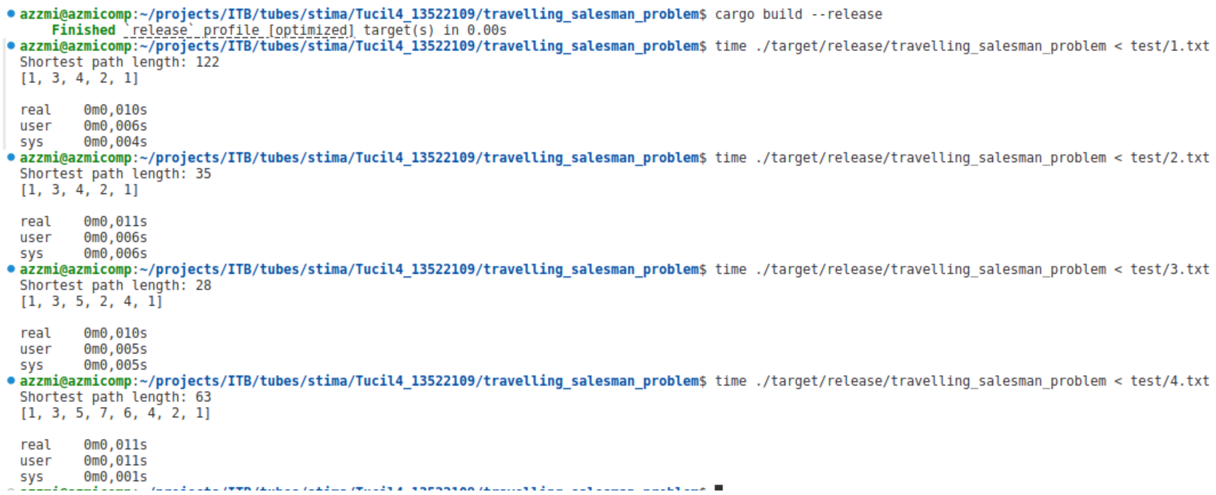
Laporan Tugas Tantangan Travelling Salesman Problem

Link Github: https://github.com/RealAzzmi/Tucil4_13522109/

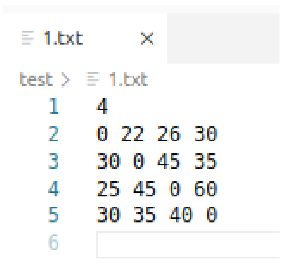
A. Code

```
use std::io;
use std::process;

#[derive(Clone, Copy)]
struct Path {
    node: usize,
    total_cost: f64,
}

impl Path {
    fn new() -> Self {
        Path {
            node: 0,
            total_cost: f64::INFINITY,
        }
    }

    fn with_cost(node: usize, total_cost: f64) -> Self {
        Path {
            node,
            total_cost,
        }
    }
}
fn main() {
    let mut input = String::new();
    io::stdin()
        .read_line(&mut input)
        .expect("Failed to read line");

    let n: i32 = input.trim().parse().expect("Please type a number!");

    if n <= 0 {
        println!("The number of nodes must be positive");
        process::exit(0);
    }

    let n: usize = n.try_into().expect("The number of nodes is too large");
    let mut adj = vec![vec![0.0; n]; n];
    for i in 0..n {
        input.clear();
        io::stdin()
            .read_line(&mut input)
            .expect("Failed to read line");
        let values: Vec<&str> = input.trim().split_whitespace().collect();

        for j in 0..n {
            if values[j] == "inf" {
                adj[i][j] = f64::INFINITY;
            } else {
                adj[i][j] = values[j]
                    .parse::<f64>()
                    .expect("Please type a valid number or 'inf'");
            }
        }
    }

    let mut dp: Vec<Vec<Path>> = vec![vec![Path::new(); n]; 1 << n];
    dp[1][0] = Path::with_cost(0, 0.0);

    for i in 2..(1 << n) {
        for j in 0..n {
            if (i & (1 << j)) == 0 {
                continue;
            }
            for k in 0..n {
                if (i & (1 << k)) == 0 || j == k {
                    continue;
                }
                if dp[i ^ (1 << j)][k].total_cost + adj[k][j] <
dp[i][j].total_cost {
                    dp[i][j].total_cost = dp[i ^ (1 << j)][k].total_cost +
adj[k][j];
                    dp[i][j].node = k;
                }
            }
        }
    }

    let mut shortest_path = f64::INFINITY;
    let mut last_node: usize = 0;
    for i in 1..n {
        let cost = dp[(1 << n) - 1][i].total_cost + adj[i][0];
        if cost <= shortest_path {
            shortest_path = cost;
            last_node = i;
        }
    }

    if n == 1 {
        println!("Shortest path length: {}", 0);
        println!("{}", 1);
    } else {
        println!("Shortest path length: {}", shortest_path);
        let mut cur = (1 << n) - 1;
        let mut path_nodes = vec![];
        path_nodes.push(1);
        for _ in 0..n {
            path_nodes.push(last_node + 1);
            let previous_node = last_node;
            last_node = dp[cur][last_node].node;
            cur ^= 1 << previous_node;
        }
        println!("{:?}", path_nodes);
    }
}
```

