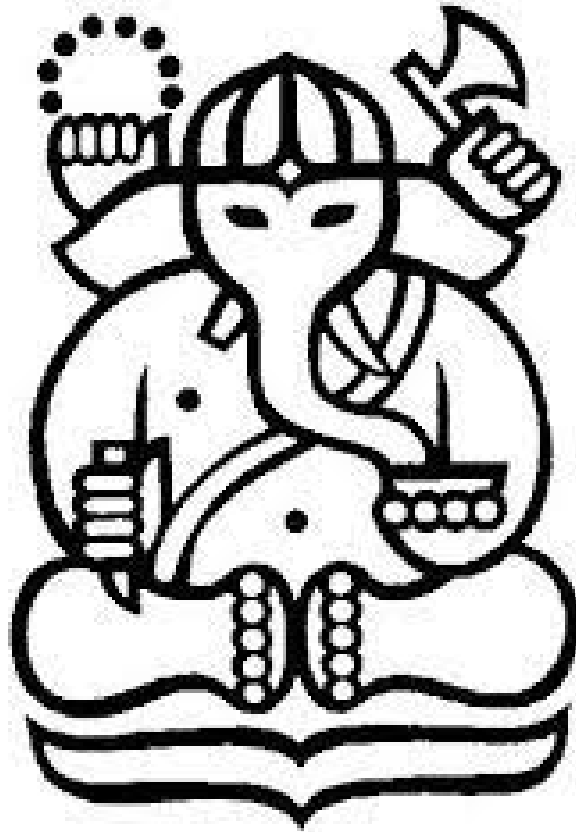


Tugas Tantangan Strategi Algoritma IF2211
Pemecahan Masalah Travelling Salesman Problem
dengan Algoritma Dynamic Programming



oleh:

Julian Caleb Simandjuntak - 13522099

Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

Daftar Isi

Daftar Isi	2
Deskripsi Masalah	3
Landasan Teori	4
Implementasi Source Code	5
Testing	10
Kesimpulan	13
Pranala	14
Daftar Pustaka	15

Deskripsi Masalah

Travelling Salesman Problem atau TSP adalah sebuah permasalahan (komputasi) untuk mencari jalur terpendek yang melewati sejumlah titik tertentu dengan syarat bahwa setiap titik harus dikunjungi tepat sekali dan rute harus kembali ke titik awal. TSP merupakan sebuah permasalahan yang dapat digambarkan sebagai sebuah graf $G(V,E)$ dengan vertices (V) merupakan node yang perlu dikunjungi dan edges (E) merupakan jalur dari suatu node ke node lain.

Tugas tantangan Stima meminta mahasiswa membuat program untuk menyelesaikan permasalahan TSP yang digambarkan sebagai graf berbobot (dalam bentuk matrix ketetanggaan) yang mengembalikan nilai terkecil beserta pathnya, menggunakan bahasa pemrograman yang belum pernah dipakai di kuliah, meliputi Rust, Ruby, REPL, atau Swift. Masukkan yang diterima berupa file txt.

Landasan Teori

Program Dinamis atau Dynamic Programming merupakan metode pemecahan masalah dengan cara menguraikan solusi menjadi sekumpulan tahapan sedemikian sehingga solusi persoalan dapat dipandang sebagai serangkaian keputusan yang saling berkaitan. Dynamic Programming digunakan untuk menyelesaikan persoalan-persoalan optimasi dengan menggunakan prinsip optimalitas, yaitu jika solusi total optimal, bagian solusi sampai tahap ke-k juga optimal.

Penggunaan Dynamic Programming dalam penyelesaian masalah TSP dimulai dengan mendefinisikan graf TSP itu sendiri, di mana masalah ini digambarkan dengan matrix ketetanggaan. Dibuat asumsi bahwa rute dimulai dan berakhir pada node 1. Dibuat pula sebuah prinsip optimalitas di mana rute yang optimal adalah jika rute tersebut melalui setiap node di dalam graf TSP tepat hanya sekali (kecuali node 1 yang menjadi awal dan akhir), rute tersebut juga menjadi rute dengan cost terkecil.

