Penerapan Dynamic Programming Dalam Penentuan Rute Wisata di Kota Cirebon

Kharris Khisunica - 13522051
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
Email: kharriskhisunica07@gmail.com

Abstract*—Kota Cirebon adalah kota yang indah dengan berbagai macam wisata kebudayaan dan kulinernya yang menarik untuk dikunjungi. Dengan banyaknya destinasi dan waktu yang terbatas, wisatawan terkadang bingung untuk merencanakan rute perjalanan mereka dan memaksimalkan pengalaman berwisata. Dalam makalah ini, penulis akan membahas mengenai salah satu cara untuk menentukan rute perjalanan berdasarkan waktu tempuh dan waktu berkunjung di setiap destinasi menggunakan pendekatan *Dynamic Programming

***Kata Kunci*—algoritma Dynamic Programming, pariwisata, rute, Cirebon**

I. PENDAHULUAN

Cirebon adalah sebuah kota yang terletak di pesisir utara Jawa Barat, Indonesia, dan dikenal sebagai kota yang kaya akan sejarah dan budaya. Dengan posisinya yang berada di jalur antara kota-kota di bagian utara Jawa Barat, Cirebon menjadi salah satu destinasi wisata yang menarik untuk dikunjungi. Kota ini menawarkan beragam atraksi, mulai dari tempat-tempat bersejarah seperti Keraton Kasepuhan, dan Taman Gua Sunyaragi, hingga kelezatan kuliner khas seperti docang, mie koclok dan nasi jamblang.

Sumber: https://cirebon.tribunnews.com/2023/10/28/kota-cirebon-berpotensi-jadi-destinasi-wisata-mice-apa-itu-ini-penjelasannya, diakses pada 04/08/2024

Dalam penentuan rute perjalanan, wisatawan terkadang bingung untuk menentukan rute perjalanan yang efisien. Banyaknya variasi tempat yang dapat dikunjungi dalam waktu yang terbatas menjadi alasan mengapa perencanaan rute yang optimal diperlukan agar wisatawan bisa memaksimalkan pengalaman berwisata tanpa menghabiskan banyak waktu di perjalanan.

Dalam makalah ini, penulis akan mengerucutkan analisis untuk kasus berikut, dimana jenis destinasi yang dituju dengan urutannya adalah nasi jamblang, hotel, tempat wisata, mie koclok, lalu docang. Untuk beberapa jenis destinasi ada beberapa variasinya juga. Alasan pengerucutan analisis ke destinasi dengan urutan tersebut karena kompleksitas yang terlalu tinggi jika memasukkan seluruh destinasi yang mungkin. Kompleksitas yang ada muncul karena persoalan ini adalah salah satu bentuk dari *Travelling Salesman Problem* (TSP), yang kompleksitasnya akan bertambah seiring bertambahnya hal yang dianalisis. Penjelasan lebih detail akan disajikan pada bagian implementasi.

II. LANDASAN TEORI

A. Graf

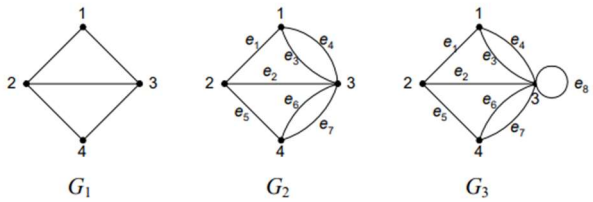
A.1 Definisi Graf

Graf adalah pasangan terurut dari pasangan G = (V,E) yang terdiri dari himpunan tidak-kosong V yang berisi *vertices*/simpul-simpul $\{v_1, v_2, \dots, v_n\}$, dan himpunan E berisi *edges*/sisi yang menghubungkan sepasang simpul $\{e_1, e_2, \dots, e_m\}$[1]. Hubungan antara sisi dan simpul dapat dinotasikan sebagai $e_p = (v_i, v_j)$ dimana $v_i, v_j \in V$ dan $e_p \in E$.

Graf digunakan untuk merepresentasikan objek-objek diskrit dan hubungan antara objek-objek tersebut.[2]

A.2 Jenis-Jenis Graf

A.2.1 Graf Sederhana dan Graf Tak-Sederhana



Gambar 2.1 a) Ilustrasi Graf Sederhana, b) Ilustrasi Graf tak-sederhana jenis Graf Ganda, c) Ilustrasi Graf tak-sederhana jenis Graf semu
Sumber:https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/19-Graf-Bagian1-2023.pdf, diakses pada 02/08/2024

Suatu graf dikatakan memiliki gelang jika dalam himpunan E

terdapat e_i sedemikian hingga $e_i = (v_j, v_j)$ untuk $e_i, e_j \in E$. Secara visual, suatu sisi disebut gelang jika dia berawal dan berakhir pada simpul yang sama. Pada gambar 1.1, contoh gelang adalah e_8 di G3 karena $e_8 = (3,3)$.

