Pengaplikasian Graf dalam Penentuan Rute Wisata dengan Biaya Minimum

Vijjasena(13517084)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
Vijjasena@students.itb.ac.id

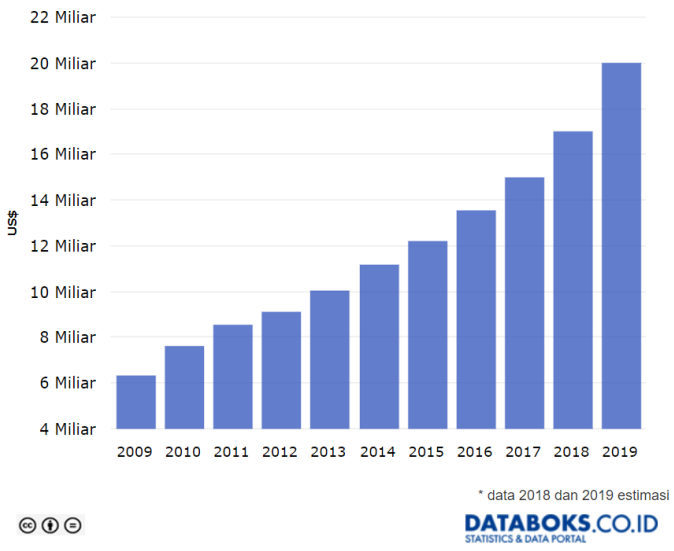
***Abstrak*—Teori Graf memiliki banyak sekali pengaplikasiannya. Sebagai contoh graf dapat ditrapkan dalam jaringan internet, pembuatan suatu peta, pembuatan game, ataupun pencarian rute. Dalam makakalah ini saya akan membahas beberapa algoritma yang dapat digunakan untuk menentukan rute wisata yang memiliki biaya minmum yang diperlukan untuk mengunjungi objek-objek wisata yang ada di suatu daerah.**

Kata kunci- graf, rute, biaya minimal, algoritma

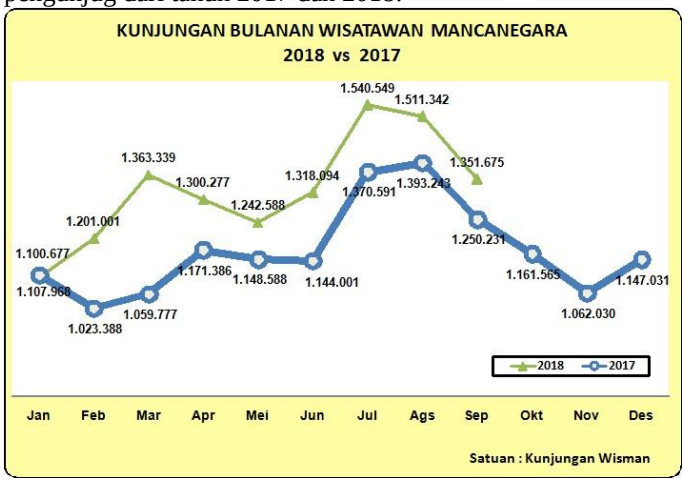
I. Pendahuluan

Sektor Pariwisata merupakan salah satu penyumbang terbesar untuk devisa Negara. Di Indonesia, kita memiliki banyak sekali tempat-tempat wisata yang banyak diminati oleh pengunjung dari berbagai belahan dunia. Seperti contoh, Pantai Kuta di Bali, Danau Toba di Sumatra Utara, Kawah Putih di Bandung, dan Kepulauan Seribu Di Jakarta.

