

**IF2211 STRATEGI ALGORITMA
TUGAS TANTANGAN
Penyelesaian Traveling Salesman Problem dengan Dynamic
Programming Menggunakan Bahasa Rust, Perl, Swift, atau
Ruby**



Dipersiapkan oleh:

Ahmad Naufal Ramadan - 13522005

**PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
JL. GANESA 10, BANDUNG 40132
2024**

Daftar Isi

Daftar Isi	2
1. Deskripsi Tugas	3
2. Source Code Program	3
main.rs	3
3. Uji Kasus	6
Tabel 3.1 Tangkapan Layar Hasil Uji Kasus	6
Lampiran	8

1. Deskripsi Tugas

Tugas Tantangan ini adalah tugas bonus dari Pak Rinaldi yang merupakan sebuah tantangan bagi kami untuk menyelesaikan persoalan Traveling Salesman Problem dengan mengimplementasikan dynamic programming yang telah diajarkan di kelas. Tantangannya adalah solusi harus dibuat menggunakan bahasa baru yang belum pernah digunakan mahasiswa IF'22 sebelumnya. Beberapa bahasa pilihan yang diperbolehkan adalah Rust, Perl, Swift, atau Ruby.

2. Source Code Program

main.rs

```
use std::collections::{BTreeSet, HashMap};
use std::env;
use std::fs::File;
use std::io::{self, BufRead};

type AdjMatrix = Vec<Vec<f64>>>;

fn read_input_file(file_path: &str) -> io::Result<(usize, AdjMatrix)> {
    let file = File::open(file_path)?;
    let mut lines = io::BufReader::new(file).lines();
    let n = lines.next().unwrap()?.parse::<usize>().unwrap();
    let mut adj_matrix = vec![vec![f64::INFINITY; n]; n];

    for i in 0..n {
        if let Some(line) = lines.next() {
            let line = line?;
            let values = line
                .split_whitespace()
                .map(|x| {
                    if x == "inf" {
                        f64::INFINITY
                    } else {
                        x.parse().unwrap()
                    }
                })
                .collect::<Vec<f64>>();
            adj_matrix[i] = values;
        }
    }
}
```

```

    }
}

Ok((n, adj_matrix))
}

fn tsp(
    i: usize,
    s: &BTreeSet<usize>,
    adj_matrix: &AdjMatrix,
    memo: &mut HashMap<(usize, BTreeSet<usize>), f64>,
) -> f64 {
    if s.is_empty() {
        return adj_matrix[i][0];
    }

    if let Some(&cost) = memo.get(&(i, s.clone())) {
        return cost;
    }

    let mut min_cost = f64::INFINITY;

    for &j in s {
        if adj_matrix[i][j] != f64::INFINITY {
            let mut s_ = s.clone();
            s_.remove(&j);
            let cost = adj_matrix[i][j] + tsp(j, &s_, adj_matrix, memo);
            min_cost = min_cost.min(cost);
        }
    }

    memo.insert((i, s.clone()), min_cost);
    min_cost
}

fn get_path(
    i: usize,
    s: &BTreeSet<usize>,
    adj_matrix: &AdjMatrix,

```

```

        memo: &mut HashMap<(usize, BTreeSet<usize>), f64>,
    ) -> Vec<usize> {
        if s.is_empty() {
            return vec![0];
        }

        let mut min_cost = f64::INFINITY;
        let mut min_path = vec![];

        for &j in s {
            if adj_matrix[i][j] != f64::INFINITY {
                let mut s_ = s.clone();
                s_.remove(&j);
                let cost = adj_matrix[i][j] + tsp(j, &s_, adj_matrix, memo);
                if cost < min_cost {
                    min_cost = cost;
                    min_path = vec![j];
                    min_path.extend(get_path(j, &s_, adj_matrix, memo));
                }
            }
        }

        min_path
    }

fn get_tour(
    n: usize,
    adj_matrix: &AdjMatrix,
    memo: &mut HashMap<(usize, BTreeSet<usize>), f64>,
) -> Vec<usize> {
    let path = get_path(0, &(1..n).collect::<BTreeSet<_>>(), adj_matrix, memo);
    let mut tour = path.clone();
    tour.insert(0, 0);
    tour
}

fn main() {
    let args: Vec<String> = env::args().collect();

```

```

    if args.len() < 2 {
        eprintln!("Usage: {} <path_to_config_file>", args[0]);
        return;
    }

    let input_file = &args[1];
    let (n, adj_matrix) = read_input_file(input_file).expect("Failed to
read input file");

    let mut memo = HashMap::new();
    let min_cost = tsp(0, &(1..n).collect::<BTreeSet<_>>(), &adj_matrix,
&mut memo);

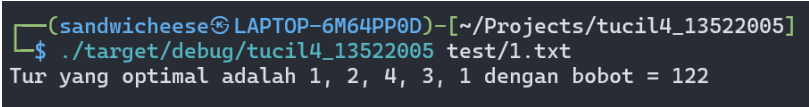
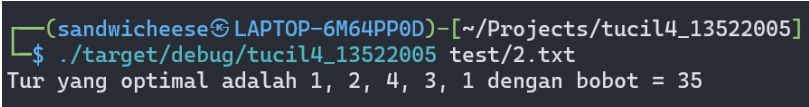
    let tour = get_tour(n, &adj_matrix, &mut memo);
    let tour: Vec<String> = tour.iter().map(|&x| (x +
1).to_string()).collect();

    println!(
        "Tur yang optimal adalah {} dengan bobot = {}",
        tour.join(", "),
        min_cost
    );
}

```

3. Uji Kasus

Tabel 3.1 Tangkapan Layar Hasil Uji Kasus

No	Test Case	Hasil Pengujian
1	4 0 22 26 30 30 0 45 35 25 45 0 60 30 35 40 0	 <pre> (sandwichese@LAPTOP-6M64PP0D) ~/Projects/tucil4_13522005 \$./target/debug/tucil4_13522005 test/1.txt Tur yang optimal adalah 1, 2, 4, 3, 1 dengan bobot = 122 </pre>
2	4 0 10 15 20 5 0 9 10	 <pre> (sandwichese@LAPTOP-6M64PP0D) ~/Projects/tucil4_13522005 \$./target/debug/tucil4_13522005 test/2.txt Tur yang optimal adalah 1, 2, 4, 3, 1 dengan bobot = 35 </pre>

	6 13 0 12 8 8 9 0	
3	5 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	<pre>(sandwichese@LAPTOP-6M64PP0D)~/Projects/tucil4_13522005] \$./target/debug/tucil4_13522005 test/3.txt Tur yang optimal adalah 1, 4, 2, 5, 3, 1 dengan bobot = 28</pre>
4	7 0 12 10 inf inf inf 12 12 0 8 12 inf inf inf 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	<pre>(sandwichese@LAPTOP-6M64PP0D)~/Projects/tucil4_13522005] \$./target/debug/tucil4_13522005 test/4.txt Tur yang optimal adalah 1, 2, 4, 6, 7, 5, 3, 1 dengan bobot = 63</pre>

Lampiran

Link Github: https://github.com/SandWithCheese/tuci4_13522005