CSC 2400 Program 2

Traveling Salesman Problem

# Terminology

The **traveling salesman problem** is the problem of finding a minimum-weight Hamilton Circuit in a complete graph.

A **complete graph** is a graph with ***n*** vertices and an edge between every two vertices.

A **Hamilton Circuit** is a circuit that uses every vertex of a graph once.

A **weighted graph** is a graph in which each edge is assigned a weight representing the time, distance, or cost of traversing that edge.

# Program Description

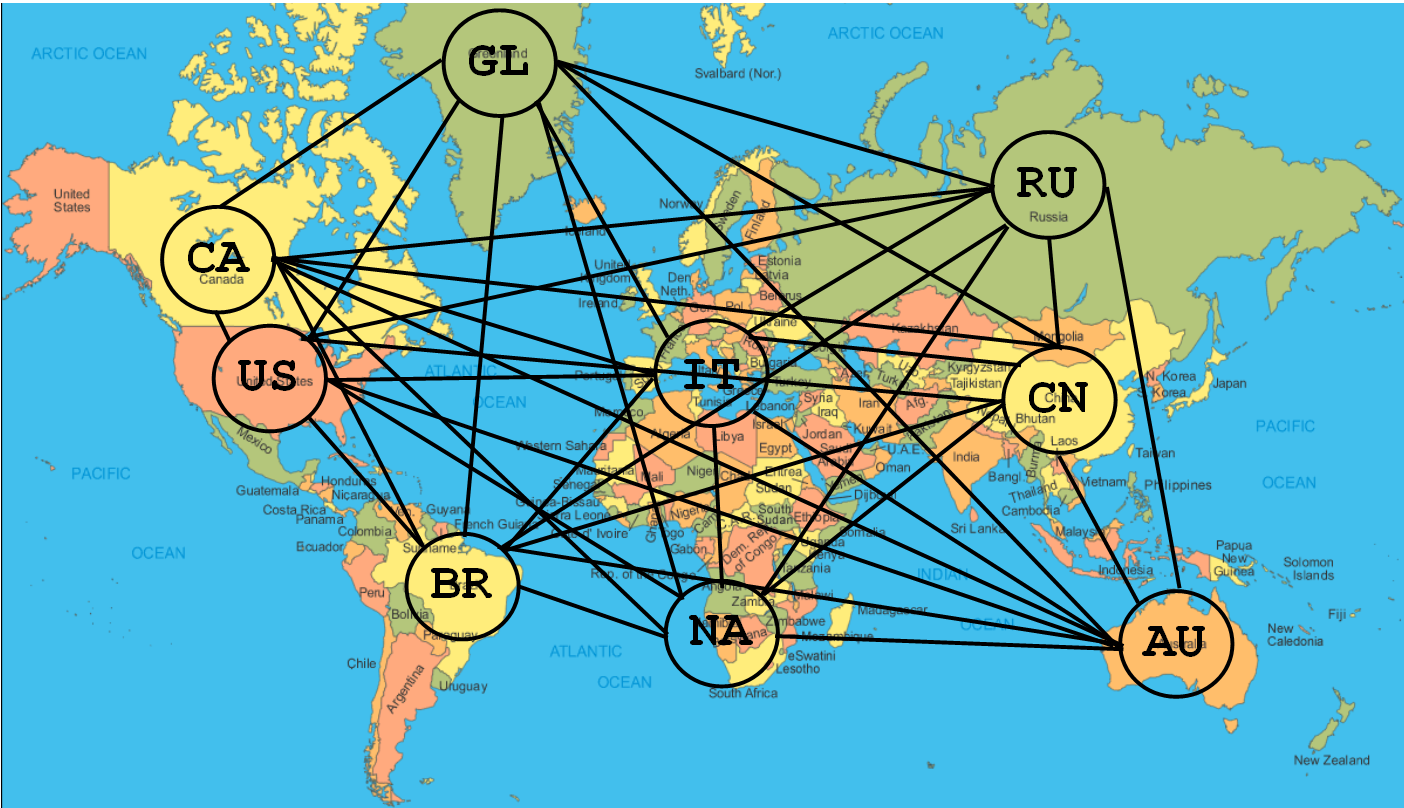
You are a famous cold-processed soap-maker. You want to travel to all of your soap factories spending the least amount of money.