Sektor pariwisata Indonesia dari tahun 2009 hingga sekarang mengalami peningkatan yang cukup signifikan dari tahun ketahunnya. Pada tahun 2018 Indonesia menghasilkan sekitar 17 miliar US dollar dan tahun 2019 diperkirakan akan meningkat sebanyak 3 miliar US Dollar berdasarkan data dibawah ini.[2]



Hal ini disebabkan oleh jumlah pengunjung mancanegara ygn berkunjung untuk menikmati keindahan Indonesia semakin bertambah dari tahun ke tahunnya berikut data pertambahan pengunjug dari tahun 2017 dan 2018.[5]



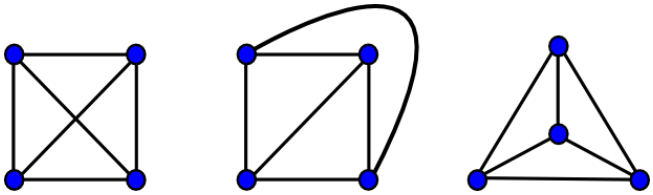
Sumber: http://www.kemenpar.go.id/asp/ringkasan.asp?c=110

Tidak hanya pengunjung dari luar negeri, penduduk dalam negeri juga menyukai berlibur baik bertujuan untuk menikmati alam, mencoba *jajanan* khas dari daerah masing masing, atau sekedar melihat budaya dari masing-masing daerah. Tetapi masalah yang sering dihadapi oleh para *travelers* ini adalah masalah biaya. Pada teori graf ataupun algoritma-algoritma lain yang menggunakan graf kita dapat menemukan biaya minimum yang perlu disiapkan agar liburan menjadi lebih hemat tanpa biaya transportasi yang berlebihan.
Dalam makalah ini penulis akan mencoba untuk menapilkan beberapa algoritma untuk menentukan biaya minimum yang diperlukan untuk mengunjungi objek-objek wisata tanpa mengunjungi tempat tersebut berulang kali.

II. Dasar Teori

2.1 Graf

2.1.1 Definisi Graf

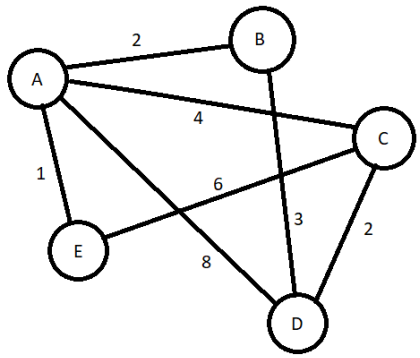


Sumber: https://www.boost.org/doc/libs/1_46_0/libs/graph/doc/planar_graphs.html

Graf Merupakan salah satu cara yang digunakan untuk merepresentasikan objek-objek diskrit dan hubungan yang terdapat diantaranya. Pada umumnya graf disimbolkan dengan G = (V,E). V pada persamaan disebut sebagai Vetrices merupakan himpunan yang tidak kosong dari sisi-sisi yang menghubungkan simpul, sedangkan E merupakan himpunan yang tidak kosong dari simpul atau *nodes* dari graf.graf memiliki beberapa jenis seperti contohnya graf planar, graf berarah, graf lingkaran, graf teratur, dan graf berbobot. Graf yang digunakan pada makalah ini adalah graf berbobot dengan bobot melambangakn biaya yang diperlukan dari satu tempat ke tempat lain. Sementara tempat dilambangkan dengan nodes ataupun simpul.[4]

2.1.2 Graf Berbobot

Graf berbobot adalah salah satu graf yang memiliki nilai untuk setiap sisi yang menghubungkan antar simpul. Nilai pada sisi terseut dapat melambangkan banyak hal seperti cintih seberapa jauh jarak yang harus dilalui dari simpul sati ke simpul yang lainnya, pada makalah ini nilai tersebut adalah biaya yang diperlukan dari satu simpi ke simpul yang lainnya. Berikt ini merupakan contoh gambar dari graf berbobot.[4]



Gambar graf berbobot

2.2 Lintasan(Path)

Lintasan yang memilikin panjan n dari simpul V_0 ke simpul akhir di V_n dalam suatu graf G berbentung barisan yang berselang-seling antara simpul dan sisi seperti berikut.

V_0, e_1, V_1, e_2, V_2 , . . . ,V_{n-1}, e_n ,V_n sehingga dapat ditulis menjadi $e_1 = (V_0 , V_1)$, $e_2 = (V_1, V_2)$, dst.

