Optimal Travel

DOCUMENTATION

Kwan, Elena W.C
DELOITTE | [COMPANY ADDRESS]

1. Introduction

The purpose of this package is provide a program that provides a solution to the travelling salesman problem based on the inputs that the user provides (i.e. number of cities within the trip and costs of travelling to and from each city). For example, if a user is looking to visit 5 cities, the program would indicate the route they should take to minimize their costs, where each city is visited exactly once and where the user returns to the origin city.

This travelling salesman problem is solved using the branch and bound approach described by the saurabhschool.org. Since this problem is computationally heavy, the program was written in c (TSP.c) for performance reasons. A web interface was written in php, which makes a call to compiled executable code of the c program.

This file contains:

> **(1)** A c-program (and related helper libraries) that calculates the lowest total travelling costs and the optimal route the user should take
> **(2)** Code for a website intended to be hosted on CS50 IDE's web server (Apache). This website runs the executable file of the c-program and is intended to be a user friendly interface for c-program.

While the program is mainly intended to be used via a web interface, the program can also be run via terminal commands. Instructions of using both interfaces are provided below.

2. Website interface

2.1. Setting up the website:

Once the package is loaded (unzipped) into your cs50 (cloud 9) ide workspace folder, perform the following:

- Navigate to that the public folder within that package via your terminal window:
  ```
  cd ~/workspace/FinalProjectWeb2/public
  ```
- **Set the appropriate permissions**
  - Ensure the public folder is world executable
    ```
    chmod a+x ~/workspace/FinalProjectWeb2/public
    ```
  - Ensure that the sub-folders within public are world executable
    ```
    chmod a+x css fonts img js
    ```
  - Ensure the files within the subdirectories are also world executable
    ```
    chmod a+r css/* fonts/* img/* js/*
    ```

- **Start Apache:**
  `apache50 start ~/workspace/FinalProjectWeb2/public`

2.2 Using the website:

Once is the website is set up, it should be viewable on `https://ide50-username.cs50.io/`, where `username` is your own username.

The website will take you through a linear flow of screens which prompt you to provide different pieces of information.

(1) Provide the total number of cities you intend to visit (including your starting city)

- For example, if you start in Boston and plan on travelling to Miami and Scottsdale, you would enter "3"
- The input should be a numeric value (i.e. "3" and not "three")
- The number of cities should be between 2 and 25 (there must be at least two cities to travel between and the algorithm only capable of handling 25 cities in less than 30 seconds and performance decreases significantly after that point).

Once you submit this form, you will be taken to a second form to provide cost information.

(2) Provide the costs of going to/from each city

- Provide the cost of going to and from each city, with the exception of going to and from the same city. Since that travel option is not applicable, the diagonal cells of the form do not require inputs.
- Please note that city 1 is the city you are starting from and the city you will return to at the end of the route. In the example described previously, conceptually city 1 would be Boston.
- All costs must be between (and inclusive of) 0 – 2147483646 for reasons described in the design document.
- Since you, the user, specifies the inputs, this cost could either be distance, plane fare, or some other "cost metric". The cost measure should be the same for all inputs (e.g. if the cost of going from city 2 to city 3 is specified in dollars, all other costs should be in dollars). However, there is no need to provide that information as an input into the calculation.

3. Terminal Interface

3.1 Running the program directly in the terminal

The c-program (TSP.c) is already compiled so only its executable file, TSP, needs to be run.

- Since the executable file is in the public folder, navigate to this folder via the terminal
  `cd ~/workspace/FinalProjectWeb2/public`

- The program takes two arguments:
  - **(1)** The total number of cities (which follow the same restrictions as described above for the web input)
  - **(2)** The name of a text file which contains the costs of going to/from each city.
    - The text file should also be saved in the public folder
    - Information in the text file must be in a certain format in order for the program to correctly read the inputs (described below)

- Execute the compiled executable file, specifying the two arguments described above

  ```
  ./TSP <total number of cities> <name of your text file>
  ```

  For example: `./TSP 5 mydata.txt`

- The program will print the output to the terminal. An example output for a journey involving 6 cities is:

  ```
  the lower bound is : 52
  optimal path is : 1 6 3 2 5 4 1
  ```

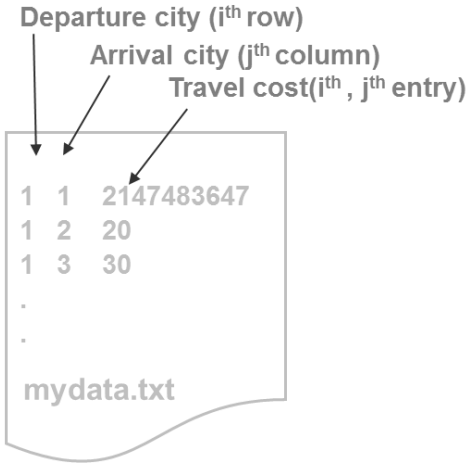
- If the c files are changed, the program can be recompiled via a 'make' command as makefile exists

  ```
  Make TSP
  ```

3.2 Text file format

- **Each row of the file contains 3 numbers**, separated by *single spaces*:
  - The first number corresponds the departure city
  - The second number corresponds to the arrival city
  - The 3rd number represents the travel costs

  No additional information should be provided.



- **The file must contain (number of cities)^2 rows**
    - In other words, values for all travel options, even logically not applicable options, must be explicitly specified. These non-applicable travel options are then logically negated by specifying the travel cost as 2147483647.

    - Thus, the options of travelling to/from the same cities must be provided as individual rows, where the travel cost is listed as 2147483647 in order to logically negate this option. For example, if there are 3 cities, the following entries must be provided:

```
1 1 2147483647
```

```
2 2 2147483647
```

```
3 3 2147483647
```

    - Rows for symmetric entries must also be provided. For example, the travel distance of going from city 2 to city 3 may be 50m. The travel distance of going from city 3 to city 2 may be identical. However, the two travel options must be explicated listed as different rows.

```
1 2 50
```

```
2 1 50
```