Untuk mengimplementasikan algoritma, misal matrix dengan baris ke-i dan kolom ke-j ($matrix[i][j]$) berarti cost rute dari node i ke node j, sisaNode adalah seluruh vertices dalam graf kecuali node 1, dengan node adalah salah satu vertices dalam graf, maka dapat didefinisikan fungsi f:

$$\text{Basis: } f(\text{node}, \emptyset) = matrix[\text{node}][1]$$

$$\text{Rekurs: } f(\text{node}, \text{sisaNode}) = matrix[\text{node}][1]$$

Dynamic Programming dapat dilakukan dengan memulai rekurs dari $\text{node} = 1$ dan sisaNode adalah seluruh vertices dalam graf kecuali node 1.

Untuk pengambilan path, dimulai dari basis, meletakkan node 1 sebagai bagian akhir dari array, lalu berdasarkan cost paling kecil, diambil node yang dikunjungi dan masukkan ke bagian depan array.

Implementasi Source Code

Source code diimplementasikan dengan menggunakan bahasa pemrograman Ruby. Pada implementasinya, bobot rute node dengan dirinya sendiri ($\text{matrix}[i][j]$ dengan $i = j$) bernilai 0 dan bobot rute node yang tidak bertetangga dengan node lain bernilai 9999.

main.rb

```
1  # Import fungsi tsp dari tsp.rb
2  require_relative 'tsp'
3
4  # Judul
5  puts "=== TRAVELING SALESMAN PROBLEM USING DYNAMIC PROGRAMMING ==="
6
7  # Instansiasi matrix ketetanggaan
8  matrix = nil
9
10 # Loop hingga filename valid
11 loop do
12   # Meminta input filename
13   print "Input filename in test folder: "
14   filename = gets.chomp
15   filepath = "test/#{filename}.txt"
16   puts
17
18   # Pembacaan file
19   begin
20     # Asumsi isi file valid
21     # Baca setiap line
22     lines = File.readlines(filepath)
23
24     # Mengubah line menjadi komponen matrix
25     matrix = lines.map do |line|
26       # Split berdasarkan spasi dan ubah menjadi integer
27       line.split.map(&:to_i)
28     end
29
30     # Debug
31     puts "Isi matrix: #{matrix}"
32     puts
33     break
34
35     # Kalau file tidak ditemukan
36     rescue Errno::ENOENT
37       puts "File #{filepath} tidak ditemukan. Silahkan coba lagi."
38
39     # Kalau ada error
40     rescue => e
41       puts "Error: #{e.message}. Silahkan coba lagi."
42     end
43   end
44 end
```

```

45 # Algoritma TSP dengan menggunakan Dynamic Programming
46 # Asumsikan perjalanan dimulai dari dan berakhir pada simpul 1
47
48 # Matrix (matrix ketetanggaan) dibaca sebagai array of array dengan baris asal node dan kolom adalah node tujuan
49 # matrix[1][2] berarti dari node 1 ke node 2
50
51 # Matrix menggunakan asumsi nilai 9999 berarti tidak bertetangga
52
53 # Instansiasi variabel yang dibutuhkan, meliputi:
54 # - nodeAwal yaitu 1
55 # - sisaNode yaitu array node selain 1
56 # - path berupa empty array
57 matrixLength = matrix.length
58 nodeAwal = 1
59 sisaNode = (2..matrixLength).to_a
60 path = []
61
62 # Memanggil fungsi f dari tsp.rb
63 tspResult = f(nodeAwal, sisaNode, path, matrix)
64
65 # tspResult merupakan tuple [cost, path]
66 # Mencetak hasil
67 puts "Lowest cost: #{tspResult[0]}"
68 puts "Path: #{tspResult[1].inspect}"
69
70
71