Sebagai contoh, tinjau gambar graf berbobot diatas. Untuk lintasan A,E,C,D adalah lintasan dengan barisan sisi (A,E), (E,C), (C,D).[4]

2.3 Shortest Path Problem

Masalah lintasan terpendek merupakan salah satu persoalan optimasi. Graf yang digunakan dalam persoalan ini adalah graf berbobot. Pada makalah ini bobot dari tiap sisi yang menyambungkan anatar simpul adalah biaya transportasi yang dibutuhkan. Berikut ini merupakan contoh persoalan dalam lintasan terpendek:

a. Lintasan terpendek antara dua buah simpul tertentu
b. Lintasan terpendek antara semua pasangan simpul
c. Lintasan terpendek dari simpul tertentu ke semua simpul yang lain
d. Lintasan terpendek antara dua buah simpul yang melalui beberapa simpul tertentu.[10]

2.4 Algortima Dijkstra

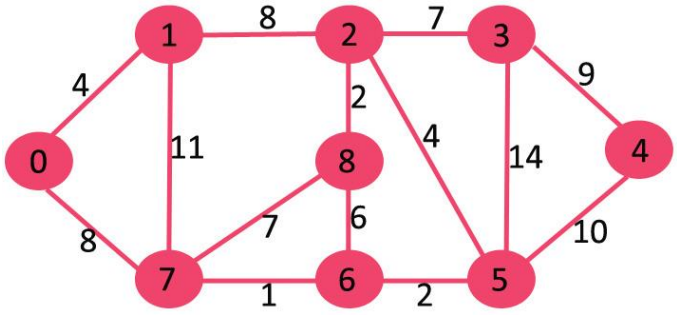
Algoritma dijkstra merupakan algoritma yang mencari lintasan terpendek dari sbuah simpul yang disebut sebagai sumber terhadap seluruh simpul yang ada di dalam graf tersebut. Pada permulaan seluruh jarak dari sumber ke simpul yang lainnya bernilai tak terhingga dan jarak simpul sumber bernilai nol(0).

Dalam pengerjaan algoritma dijkstra dibutuhkan suatu himpunan dengan nama misalkan S untuk menampung simpul yang akan diambil dari graf awal(selanjutnya akan disimbolkan dengan G) dengan nilai lintasan terpendek sudah tetap dan tidak akan berubah nantinya. Berikut ini cara kerja dari algoritma dijkstra:

1. Pilih sebuah simpul u yang tidak terdapat di S dan memiliki jarak/bobot yang paling kecil dari yang lainnya.
2. Masukan simpul tersebut ke dalam S
3. Perbarui semua jarak dari semua simpul tetangga dari simpul u. untuk perbarui setiap simpul tetangga v, jika jumlah jarak u dan besar lintasan u-v lebih kecil dari jarak v sebelumnya, maka besar jarak dari simpul v harus diperbarui.

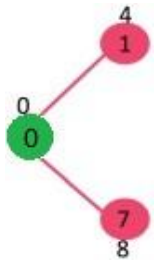
Berikut adalah contoh pengerjaan dari algoritma dijkstra.

Disiapkan sebuah graf dengan bentuk dan lintasan seperti berikut

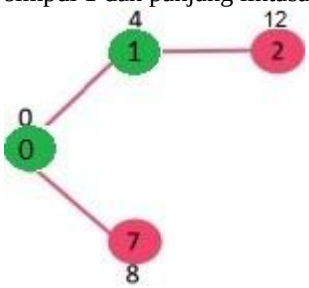


Pada awalnya himpunan S adalah kosong nilai dari setiap simpul adal tak terhingga kecuali untuk simpul 0 bernilai 0. Pada langkah pertama akan dipilih simpul dengan nilai yang paling kecil, yaitu simpul 0. Oleh karena itu simpul 0 dipindahkan ke himpunan S sehingga menjadi S{0}.

