Penerapan Algoritma Pemrograman Dinamis Terhadap Persoalan Travelling Salesman Problem

Michael Philip Gunadi / 1351912
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
mpg066@gmail.com

***Abstract*—Travelling Salesman Problem (TSP) merupakan persoalan klasik dalam dunia ilmu komputer, dan merupakan masalah mayoritas orang dengan tuntutan mobilitas yang tinggi. Persoalan ini juga merupakan persoalan utama untuk penyedia jasa layanan antar barang untuk dapat mengirimkan seluruh barang ke seluruh destinasi seefektif mungkin. Makalah ini bertujuan untuk menjelaskan penerapan algoritma pemrograman dinamis terhadap persoalan TSP untuk membantu penyelesaian persoalan TSP.**

Kata Kunci—pemrograman dinamis; efektif; maksimal; optimal;

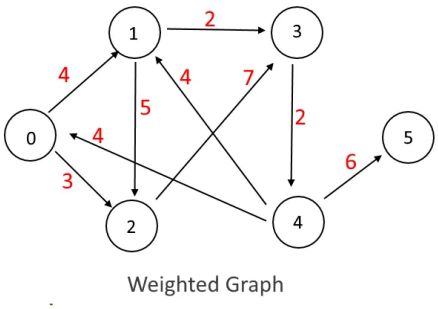
I. PENDAHULUAN

Travelling Salesman Problem (TSP) adalah suatu persoalan yang terkenal di dunia ilmu komputer, persoalan ini merupakan persoalan untuk mencari rute terpendek dan terefektif untuk mendatangi seluruh tempat yang ada. Persoalan ini menjadi menarik karena diantara banyaknya rute yang mungkin, dicari sebuah rute yang paling efektif tanpa menghiraukan tujuan utamanya : menentukan rute terpendek dan terefektif untuk mendatangi keseluruhan tempat. Persoalan ini menjadi terkenal karena persoalan ini sangat mudah untuk dijelaskan namun sangat sulit untuk diselesaikan. Persoalan TSP merupakan persoalan utama dari perusahaan penyedia jasa layanan kirim barang seperti Anteraja, JNE, dan lain sebagainya, optimasi sedikit dalam penentuan rute pengiriman barang mungkin dapat menghemat waktu dan biaya.

Anteraja, sebuah perusahaan terbuka dengan kode emiten ASSA ini merupakan perusahaan pengiriman barang yang baru naik daun. Perusahaan ini mencatatkan bahwa jumlah barang yang dikirimkan sudah mencapai angka 500.000 parsel/hari. Untuk perusahaan yang sudah lama berdiri seperti JNE, pada Januari 2020 saja saat dikabarkan adanya penurunan parsel yang dikirim per hari, JNE masih mencatatkan angka 1,4 juta parcel/hari yang dikirimkan ke seluruh Indonesia. Dengan menyelesaikan persoalan akan mana lokasi yang akan dituju duluan dan dituju belakangan, perusahan-perusahaan besar dengan demand yang besar ini dapat menghemat banyak waktu dan uang juga dapat meningkatkan kepuasan pelanggan.

Gambar 1. Kurir AnterAja sedang melakukan pengiriman
Sumber : investasi.kontan.co.id

Memang pada praktiknya ada banyak hal diluar perkiraan yang dapat mempengaruhi keefektivan dari suatu rute pengantaran barang yang telah dipilih, seperti ketidakberadaan pelanggan di rumah tempat alamat barang tersebut dikirimkan, cuaca yang tidak menentu, kondisi jalan yang sering tidak menentu dan lain sebagainya. Namun untuk makalah ini, akan dibahas benar-benar faktor eksak yang dapat diperhitungkan untuk menentukan apakah suatu rute merupakan rute yang memiliki potensi untuk menjadi rute yang paling efektif untuk suatu pengantaran, seperti jumlah lokasi tujuan dan "harga" yang dibutuhkan untuk mencapai suatu titik lokasi dari suatu titik lokasi yang spesifik (selayaknya persoalan TSP pada umumnya yang merupakan persoalan graf berarah berbobot).