```

tsp.rb

```
1 # Algoritma didefinisikan dengan menggunakan sebuah fungsi f yang menghitung jarak terpendek dengan
2 # basis jika semua node sudah dikunjungi, maka kembali ke node 1,
3 # Basis:
4 # f(nodeTerakhir, sisaNode) = jarak[nodeTerakhir, 1]
5 # dengan sisaNode adalah array kosong karena semua node telah dikunjungi
6 # Rekurens:
7 # f(nodeTerakhir, sisaNode) = jarak[nodeTerakhir, node] + f(node, sisaNodeTanpaNode)
8 # dengan node berlaku untuk setiap elemen dari sisaNode dan sisaNodeTanpaNode adalah sisaNode - node
9
10 # Sebagai tambahan, untuk menyimpan path dan memasukkan matrix, fungsi f dimodifikasi menjadi
11 # f(nodeTerakhir, sisaNode, nodePath, matrix) yang mengembalikan tuple [cost, path]
12 # Penjelasan lebih lanjut terdapat di dalam fungsi
13
14 # Fungsi f
15 def f(nodeTerakhir, sisaNode, nodePath, matrix)
16   # Debug
17   puts "Node saat ini: #{nodeTerakhir}"
18   puts "Sisa node: #{sisaNode}"
19   puts
20
21   # Basis
22   # Jika tidak ada node yang bisa dikunjungi, kembali ke node awal yaitu 1
23   if sisaNode.empty?
24     # Memasukkan path dengan nodeTerakhir -> 1
25     path = nodePath.dup
26     path.unshift(1)
27     path.unshift(nodeTerakhir)
28
29     # Mengembalikan jarak node terakhir ke 1 dan pathnya
30     return [matrix[nodeTerakhir - 1][1 - 1], path]
31   end
32 end
```