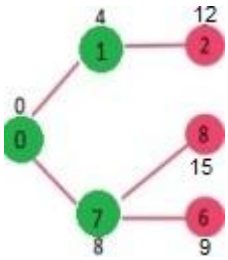
Langkah selanjutnya adalah memperbarui semua simpul yang bertetagga dengan simpul 0, yaitu simpul 1 dan simpul 7. Untik simpul 1 yang memiliki nilai tak terhinga, jumlah dari nilai simpul 0 dan lintasan antara 0-1 yang bernilai 4 tentu lebih kecil sehingga nilai simpul 1 sekarang adalah 4. Utuk simpul 7 yang juga bernilai tak hingga, jumlah dari nilai simpul 0 dan lintasan antara 0-7 yang bernilai 8 tentu lebih kecil sehingga nilai simpul 7 sekarang adalah 8. Kedua nilai dapat dilihat pada gambar berikut ini.



Setelah menyelesaikan ketiga langkah tersebut, kita akan kembali memilih simpul yang memiliki nilai paling kecil dari seluruh simpul yang ada yaitu simpul 1 dengan nilai 4. Simpul 1 lalu dipindahkan ke dalam himpunan S sehingga menjadi S{0,1}. Kemudian tetangga dari simpul 1 yaitu simpul 2 akan diperbarui nilainya dari tak hingga menjadi jumlah dari nilai simpul 1 dan panjang lintasan antara 1-2 yaitu 12.



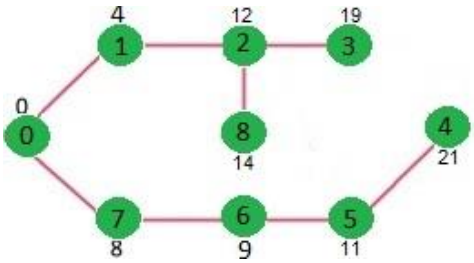
Kembali lagi ke langkah pertama, dipilih simul dengan nilai yang paling kecil yaitu simpul 7 karena simpul yang lain bernilai tak terhingga kecuali simpul 2 yang bernilai 12. Simpul 7 akan dipindahkan ke himpunan S sehingga menjadi S{0,1,7}. Tetangga dari simpul 7 akan diperbarui nilainya, simpu l 8 akan bernilai 15 berasal dari nilai simpul 7 yaitu 8 dan panjang lintasan yaitu 7. Simpul 6 akan bernilai 9.



Dipilih simpul yang memiliki nilai yang paling kecil dari seluruh simpul yang ada yaitu simpul 6. Pindahkan simpul 6 ke himpunan S sehingga menjadi S{0,1,7,6}. Tetangga dari simpul 6 harus diperbarui. Simpul 8 sebelumnya benilai 15, jumlah dari nilai simpul 6 dan nilai lintasan antara simpul 6-8 adalah 15 sehingga nilai dari simpul 8 tetap sedangkan nilai dari simpul 5 akan berubah menjadi 11 karena sebelumnya bernilai tak terhingga.

Apabila saat memprebarui simpul tetangga B, ternyata jumlah dari nilai simpul yang dipindahkan A ke himpunan S dan lintasan dari simpul A-B bernilai lebih besar dari nilai simpul B sebelumnya, maka nilai simpul B tidak diubah.

Apabila proses diatas diulang hingga simpul terakhir dipindahkan ke himpunan S maka akan menjadi seperti dibawah. Dengan nilai dari masing masing simpul adalah lintasan terpendek dari simpul sumber yaitu 0.[9]

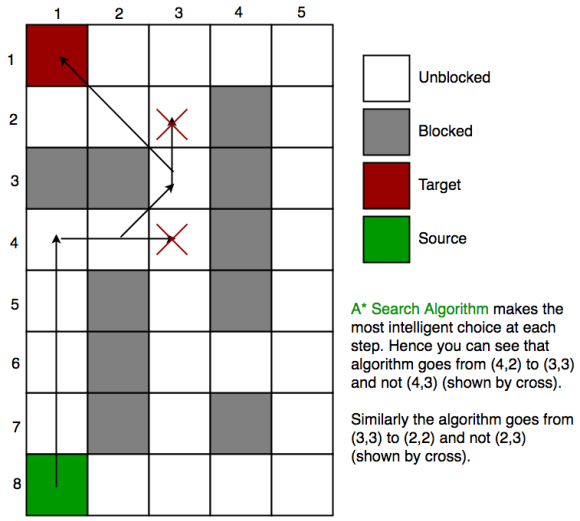


Berikut ini adalah pseudocode dari algoritma dijkstra.[8]

