

# **Laporan Tugas Tantangan IF2211 Strategi Algoritma**

**Semester II Tahun Akademik 2023/2024**

Penyelesaian Travel Salesman Problem dengan Dynamic Programming pada  
Bahasa Pemrograman Ruby



Disusun oleh:

Muhammad Gilang Ramadhan

13520137

K01

**Program Studi S1 Teknik Informatika**  
**Sekolah Teknik Elektro dan Informatika**  
**Institut Teknologi Bandung**  
**2024**

## ANALISIS DAN PENJELASAN ALGORITMA

Dalam pendekatan DP untuk TSP, kita menggunakan dua parameter utama untuk mendefinisikan state:

- **Mask:** Sebuah bitmask yang menunjukkan kota-kota yang sudah dikunjungi.
- **Pos:** Kota saat ini di mana penjual berada.

State ini dinyatakan sebagai  $dp[mask][pos]$ , yang merepresentasikan biaya minimum untuk mengunjungi semua kota yang ditandai oleh mask dan berakhir di kota pos.

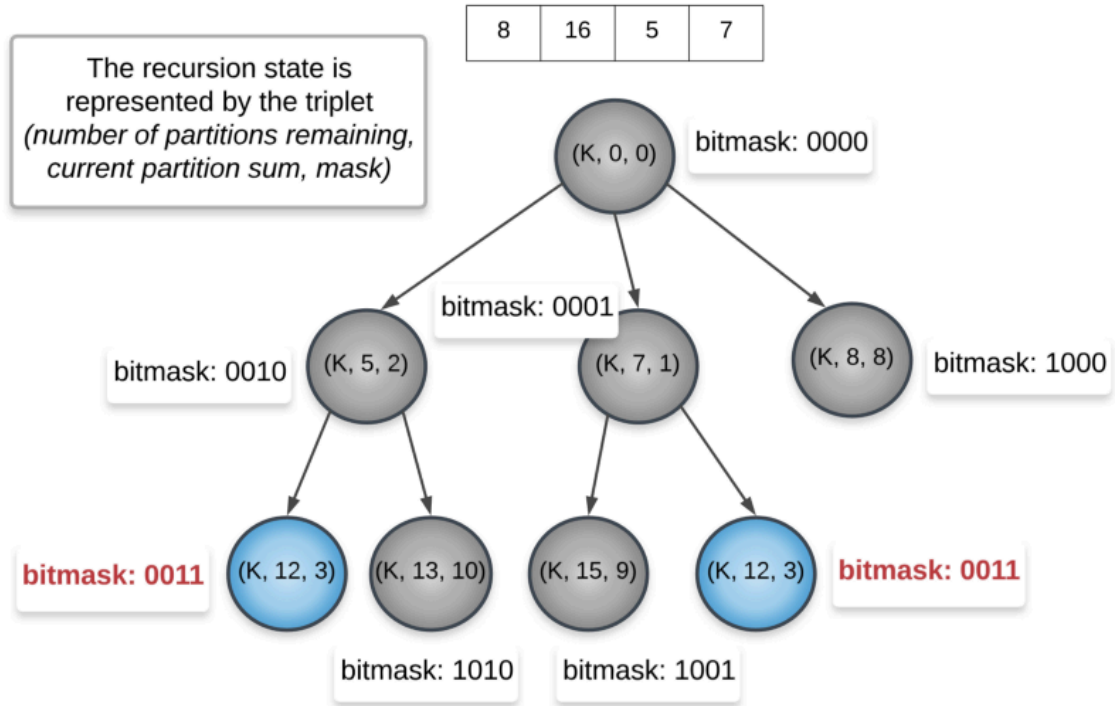
Inisialisasi dilakukan dengan mengatur nilai awal state:

```
dp[1 << start][start] = 0
```

Untuk menghitung nilai  $dp[mask][pos]$ , kita mempertimbangkan semua kota yang belum dikunjungi dan menghitung biaya perjalanan dari kota pos ke kota tersebut ditambah biaya perjalanan dari kota tersebut kembali ke kota asal.

```
dp[mask][pos] = min(dp[mask][pos], adj[pos][v] + dp[mask | (1 << v)][v])
```

Kita dapat melakukan iterasi melalui semua kombinasi bitmask dan semua posisi untuk memperbarui nilai dp sesuai dengan persamaan rekurens yang telah ditentukan.