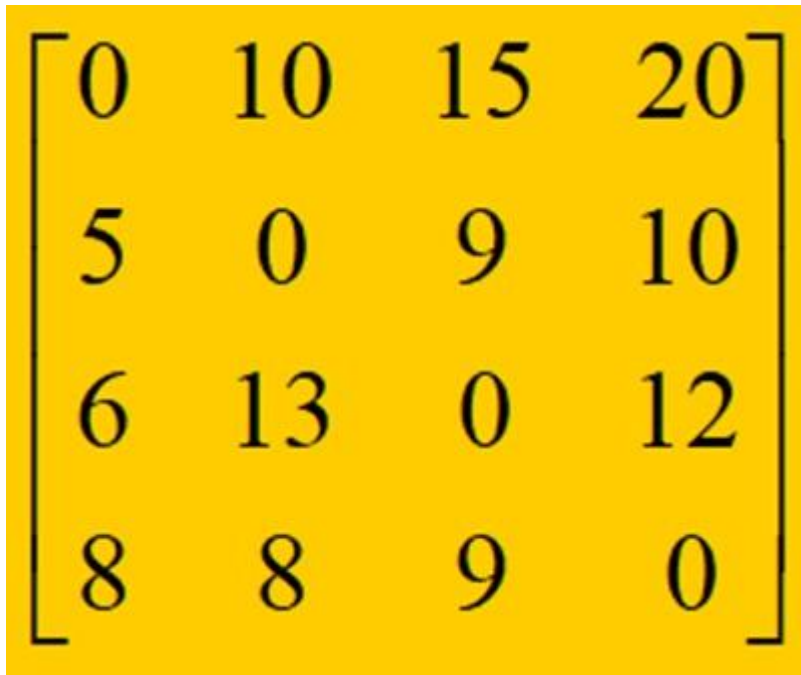
```

33     # Instansiasi variabel untuk menampung semua kemungkinan
34     tempCost = []
35     tempPath = []
36
37     # Rekurens
38     # Jika masih ada node yang bisa dikunjungi, untuk setiap node
39     sisaNode.each do |node|
40         # Debug
41         puts "Pergi ke node: #{node}"
42
43         # Menghapus node dari sisaNode
44         sisaNodeTanpaNode = sisaNode.dup
45         sisaNodeTanpaNode.delete(node)
46
47         # Melakukan rekurens
48         nextMove = f(node, sisaNodeTanpaNode, nodePath, matrix)
49
50         # Append cost dan path ke dalam variabel penampung
51         tempCost << (matrix[nodeTerakhir - 1][node - 1] + nextMove[0])
52         tempPath << nextMove[1]
53     end
54
55     # Mencari cost paling kecil
56     cost = tempCost.min
57
58     # Mencari path yang tepat berdasarkan cost yang dipilih
59     idx = tempCost.index(cost)
60     path = tempPath[idx]
61
62     # Masukkan node sekarang ke dalam path yang terpilih
63     path.unshift(nodeTerakhir)
64
65     # Mengembalikan [cost, path]
66     return [cost, path]
67 end
68

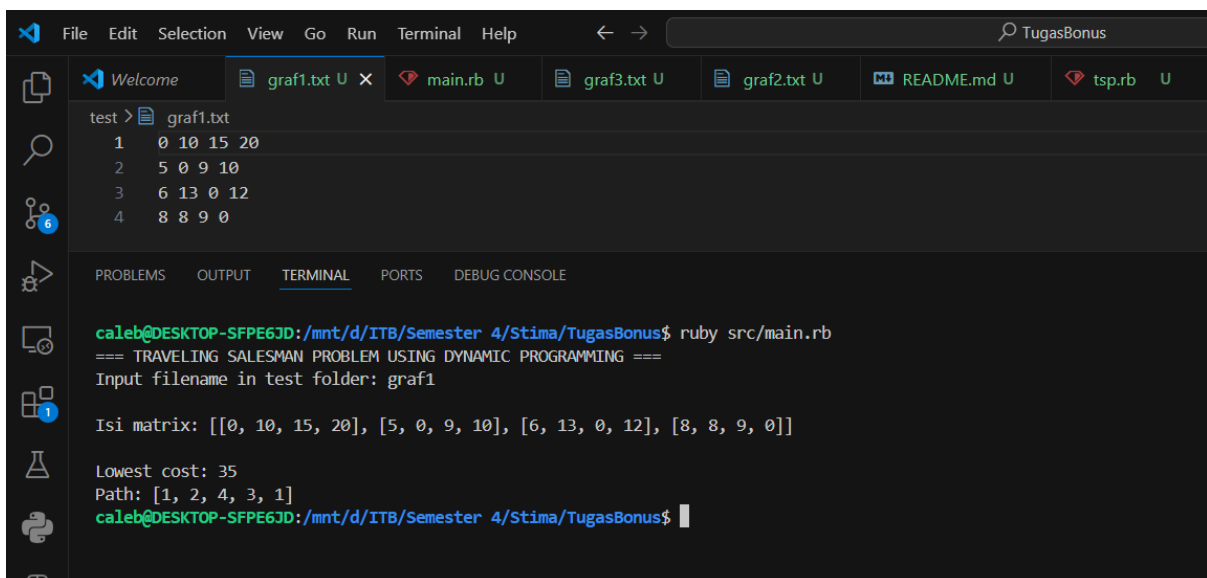
```