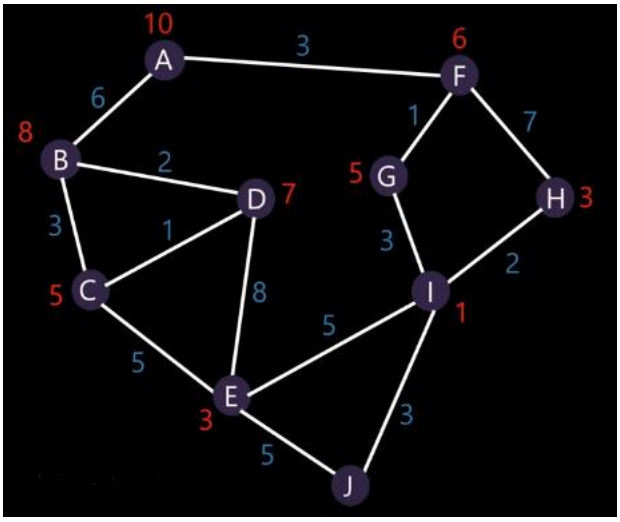
```
1  function Dijkstra(Graph, origin, destination):
2      for each vertex v in Graph:           // Initializations
3          dist[v] := infinity,    previous[v] := undefined
4      dist[origin] := 0                     // Distance from origin to origin is 0
5      Q := the set of all nodes in Graph // All nodes in the graph are unoptimized - thus are in Q

6      while Q is not empty:                 // The main loop
7          u := vertex in Q with smallest dist[]
8          if dist[u] = infinity:     break                    // There is no route from origin to destination
9          if u = destination:        break                    // reach the destination
10         remove u from Q
11         for each neighbor v of u:         // where v has not yet been removed from Q.
12             alt := dist[u] + cost_between(u, v)
13             if alt < dist[v]:             //If the distance is less than the previously recorded distance
14                 dist[v] := alt,       previous[v] := u

15     //read the shortest path
16     S := empty sequence
17     u := destination
18     while previous[u] is defined:
19         insert u at the beginning of S
20         u := previous[u]

21     return S
```

2.5 Algoritma A*(A Star)

Seperti algoritma dijkstra, algoritma A star merupakan algoritma yang paling sering digunakan untuk mencari lintasan yang paling pendek dari satu titik ke titik lainnya.



Cara kerja dari algoritma A star tidak jauh berbeda dengan algoritma dijkstra. Misalkan ada graf G sebagi berikut.



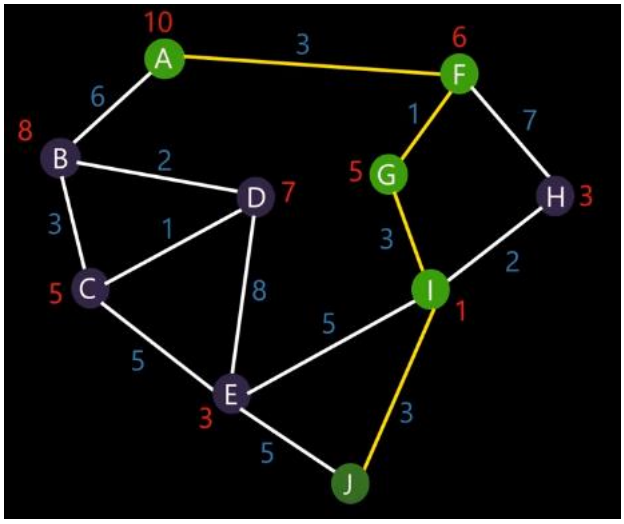
Pada graf G nilai yang berwarna merah adalah nilai yang dimiliki oleh simpul tersebut sedangkan nilai yang berwarna biru adalah nilai lintasan dari suatu simpul ke simpul lainnya. Untuk mengambil keputusan simpul mana yang akan diambil selanjutnya adalah melaui fungsi
F(n) = g(n) + h(n)
Dengna g(n) dalah besar lintasan dari simpul awal ke simpul n dan h(n) adalah nilai yang dimiliki simpul n. misalkan dari simpul A ke J akan dilakukan pencarian lintasan yang paling pendek.

