang memisahkan. Oleh karena itu, kelompok kami akan melakukan pencarian jalur pengiriman informasi dengan proses *Network Routing* yang bertujuan untuk mendapatkan kondisi pengiriman informasi yang paling optimal antar *user*.

Untuk proses *Network Routing* ini, kelompok kami akan memanfaatkan algoritma dalam pengerjaannya. Algoritma yang akan kelompok kami pakai yaitu algoritma Dijkstra. Algoritma Dijkstra merupakan algoritma yang mencari rute jaringan dengan memanfaatkan *cost* yang paling rendah untuk menghubungkan *user* dengan *server*. *Cost* pada kasus ini sendiri berupa jarak dan juga berbagai penghalang yang menghambat proses pengiriman informasi dalam jaringan (*packet*). Pada *graph* yang akan dirancang, *node* yang ada pada awal dan akhir menggambarkan *user* pertama sebagai titik *graph* awal dan *user* lainnya sebagai titik *graph* akhir (destinasi *node* yang dituju) serta *node* lainnya menggambarkan *router-router* yang berfungsi sebagai penghubung. Selain itu terdapat juga *edge* yang menggambarkan *link* antar *node* serta *weight* pada tiap *edge* menggambarkan *cost* untuk tiap *link.* Sehingga nanti akan dirancang peta route network pada Institut Teknologi Bandung kampus Jatinangor seperti gambar di bawah ini dan dicari rute tercepat dalam mengirimkan *packet's* antar *user*.

Gambar 1.1 Peta area router di ITB kampus Jatinangor [1]

II. DASAR TEORI

Algoritma Dijkstra merupakan algoritma yang dipakai untuk menyelesaikan permasalahan jarak terpendek dari sebuah *graph* yang berarah. Cara kerja dari algoritma ini sendiri adalah dengan membuat jalur ke satu simpul yang optimal pada setiap langkahnya[2]. Sehingga, pada langkah tertentu sudah diketahui beberapa jalur terpendeknya.

Cara kerja algoritma ini dimulai dari proses pengecekan jarak yang diketahui saat ini dari *node* sumber ke *node* lainnya dan secara dinamis memperbarui nilai beban jika jalur yang lebih pendek ditemukan. Dalam algoritma ini, semua nodes memiliki informasi yang sama.

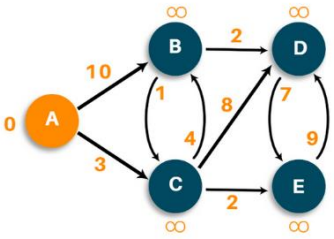
Node yang telah dicek kemudian ditandai sebagai dikunjungi dan ditambahkan ke jalur jika jarak antara *node* tersebut dan *node* sumber terpendek. Ini berlanjut sampai semua *node* telah dilakukan pengecekan, dan akhirnya nilai jalur terpendek dari *node* sumber ke semua *node* lain (nilai beban terkecil) didapatkan dan kemudian pengiriman informasi jaringan (*packet*) dilakukan mengikuti hasil algoritma yang didapatkan.

Untuk beban yang ada dalam algoritma sendiri haruslah merupakan beban dengan nilai positif sehingga perhitungan algoritma dapat dilakukan. Apabila nilai beban adalah negatif, algoritma tidak akan bekerja dan mengalami error [3].

Sistematika tahapan dari cara kerja algoritma Dijkstra sendiri kurang lebih sebagai berikut:

1) Menetukan titik simpul awal yang kemudian diberi jarak dengan simpul terdekatnya satu per satu yang pencarian titik-titiknya akan dikembangkan dengan algoritma ini.

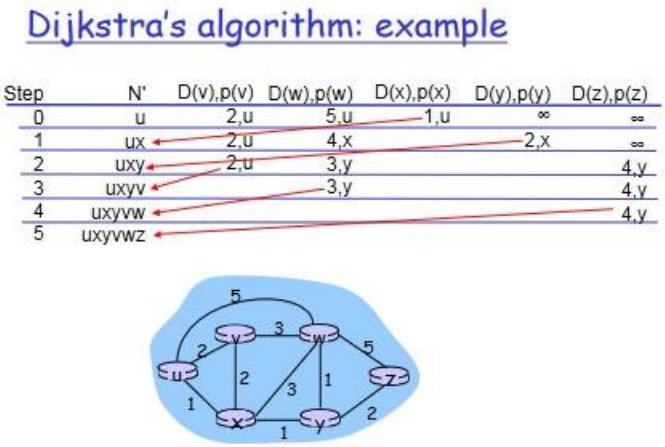
2) Memberi nilai jarak pada setiap titik. Beri nilai 0 pada simpul awal dan nilai tak hingga pada simpul lainnya yang belum terisi. Titik sementara dapat ditetapkan sebagai himpunan priority-queue dengan bentuknya yaitu O(E log V) dengan E sebagai simpul sedangkan V merupakan garis sisi.