Suatu graf dikatakan memiliki sisi ganda jika dalam himpunan E terdapat $e_i = e_j = (v_p, v_q), i \neq j$ untuk $v_p, v_q \in V$ dan $e_i, e_j \in E$ atau bisa dikatakan elemen himpunan E tidak unik. Secara visual, suatu graf disebut memiliki sisi ganda jika ada dua sisi yang berawal dari simpul yang sama dan berakhir [ada simpul yang sama juga. Pada gambar 1.1, contoh sisi ganda adalah e_6 dan e_7 karena $e_6 = e_7 = (3,4)$ dan $6 \neq 7$.

Berdasarkan ada tidaknya gelang atau sisi ganda pada suatu graf, maka graf digolongkan menjadi dua jenis:

1. Graf sederhana (*simple graph*).

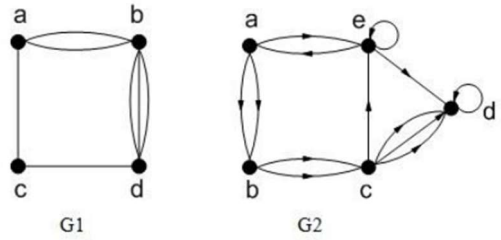
Suatu graf dikatakan sederhana jika graf tersebut tidak memiliki gelang dan tidak ada sisi ganda.

2. Graf tak-sederehana (*unsimple-graph*)

Suatu graf dikatakan tak-sederehana jika graf mengandung gelang dan/atau sisi ganda. Graf tak-sederhana bisa dibedakan menjadi dua jenis:

a. Graf ganda (*multi-graph*), yang memiliki sisi ganda.
b. Graf semu (*pseudo-graph*), yang memiliki sisi gelang.

A.2.2 Graf Tak-Berarah dan Graf Berarah



Gambar 2.2 a) Ilustrasi Graf Tak-Berarah, b) Ilustraksi Graf Berarah
Sumber:https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/19-Graf-Bagian1-2023.pdf, diakses pada 02/08/2024

Berdasarkan orientasi arah pada sisi, graf dapat dibedakan menjadi 2 jenis:

1. Graf tak-berarah (*undirected graph*)

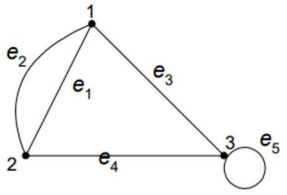
Suatu graf dikatakan tak berarah jika suatu graf tidak memiliki orientasi arah dan $(v_i, v_j) = (v_j, v_i)$ untuk $v_i, v_j \in V$. Secara visual, sisi dari graf tak-bearah hanya berupa garis yang menghubungkan dua simpul.

2. Graf berarah (*directed graph*)

Suatu graf dikatakan berarah jika setiap sisi pada graf tersebut memiliki orientasi arah. Hal ini berarti $(v_i, v_j) \neq (v_j, v_i)$ untuk $v_i, v_j \in V$. Karena (v_i, v_j) menunjukan sisi yang berasal dari v_i dan menunjuk ke v_j dan (v_j, v_i) menunjukan sisi yang berasal dari v_j dan menunjuk ke v_i.

A.3 Teriminologi Graf

Beberapa terminologi graf yang akan dipakai di dalam makalah ini:



Gambar 2.3 Contoh Graf
Sumber:https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/19-Graf-Bagian1-2023.pdf, diakses pada 02/08/2024

1. Ketetanggan (*Adjacent*)

Dua buah simpul dikatakan *bertetangga* bila keduanya terhubung secara langsung. Secara matematis, simpul $v_i, v_j \in V$ dikatakan bertetangga bila ditemukan $e_p \in E$ sedemikian hingga $e_p = (v_i, v_j)$. Secara visual, v_i dan v_j bertetangga bisa ada sisi yang menghubungkan kedua simpul. Contohnya pada gambar 1.3, simpul 1 dan 3 bertetangga karena ada simpul e_3 yang menghubungkan kedua simpul tersebut.

