

TUGAS KECIL 3 (BONUS)
IF2211 STRATEGI ALGORITMA
SEMESTER II TAHUN 2023/2024

**“Penyelesaian *Travelling Salesman Problem*
dengan Algoritma *Dynamic Programming*”**



OLEH:

Ahmad Hasan Albana 13522041

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2024

KATA PENGANTAR

Puji syukur kita panjatkan ke hadirat Allah SWT, karena atas rahmat dan karunia-Nya, kami dapat menyelesaikan makalah ini dengan judul "Penyelesaian Travelling Salesman Problem dengan Algoritma Dynamic Programming". Makalah ini disusun sebagai salah satu tugas mata kuliah Strategi Algoritma, dengan fokus pada penerapan konsep Algoritma Dynamic Programming.

Penyusunan makalah ini tidak lepas dari bantuan berbagai pihak. Kami mengucapkan terima kasih kepada dosen mata kuliah Strategi Algoritma yang telah memberikan bimbingan dan panduan selama proses pembuatan makalah ini.

Semoga makalah ini dapat memberikan kontribusi positif dalam pemahaman dan pengembangan suatu aplikasi dari konsep Algoritma Dynamic Programming. Akhir kata, kami menyampaikan permohonan maaf atas segala keterbatasan dalam makalah ini, dan kami menerima dengan terbuka segala kritik dan saran yang bersifat membangun.

Bandung, 15 Mei 2024,

Ahmad Hasan Albana

(13522041)

DAFTAR ISI

KATA PENGANTAR.....	1
DAFTAR ISI.....	2
BAB I.....	3
1.1. Abstraksi.....	3
BAB II.....	4
2.1. Algoritma Dynamic Programming.....	4
BAB III.....	5
3.1 Implementasi Kode.....	5
BAB IV.....	7
4.1 Hasil Pengujian.....	7
4.2 Pembahasan Hasil Pengujian.....	7
DAFTAR PUSTAKA.....	8
LAMPIRAN.....	9
Repository.....	9

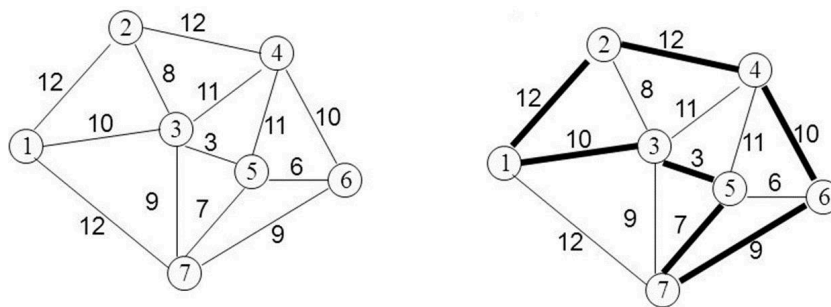
BAB I

DESKRIPSI MASALAH

1.1. Abstraksi

Travelling salesman problem atau TSP adalah tantangan untuk menemukan rute terpendek dan efisien bagi seseorang sesuai daftar tujuan tertentu. TSP pertama kali diperkenalkan pada tahun 1930-an oleh Karl Menger, seorang ahli matematika dan ekonomi. Menger menyebutnya sebagai “Messenger Problem” yakni masalah yang dihadapi oleh pengirim surat dan banyak pengelana.

TSP berusaha menjawab pertanyaan tentang rute terpendek mana yang harus dilalui oleh sales tersebut sehingga dia hanya mengunjungi setiap lokasi sekali saja sebelum kembali ke titik awal.



Gambar 1.1 Graf Travelling Salesman Problem
(Sumber: <https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Program-Dinamis-2020-Bagian2.pdf>)

Pada Tugas Kecil 3 (Bonus) ini, penulis merancang program untuk menerapkan algoritma *Dynamic Programming* yang telah dipelajari di kelas untuk mencari solusi optimum dari permasalahan tersebut.

