Tennessee Tech. University — Jeffrey Graves
Design of Algorithms — Program 3
CSC 2400-002 — Due: March 10, 2023

Brute Force Traveling Salesman Problem

The Traveling Salesman Problem

Given a collection of cities, along with pairwise distances between the cities, what is the shortest route that visits each city exactly once and returns the the starting city? This question is commonly referred to as the *traveling salesman problem* (TSP). The traveling salesman problem can be reformulated as finding a Hamiltonian cycle of least cost in a weighted graph — a Hamiltonian cycle is a cycle that includes every vertex in the graph. This assignment will focus on a brute-force solution to the traveling salesman problem.

Program Specifications

You will write a C++ program to implement a brute-force, permutation-based solution for the traveling salesman problem. Your program will take in a single command line input specifying a file to read. The input file will contain *one or more* lines, each of which specifies a directed edge the form "*src dst wt*" where *src* and *dst* are non-negative integers indicating the source and destination vertices of the edge, and *wt* is the weight of the edge. I will provide you with sample input files for testing, but it is your responsibility make sure that your program behaves correctly on any valid input file. Your program will output the cost of the minimum Hamiltonian cycle. See the end of this handout for example output.

You will need to implement your own adjacency matrix or adjacency list structure to store the graph. Take care not to over engineer your data structure — you do not need to write a full class to implement your graph data structure. I suggest that you rely on standard template library containers for your implementation (see https://cplusplus.com/reference/stl/). For instance

- `std::vector<std::vector<vertex_t> > adj_matrix;`
- `std::vector<std::list<vertex_t> > adj_list;`

Regardless of the data structure you choose, do not hard-code limits on your data structure size. Also, please note that variable-length arrays (VLAs) are not part of the C++ standard and should not be used. For your solution, you make use the `std::next_permutation` function.

Submission and Grading

You must use `skeleton3.cpp` (see iLearn) as a starting point for your program, and complete the `TSP` function; feel free to create any additional helper functions or include any additional standard libraries that you need, but do not modify any other existing functions. Your source code should be contained in a single file and should be named after your TTU email address excluding the "@tntech.edu" (e.g., `jagraves21.cpp`). All submissions will be made on iLearn — please do not zip or compress your files. Make sure to follow best coding practices (proper naming conventions, useful comments, etc.). Your program should compile without errors or warnings. Programs will be compiled using the following command:

```
g++ -Wall -pedantic -std=c++11 [source file]
```

Sample Output

The following lines contain sample input and expected output to your programs. Please note that these examples are not exhaustive, and you should verify your programs with additional test cases.

```
$ ./a.out graph1.txt
18

$ ./a.out graph2.txt
14

$ ./a.out graph3.txt
No Hamiltonian cycle exists.
```