2. Bersisian (*Incidency*)

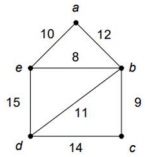
Sebuah simpul dikatakan *bersisian* dengan suatu sisi jika simpul tersebut adalah salah satu dari dua simpul pembentuk sisi tersebut. Secara matematis, simpul v_i dikatakan bersisian dengan e jika $e = (v_i, v)$ atau $e = (v, v_i)$ untuk suatu $v \in V$ dengan $v_i \in V$ dan $e \in E$. Secara visual, v_i bersisian dengan e jika v_i adalah salah satu titik pembentuk e_p untuk $v_i \in V$ dan $e \in E$. Contohnya pada gambar 1.3, simpul 1 dan sisi e_1 bersisian karena simpul 1 adalah salah satu pembentuk e_1.

3. Lintasan (*Path*)[5]

Lintasan pada graf G adalah barisan terbatas tak-kosong L = $v_0 e_1 v_1 e_2 v_2 \ldots e_k v_k$, dimana $v_i \in V, 0 \leq i \leq n$, dan $e_j \in E, 1 \leq j \leq m$. Dalam lintasan, simpul dan sisi disusun berselang-seling sedemikian hingga, untuk $1 \leq i \leq k$, $e_i = (v_{i-1}, v_i)$. Lintasan L bisa disebut sebagai lintasan dari v_0 ke v_k, dimana v_0 disebut simpul awal dan v_k disebut simpul tujuan dari L dan k adalah panjang dari lintasan L. Contoh lintasan pada gambar 1.3 adalah misal v_0 = simpul 1 dan v_n = simpul 2. Maka L yang terbentuk bisa L = $1e_1 2$ dengan panjang 1 atau $L = 1e_3 3e_4 2$ dengan panjang 2.

4. Graf Berbobot (*Weighted Graph*)

Graf berbobot adalah graf yang setiap sisinya diberi harga (bobot).



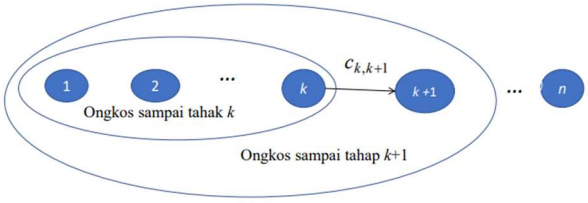
Gambar 2.4 Graf Berbobot
Sumber:https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/19-Graf-Bagian1-2023.pdf, diakses pada 02/08/2024

B. Dynamic Programming

Dynamic Programming atau Pemograman Dinamis adalah metode pemecahan masalah dengan cara menguraikan permasalahan menjadi sekumpulan tahapan yang lebih sederhana, sedemikian hingga solusi dari persoalan tersebut dapat dipandang sebagai serangkaian keputusan yang saling berkaitan.

Penguraian masalah dibuat dengan cara mendefinisikan fungsi nilai f_1, f_2, ..., f_n yang menerima masukan s sebagai *state* dari sistem pada saat i dari 1 sampai n

Pada pemograman dinamis, rangkaian keputusan yang optimal dibuat dengan menggunakan Prinsip Optimalisasi. Prinsip Optimalisasi sendiri adalah jika solusi total optimal, maka bagian solusi sampai tahap ke-k juga optimal. Prinsip ini berarti jika kita ingin mencari bobot optimal dari tahap k ke tahap $k+1$, kita bisa menggunakan hasil optimal dari tahap k tanpa harus kembali ke tahap awal lagi.



Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Program-Dinamis-2020-Bagian1.pdf, diakses pada 04/08/2024

Karakteristik dari persoalan pemograman dinamis:

- Persoalan dapat dibagi menjadi beberapa tahap, dimana setiap tahap hanya diambil satu keputusan
- Masing-masing tahap terdiri dari sejumlah status yang berhubungan dengan tahap tersebut. Status merupakan kemungkinan-kemungkinan masukan yang ada pada suatu tahap.
- Hasil dari keputusan yang diambil pada setiap tahap ditransformasikan dari status yang bersangkutan ke status berikutnya pada tahap berikutnya
- Cost pada suatu tahap meningkat secara teratur seiring bertambahnya jumlah tahapan, dengan cost pada suatu tahap bergantung dengan cost tahap-tahap sebelumnya dan cost tahap berikutnya.
- Adanya hubungan rekursif untuk mengidentifikasi keputusan terbaik untuk setiap kasus pada tahap k yang akan memberikan keputusan terbaik untuk setiap status pada tahap $k+1$.

