

# **Group Project**

## **World Airline Route Search CSCE 2110 - Foundation of Data Structures**

### **Introduction**

In this project, you will be designing and implementing algorithms to store and search graphs. World Airline (WA) flies to many destinations worldwide including: Moscow, Seoul, Tokyo, Hong Kong, and London. Complete detailed information regarding airline flight routes can be found at [/nfs/home/UNT/fi0047/public/csce2110/project](http://nfs/home/UNT/fi0047/public/csce2110/project) folder in CELL Linux Server. The flight.txt contains a “from city” and its destination cities that WA flights to. Note that the flight.txt is a synthetically generated datasets by a graph data generator, graphGen (graphGen.cpp). You may want to take a copy of the graph generator and generate a few more graphs with different number of cities to test your algorithms.

### **Your task**

Your task is to design and implement algorithms to answer the following questions:

1. I am in city “A”, can I fly to city “B” with less than x connections? Give me the route with the smallest number of connections or tell me there is no such a route.
2. Give me the route with the smallest number of connections from city “A” to city “D” through city “B” and “C”. (the order of “B” and “C” is not important). Or tell me there is no such a route
3. I want to start from city “A”, visit all the cities that can be reached and then come back to “A” using as less connections as possible. Give me a route or tell me there is no such a route. (note: once you come back to “A”, you do not go out anymore).
4. I am in city “A”, my friend John is in a different city “B”, and my other friend Ann is in yet another different city “C”. We want to find a city different from the three cities we are in to meet so that the total number of connections among three of us is minimized. Tell me the city we should fly to and the routes for us or tell me there is no such a city.

You will need to process the documents and store their content in the data structures that you selected. Next, for every question, you will search your data structure using an algorithm of your choice. You have the freedom to select the data structures and algorithms that you consider to be more efficient for the tasks. Of course, you will have to justify your decisions. No fancy user interface is expected (fancy interface may be counted towards extra credits though). Your program should take a few parameters with

the first as the number of the question (1-4). The rest of the parameters and output of the four questions (1-4) are described as follows assuming your program is called routeSearch:

1. usage: routeSearch 1 <city\_A> <city\_B> <num\_connection>  
sample output:  
    city\_A to city\_x to city\_y ... to city\_B  
    total connection: 4
2. usage: routeSearch 2 <city\_A> through <city\_B> and <city\_C> to <city\_D>  
sample output:  
    city\_A to city\_x to city\_C to city\_y ... to city\_B to city\_D  
    smallest number of connection: 4
3. usage: routeSearch 3 <city\_A>  
sample output:  
    city\_A to city\_x to city\_y ... to city\_z to city\_A  
    smallest number of connection: 4
4. usage: routeSearch 4 <city\_A> <city\_B> <city\_C>  
sample output:  
    You three should meet at city\_D  
    Route for first person: city\_A to city\_x ... to city\_D (3 connections)  
    Route for second person: city\_B to city\_y ... to city\_D (1 connections)  
    Route for third person: city\_C to city\_y ... to city\_D (0 connections)  
    Total number of connection: 4

You can choose to work alone or work with a team of three (max). If you choose to work in a team, you all will receive the same score for the project. However, you will need to explicitly specify each person's portion of work in the appendix of your written report. This information is used only when a student's final letter grade is on the boundary.

**Note:** if it takes too long for your program to finish the tasks, try to generate smaller graphs using the provided graphGen.cpp.

## **What to turn in**

There are three main parts in this project, all of them contributing to the final project grade.

1. You will have to write a project report (about 2-4 pages – single space – 12pt font) that includes:

- design issues, what are the problems and how you solve them
- data structures you use, the decision behind selecting them
- algorithms you employ, again with a justification of your decision
- particular emphasis should be placed on the running time of your algorithm, please show the asymptotic costs for each of your algorithm
- optimization issues: what could you do to further optimize your algorithm
- you need to specifically address the problem of scalability: would your implementation be efficient in the case of very large species collections for larger scale geographic area?
- any other remarks about your design and implementation

2. You will have to send in a fully working program, written in C/C++. We will test your algorithms extensively by generating graphs with different characteristics. It is mandatory that you include a README file, as detailed as possible, including compilation and running instructions. It is also mandatory that your programs are fully documented, that is they should include detailed comments on what is included in each file and what each method does. Create a zip file (**csce2110\_project\_team#.zip**) including all of your programs, report, README, etc. and upload it to PROJECT drop box in Canvas by due date and time.

3. In class presentation. You will have to prepare a short presentation, to last about 10-15 minutes (questions included), where you will present the design and implementation decisions you made in this project. Make use of examples, compare with other possible approaches, and use any other means you wish to make your point (that your design is the best). You should prepare PowerPoint slides and put them to a web accessible place or bring it on a USB pen drive. Project presentations will take place in class during the week of December 1, 2025.

## **Notes**

\* No late submissions are accepted!