Pada simpul A terdapat dua tetangga yaitu B dan F untuk simpul F memiliki F(F) = 3 + 6 yaitu 9. Sedangkan simpul B memiliki F(B) = 6 + 8 yaitu 14 antara simpul B dan F, F memiliki jumlah yang lebih kecil maka simpul selanjutnya yang diambil adalah simpul F. sehingga lintasan yang telah diambil menjadi A-F.

Pada simpul F, terdapat dua tetangga simpul yaitu G dan simpul H. untuk simmpul G memiliki F(G) = (1+3) + 5 yaitu 9 dan untuk simpul H memiliki F(H) = (3+7) + 3 yaitu 13, antara simpul G dan simpul H. Simpul G memiliki nilai yang lebih kecil sehingga lintasan berikutnya yang diambil adalah G dan lintasan menjadi A-F-G.

Pada simpul G, hanya memiliki satu simpul tetangga yaitu I. simpul I memiliki F(I) = (3+1+3) + 1 yaitu 8 dan lintasan selanjutnya menjadi A-F-G-I.

Pada simpul I, I memiliki tiga simpul tetangga yaitu H, E, dan J. untuk simpul H memiliki F(H) = (3+1+3+2) + 3 yaitu 12 dan simpul E memiliki F(E) = (3+1+3+5) + 3 yaitu 15 dan untuk simpul F(J) = (3+1+3+3) + 0 yaitu 10. Karena F(J) memiliki nilai yang paling dari F(H) dan F(E) sehingga lintasan selanjutnya yag akan diambil adalah F(J). karena J adalah tujuan akhir dari pencarian lintasan maka pencarian lintasan telah selesai dengan lintasan akhir adalah A-F-G-I-J. dengan gambar seperti berikut.[7]