Berdasarkan pendekatan penyelesaiannya, pemograman dinamis bisa dibagi menjadi dua jenis:

- Program dinamin maju (*forward* / *top-down*)

Pendekatan ini akan melakukan perhitungan dari tahap $1,2,\ldots,n-1,n$.

- Pemograman dinamis mundur (*backward* / *bottom-up*)

Pendekatan ini akan melakukan perhitungan dari tahap $n,n-1,\ldots,2,1$.

Langkah-langkah dalam penyelesaian persoalan dengan algoritma pemograman dinamis:

- Karakteristikkan struktur solusi optimal.
  Di tahap ini, tahap, variabel keputusan, dan status akan di definisikan dan akan dibentuk bentuk dari solusi optimal.

- Definisikan solusi optimal secara rekursif
  Di tahap ini, definisikan fungsi nilai rekursif yang akan menghasilkan solusi paling optimum dari persoalan yang ada.

- Hitung nilai solusi optimal secarar maju atau mundur
  Di tahap ini, akan dilakukan perhitungan dengan tabel untuk mencari fungsi nilai pada setiap tahap dan nilainya dengan pendekatan *forward* atau *backward*.

- Rekonstruksi solusi optimal
  Di tahap ini, cari dan temukan pilihan yang paling optimum di setiap tahap.

III. PENERAPAN

A. Pemetaan Masalah

Dalam kasus yang diangkat oleh penulis, terdapat 5 destinasi yang ingin dituju dengan masing-masing destinasi memiliki variannya tersendiri. Berikut adalah jenis destinasi beserta urutannya:

1) Warung makan nasi jamblang
2) Hotel
3) Wisata Cirebon
4) Warung makan mie koclok
5) Warung makan docang

Setiap urutan destinasi di atas akan menggambarkan tahap dari *Dynamic Programming*, dengan urutan ke-i menggambarkan pengambilan keputusan di tahap ke-$i+1$.

Titik awal dari rute yang akan dihitung adalah pintu keluar gerbang tol Plumbon 2. Pemilihan pintu keluar ini sebagai titik awal karena pintu keluar tol ini adalah pintu keluar terdekat ke kota Cirebon jika penulis berangkat dari Kota Bandung.

Titik akhir dari rute adalah Hotel yang sudah dipilih. Tetapi karena sudah tidak ada urgensi untuk sampai ke tempat tujuan dengan waktu minimal, maka titik akhir rute hanya akan berupa *goal state* dan cost ke goal state adalah 0. Cost bernilai 0 karena waktu dari state sebelum goal ke goal tidak lagi diperhitungkan.

Beberapa variasi dari setiap titik destinasi adalah sebagai berikut beserta perkiraan harga dan waktu berkunjung:

1) Warung makan nasi jamblang:

Nama Tempat	Harga rata-rata (Rp)	Waktu berkunjung (Menit)
Bu Nur	30000	45
Mang Dul	25000	45
Ibad Otoy	25000	45

2) Hotel

Nama Tempat	Harga (Rp)	Waktu berkunjung (Menit)
Neo Hotel	389000	0
Citra Dream	348000	0
Grand Dian	407000	0

Waktu berkunjung hotel = 0 karena total waktu di dalam hotel tidak perlu diperhitungkan.

3) Wisata Cirebon

Nama Tempat	Harga (Rp)	Waktu berkunjung (Menit)
Taman Gua Sunyaragi	15000	75
Kraton Kasepuhan	25000	90
Kampung Sabin	30000	90

4) Mie Koclok

Nama Tempat	Harga (Rp)	Waktu berkunjung (Menit)
Mas Edi	17000	30
Panjunan	20000	30

5) Docang

Nama Tempat	Harga (Rp)	Waktu berkunjung (Menit)
Ibu Wiwi	12000	30
Pak Kumis	9000	30

Berikut adalah waktu tempuh dari satu destinasi ke destinasi lainnya:

1) Waktu tempuh dari pintu keluar tol Plumbon 2 ke warung makan nasi jamblang

Asal \ Tujuan	Bu Nur	Mang Dul	Ibad Otoy
Pintu Keluar	21	21	16

2) Waktu tempuh dari warung makan nasi jamblang ke hotel

Asal \ Tujuan	Neo Hotel	Citra Dream	Grand Dian
Bu Nur	7	4	4
Mang Dul	9	2	7
Ibad Otoy	14	13	17