Gambar 2. Contoh graf berarah berbobot
Sumber : algorithms.tutorialhorizon.com

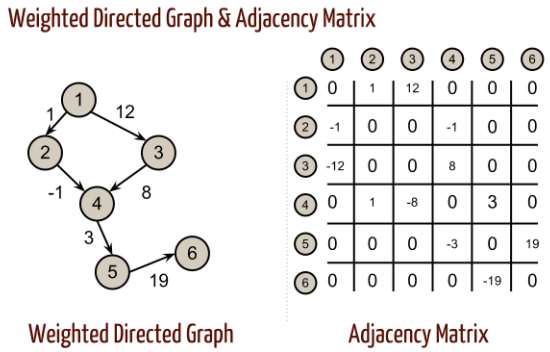
II. LANDASAN TEORI

A. *Graf*

Graf merupakan sebuah struktur data khusus dalam dunia ilmu komputer, graf terdiri dari 2 komponen utama yaitu *nodes* dan *edges* dimana *nodes* menyatakan sebuah titik yang ada dalam sebuah graf, dan edges merupakan *tuple* dengan minimal 2 elemen yang menandakan kedua *nodes* yang terhubung, dan biasanya elemen ketiganya merupakan *cost* atau biaya yang dibutuhkan untuk pergi dari suatu *node* ke *node* lain.

1. Properti Umum Struktur Data Graf

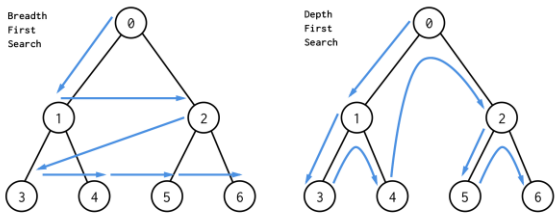
   Struktur data graf pada umumnya digambarkan dengan menggunakan matriks ketetanggaan dimana isi dari matriks ke (r,c) menandakan *cost edge* dari *node* r menuju *node* c, apabila *cost*-nya 0 berarti tidak ada edge dari *node* r menuju *node* c. Dalam kasus graf berbobot berarah, matriks ketetanggaan yang terbentuk belum tentu merupakan matriks yang simetri, dimana matriks(r,c) belum tentu sama dengan matriks(c,r).



Gambar 3. Gambar graf berbobot berarah dan representasinya dalam matriks ketetanggaan
Sumber : www.bournetocode.com

2. Metode Penelusuran Graf

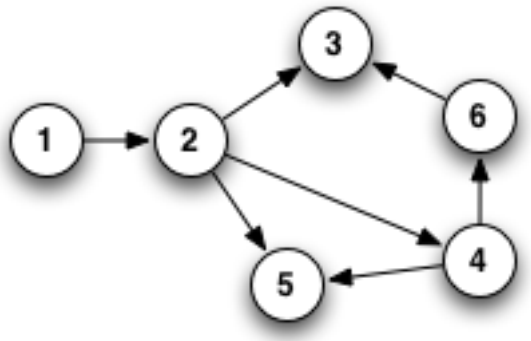
   Ada dua algoritma utama untuk menyelusuri suatu graf, yaitu dengan DFS (Depth-First-Search) dan BFS (Breadth-First-Search). Algoritma DFS adalah algoritma untuk menelusuri seluruh *node dalam graf* dengan mengunjungi *node* yang merupakan anak dari *node* yang saat ini dikunjungi hingga tidak ada lagi *node* anak yang dapat dikunjungi (kasus mengunjungi *node* daun), kemudian diteruskan dengan mengunjungi anak lain dari node sebelumnya. Sedangkan algoritma BFS adalah algoritma untuk menelusuri seluruh *node* dalam graf yang merupakan anak dari generasi yang sama dengan *node* yang sedang dikunjungi saat ini.



Gambar 4. Ilustrasi penelusuran graf dengan algoritma DFS dan BFS
Sumber : Hackerearth.com

3. Graf Berarah

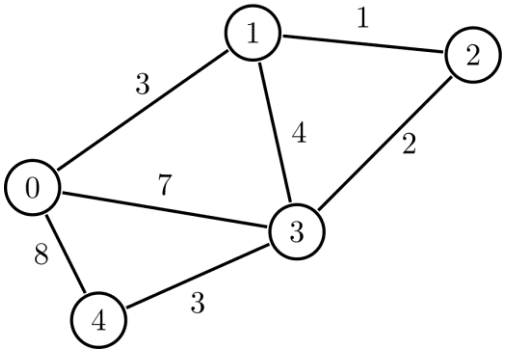
   Graf berarah merupakan jenis graf khusus dimana ketika terdapat *edge* dari *node* a ke *node* b, belum tentu ada *edge* dari *node* b ke *node* a. Hal ini serupa dengan kota yang memiliki jalan sejalur dimana hanya memungkinkan arus kendaraan dari satu tempat ke tempat lain namun tidak sebaliknya.