Testing

test 1



0	10	15	20
5	0	9	10
6	13	0	12
8	8	9	0



```
File Edit Selection View Go Run Terminal Help
Welcome | graf1.txt U x | main.rb U | graf3.txt U | graf2.txt U | README.md U | tsp.rb U
test > graf1.txt
1 0 10 15 20
2 5 0 9 10
3 6 13 0 12
4 8 8 9 0

PROBLEMS OUTPUT TERMINAL PORTS DEBUG CONSOLE

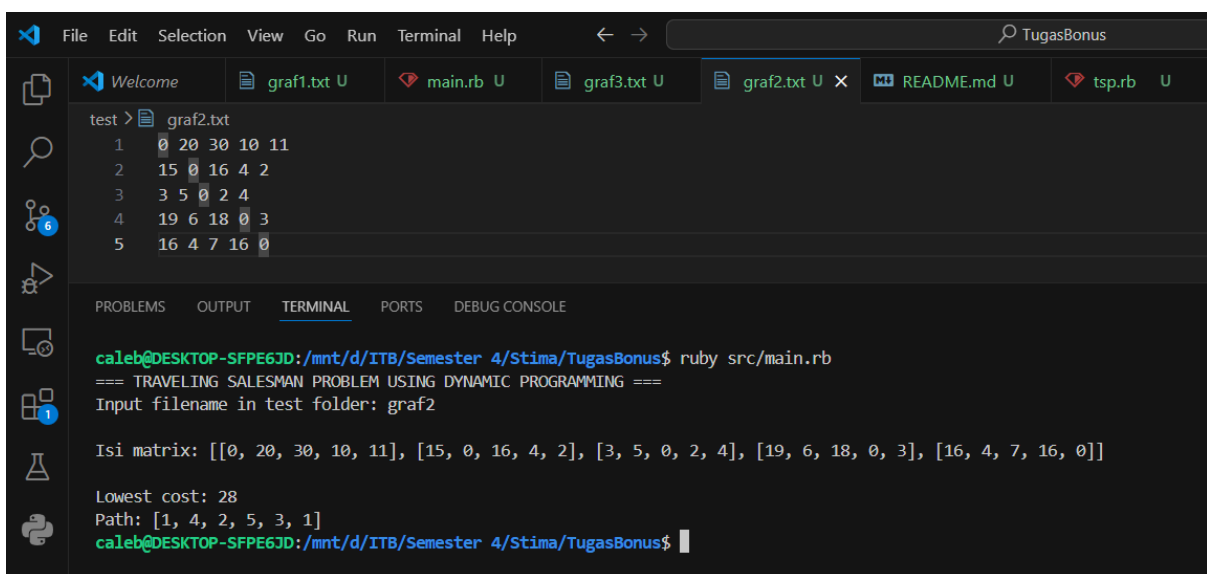
caleb@DESKTOP-SFPE6JD:/mnt/d/ITB/Semester 4/Stima/TugasBonus$ ruby src/main.rb
=== TRAVELING SALESMAN PROBLEM USING DYNAMIC PROGRAMMING ===
Input filename in test folder: graf1

Isi matrix: [[0, 10, 15, 20], [5, 0, 9, 10], [6, 13, 0, 12], [8, 8, 9, 0]]

Lowest cost: 35
Path: [1, 2, 4, 3, 1]
caleb@DESKTOP-SFPE6JD:/mnt/d/ITB/Semester 4/Stima/TugasBonus$
```