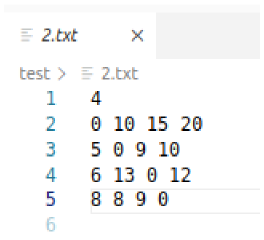
B. Tests



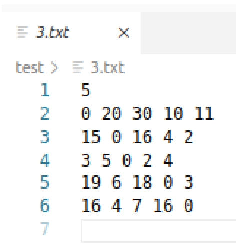
```
• azzmi@azmicomp:~/projects/ITB/tubes/stima/Tucil4_13522109/travelling_salesman_problem$ cargo build --release
    Finished `release` profile [optimized] target(s) in 0.00s
• azzmi@azmicomp:~/projects/ITB/tubes/stima/Tucil4_13522109/travelling_salesman_problem$ time ./target/release/travelling_salesman_problem < test/1.txt
  Shortest path length: 122
  [1, 3, 4, 2, 1]

  real    0m0,010s
  user    0m0,006s
  sys     0m0,004s
• azzmi@azmicomp:~/projects/ITB/tubes/stima/Tucil4_13522109/travelling_salesman_problem$ time ./target/release/travelling_salesman_problem < test/2.txt
  Shortest path length: 35
  [1, 3, 4, 2, 1]

  real    0m0,011s
  user    0m0,006s
  sys     0m0,006s
• azzmi@azmicomp:~/projects/ITB/tubes/stima/Tucil4_13522109/travelling_salesman_problem$ time ./target/release/travelling_salesman_problem < test/3.txt
  Shortest path length: 28
  [1, 3, 5, 2, 4, 1]

  real    0m0,010s
  user    0m0,005s
  sys     0m0,005s
• azzmi@azmicomp:~/projects/ITB/tubes/stima/Tucil4_13522109/travelling_salesman_problem$ time ./target/release/travelling_salesman_problem < test/4.txt
  Shortest path length: 63
  [1, 3, 5, 7, 6, 4, 2, 1]

  real    0m0,011s
  user    0m0,011s
  sys     0m0,001s
```



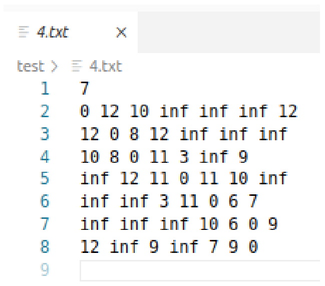
```
4
0 22 26 30
30 0 45 35
25 45 0 60
30 35 40 0
```



```
4
0 10 15 20
5 0 9 10
6 13 0 12
8 8 9 0
```



```
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
```



```
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
```