Gambar 5. Ilustrasi Graf Berarah
Sumber : Stackoverflow

4. Graf Berbobot

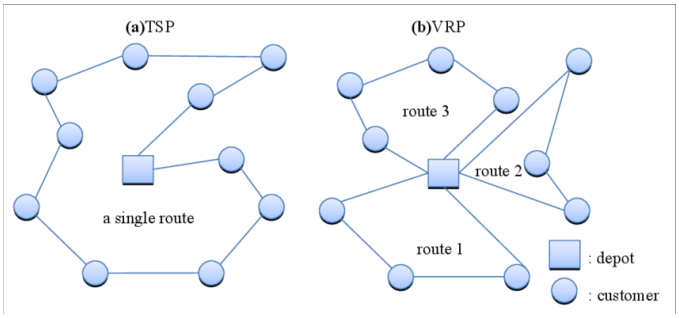
   Graf berbobot adalah graf khusus dimana untuk setiap *edge* yang terdaftarkan memiliki nilai yang menyatakan *cost* untuk menelusuri *edge* tersebut. Untuk *node* yang tidak terhubung, nilai edgenya akan bernilai 0.



Gambar 6. Ilustrasi Graf Berbobot
Sumber : hyperskill.org

B. *Travelling Salesman Problem*

Travelling salesman problem (TSP) merupakan persoalan terkenal dalam dunia informatika. Persoalan ini secara sederhana merupakan persoalan untuk mengunjungi seluruh *node* yang ada dalam graf berbobot berarah dengan rute yang paling efektif dan optimal. Persoalan ini merupakan persoalan yang sangat mudah untuk dijelaskan namun sangat sulit untuk diselesaikan. Ada banyak pendekatan untuk menyelesaikan persoalan TSP, pendekatan yang paling naif adalah dengan menggunakan brute-force dimana kita menguji coba seluruh kemungkinan rute yang mungkin untuk mendapatkan rute yang paling optimal. Pendekatan lain yang dapat membantu adalah pendekatan branch and bound, dimana kita memilih rute yang terpendek untuk node tertentu, dan mengulangi terus prosedur ini hingga seluruh node dalam graf dikunjungi.



Gambar 7. Ilustrasi Persoalan Travelling Salesman Problem
Sumber : ResearchGate

C. *Pemrograman Dinamis*

Pemrograman dinamis secara sederhana merupakan optimisasi untuk persoalan rekursif/iteratif. Pemrograman dinamis memungkinkan optimisasi persoalan rekrusif/iteratif dengan memecah persoalan besar menjadi persoalan-persoalan kecil kemudian menyimpan solusi persoalannya untuk kemudian dipakai sebagai bekal untuk menyelesaikan persoalan utamanya. Pemrograman dinamis memiliki 2 pendekatan utama untuk menyelesaikan suatu persoalan, yaitu dengan metode tabulasi dan memoisasi. Metode tabulasi secara sederhana merupakan metode untuk menyelesaikan suatu persoalan secara *bottom-up* yaitu dengan menyelesaikan sub-masalah sub-masalah yang ada dan kemudian menyimpan hasilnya untuk menjadi bekal untuk menyelesaikan masalah utamanya. Metode memoisasi secara sederhana merupakan metode untuk menyelesaikan suatu persoalan secara top-bottom yaitu dengan memecah masalah utama menjadi masalah-masalah yang lebih kecil sehingga apabila masalah-masalah kecil tersebut selesai, maka masalah utamanya besar.

