PENDAHULUAN

Pada implementasi permasalahan TSP dengan Dynamic programming, saya menggunakan bahasa rust untuk mengimplementasikan tantangan ini. Dalam Program saya, terdapat 3 file utama yaitu main.rs untuk melakukan penginputan dan juga pemanggilan fungsi, kemudian terdapat juga tsp.rs yang dimana terdapat kode program utama untuk permasalahan tsp, dan terakhir matrix_reader.rs yang berfungsi untuk membaca file txt untuk penginputannya.

IMPLEMENTASI PROGRAM

Main.rs

```rust
mod matrix_reader;
mod tsp;
use tsp::{tsp_dp};
use matrix_reader::read_matrix_from_file;

use std::io::{self};

fn main() -> io::Result<()> {
    println!("Enter the filename containing the matrix:");
    let mut input = String::new();
    io::stdin().read_line(&mut input)?;
    let filename = input.trim();

    let dist = read_matrix_from_file(filename)?;
    let n = dist.len();

    let (result, route) = tsp_dp(&dist, n);

    if result == i32::MAX {
        println!("No feasible path found due to infinite or invalid weights.");
    } else {
        println!("\nThe minimum cost is: {}", result);
        println!("Path taken (starting from city 1): {}",
            route.iter().map(|&x| (x + 1).to_string()).collect::<Vec<_>>().join(" -> "));

        if route.len() > 1 {
            println!("\nDetailed route:");
            for i in 0..route.len() - 1 {
                let from = route[i];
                let to = route[i + 1];
                println!("Travel from city {} to city {}: {} units",
                         from + 1, to + 1, dist[from][to]);
            }
        }
    }

    Ok(())
}
```

tsp.rs

```rust
const INFINITY: i32 = i32::MAX;

pub fn tsp_dp(dist: &[Vec<i32>], n: usize) -> (i32, Vec<usize>) {
    let mut memo = vec![vec![i32::MAX; 1 << n]; n];
    let mut path = vec![vec![usize::MAX; 1 << n]; n];

    let min_cost = tsp(0, 1, n, dist, &mut memo, &mut path);
    let route = find_path(n, &path, 0);

    (min_cost, route)
}

pub fn tsp(pos: usize, visited: usize, n: usize, dist: &[Vec<i32>], memo: &mut Vec<Vec<i32>>, path: &mut Vec<Vec<usize>>) -> i32 {
    if visited == (1 << n) - 1 {
        return dist[pos][0];
    }

    if memo[pos][visited] != i32::MAX {
        return memo[pos][visited];
    }

    let mut res = i32::MAX;
    for i in 0..n {
        if (visited & (1 << i)) == 0 && dist[pos][i] != INFINITY {
            let next_cost = tsp(i, visited | (1 << i), n, dist, memo, path);
            if next_cost != INFINITY {
                let cost = dist[pos][i] + next_cost;
                if cost < res {
                    res = cost;
                    path[pos][visited] = i;
                }
            }
        }
    }

    memo[pos][visited] = res;
    res
}
```

```rust
pub fn find_path(n: usize, path: &[Vec<usize>], start: usize) -> Vec<usize> {
    let mut route = vec![start];
    let mut current = start;
    let mut visited = 1 << start;

    while visited != (1 << n) - 1 {
        let next = path[current][visited];
        route.push(next);
        visited |= 1 << next;
        current = next;
    }
    route.push(start);
    route
}
```

matrix_reader.rs

```rust
use std::fs::File;
use std::io::{self, BufRead, BufReader};

const INFINITY: i32 = i32::MAX;

pub fn read_matrix_from_file(filename: &str) -> io::Result<Vec<Vec<i32>>> {
    let file = File::open(filename)?;
    let reader = BufReader::new(file);
    let mut matrix = Vec::new();

    for line in reader.lines() {
        let line = line?.trim().to_string();
        let numbers: Vec<i32> = line.split_whitespace()
            .map(|num| {
                if num == "∞" {
                    INFINITY
                } else {
                    num.parse().unwrap_or_else(|_| {
                        eprintln!("Error parsing number: {}", num);
                        std::process::exit(1);
                    })
                }
            })
            .collect();
        matrix.push(numbers);
    }
    Ok(matrix)
}
```

PENGUJIAN

1. **Test case 1**

```
testcase_1.txt
src > testcase_1.txt
0 10 15 20
5 0 9 10
6 13 0 12
8 8 9 0
```

```
PS C:\Users\User\Documents\ITB\Sem 4\Stima\TSP_with_DP\src> cargo run
    Finished `dev` profile [unoptimized + debuginfo] target(s) in 0.01s
     Running `C:\Users\User\Documents\ITB\Sem 4\Stima\TSP_with_DP\target\debug\tsp_algo.exe`
Enter the filename containing the matrix:
testcase_1.txt

The minimum cost is: 35
Path taken (starting from city 1): 1 -> 2 -> 4 -> 3 -> 1

Detailed route:
Travel from city 1 to city 2: 10 units
Travel from city 2 to city 4: 10 units
Travel from city 4 to city 3: 9 units
Travel from city 3 to city 1: 6 units
```

2. **Test case 2**

```
testcase_2.txt
src > testcase_2.txt
∞ 20 30 10 11
15 ∞ 16 4 2
3 5 ∞ 2 4
19 6 18 ∞ 3
16 4 7 16 ∞
```

```
PS C:\Users\User\Documents\ITB\Sem 4\Stima\TSP_with_DP\src> cargo run
    Finished `dev` profile [unoptimized + debuginfo] target(s) in 0.01s
     Running `C:\Users\User\Documents\ITB\Sem 4\Stima\TSP_with_DP\target\debug\tsp_algo.exe`
Enter the filename containing the matrix:
testcase_2.txt

The minimum cost is: 28
Path taken (starting from city 1): 1 -> 4 -> 2 -> 5 -> 3 -> 1

Detailed route:
Travel from city 1 to city 4: 10 units
Travel from city 4 to city 2: 6 units
Travel from city 2 to city 5: 2 units
Travel from city 5 to city 3: 7 units
Travel from city 3 to city 1: 3 units
```

3. Test case 3

testcase_3.txt

src > testcase_3.txt

```
0   12  10  ∞   ∞   ∞   12
12  0   8   11  ∞   ∞   ∞
10  8   0   ∞   3   ∞   9
∞   11  ∞   0   11  10  ∞
∞   ∞   3   11  0   6   7
∞   ∞   ∞   10  6   0   9
12  ∞   9   ∞   7   9   0
```

```
PS C:\Users\User\Documents\ITB\Sem 4\Stima\TSP_with_DP\src> cargo run
    Finished `dev` profile [unoptimized + debuginfo] target(s) in 0.01s
     Running `C:\Users\User\Documents\ITB\Sem 4\Stima\TSP_with_DP\target\debug\tsp_algo.exe`
Enter the filename containing the matrix:
testcase_3.txt

The minimum cost is: 62
Path taken (starting from city 1): 1 -> 2 -> 4 -> 6 -> 7 -> 5 -> 3 -> 1

Detailed route:
Travel from city 1 to city 2: 12 units
Travel from city 2 to city 4: 11 units
Travel from city 4 to city 6: 10 units
Travel from city 6 to city 7: 9 units
Travel from city 7 to city 5: 7 units
Travel from city 5 to city 3: 3 units
Travel from city 3 to city 1: 10 units
```

LAMPIRAN

Link Github :
https://github.com/Filbert88/TSP_with_DP