Gambar 2.1 Nilai yang diberikan pada setiap titik simpul pada algoritma Dijkstra[4]

3) Mengatur simpul awal menuju simpul berikutnya yang belum dilalui lalu menghitung jaraknya antara keduanya. Jika jaraknya lebih kecil dari jarak yang telah diukur sebelumnya, simpan ulang data dengan jarak baru yang lebih singkat ini.

4) Saat simpul dengan jarak terkecil telah didapatkan, simpul akan ditandai dan tidak akan dilakukan pengecekan ulang lagi. Pengecekan akan dilakukan terus-menerus hingga mencapai simpul dari titik akhir yang diinginkan [2].

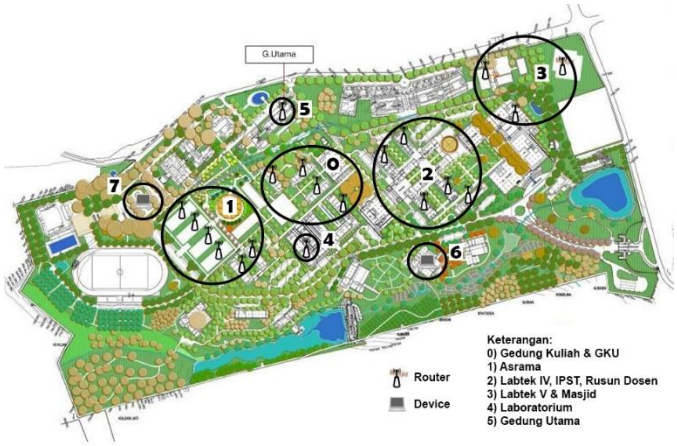


Gambar 2.2 Contoh penerapan Algoritma Dijkstra untuk mencari nilai terpendek dari *nodes* U ke *nodes* Z [5]

III. METODOLOGI

Metodologi yang dipergunakan dalam tugas besar kelompok kami adalah diawali dengan penggambaran peta dari ITB Jatinangor yang telah diberi keterangan posisi-posisi Router-nya, sehingga penggambaran jaringan internet dan posisi *device user* yang ada di ITB Jatinangor menjadi jelas. Dari gambaran peta ini kemudian akan dicari proses pengiriman informasi antar *user* yang paling optimal dalam kampus ITB Jatinangor dengan memanfaatkan algoritma Dijkstra dan program penentuan jaringan internet akan disusun dalam bahasa pemrograman Python.

Pada aplikasi pengerjaannya, router-router akan dianggap sebagai *node* dan kemudian dihubungkan oleh edge. Pada setiap edge yang menghubungkan *node* (router) akan diberi nilai masing-masing sesuai dengan *cost* yang dimiliki. *Cost* akan merepresentasikan jarak antar *node* dan hambatan yang mungkin dialami oleh *node* dalam mengirimkan informasi (packet) kepada *node* yang lain. Hambatan yang ada ditentukan berdasarkan kondisi lingkungan dalam ITB Jatinangor juga (misal pohon, tembok, gedung). Semakin besar *cost* yang dimiliki oleh *edge*, maka akan semakin sulit pula dalam mengirimkan paket-paket informasi tersebut.