Sumber: [https://cdn-media-1.freecodecamp.org/images/1\\*uc66c1lYHTXfbnMLca2bXA.png](https://cdn-media-1.freecodecamp.org/images/1*uc66c1lYHTXfbnMLca2bXA.png)

Untuk melacak jalur yang dilalui, kita menggunakan array `next_node` yang menyimpan kota berikutnya untuk setiap state:

```
next_node[mask][pos] = v
```

Ini memungkinkan kita untuk merekonstruksi jalur optimal setelah menyelesaikan perhitungan dp.

**Kompleksitas waktu dari algoritma ini adalah  $O(n^2 * 2^n)$** , di mana  $n$  adalah jumlah kota. Hal ini disebabkan oleh fakta bahwa ada  $2^n$  kombinasi bitmask dan untuk setiap bitmask kita melakukan iterasi  $n$  kali untuk memperbarui nilai dp.

## SOURCE CODE PROGRAM DALAM BAHASA PEMROGRAMAN RUBY

### tsp\_solver.rb

```
require 'benchmark'
INF = 1_000_000_000
MAXN = 20
def tsp(mask, pos, adj, dp, next_node, n)
  if mask == (1 << n) - 1
    return adj[pos][0] # Return to the start point
  end
  return dp[mask][pos] if dp[mask][pos] != -1
  ans = INF
  (0...n).each do |v|
    if (mask & (1 << v)) == 0
      curr = adj[pos][v] + tsp(mask | (1 << v), v, adj, dp, next_node, n)
      if curr < ans
        ans = curr
        next_node[mask][pos] = v # Store the next node in the path
      end
    end
  end
  dp[mask][pos] = ans
  ans
end
def print_path(start, next_node, n)
  mask = 1
  pos = start
  print "Path: #{pos + 1} "
  while mask != (1 << n) - 1
    pos = next_node[mask][pos]
    mask |= (1 << pos)
    print "→ #{pos + 1} "
  end
  puts "→ 1" # Return to the start point
end
def print_matrix(matrix, n, m)
  puts "Adjacency Matrix:"
  (0...n).each do |i|
    (0...m).each do |j|
```

```

        print matrix[i][j] == INF ? "INF".ljust(5) :
matrix[i][j].to_s.ljust(5)
    end
    puts
end
end
def solve
    puts "Choose a test case (1, 2, or 3):"
    choice = gets.chomp.to_i
    case choice
    when 1
        n = 4
        m = 4
        foo = [
            0, 10, 15, 20,
            5, 0, 9, 10,
            6, 13, 0, 12,
            8, 8, 9, 0
        ] # Test case 1
    when 2
        n = 5
        m = 5
        foo = [
            INF, 20, 30, 10, 11,
            15, INF, 16, 4, 2,
            3, 5, INF, 2, 4,
            19, 6, 18, INF, 3,
            16, 4, 7, 16, INF
        ] # Test case 2
    when 3
        n = 7
        m = 7
        foo = [
            0, 12, 10, INF, INF, INF, 12,
            12, 0, 8, 12, INF, INF, 12,
            10, 8, 0, 11, 3, INF, 9,
            INF, 12, 11, 0, 11, 10, INF,
            INF, INF, 3, 11, 0, 6, 7,
            INF, INF, INF, 10, 6, 0, 9,
            12, INF, 9, INF, 7, 9, 0
        ] # Test case 3
    end
end

```

```

else
  puts "Invalid choice"
  return
end
adj = Array.new(n) { Array.new(m, 0) }
dp = Array.new(1 << MAXN) { Array.new(MAXN, -1) }
next_node = Array.new(1 << MAXN) { Array.new(MAXN, -1) }
(0...n).each do |i|
  (0...m).each do |j|
    adj[i][j] = foo[i * m + j]
  end
end
print_matrix(adj, n, m)
min_cost = nil
time = Benchmark.realtime do
  min_cost = tsp(1, 0, adj, dp, next_node, n)
end
puts "The shortest path has length #{min_cost}"
puts "Execution time: #{time.round(6)} seconds"
print_path(0, next_node, n) # Print the path starting from node 0
end
solve

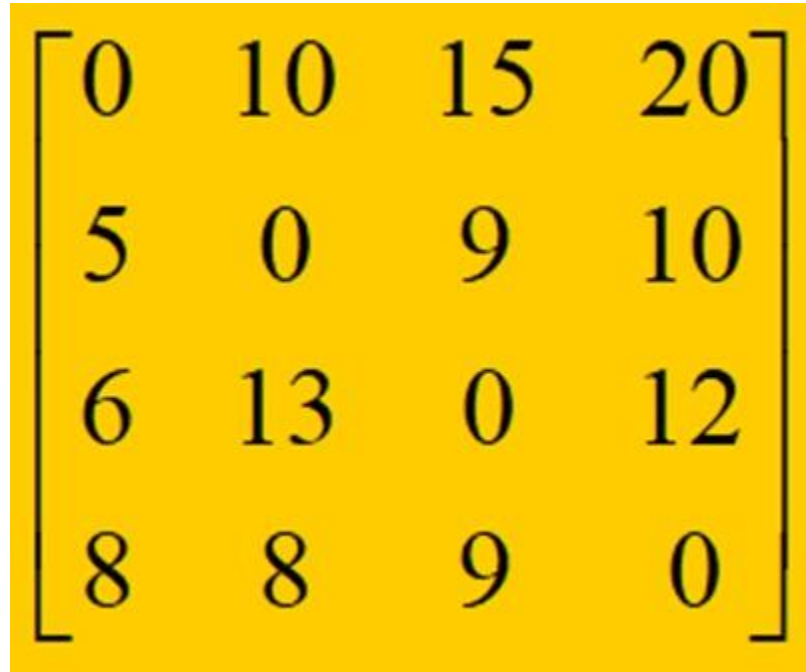
```