test 2

∞	20	30	10	11
15	∞	16	4	2
3	5	∞	2	4
19	6	18	∞	3
16	4	7	16	∞



```
File Edit Selection View Go Run Terminal Help
Welcome graf1.txt U main.rb U graf3.txt U graf2.txt U x README.md U tsp.rb U

test > graf2.txt
1 0 20 30 10 11
2 15 0 16 4 2
3 3 5 0 2 4
4 19 6 18 0 3
5 16 4 7 16 0

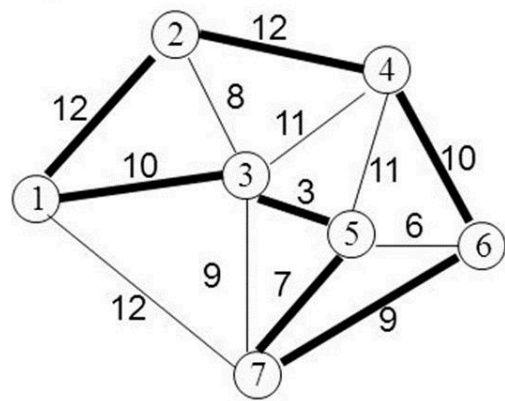
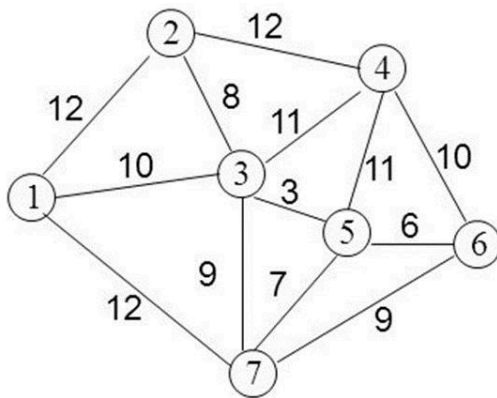
PROBLEMS OUTPUT TERMINAL PORTS DEBUG CONSOLE

caleb@DESKTOP-SFPE6JD:/mnt/d/ITB/Semester 4/Stima/TugasBonus$ ruby src/main.rb
=== TRAVELING SALESMAN PROBLEM USING DYNAMIC PROGRAMMING ===
Input filename in test folder: graf2

Isi matrix: [[0, 20, 30, 10, 11], [15, 0, 16, 4, 2], [3, 5, 0, 2, 4], [19, 6, 18, 0, 3], [16, 4, 7, 16, 0]]

Lowest cost: 28
Path: [1, 4, 2, 5, 3, 1]
caleb@DESKTOP-SFPE6JD:/mnt/d/ITB/Semester 4/Stima/TugasBonus$
```

test 3



```

File Edit Selection View Go Run Terminal Help
TugasBonus
Welcome graf1.txt U main.rb U graf3.txt U graf2.txt U README.md U tsp.rb U
test > graf3.txt
1 0 12 10 9999 9999 9999 12
2 12 0 8 12 9999 9999 9999
3 10 8 0 11 3 9999 9
4 9999 12 11 0 11 10 9999
5 9999 9999 3 11 0 6 7
6 9999 9999 9999 10 6 0 9

PROBLEMS OUTPUT TERMINAL PORTS DEBUG CONSOLE
caleb@DESKTOP-SFPE63D:/mnt/d/ITB/Semester 4/Stima/TugasBonus$ ruby src/main.rb
=== TRAVELING SALESMAN PROBLEM USING DYNAMIC PROGRAMMING ===
Input filename in test folder: graf3

Isi matrix: [[0, 12, 10, 9999, 9999, 9999, 12], [12, 0, 8, 12, 9999, 9999, 9999], [10, 8, 0, 11, 3, 9999, 9], [9999, 12, 11, 0, 11, 10, 9999], [9999, 9999, 3, 11, 0, 6, 7], [9999, 9999, 9999, 10, 6, 0, 9], [12, 9999, 9, 9999, 7, 0, 0]]

Lowest cost: 63
Path: [1, 2, 4, 6, 7, 5, 3, 1]
caleb@DESKTOP-SFPE63D:/mnt/d/ITB/Semester 4/Stima/TugasBonus$

```

Kesimpulan

Permasalahan TSP dapat diselesaikan dengan menggunakan Dynamic Programming memanfaatkan fungsi rekursif, mencari nilai minimal setiap tahapan sehingga menjadi optimal dan menjumlahkan tiap tahapan, menjamin solusi akhirnya optimal.

Pranala

Link Github:

https://github.com/Julian-Caleb/Tugas_Tantangan_Stima_13522099

Daftar Pustaka

<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Program-Dinamis-2020-Bagian1.pdf>

<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Program-Dinamis-2020-Bagian2.pdf>