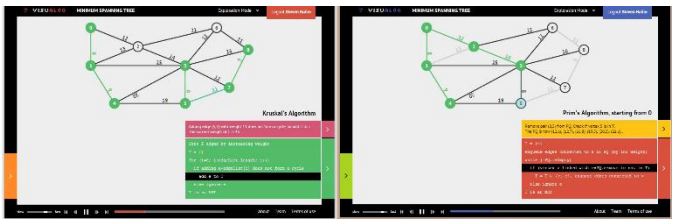


Gambar 3.1 Peta yang telah diberi tanda *node* berdasarkan letak posisinya di ITB Jatinangor [1]

Dalam kasus routing asli, terdapat input yang ada dalam setiap router yaitu alamat IP router yang bersifat unik yang berfungsi sebagai nama *node*nya dan juga untuk pengguna (*user*) sendiri juga memiliki alamat IP yang bervariasi pada setiap device-nya. Alamat IP ini dijadikan sebagai penanda yang nantinya akan dituju sehingga packet informasi tidak akan masuk ke alamat yang salah. Akan tetapi, dalam kajian tugas besar ini, router akan diwakilkan dengan nama yang sederhana untuk memudahkan pengerjaan tugas.

Untuk program dari pemetaan jaringan internetnya, kelompok kami akan melakukan pembuatan skema algoritmanya terlebih dahulu dengan menggunakan aplikasi Python. Hasil akhir yang didapatkan dari program Python ini sendiri adalah berupa nilai terdekat yang dihasilkan dari jarak *node* awal yaitu *device* dari *user* pertama dengan *node* tujuan yaitu *device* dari *user* lainnya.

Pengujian dari aplikasi Python yang kami buat akan menggunakan sebuah website yang bernama VisuAlgo. Website ini sendiri berfungsi untuk memberikan gambaran visual cara kerja dari algoritma Dijkstra yang dibuat dalam aplikasi Python sebelumnya. Tujuan dari penggunan website ini adalah untuk memvalidasi nilai yang didapat dari Python serta memberikan gambaran dari model *graph* pada pemetaan.



Gambar 3.2 Gambar tampilan dari website VisuAlgo[6]

Hasil akhir yang diharapkan dari kajian tugas besar ini adalah dihasilkannya urutan *node-node* yang akan dilalui dengan *cost* paling rendah atau dengan kata lain yang menempuh jarak paling dekat antara dua device yang dimiliki oleh dua *user*. Dalam kasus asli, juga dapat dihasilkan waktu yang ditempuh dari *node* asal ke *node* tujuan, namun dalam kajian ini hanya akan dihasilkan *cost* total yang dibutuhkan untuk perjalanan dari *node* asal ke *node* tujuan. Berdasarkan hasil output inilah kemudian dapat ditemukan kondisi jaringan internet yang menghubungkan antara router dan device yang paling optimal.