Program ini dibuat menggunakan bahasa Ruby dan dijalankan melalui CLI, serta menerima input berupa path menuju file yang memuat Matriks Ketetanggaan dari graf TSP.

Setelah menerima input, program akan melakukan parsing file menjadi matriks dan kalkulasi menggunakan algoritma *dynamic programming*, lalu menampilkan bobot atau jarak minimum sekaligus jalur-jalur yang dilalui.

BAB II

IMPLEMENTASI ALGORITMA

2.1. Algoritma *Dynamic Programming*

Algoritma *dynamic programming* adalah salah satu pendekatan algoritma untuk menyelesaikan berbagai persoalan komputasi. Ide utama dari algoritma ini yaitu memecahkan masalah dengan cara menguraikan solusi menjadi sekumpulan tahapan (stage).

Pada program dinamis, rangkaian keputusan yang optimal dibuat dengan menggunakan Prinsip Optimalitas. Prinsip optimalitas berarti bahwa jika kita bekerja dari tahap k ke tahap $k + 1$, kita dapat menggunakan hasil optimal dari tahap k tanpa harus kembali ke tahap awal.

Pada program ini, algoritma *dynamic programming* untuk penyelesaian *Travelling Salesman Problem* diimplementasikan sebagai berikut.

Misalkan $G = (V, E)$ adalah graf lengkap berarah dengan sisi-sisi yang diberi harga $C_{ij} > 0$ dan $V = n$ dengan $n > 1$. Setiap simpul diberi nomor $[1, n]$. Asumsikan perjalanan (tur) dimulai dan berakhir pada simpul 1, maka lintasan dari simpul k ke simpul 1 tersebut melalui setiap simpul di dalam $V - \{1, k\}$ tepat hanya sekali.

Prinsip Optimalitas: jika tur tersebut optimal maka lintasan dari simpul k ke simpul 1 juga menjadi lintasan k ke 1 terpendek yang melalui simpul-simpul di dalam $V - \{1, k\}$.

Misalkan $f(i, S)$ adalah bobot lintasan terpendek yang berawal dari simpul i , yang melalui semua simpul di dalam S dan berakhir pada simpul 1, maka akan didapatkan persamaan tersebut sebagai berikut.

$$f(i, \emptyset) = C_{i1}, 2 \leq i \leq n \text{ (basis)}$$
$$f(i, S) = \min_{(j \in S)} \{C_{ij} + f(j, S - \{j\})\} \text{ (rekurens),}$$

dengan syarat jika i tidak bertetangga dengan j , maka bobot tidak akan dikalkulasi dan bukan merupakan kandidat minimum.

BAB III

IMPLEMENTASI KODE PROGRAM

3.1 Implementasi Kode

```
def rekursif(graph, i, list)
  if (list == [])
    if graph[i-1][0] == 0 # Case: node is not accessible
      return [Float::INFINITY, [1]]
    else
      return [graph[i-1][0], [1]]
    end
  else
    min_dist = Float::INFINITY

    for x in list
      if graph[i-1][x-1] == 0 # Case: node is not
accessible
        next
      end

      rek = rekursif(graph, x, list - [x])
      temp = graph[i-1][x-1] + rek[0]
      if (temp < min_dist)
        min_dist = temp
        path = [x] + rek[1]
      end
    end
    return [min_dist, path]
  end
end

def tsp(graph)
  num_cities = graph.size
  result = rekursif(graph, 1, (2..num_cities).to_a)
  puts "Bobot minimum: #{result[0]}"
  puts "Path yang dilalui: #{[1] + result[1]}"
end

# Main Program
## Get filepath
```

```

puts "Masukkan nama file testing: "
path = "../test/" + gets.chomp

## Read file
fileobject = File.new(path, mode="r")
lines = fileobject.readlines

## Convert into matrix
graph = Array.new
for x in lines
  graph = graph.push(x.split(' ').map{|a| Integer(a)})
end
fileobject.close()

## Calling function
tsp(graph)