III. PEMBAHASAN

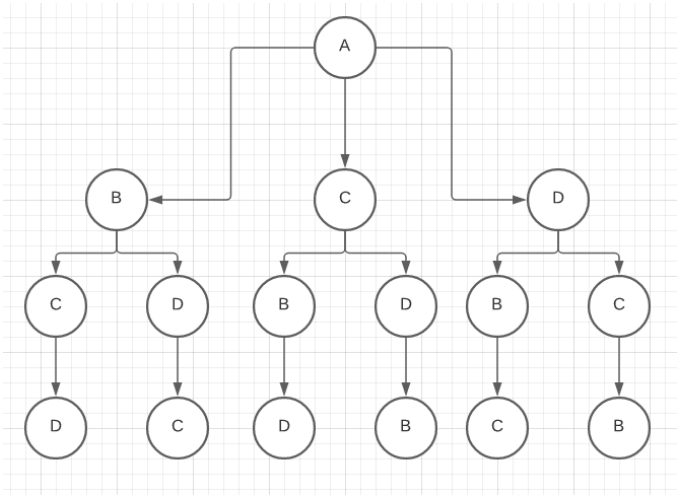
Agar kita dapat lebih mudah memahami persoalan ini, maka kita akan menggunakan matriks ketetanggaan sebagai berikut untuk kita analisa agar kita mendapatkan rute terpendek untuk mengunjungi seluruh *node* yang ada dalam graf ini. Agar graf yang terbentuk lebih menggambarkan kondisi yang ada dalam kehidupan nyata, maka kita akan menggunakan graf berbobot berarah.

Nodes	A	B	C	D
A	0	10	15	20
B	5	0	9	10
C	6	13	0	12
D	8	8	9	0

Tabel 1. Tabel matriks ketetanggaan

Perhatikan bahwa matriks ketetanggaan diatas memiliki 4 buah *node* yaitu node A,B,C,D dan perhatikan bahwa matriks ketetanggaan tersebut bukanlah matriks simetris yang berarti *cost* dari *node* A menuju *node* B dapat memiliki *cost* yang berbeda untuk jalur sebaliknya dari *node* B ke *node* A. Persoalan TSP dapat kita selesaikan melalui pendekatan pemrograman dinamis.

Asumsikan pada awalnya "kurir" yang akan mengantar barang tersebut ada pada *node* A, maka dari itu *nodes* lainnya yang harus ia kunjungi adalah *node* B,C, dan juga D. Dengan mengetahui rute terpendek untuk menelusuri *nodes* B,C, dan juga D dari titik awal *node* A, kita dapat menyelesaikan persoalan TSP ini. Namun sayangnya rute apabila kita memutuskan untuk mengunjungi *node* B terlebih dahulu akan menghasilkan hasil yang berbeda apabila kita mengunjungi *node* C atau *node* D terlebih dahulu setelah kita mengunjungi *node* A, maka dari itu kita dapat membuat *tree* sederhana seperti yang tertera dibawah ini untuk menentukan seluruh kemungkinan rute yang ada. Lalu andaikan jika kita memutuskan bahwa *node* berikutnya yang dipilih untuk dikunjungi adalah *node* B dari *node* A, maka dengan menemukan rute terpendek untuk menelusuri *node* C dan D melalui B, maka persoalan TSP tersebut selesai. Dengan melakukan pendekatan *brute force* seperti ini, kita dapat menyelesaikan persoalan TSP ini dengan kompleksitas eksponensial yang mana sangat tidak baik.



Gambar 8. Diagram seluruh kemungkinan rute yang mungkin apabila kurir memulai perjalanan dari *node* A

Pendekatan persoalan menggunakan *brute force* ini dapat membantu kita untuk membentuk rumus umum yang dapat kita manfaatkan untuk penyelesaian persoalan ini dengan pendekatan pemrograman dinamis. Perhatikan bahwa *cost* atau

biaya untuk mengunjungi *node* B,C dan D dari node A adalah minimum dari nilai node A menuju node X dimana node X merupakan salah satu node dari node B,C, dan D, ditambah dengan *cost* dari node X tersebut menuju ke node lain selain node X, dalam hal ini node selain X adalah himpunan node {B,C,D}-{X}. Maka secara formal kita dapat menuliskan formula penyelesaian persoalan TSP ini sebagai berikut :

g(1,{2,3,4}) = min({cost(1,X) + g(X, {2,3,4}-{X})})

- g(a,b) : fungsi yang mengembalikan nilai jarak dari rute yang paling optimal untuk mengunjungi seluruh node yang ada dalam himpunan b dari *node* a.
- cost(a,b) : fungsi yang mengembalikan nilai yang dibutuhkan untuk berjalan dari node a, menuju node b.