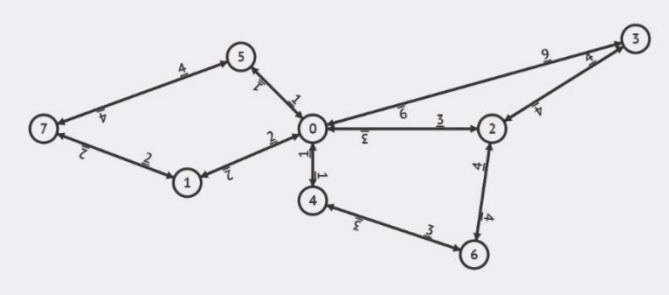
IV. HASIL EKSPERIMEN

Hasil eksperimen yang didapatkan dari kajian tugas besar ini yaitu berupa program Python yang berhasil untuk menyusun nilai terdekat yang dapat ditempuh dari *node* router menuju ke *node device* yang dipergunakan oleh *user*, untuk source code algoritma Pythonnya kurang lebih sebagai berikut:

```
class Graph():
    def __init__(self, nodes): #Fungsi untuk menginisialisasi nilai pertama dari graph
        self.distArray = [0 for i in range(nodes)]     #inisialisasi array jarak
        self.vistSet = [0 for i in range(nodes)]       #inisialsasi nodes yang dikunjungi
        self.V = nodes                                 #inisialisasi jumlah dari nodes
        self.INF = 1000000                             #inisialisasi nilai tak hingga
        self.graph = [[0 for column in range(nodes)]   #inisialisasi graph untuk matriks
                      for row in range(nodes)]

    def dijkstra(self, srcNode):
        for i in range(self.V):
          #inisialisasi jarak menuju tak hingga terlebih dahulu
          self.distArray[i] = self.INF
          #set node yang dikunjungi dianggap boolean yang bernilai False terlebih dahulu
          self.vistSet[i] = False
        self.distArray[srcNode] = 0       #insialisasi jarak pertama adalah 0
        for i in range(self.V):

            #Menentukan nilai jarak minimum dari nodes yang belum diproses
            #u selalu sama dengan Node yang diiterasi pertama kali
            u = self.minDistance(self.distArray, self.vistSet)

            #Set nilai minimum pada node yang dikunjungi
            self.vistSet[u] = True

            #Mengupdate dist[v] jika tidak dalam vistSet,
            #ada edge dari u ke v dan total beban dari src ke v melewati u
            #lebih kecil dari nilai dist[v] yang sekarang
            for v in range(self.V):
                if self.graph[u][v] > 0 and self.vistSet[v] == False and
                   self.distArray[v] > self.distArray[u] + self.graph[u][v]:

                        self.distArray[v] = self.distArray[u] + self.graph[u][v]

        self.printSolution(self.distArray, srcNode)
    #Fungsi untuk menemukan node dengan nilai jarak minimum,
    #dari himpunan node yang belum termasuk dalam jalur graph
    def minDistance(self, distArray, vistSet):

        #Inisialisasi nilai minimum untuk node berikutnya
        min = self.INF

        #Melakukan pencarian dari node yang belum dikunjungi
        for v in range(self.V):
            if distArray[v] < min and vistSet[v] == False:
                min = distArray[v]
                min_index = v

        return min_index

    def printSolution(self, distArray, srcNode):
        print (f'Node \t Jarak dari {srcNode}')
        for i in range(self.V):
            print (i, "\t\t\t", distArray[i])

#Menampilkan tabel Dijkstra berdasarkan kondisi pemetaan
ourGraph = Graph(8)
ourGraph.graph = [[0, 2, 3, 9, 1, 1, 0, 0],
        [2, 0, 0, 0, 0, 0, 0, 2],
        [2, 0, 0, 4, 0, 0, 4, 0],
        [9, 0, 4, 0, 0, 0, 0, 0],
        [1, 0, 0, 0, 0, 0, 3, 0],
        [1, 0, 0, 0, 0, 0, 0, 4],
        [0, 0, 2, 0, 3, 0, 0, 0],
        [0, 2, 0, 0, 0, 4, 0, 0]
        ];


node = int(input("Masukkan node sumber: "))
ourGraph.dijkstra(node)
```

Gambar 4.1 Source Code pemetaan dengan algoritma Dijkstra

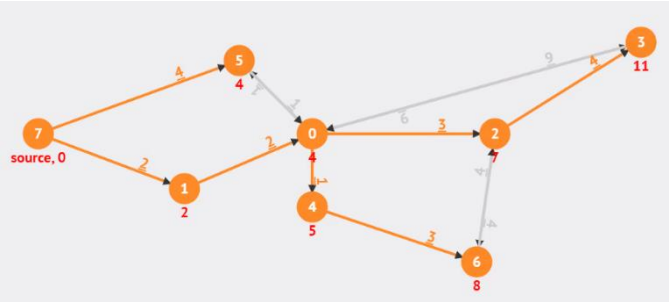
Untuk model visualisasi dari program *graph* yang ada di Python, setelah divisualisasikan dengan menggunakan website VisuAlgo gambar dari model *graph* pemetaan *Network Routing* yang ada di ITB Jatinangor adalah sebagai berikut:



Gambar 4.2 Pemetaan *router graph* pada ITB Jatinangor dengan menggunakan website VisuAlgo

Dalam visualisasi *graph* ini, *node* 7 dijadikan sebagai *node* titik awal yaitu sebagai posisi dari *device* yang dipergunakan oleh *user* 1. Sedangkan, untuk *node* 6 dijadikan sebagai *node* titik akhir yaitu sebagai posisi dari *device* yang dipergunakan oleh *user* 2. Untuk *node-node* lainnya mulai dari *node* 0 sampai dengan *node* 5, *node-node* ini merupakan *node* untuk *router* yang berfungsi sebagai alat untuk membantu proses pengiriman informasi antar *user*.