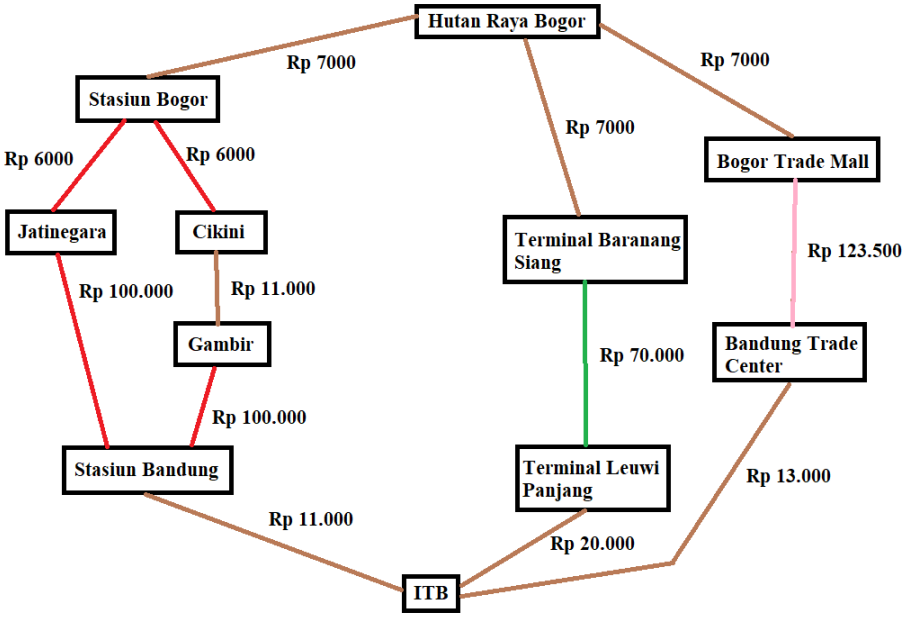


Berikut ini adalah pseudocode dari algoritma A*[6]

```
Input : A graph G(V,E)  with source node start and goal node end.

Output : Least cost path from start to end.
Steps:

Initialise

        open_list = { start }
        closed_list={}
        g(start) = 0
        h(start) = heuristic_function(start,end))
        f(start) = g(start)+h(start)

While open_lisy is not empty
        m=Node on top of open_list, with least f
        If m == end
                return
        Remove m fro open_list
        Add m to closed_ist
        For each n in child(m)
                ff n in closed_list
                        Continue
        Cost=g(m)+distance(m,n)
        If n in open_list and cost<g(n)
                remove n from open_list as new path is better
        if n in closed_list and cost<g(n)
                remove n from closed_list
        if n not in open_list and n not in closed_list
                add n to open_list
                g(n)=cost
                h(n)= heuristic_function(n,end)
                f(n)=g(n)+h(n)
```

III. Penerapan Algoritma untuk Menentukan Biaya Minimum

Untuk mensimulasikan penentuak biaya dari satu tempat ke tempat lain, penulis mengambil contoh perjalanan dari ITB ke Hutan Raya Bogor dengan graf yang bisa dilihat sebagai berikut.



Pada graf di atas terdapat juga transportasi yang digunakan untuk menempuh perjalanan dari satu simpul ke simpul lainnya. Sisi berbobot merupakan biaya yang diperlukan. Perlu diketahui jumlah biaya-biaya yang tercanum pada grah hanya estimasi dari harga. Hal ini dikarenakan harga dapat berubah sewaktu-waktu tergantung dari situasi dan kondisi yang berlangsung. Semua estimasi harga didapatkan melalui aplikasi go-jek, situs KAI, dan aplikasi traveloka.

Untuk pewarnaan pada sisi, warna merah merupakan transportasi kereta, warna hijau merupakan traportasi dengan bis, warna pin adalah jasa travel, dan warna cokelat merupakan transportasi ojek.

Untuk memudahkan graf tersebut dianalisis nama simpul-simpul yang ada dijadikan nomor dengan konversi sebagai berikut.

Nomor	Simpul
1	ITB
2	Stasiun Bandung
3	Jatinegara
4	Gambir
5	Cikini
6	Stasiun Bogor
7	Stasiun Leuwi Panjang
8	Stasiun Baranang siang
9	Bandung Trade Center
10	Bogor Trade Mall
11	Hutan Raya Bogor

Untuk mencari biaya minimum dari graf tersebut dapat digunakan dua contoh algoritma yang telah dijabarkan pada dasar teori. Untuk pengaplikasian dengan algoritma dijkstra dapat dilihat penjabarannya pada tabel berikut ini.

Langkah	Simpul	lintasan yang sudah dikujungi										Jarak dari Simpul Awal										
		1	2	3	4	5	6	7	8	9	11	1	2	3	4	5	6	7	8	9	10	11
1	1	1	0	0	0	0	0	0	0	0	0	0	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	2	1	1	0	0	0	0	0	0	0	0	0	11k	∞	∞	∞	∞	20k	∞	13k	∞	∞
3	9	1	1	0	0	0	0	0	0	1	0	0	11k	111K	111K	∞	∞	20k	∞	13k	∞	∞
4	7	1	1	0	0	0	0	1	0	1	0	0	11k	111K	111K	∞	∞	20k	∞	13k	136,5K	∞
5	8	1	1	0	0	0	0	1	1	1	0	0	11k	111K	111K	∞	∞	20k	90K	13k	136,5K	∞
6	11	1	1	0	0	0	0	1	1	1	1	0	11k	111K	111K	∞	∞	20k	90K	13k	136,5K	97K
7	4	1	1	0	1	0	0	1	1	1	1	0	11k	111K	111K	∞	114K	20k	90K	13k	114K	97K
8	5	1	1	0	1	1	0	1	1	1	1	0	11k	111K	111K	122K	114K	20k	90K	13k	114K	97K
9	3	1	1	1	1	1	0	1	1	1	1	0	11k	111K	111K	122K	114K	20k	90K	13k	114K	97K
10	6	1	1	1	1	1	1	1	1	1	1	0	11k	111K	111K	122K	114K	20k	90K	13k	114K	97K