3) Waktu tempuh dari hotel ke tempat wisata Cirebon

Asal \ Tujuan	Gua Sunyaragi	Kraton Kasepuhan	Kampung Sabin
Neo Hotel	18	7	41
Citra Dream	10	9	37
Grand Dian	13	7	41

4) Waktu tempuh antar tempat wisata Cirebon

Asal \ Tujuan	Gua Sunyaragi	Kraton Kasepuhan	Kampung Sabin
Gua Sunyaragi	0	14	35
Kraton Kasepuhan	12	0	44
Kampung Sabin	30	43	0

5) Waktu tempuh dari tempat wisata Cirebon ke warung makan mie koclok

Asal \ Tujuan	Mas Edi	Panjunan
Gua Sunyaragi	11	15
Kraton Kasepuhan	4	6
Kampung Sabin	37	40

6) Waktu tempuh dari warung makan mie koclok ke warung makan docang

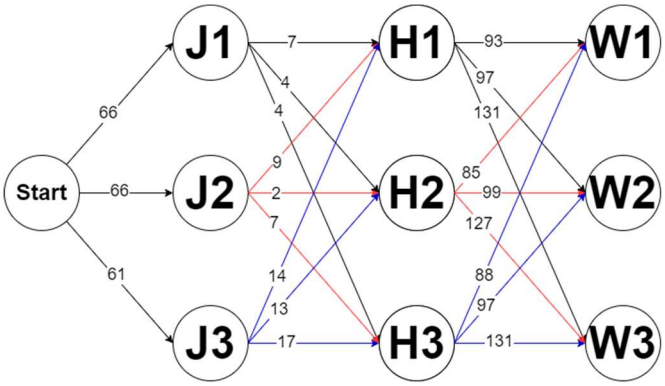
Asal \ Tujuan	Ibu Wiwi	Pak Kumis
Mas Edi	3	5
Panjunan	4	6

Beberapa kondisi yang harus dipenuhi:
- Setiap jenis destinasi harus dikunjungi tepat 1 kali
- Hanya boleh memilih satu jenis destinasi kecuali untuk tempat wisata Cirebon
- Untuk tempat wisata Cirebon, setiap destinasi harus dipilih tepat 1 kali
- Waktu yang diperlukan untuk mencapai seluruh destinasi minimal.

B. Pemetaan Permasalahan ke Dynamic Programming

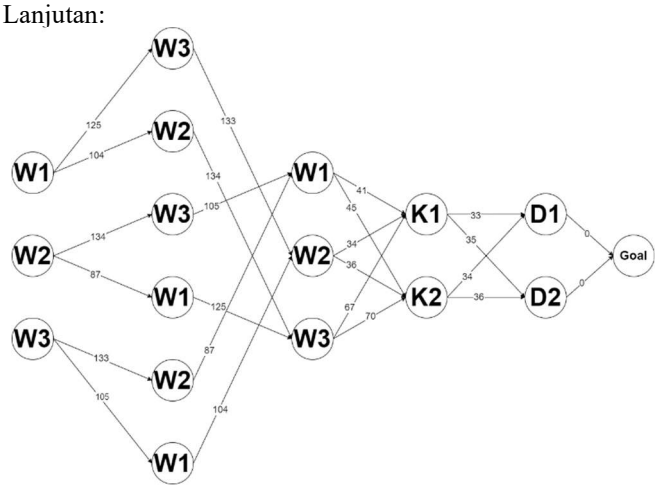
Bobot dari setiap sisi menggambarkan waktu tempuh antar tempat + waktu berkunjung di tempat tersebut.

Graf:



Gambar 3.1 Graf Multitahap Persoalan – Bagian 1
Sumber: Dokumentasi Pribadi

Lanjutan:



Gambar 3.2 Graf Multitahap Persoalan – Bagian 2
Sumber: Dokumentasi Pribadi

Keterangan:
J1 = Nasi Jamblang Ibu Nur
J2 = Nasi Jamblang Mang Dul
J3 = Nasi Jamblang Ibad Otoy

H1 = Neo Hotel
H2 = Hotel Citra Dream
H3 = Hotel Grand Dian

W1 = Taman Gua Sunyaragi
W2 = Kraton Kasepuhan
W3 = Kampung Sabin

K1 = Mie Koclok Mas Edi
K2 = Mie Koclok Panjunan

D1 = Docang Ibu Wiwi
D2 = Docang Pak Kumis

Solusi yang ingin dicari adalah rute yang menghasilkan waktu total minimum.