#### Daftar Fungsi yang digunakan pada program:

Function	Kegunaan
<b>tsp</b>	Melakukan komputasi TSP dengan Dynamic Programing secara rekursif
<b>print_path</b>	Menampilkan path yang merupakan solusi TSP
<b>print_matrix</b>	Menampilkan matrix inputan untuk TSP
<b>solve</b>	Melakukan seluruh komputasi untuk program pada file (Main function)

## SKRINSUT HASIL PENGUJIAN

### 1. Test Case 1



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

Output:

```
[nodrop@Gilangs-Air src % ruby tsp_solver.rb
Choose a test case (1, 2, or 3):
1
Adjacency Matrix:
0    10    15    20
5     0     9    10
6    13     0    12
8     8     9     0
The shortest path has length 35
Execution time: 1.8e-05 seconds
Path: 1 -> 2 -> 4 -> 3 -> 1
[nodrop@Gilangs-Air src %]
```

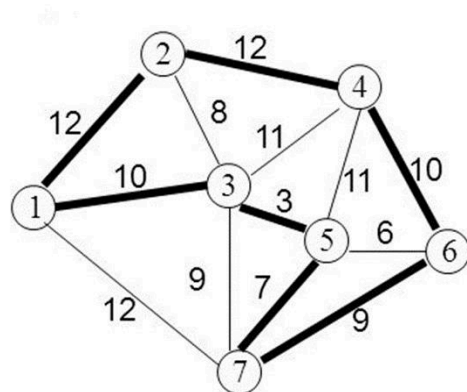
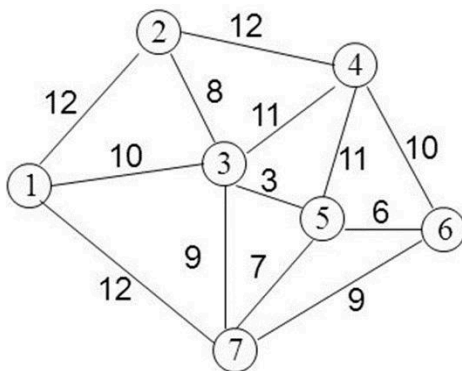
### 2. Test Case 2

$\infty$	20	30	10	11
15	$\infty$	16	4	2
3	5	$\infty$	2	4
19	6	18	$\infty$	3
16	4	7	16	$\infty$

**Output:**

```
[nodrop@Gilangs-Air src % ruby tsp_solver.rb
Choose a test case (1, 2, or 3):
2
Adjacency Matrix:
INF 20 30 10 11
15 INF 16 4 2
3 5 INF 2 4
19 6 18 INF 3
16 4 7 16 INF
The shortest path has length 28
Execution time: 5.1e-05 seconds
Path: 1 -> 4 -> 2 -> 5 -> 3 -> 1
```

### 3. Test Case 3





**Output:**

```
[nodrop@Gilangs-Air src % ruby tsp_solver.rb
Choose a test case (1, 2, or 3):
3
Adjacency Matrix:
0      12      10      INF      INF      INF      12
12      0       8      12      INF      INF      12
10      8       0      11       3      INF       9
INF     12      11      0       11      10      INF
INF     INF      3      11      0       6       7
INF     INF     INF     10      6       0       9
12      INF      9      INF      7       9       0
The shortest path has length 63
Execution time: 0.000265 seconds
Path: 1 -> 2 -> 4 -> 6 -> 7 -> 5 -> 3 -> 1
```

## **LAMPIRAN**

Link Repository Github: [https://github.com/gilangr301102/Challenge\\_13520137](https://github.com/gilangr301102/Challenge_13520137)