Tabel : Langkah Penerapan Algoritma Dijkstra dalam penentuan biaya transportasi minimum (0 sebagai belum dikujungi, 1 sebagai sudah dikunjungi)

Hasil dari penerapan algoritma dijkstra adalah biaya tranportasi minimum dari simpul awal nomor 1 yaitu ITB dan titik akhir hutan raya bogor yang diwakilkan dengan nomor 11 dengan jumlah biaya minimum sebesar 97.000 rupiah. Dengan lintasan yang ditempuh adalah.

(1) ITB
(7) Terminal Leuwi Panjang
(8) Terminal Baranang Siang
(11) Hutan Raya Bogor

IV. Kesimpulan

Teori Graf dapat diaplikasikan ke dalam Perancangan rute wisata yang memiliki biaya transportasi yang minimum. Dengan tiap simpul pada graf dinyatakan sebagai pemberhentian atau tempat singgah untuk sampai ke tujuan akhir. Sedangkan bobot sisi yang menghubungkan simpul-simpul dinyatakan sebagai biaya transportasi yang diperlukan untuk berpindah dari simpul satu ke yang lainnya.

Algoritma dijkstra dan A* sangat berguna dalam pencarian lintasan terpendak. Algoritma dijkstra terbukti dapat menghasilkan lintasan yang mempunyai biaya yang paling minimun untuk setiap simpul pada graf. Dengan paduan rute wisata dengan biaya minimun=m diharapkan pembaca dapat menentukan rute perjalan yang lebih hemat.

V. Ucapan Terima Kasih

Penulis pertama ingin mengucapkan puji syukur kepada Tuhan Yang Maha Esa karena berkat rahmat dan berkatNya Penulis dapat menyelesaikan makalah ini. Penulis juga ingin berterima kasih kepada orang tua karena selalu memberikan dukungan dan arahan kepada Penulis. Penulis juga ingin mengucapkan terimakasih kepada Dr. Judhi Santoso, dosen yang mengajarkan matematika diskrit, termasuk di dalamnya teori graf dan pengolahannya. Terakhir Penulis juga ingin berterima kasih kepada rekan-rekan yang telah membantu pembuatan makalah ini.

VI. Daftar Referensi

[1] http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2009-2010/Makalah0910/MakalahStrukdis0910-075.pdf Diakses pada 8 desember 2018
[2] https://databoks.katadata.co.id/datapublish/2018/09/10/berapa-pendapatan-devisa-dari-sektor-pariwisata-indonesia Diakses pada 8 desember 2018
[3] https://www.boost.org/doc/libs/1_46_0/libs/graph/doc/planar_graphs.html Diakses pada 8 desember 2018
[4] Munir, R. Matematika Diskrit. Bandung: Informatika Bandung,2005.
[5] http://www.kemenpar.go.id/asp/ringkasan.asp?c=110 Diakses pada 8 desember 2018
[6] https://www.researchgate.net/figure/A-search-algorithm-Pseudocode-of-the-A-search-algorithm-operating-with-open-and-closed_fig8_232085273 Diakses pada 8 desember 2018
[7] https://www.geeksforgeeks.org/a-search-algorithm/ Diakses pada 8 desember 2018
[8] https://www.researchgate.net/figure/The-pseudo-code-of-Dijkstras-algorithm-adapted-from Wikipedia-2011_fig2_233025836 Diakses pada 8 desember 2018
[9] https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/ Diakses pada 8 desember 2018
[10] http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2014-2015/Makalah2015/Makalah_IF221_Strategi_Algoritma_2015_009.pdf Diakses pada 8 desember 2018

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, 8 Desember 2018

Vijjasena(13517084)