```

Program pada awalnya akan meminta input berupa file yang berisikan matriks ketetanggaan graf TSP dengan konvensi bahwa jika nilai matriks adalah 0, maka simpul tersebut tidak bertetanggaan, lalu mengkonversinya menjadi suatu variabel matriks yang dapat diolah program. Setelah itu, program akan memanggil fungsi tsp dan memberikan matriks ketetanggaan yang telah dibuat sebelumnya.

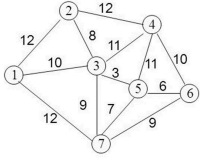
Fungsi tsp adalah fungsi yang berguna untuk melakukan inisiasi proses rekursif dan juga menampilkan solusi yang telah didapatkan. Sedangkan fungsi rekursif adalah fungsi yang mengimplementasikan rumus yang telah kita dapatkan pada Bab II, dengan pengecekan tambahan saat suatu simpul tidak saling bertetangga yang ditangani dengan membuat nilai simpul tersebut menjadi INFINITY sehingga tak mungkin terpilih sebagai simpul yang dipilih selanjutnya.

BAB IV

HASIL PENGUJIAN DAN ANALISIS

4.1 Hasil Pengujian

Berikut adalah hasil pengujian dengan menggunakan data uji yang berada pada folder *test*.

DYNAMIC PROGRAMMING	
<i>test1.txt</i>	
$\begin{bmatrix} 0 & 10 & 15 & 20 \\ 5 & 0 & 9 & 10 \\ 6 & 13 & 0 & 12 \\ 8 & 8 & 9 & 0 \end{bmatrix}$	<pre>Masukkan nama file testing: test1.txt Bobot minimum: 35 Path yang dilalui: [1, 2, 4, 3, 1]</pre>
<i>test2.txt</i>	
$\begin{bmatrix} \infty & 20 & 30 & 10 & 11 \\ 15 & \infty & 16 & 4 & 2 \\ 3 & 5 & \infty & 2 & 4 \\ 19 & 6 & 18 & \infty & 3 \\ 16 & 4 & 7 & 16 & \infty \end{bmatrix}$	<pre>Masukkan nama file testing: test2.txt Bobot minimum: 28 Path yang dilalui: [1, 4, 2, 5, 3, 1]</pre>
<i>test3.txt</i>	
	<pre>Masukkan nama file testing: test3.txt Bobot minimum: 63 Path yang dilalui: [1, 2, 4, 6, 7, 5, 3, 1]</pre>

4.2 Pembahasan Hasil Pengujian

Berdasarkan hasil pengujian diatas, terlihat bahwa hasil implementasi algoritma *dynamic programming* telah sesuai dengan yang diharapkan. Terbukti dengan sesuainya seluruh *testcase* yang diberikan dengan hasil yang ditampilkan oleh program.

DAFTAR PUSTAKA

1. Munir, Rinaldi. “Program Dinamis (Dynamic Programming) Bagian 1”
(online).(<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Program-Dinamis-2020-Bagian1.pdf>, diakses pada 15 Mei 2024).
2. Munir, Rinaldi. “Program Dinamis (Dynamic Programming) Bagian 2”
(online).(<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Program-Dinamis-2020-Bagian2.pdf>, diakses pada 15 Mei 2024).
3. Varta, Bhumi. “Traveling Salesman Problem: Definisi dan Implementasi”
(online).(<https://bvarta.com/id/traveling-salesman-problem-definisi-dan-implementasi/>, diakses pada 15 Mei 2024).

LAMPIRAN

Repository

Link Repository dari Tugas Kecil 03 (Bonus) IF2211 Strategi Algoritma adalah sebagai berikut.

https://github.com/Bana-man/TSP_Solver_13522041.git