Apabila visualisasi tersebut dijalankan, maka akan muncul gambaran dari pemetaan node secara keseluruhan yang menggambarkan jarak antara *node* 7 sebagai *node* awal (*user* 1) dengan *node* 6 sebagai *node* akhir (*user* 2). Gambaran yang didapat adalah sebagai berikut:



Gambar 4.2 Pemetaan *router graph* pada ITB Jatinangor dengan menggunakan website VisuAlgo

Dari gambar, akan dilakukan pemetaan jarak dari *node* asal yaitu *node* 7 menuju ke *node* akhir tujuan yaitu *node* 6. Dari berbagai macam jarak inilah kemudian akan ditinjau nilai yang terdekat antara *node* 7 sebagai *user 1* dan *node* 6 sebagai *user* 2 yang menunjukkan nilai beban yang paling optimal untuk proses pengiriman informasi antar *user*.

Untuk perhitungan jarak yang didapat dari algoritma Dijkstra dan berdasarkan visualisasi *graph* dari VisuAlgo adalah sebagai berikut, bisa dihitung dengan menggunakan table penentuan langkah-langkah peninjauan algoritma Dijkstran sebagai berikut:

Tabel 4.1 Tabel langkah-langkah penentuan jarak terdekat dari posisi node utama (*device*) terhadap *router* internet

Step	N'	D(0), P(0)	D(1), P(1)	D(2), P(2)	D(3), P(3)	D(4), P(4)	D(5), P(5)	D(6), P(6)
0	7	∞	**2,7**	∞	∞	∞	4,7	∞
1	71	4,1		∞	∞	∞	**4,7**	∞
2	715	**4,1**		∞	∞	∞		∞
3	7150			7,0	13,0	**5,0**		∞
4	71504			**7,0**	13,0			8,4
5	715042				11,2			**8,4**
6	7150426				**11,2**			
7	71504263							
HASIL		**4,1**	**2,7**	**7,0**	**11,2**	**5,0**	**4,7**	**8,4**

Penjelasan perhitungan:

1) Untuk step 0, pengecekan dilakukan dari *node* asalnya yaitu *node* 7, tinjau semua titik yang berkaitan dengan *node* asal, untuk graph yang terhubung dan berarah maka akan memiliki nilai beban tertentu, sedangkan graph yang tidak terhubung langsung dengan *node* 7 dianggap bernilai tak hingga (∞).
2) Dari step 0, didapatkan untuk nilai beban terendah adalah yang menuju *node* 1 sebesar 2 satuan sehingga pengecekan berlanjut dari *node* 1.
3) Untuk step 1, pengecekan berlanjut dari node 1 dan meninjau *node* sebelumnya dan juga *node* baru yang terhubung dengan *node* 1 yaitu *node* 0. Didapatkan nilai node terendah adalah yang berasal dari *node* asal 7 menuju ke *node* 5 sehingga pengecekan berlanjut dari *node* 5.
4) Masuk ke step 2, peninjauan dilakukan pada *node* 0 karena hanya *node* ini yang berkaitan dengan *node* 1 dan *node* 5. Didapatkan nilai untuk menuju *node* 0 sebesar 4 satuan sehingga pengecekan berlanjut dari *node* 0.
5) Pada step 3, peninjauan dilakukan pada *node-node* yang terhubung dengan *node* 0 yaitu *node* 2, 3, dan 4. Dari pengecekan ini, didapatkan nilai *node* terendah adalah yang menuju *node* 4 yaitu sebesar 5 satuan dari *node* awal sehingga pengecekan berlanjut dari *node* 5.
6) Kemudian untuk step 4, nilai terendah didapatkan pada *node* yang menuju ke *node* 2 yaitu sebesat 7 satuan dari *node* awal. Sehingga pengecekan dilakukan berlanjut dari *node* 2.
7) Masuk ke step 5, dari hasil peninjauan didapatkan nilai terendah untuk *node* yang menuju ke arah *node* 6

yaitu bernilai sebesar 8 satuan. Pada tahap ini sebenarnya pengecekan sudah dapat dihentikan jika berdasarkan pengecekan oleh manusia. Akan tetapi, karena pengecekan dilakukan oleh algoritma Dijkstra, maka algoritma akan melakukan pengecekan pada seluruh *nodes* yang ada sehingga *node* 6 kemudian masih terjadi pengecekan yang berlanjut. Selain itu, pada *node* 3 sendiri kemudian dilakukan perbaharuan nilai karena ditemukan arah *graph* dengan beban yang lebih kecil.