You live in the United States (US) so you will always **begin** and **end** your tour with United States (US). The following is a list of countries that contain one of your factories in addition to United States and the country’s abbreviation used in your program:

* Australia, **AU**
* Brazil, **BR**
* Canada, **CA**
* China, **CN**
* Greenland, **GL**
* Italy, **IT**
* Namibia, **NA**
* Russia, **RU**

The following image is a world map that indicates the location of each country:



The following table lists the cost to travel to/from each country with an airplane flight:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **AU** | **BR** | **CA** | **CN** | **GL** | **IT** | **NA** | **RU** | **US** |
| **US** | $ 807.00 | $ 628.00 | $ 170.00 | $ 677.00 | $ 1,277.00 | $ 539.00 | $ 882.00 | $ 548.00 | $ 0 |
| **CA** | $1,041.00 | $ 620.00 | $ 0 | $ 649.00 | $ 883.00 | $ 774.00 | $ 938.00 | $ 839.00 | $ 170.00 |
| **GL** | $3,038.00 | $4,505.00 | $ 883.00 | $3,496.00 | $ 0 | $10,000.00 | $10,000.00 | $10,000.00 | $1,277.00 |
| **BR** | $3,145.00 | $ 0 | $ 620.00 | $4,617.00 | $ 4,505.00 | $ 560.00 | $ 650.00 | $ 741.00 | $ 628.00 |
| **IT** | $1,494.00 | $ 560.00 | $ 774.00 | $1,009.00 | $10,000.00 | $ 0 | $ 920.00 | $ 230.00 | $ 539.00 |
| **NA** | $1,522.00 | $ 650.00 | $ 938.00 | $1,142.00 | $10,000.00 | $ 920.00 | $ 0 | $ 1,101.00 | $ 882.00 |
| **RU** | $1,481.00 | $ 741.00 | $ 839.00 | $ 773.00 | $10,000.00 | $ 230.00 | $ 1,101.00 | $ 0 | $ 548.00 |
| **CN** | $1,867.00 | $4,617.00 | $ 649.00 | $ 0 | $ 3,496.00 | $ 1,009.00 | $ 1,142.00 | $ 773.00 | $ 677.00 |
| **AU** | $ 0 | $3,145.00 | $1,041.00 | $1,867.00 | $ 3,038.00 | $ 1,494.00 | $ 1,522.00 | $ 1,481.00 | $ 807.00 |

# Program Specifications

* You must use the files that are provided (**given.cpp**, **GraphMatrix.h**, **flight.txt**).
* Rename **given.cpp** to **program2.cpp**.
* Write a **C++** program to implement the **Traveling Salesman Algorithm** where you will find the minimum cost of traveling to each country once, starting from and ending at United States.
* Use the **Lexicographic Permute** **Algorithm** to generate all tours possible. I have provided you with a program that implements the lexicographic permute algorithm, which you will have to modify for your program.
* I should be able to compile your code with **g++ -Wall program2.cpp**.

# Algorithm Steps

With the Traveling Salesman Problem, the first step is to generate all permutations of the problem instance. In this program, it is easiest to calculate the cost of each permutation (tour) as soon as you generate the permutation. So, for each iteration of the algorithm you will:

1. Generate the new permutation based on lexicographic permute algorithm (find i, find j, swap, reverse).
2. Save this new permutation along with its total cost (so you also calculate the cost at the same time as saving the permutation. Where do you save? In a Tour structure array. Print out the permutation & its cost to the screen.

Then, after all tours are generated (which should be 40,320 tours), you find the lowest cost tour and print it out.

# Suggestions

Below I describe the functions that I created in my implementation of this program and their purpose. You do not have to have these functions but if you are struggling with how to implement the program, then you may find this helpful.

### main function

I have given you most of my main function. After the for loop to assign the subset of the country strings into countries array, I then call my **lexicographic** function, sending the countries array, the integer 8 which indicates the size of this array, the tourOptions array, and the matrix. Then, I print the solution by calling the **findLowest** function. Last, I delete all dynamically allocated variables/arrays.

### lexicographic function

You are given a sample program that implements the lexicographic permutation algorithm. You will have to modify the lexicographic function where at the end of the for loop when you have generated a new permutation and print it out by calling the **printStringArray** function, call the **saveTour** function, sending the tourOptions array, the string array which has the current permutation generated, and the matrix.

### saveTour function

The purpose of the **saveTour** function is to save the string array with the permutation to the Tour array (appending “US” to the front and end) and also to add up the weights of the edges in this permutation and save it to the Tour array as well.

### findLowest function

The **findLowest** function will just search through the Tour array and find the lowest cost tour. After the for loop which finds the lowest cost, it prints out the lowest cost amount and the string array containing the tour (permutation) that gave this result.

SAMPLE OUTPUT

The sample output for this program can be found in a text file named **SampleOutput.txt**.

# Submission

Zip all required files to compile & run your program (including flight.txt) and upload your submission to ilearn in an assignment folder named PROGRAM.

# Grading

|  |  |
| --- | --- |
| **20 points** | Program compiles without errors or warnings. **(20 points if it compiles, 0 if not)** |
| **20 points** | The program correctly selects the lowest-cost tour |
| **50 points** | The traveling salesman algorithm was followed and implemented correctly   * find all tours and add up each tour’s weight * find the smallest-weight tour of all tours |
| **10 points** | Programming Practices / Readable Code  (proper naming conventions, useful comments, consistent indentions in code) |
| **100 points** | TOTAL |