Maka dalam persoalan TSP ini, apabila kita memulai dari *node* A, maka kita mendapat persamaan awalnya adalah

g(A,{B,C,D}) = min({cost(A,X) + g(X, {B,C,D}-{X})})

Dimana nilai X merupakan *node* yang belum kita ketahui secara pasti yang mana namun kita tau bahwa kandidat dari X adalah {B,C,D}. Karena persamaan ini nampak tidak dapat diselesaikan, maka kita dapat menyelesaikan persoalan ini menggunakan pendekatan *bottom-up*. Apabila kita fokus kepada 2 generasi terbawah dari *tree* yang diilustrasikan diatas, kita dapat mendapatkan bahwa ada 6 jalur yang mungkin dari kiri ke kanan yaitu :

- g(C,{D}) = 12
- g(D,{C}) = 9
- g(B,{D}) = 10
- g(D,{B}) = 8
- g(B,{C}) = 9
- g(C,{B}) = 13

Perhatikan bahwa dengan data diatas, kita sudah setengah jalan menyelesaikan persoalan TSP ini, setidaknya kita sudah mengetahui untuk 2 *node* terakhir, mana jalur yang paling efisien. Perhatikan bahwa untuk setiap pasangan 2 *node* kemudian akan mengerucut menjadi satu jalur, seperti g(C,{D}) dengan g(D,{C}) merupakan 2 kemungkinan jalur untuk menelusuri *node* C dan D, perhatikan bahwa 2 jalur ini tidak mungkin sama-sama merupakan jalur yang efektif untuk mengunjungi *node* C dan D dari node B. Maka dengan memasukan rumus umum yang telah kita tuliskan diatas, kita dapat mengeliminasi salah satu kemungkinan jalur dari daftar kandidat rute yang efektif. Maka dari itu untuk awalan node B dan node berikutnya yang ingin dikunjungi adalah node {C,D} kita menemukan kesimpulan :

g(B, {C,D}) = min({Cost(B,X) + g(X, {C,D}-{X})})

Sejauh ini nilai X yang mungkin adalah anggota dari *nodes* yang ingin dikunjungi, yaitu C atau D. Apabila kita mensubtitusikan X dengan kedua node tersebut, maka kita akan memperoleh :

Untuk (X = C)

g(B, {C,D}) = Cost(B,C) + g(C,{D})

g(B, {C,D}) = 9 + 12 = 21

Untuk (X = D)

g(B, {C,D}) = Cost(B,D) + g(D,{C})

g(B, {C,D}) = 10 + 9 = 19

Dari sini kita dapat mengambil kesimpulan bahwa apabila kita ingin menelusuri *nodes* C dan D dari *node* awal B, maka lebih efektif apabila kita mengunjungi *node* D terlebih dahulu baru kemudian *node* C, daripada kebalikannya. Maka dari itu kita sudah membentuk sebagian akhir dari solusi persoalan TSP ini yaitu apabila kita memulai dari *node* B dan *nodes* yang ingin dikunjungi adalah *node* C dan D, maka rute yang terbentuk adalah B-D-C. Apabila kita mengulangi proses yang serupa untuk kesluruhan *nodes* lainnya, maka kita akan memperoleh

Untuk mengunjungi B,D dari node C

(X=B)

g(C, {B,D}) = Cost(C,B) + g(B,{D})

g(C, {B,D}) = 13 + 10 = 23

(X=D)

g(C, {B,D}) = Cost(C,D) + g(D,{B})

g(C, {B,D}) = 12 + 8 = 20

Untuk mengunjungi B,C dari node D

(X=B)

g(D, {B,C}) = Cost(D,B) + g(B,{C})

g(D, {B,C}) = 8 + 9 = 17

(X=C)

g(D, {B,C}) = Cost(D,C) + g(C,{B})