C. Penyelesaian Permasalahan dengan Dynamic Programming

C.1 Implementasi Matematis

Langkah-Langkah Pengembangan Algoritma *Dynamic Programming:*

1. Karakteristik Struktur Solusi Optimal

Misalkan x_1, x_2, ... x_8 adalah simpul yang akan dikunjungi pada setiap tahap, dengan tahap k mengunjungi simpul x_k untuk $k = \{1,2,\dots,8\}$. Pendekatan yang dipakai adalah *dynamic programming* maju. Maka rute yang dilalui adalah

$$x_1 \rightarrow x_2 \rightarrow \cdots \rightarrow x_8 \rightarrow x_9 = goal$$

Pada persoalan ini, Tahap (k) adalah proses memilih simpul tujuan, dimana terdapat 8 tahap. States (s) yang berhubungan dengan masing-masing tahap adalah simpul-simpul di dalam graf di atas.

2. Definisikan Hubungan Rekursif Solusi Optimal

Relasi rekurens yang menyatakan waktu tempuh terpendek pada setiap tahap:

$f_1(s) = C_{x1,s}$ {basis}
$f_k(s) = \min\{f_{k-1}(x_k) + C_{xk,s}\}$ {rekurens}
$k = 2,3,\dots,8$

Keterangan:
- x_k : peubah keputusan pada tahap k
- s : status pada setiap tahap
- $C_{xk,s}$: cost sisi dari x_k ke s
- $f_k(s)$: nilai minimum dari $f_k(x_k, s)$
- $f_{k-1}(x_k)$: nilai minimum tahap sebelumnya dari x_k ke s

3. Hitung Nilai Solusi Optimal

3.A Nilai Solusi Teoritis
Tahap 1:

s	Solusi Optimum	
	$f_1(s)$	x_1^*
J1	66	Start
J2	66	Start
J3	61	Start

Tahap 2:

x_2 / s	$f_1(x_2) + c_{x2,s}$			Solusi Optimum	
	J1	J2	J3	$f_2(s)$	x_2^*
H1	73	75	75	73	J1
H2	70	68	74	68	J2
H3	70	73	78	70	J1

Tahap 3:

x_3 / s	$f_2(x_3) + c_{x3,s}$			Solusi Optimum	
	H1	H2	H3	$f_3(s)$	x_3^*

W1	166	153	158	153	H2
W2	170	167	167	167	H2/H3
W3	204	195	201	195	H2

Tahap 4:

x_4 / s	$f_3(x_4) + c_{x4,s}$			Solusi Optimum	
	W1	W2	W3	$f_4(s)$	x_4^*
W3	278	-	-	278	W1
W2	257	-	-	257	W1
W3’	-	301	-	301	W2
W1	-	254	-	254	W2
W2’	-	-	328	328	W3
W1’	-	-	300	300	W3

Tahap 5:

x_5 / s	$f_4(x_5) + c_{x5,s}$						Solusi Optimum	
	W3	W2	W3’	W1	W2’	W1'	$f_5(s)$	x_5^*
W1			406		415		406	W3’
W2	411					404	404	W1’
W3		391		379			379	W1

Tahap 6:

x_6 / s	$f_5(x_6) + c_{x6,s}$			Solusi Optimum	
	W1	W2	W3	$f_6(s)$	x_6^*
K1	447	438	446	438	W2
K2	451	440	449	440	W2

Tahap 7:

x_7 / s	$f_6(x_7) + c_{x7,s}$		Solusi Optimum	
	K1	K2	$f_7(s)$	x_7^*
D1	471	474	471	K1
D2	473	476	473	K1

Tahap 8:

x_8 / s	$f_7(x_8) + c_{x8,s}$		Solusi Optimum	
	D1	D2	$f_8(s)$	x_8^*
Goal	471	473	471	D1

4. Rekonstruksi Solusi Optimal

Solusi optimal bisa didapatkan dari membaca tabel-tabel di atas:

Goal ← D1 ← K1 ← W2 ← W1’ ← W3 ←H2 ← J2 ← Start

Maka rute yang terbentuk adalah

Start → Warung nasi jamblang Mang Dul → Hotel Citra Dream → Kampung Sabin → Taman Gua Sunyaragi → Kraton Kasepuhan → Warung mie koclok Mas Edi → Warung docang Ibu Wiwi → Goal.
Dengan total waktu adalah 471 menit, atau 7 jam 51 menit.