8) Masuk ke step 6, nilai yang terendah hanya tersisa pada *node* 3 yaitu sebesar 11 satuan sehingga pengecekan masuk ke node 3.
9) Pada step 7, semua *node* telah ditinjau sehingga pengecekan berhenti dan *cost* minimum yang diperlukan untuk mengirimkan informasi dari *user* 1 menuju ke *user* 2 telah didapatkan.

Dari penjelasan tersebut, didapatkan bahwa untuk proses pengiriman informasi antar *user* yang paling optimal dari *node* asal yaitu *node* 7 yang merupakan *node* dari *device* pada *user* 1 adalah dengan melalui rute menuju *node* 1 terlebih dahulu, lalu menuju ke *node* 0, kemudian mengarah pada *node* 4 dan akhirnya sampai pada *node* tujuan yaitu *node* 6 sebagai *node* dari *device* pada *user* 2.

Jika dibandingkan dengan peta ITB Jatinangor, jalur paling optimal yang dapat dilalui untuk proses pengiriman informasi dari *device user* 1 sebagai pengirim adalah melalui asrama terlebih dahulu, lalu menuju gedung kuliah & GKU, setelah itu mengarah pada laboratorium dan akhirnya sampai pada *device user* 2 sebagai penerima informasi.

V. KESIMPULAN

Kesimpulan yang didapat dari eksperimen ini adalah bahwa untuk menentukan kondisi pengiriman informasi yang paling optimal yang ada di ITB Jatinangor dapat menggunakan algoritma Dijkstra karena algoritma ini memiliki sistem pemetaan yang sudah canggih sehingga sangat bermanfaat untuk menentukan pemetaan jarak terpendek untuk pengiriman informasi dari *user* satu menuju ke *user* lainnya.

Algoritma Dijkstra akan melakukan pemetaan dari berbagai router yang ada di ITB Jatinangor dan kemudian menentukan jarak yang paling optimal antara *router* dengan *user*. Untuk pemakaiannya sendiri, algoritma Djikstra memiliki beberapa syarat agar dapat berjalan dengan lancer yaitu graf yang ada harus memiliki nilai bobot positif dan memiliki arah tertentu.

VI. REFERENSI

[1] https://jatinangor.itb.ac.id/en/akses-internet/ , diakses pada 14 April 2022 pukul 19.45 WIB
[2] https://mti.binus.ac.id/2017/11/28/algoritma-dijkstra/ , diakses pada 14 April 2022 pukul 20.35 WIB
[3] https://www.section.io/engineering-education/dijkstra-python/ , diakses pada 14 April 2022 pukul 21.30 WIB
[4] https://favtutor.com/blogs/dijkstras-algorithm-cpp , diakses pada 14 April 2022 pukul 21.50 WIB
[5]https://www.cs.usfca.edu/~srollins/courses/cs336/web/slides/routing_algorithms.pdf , diakses pada 14 April 2022 pukul 22.30 WIB
[6] https://visualgo.net/id , diakses pada 16 April 2022 pukul 12.00 WIB

VII. HAL-HAL YANG DIPELAJARI

1) Mampu untuk mengaplikasikan algoritma Dijkstra untuk menentukan pengiriman informasi antar *user* tercepat
2) Mampu menyusun dan menjelaskan sistem *Graph Algorithm* beserta dengan komponen-komponen didalamnya
3) Mampu untuk memahami sistem *Network Routing* secara umum
4) Mampu untuk bekerjasama dalam tim sebagai satu kelompok

VIII. PEMBAGIAN KERJA ANGGOTA KELOMPOK

1) Muhamad Salman Hakim Alfarisi (16521513): mengurus pembuatan ppt untuk presentasi kelompok
2) Christophorus Dharma Winata (16521514): mengurus pembuatan ppt untuk presentasi kelompok
3) Ahmad Rivai Yahya (16521523): mengurus laporan dari tugas besar untuk bagian pendahuluan dan metodologi
4) Wan Aufa Azis (16521525): mengurus proses pembuatan graph dan algoritma python
5) Jason Rivalino (16521541): mengurus laporan dari tugas besar untuk bagian dasar teori, hasil percobaan, dan kesimpulan
6) Afnan Edsa Ramadhan (16521542): mengurus proses pembuatan graph dan algoritma python