g(D, {B,C}) = 9 + 13 = 22

Maka dari penjabaran di atas, kita dapat mengambil kesimpulan bahwa apabila kita ingin mengunjungi *nodes* C,D dari *node* B, maka rute yang dipilih adalah rute B-D-C dengan *total cost* : 19, sedangkan apabila kita ingin mengunjungi *nodes* B,D dari *node* C maka rute yang dipilih adalah rute C-D-B dengan *total cost* : 20, kemudian apabila kita ingin mengunjungi *nodes* B,C dari *node* D maka rute yang akan dipilih adalah D-B-C dengan *total cost* : 17. Sampai titik ini kita telah berhasil menyelesaikan persoalan TSP ini setengah jalan. Langkah terakhir adalah memasukkan *node* awal sebagai penentu rute mana yang akan diambil.

Untuk mengunjungi B,C,D dari node A

(X=B)

g(A, {B,C,D}) = Cost(A,B) + g(B,{C,D})

g(A, {B,C,D}) = 10 + 19 = 29

(X=C)

g(A, {B,C,D}) = Cost(A,C) + g(C,{B,D})

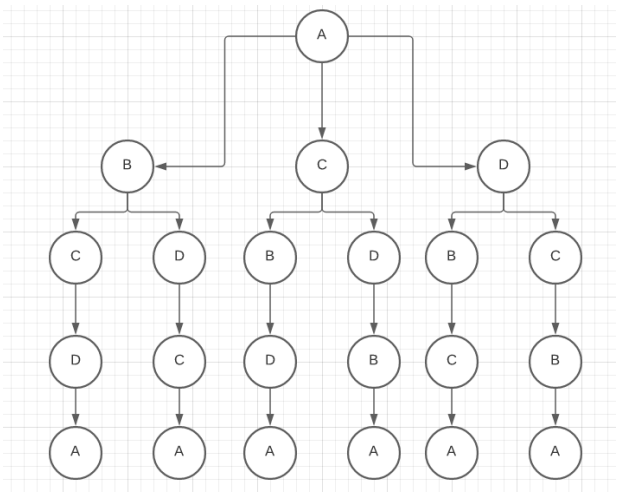
g(A, {B,C,D}) = 15 + 20 = 35

(X=D)

g(A, {B,C,D}) = Cost(A,D) + g(D,{B,C})

g(A, {B,C,D}) = 20 + 17 = 37

Hingga titik ini, persoalan TSP dengan 4 *nodes* yaitu A,B,C dan D, dengan matriks ketetanggaan seperti yang ada di atas telah selesai dengan rute optimal untuk mengunjungi keseluruhan *node* yang ada dari *node* A adalah 29 dengan rute A-B-D-C. Namun pertanyaannya apakah jawaban tersebut valid untuk setiap awalan? Jawabannya tidak. Apabila kita memulai dari *node* A dan hendak menelusuri node B,C,D, maka jawaban yang kita peroleh adalah 29 dengan rute A-B-D-C, namun apabila kita tidak memulai dari *node* A maka jawaban tersebut tidak lagi valid. Namun bukan berarti apa yang kita lakukan sejauh ini tidak berarti, untuk mengatasi kasus apabila kita tidak tau *node* awal, atau *node* awal tidak ditentukan, kita dapat melakukan sedikit perubahan pada jawaban kita dengan mencari rute siklis yang paling optimal dengan cara menambah 1x lagi penjabaran formula tersebut hingga kembali ke titik awal (dalam kasus ini kembali lagi ke *node* A), maka pohon kemungkinan rute yang kita dapatkan menjadi sebagai berikut.



Gambar 9. Diagram seluruh kemungkinan rute yang mungkin untuk persoalan 4 nodes

Maka secara singkat kita dapat memperoleh penjabaran sebagai berikut setelah penyesuaian :

- g(D,{A}) = 8
- g(C,{A}) = 6
- g(B,{A}) = 5

Kemudian untuk penjabaran tingkat berikutnya kita akan memeproleh :

- g(C,{D,A}) = 20
- g(D,{C,A}) = 15
- g(B,{D,A}) = 18
- g(D,{B,A}) = 13
- g(B,{C,A}) = 15
- g(C,{B,A}) = 18

Kemudian untuk tingkat atasnya, kita akan memperoleh penjabaran :