Dari rute, juga bisa didapatkan bahwa total biaya yang harus dikeluarkan adalah Rp. 472.000 dengan catatan tidak ada tambahan biaya lain selain dari biaya yang sudah tertera di tabel di atas.

C.2 Implementasi Program

Source code akan dibuat dalam bahasa pemograman Java, dan menerapkan *Dynamic Programming*. Program akan menerima dua file .txt. destionation.txt berisi destinasi yang akan dituju beserta harganya (Gambar 3.3) dan adjacent.txt berisi simpul-simpul graf berarah dengan bobot nya (bobot = waktu perjalanan + waktu tinggal) (Gambar 3.4)

```
J1 Nasi_Jamblang_Ibu_Nur 30000
J2 Nasi_Jamblang_Mang_Dul 25000
J3 Nasi_Jamblang_Ibad_Otoy 25000
H1 Neo_HoteL 389000
H2 Hotel_Citra_Dream 348000
H3 Hotel_Grand_Dian 407000
W1 Taman_Gua_Sunyaragi 15000
W2 Kraton_Kasepuhan 25000
W3 Kampung_Sabin 30000
K1 Mie_Koclok_Mas_Edi 17000
K2 Mie_Koclok_Panjunan 20000
D1 Docang_Ibu_Wiwi 12000
D2 Docang Pak Kumis 9000
```

Gambar 3.3 destiantion.txt
Sumber: repositori pribadi

```
Start J1 66
Start J2 66
Start J3 61
J1 H1 7
J1 H2 4
J1 H3 4
J2 H1 9
J2 H2 2
J2 H3 7
J3 H1 14
J3 H2 13
J3 H3 17
H1 W1 93
H1 W2 97
H1 W3 131
H2 W1 85
H2 W2 99
H2 W3 127
H3 W1 88
H3 W2 97
H3 W3 131
W1 W12 104
W1 W13 125
W2 W21 87
W2 W23 134
W3 W31 105
W3 W32 133
W13 W132 133
W12 W123 134
W23 W231 105
W21 W213 125
W32 W321 87
W31 W312 104
W231 K1 41
W321 K1 41
W231 K2 45
W321 K2 45
W132 K1 34
W312 K1 34
W132 K2 36
W312 K2 36
W123 K1 67
W213 K1 67
W123 K2 70
W213 K2 70
K1 D1 33
K1 D2 35
K2 D1 34
K2 D2 36
D1 Goal 0
D2 Goal 0
```

Gambar 3.4 adjacent.txt
Sumber: Repositori pribadi

Dengan menggunakan dua file di atas, maka bisa didapatkan data Node Destinasi dan Node tetangga.

Source code lengkapnya dapat diakses di https://github.com/Kharris-Khisunica/MakalahStima_13522051.git. Berikut adalah snippet untuk algoritma *dynamic programming* untuk menyelesaikan permasalahan.

```
static int findOptimalRoute(String current) {
    if (dp.containsKey(current)) {
        return dp.get(current);
    }

    if (current.equals(anObject:"Goal")) {
        dp.put(current, value:0);
        return 0;
    }

    int minTime = Integer.MAX_VALUE;
    Node currentNode = graph.get(current);
    String bestNextNode = null;

    for (Map.Entry<Node, Integer> neighborEntry : currentNode.neighbors.entrySet()) {
        Node neighbor = neighborEntry.getKey();
        int travelTime = neighborEntry.getValue();
        int newTime = travelTime + findOptimalRoute(neighbor.name);

        if (newTime < minTime) {
            minTime = newTime;
            bestNextNode = neighbor.name;  // Track the node that leads to the optimal path
        }
    }

    if (bestNextNode != null) {
        previous.put(current, bestNextNode);  // Store the best route leading from current to next node
    }

    dp.put(current, minTime);
    return minTime;
}
```

Gambar 3.5: Snippet Algortima Dynamic Programming
Sumber: Repositori Pribadi

Langkah:

- Jika *current* = "Goal" maka keluarkan hasil dan return waktu/cost = 0.
- Untuk setiap state, akan dicari simpul dengan cost total termurah. Jika ditemukan, maka simpul tersebut akan menjadi bestNextNode lalu disimpan di Map *previous* untuk pengambilan rute. Simpan cost total termurah untuk mencapai *current* lalu return cost total termurah.

Untuk kasus permasalahan yang diangkat dalam makalah ini, output dari program ini adalah:

```
PS C:\Users\Asus\Downloads\Proyek Liburan\MakalahStima_13522051\src> java TravelRo
uteDP.java
Minimum total travel and stay time: 471 minutes.
Optimal Route:
Start -> Nasi Jamblang Mang Dul -> Hotel Citra Dream -> Kampung Sabin -> Taman Gua
 Sunyaragi -> Kraton Kasepuhan -> Mie Koclok Mas Edi -> Docang Ibu Wiwi -> Goal
Total Price: Rp.472000
```

Gambar 3.6: Sinppet Luaran Hasil
Sumber: Repositori Pribadi

Perhatikan bahwa hasil yang didapat dari program sama dengan hasil yang didapat dari perhitungan matematis. Sehingga bisa disimpulkan bahwa perhitungan yang ada sudah benar dan didapati bahwa rute dengan total waktu minimal adalah

Start → Warung nasi jamblang Mang Dul → Hotel Citra Dream → Kampung Sabin → Taman Gua Sunyaragi → Kraton Kasepuhan → Warung mie koclok Mas Edi → Warung docang Ibu Wiwi → Goal.

Dengan total waktu 471 Menit dan total harga Rp. 472.000

IV. KESIMPULAN

Dari makalah ini bisa disimpulkan bahwa dengan pemilihan destinasi dan dengan urutan yang ada, bisa didapatkan rute yang paling optimal secara waktu adalah Start → Warung nasi jamblang Mang Dul → Hotel Citra Dream → Taman Gua Sunyaragi → Kampung Sabin → Kraton Kasepuhan → Warung mie koclok Mas Edi → Warung docang Ibu Wiwi → Hotel. Rute yang dihasilkan ini memiliki cost total 471 menit atau 7 jam 51 menit dan memakan biaya sebesar Rp. 472.000. Walaupun dalam makalah ini, bobot antar simpul hanya mempertimbangkan waktu, biaya yang dikeluarkan juga cukup mendekati angka minimal. Penerapan algoritma ini dalam penentuan rute dalam menentukan wisata di Cirebon bisa memberi pemahaman lebih baik dalam cara menentukan rute yang paling dekat dan membantu untuk memaksimalkan jumlah tempat wisata yang dikunjungi dengan waktu minimal.

V. SARAN

Saran untuk peneliti selanjutnya adalah untuk mencoba algoritma-algoritma lain yang lebih efisien untuk membentuk rute. Selain itu, peneliti selanjutnya bisa mengintegrasikan persoalan di makalah ini dengan Google Maps API sehingga pemilihan tempat bisa lebih beragam, dan tidak hanya terbatas di daerah Cirebon dan di tempat wisata tertentu. Peneliti berikutnya bisa juga menambahkan parameter lain sebagai bobot dari graf tahap yang ada, yang mungkin bisa menghasilkan hasil yang berbeda bergantung pada kebutuhan peneliti.

VI. UCAPAN TERIMA KASIH

Pertama-tama, penulis mengucapkan puji dan Syukur kepada Tuhan Yang Maha Esa karena dengan pertolonganNya, Makalah berjudul "Penerapan *Dynamic Programming* Dlama Penentuan Rute Wisata di Kota Cirebon" dapat terselesaikan dengan baik. Penulis juga ingin mengucapkan terimakasih kepada Pak Rinaldi dan tim dosen lainnya yang telah mengajar mata kuliah IF2211 Strategi Algoritma dan telah membimbing penulis. Selain itu, penulis juga ingin berterimakasih kepada teman yang sudah belajar bersama dan membimbing penulis. Kemudian, penulis juga ingin berterimakasih kepada keluarga penulis yang selalu mendukung penulis sehingga makalah ini bisa diselesaikan dengan baik. Semoga makalah ini bisa membawa manfaat bagi banyak orang.

REFERENSI

[1] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/ diakses pada 02/08/2024
[2] https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/ diakses pada 02/08/2024
[3] https://travel.kompas.com/read/2023/12/13/175000627/10-wisata-cirebon-yang-hits-buat-liburan-tahun-baru-2024 diakses pada 02/08/2024
[4] https://travel.kompas.com/read/2023/12/13/175000627/10-wisata-cirebon-yang-hits-buat-liburan-tahun-baru-2024-?page=all diakses pada 02/08/2024
[5] https://github.com/Kharris-Khisunica/MakalahStima_13522051.git

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, 04 Agustus 2024

Kharris Khisunica, 13522051