Untuk mengunjungi C,D dari node B

Untuk (X = C)

g(B, {C,D,A}) = Cost(B,C) + g(C,{D,A})

g(B, {C,D,A}) = 9 + 20 = 29

Untuk (X = D)

g(B, {C,D,A}) = Cost(B,D) + g(D,{C,A})

g(B, {C,D,A}) = 10 + 15 = 25

Untuk mengunjungi B,D dari node C

(X=B)

g(C, {B,D,A}) = Cost(C,B) + g(B,{D,A})

g(C, {B,D,A}) = 13 + 18 = 31

(X=D)

g(C, {B,D,A}) = Cost(C,D) + g(D,{B,A})

g(C, {B,D,A}) = 12 + 13 = 25

Untuk mengunjungi B,C dari node D

(X=B)

g(D, {B,C,A}) = Cost(D,B) + g(B,{C,A})

g(D, {B,C,A}) = 8 + 15 = 23

(X=C)

g(D, {B,C,A}) = Cost(D,C) + g(C,{B,A})

g(D, {B,C,A}) = 9 + 18 = 27

Kemudian untuk tingkat teratasnya, kita dapat memperoleh penjabaran sebagai berikut

Untuk mengunjungi B,C,D dari node A

(X=B)

g(A, {B,C,D,A}) = Cost(A,B) + g(B,{C,D,A})

g(A, {B,C,D,A}) = 10 + 25 = 35

(X=C)

g(A, {B,C,D,A}) = Cost(A,C) + g(C,{B,D,A})

g(A, {B,C,D,A}) = 15 + 20 = 31

(X=D)

g(A, {B,C,D,A}) = Cost(A,D) + g(D,{B,C,A})

g(A, {B,C,D,A}) = 20 + 27 = 47

Maka dari semua penjabaran diatas, kita dapat mengambil kesimpulan bahwa rute optimal untuk mengunjungi seluruh *nodes* yang ada dan kembali ke *node* awal adalah rute dengan bobot 35 dan arah A-B-D-C-A.

IV. KESIMPULAN

Dengan pendekatan pemrograman dinamis terhadap persoalan TSP, kita dapat meminimalisir jumlah pengecekkan dan pengulangan yang terjadi sehingga kita dapat memperoleh rute yang paling efektif dengan lebih baik. Algoritma pemrograman dinamis dalam kasus ini menitikberatkan pemecahan masalah menjadi masalah-masalah yang lebih kecil, dan menyelesaikan persoalan-persoalan yang kecil itu untuk kemudian memakai solusinya sebagai bekal untuk menyelesaikan persoalan yang lebih besar.

V. PENUTUP

Segala puji syukur bagi Tuhan yang maha esa yang telah memberikan kemamuan pada penulis sehingga dapat menyelesaikan makalah ini dengan tepat waktu. Penulis juga mengucapkan terima kasih atas semua dukungan dan doa dari orang tua dan orang-orang terdekat. Tidak lupa juga penulis mengucapkan terima kasih kepada Bapak Dr. Ir. Rinaldi Munir, MT. yang telah membimbing penulis sepanjang semester genap di mata kuliah Strategi Algoritma. Akhir kata, penulis menyadari masih terdapat kekurangan dan kesalahan kata dalam makalah ini, penulis berharap makalah ini dapat bermanfaat bagi para pembacanya.

LINK YOUTUBE

https://youtu.be/kSupJGUlGJw

REFERENCES

[1] https://market.bisnis.com/read/20210421/192/1384313/kinerja-anteraja-melesat-pendapatan-assa-makin-tebal . Diakses pada 9 Mei 2021

[2] https://blog.routific.com/travelling-salesman-problem#:~:text=The%20Travelling%20Salesman%20Problem%20(TSP)%20is%20the%20challenge%20of%20finding,a%20list%20of%20specific%20destinations.&text=TSP%20has%20commanded%20so%20much,yet%20so%20difficult%20to%20solve . Diakses pada 9 Mei 2021

[3] https://money.kompas.com/read/2020/01/07/194200526/banjir-pengiriman-barang-jne-di-jabodetabek-sempat-turun-10-15-persen . Diakses pada 9 Mei 2021

[4] https://bournetocode.com/projects/AQA_A_Theory/pages/graph.html . Diakses pada 9 Mei 2021

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, 10 Mei 2021

Michael Philip G - 13519121