

# Graph Searching

## Deadline Friday week 11

### Introduction

This exercise should be done individually. The deadline for this coursework is **16:30 on the Friday of week 11**. You should post the coursework in the submission box stapled to a cover sheet.

Parts of the coursework should be demonstrated in the lab to the lab helpers. You will need to tackle each part of the coursework in turn over a number of weeks to spread the load. The demonstration of all parts must be complete by the last lab session you attend in week 11.

This coursework is worth 50% of the total coursework mark for this module.

The coursework aims to reinforce your understanding of module material, specifically the following learning objectives:

- Gain an understanding of a range of graph classes and their use to represent realistic data.
- Gain further experience in object-oriented software engineering with a non-trivial class hierarchy: specifically selecting an appropriate class; reusing existing classes; extending existing classes.
- Using generic code: reusing generic graph classes, and parameterising a class with different types.
- You will also gain general software engineering experience, specifically downloading and using Open Source software, using a general method for a specific purpose, and issues with reusing existing code.
- Gain further experience with Java programming.

### Context

Your task is to use a graph to represent airline data, and to support searching. You should carefully test the code you write and may add to the graph to do so.

**N.B.** You should carefully test all of the code you write, generating new data files as necessary, and *include the result of your tests in the report*.

### Coursework Details

#### Part A: Download and Familiarisation with JGraphT

In this part you will download a personal copy of the Open Source `jgrapht` graph library. Having a single central copy of the library for all students on the course would save space, but it's a valuable experience for you to use an Open Source code repository.

## A.1 Graph Package Download & Setup

Download the Open Source jgrapht graph library from SourceForge by following the instructions on:

<http://jgrapht.sourceforge.net/>

The following instructions are for jgrapht-0.7.1 on a Unix machine using bash. Decompress the tarball, e.g.

```
linux22% gunzip jgrapht-0.7.1.tar.gz
```

Extract the tarball, e.g.

```
linux22% tar -xvf jgrapht-0.7.1.tar
```

Delete the tarball to save 6Mb!, e.g.

```
linux22% rm jgrapht-0.7.1.tar
```

Add the jgrapht demo and jar files to your class path at the end of your .profile file in your home directory. In the following command you should replace `yourlogin` with your login, and `yourpath` with the directory path to where you have installed jgrapht. That is, for Zedekia Zachary,

```
export CLASSPATH=/u1/cs2/yourlogin/yourpath/  
jgrapht-0.7.1/jgrapht-jdk1.5.jar:  
/u1/cs2/yourlogin/yourpath/  
jgrapht-0.7.1/src/org/ jgrapht/demo:.$CLASSPATH
```

might become

```
export CLASSPATH=/u1/cs2/zz24/java/cs2assignments/jgrapht/  
jgrapht-0.7.1/jgrapht-jdk1.5.jar:  
/u1/cs2/zz24/java/cs2assignments/jgrapht/  
jgrapht-0.7.1/src/org/jgrapht/demo:.$CLASSPATH
```

Execute your new .profile, e.g.

```
linux22% source ~/.profile
```

## A.2 Demonstration Programs

Go to the jgrapht source directory, e.g.

```
cd ../jgrapht/jgrapht-0.7.1/src/
```

Compile and execute the 1st demo program:

```
javac org/jgrapht/demo/HelloJGraphT.java  
java org/jgrapht/demo/HelloJGraphT
```

Execute other demo programs, e.g. PerformanceDemo - takes some minutes!

### A.3 View Javadocs

You will need to use the `jgrapht` javadocs to locate appropriate classes and methods. Browse to <http://jgrapht.sourceforge.net/javadoc/>

### A.4 Editing & Compiling Graph Programs

To edit a program

```
cd ../jgrapht/jgrapht-0.7.1/src/org/jgrapht/demo
```

Edit `HelloJGraphT.java`, adding a new edge to one of the graphs.

Return to the source directory, e.g.

```
cd ../jgrapht/jgrapht-0.7.1/src/
```

Compile and run your revised program

```
javac org/jgrapht/demo/HelloJGraphT.java  
java org/jgrapht/demo/HelloJGraphT
```

Congratulations, you're ready to start writing graph programs.

### A.5 Viewing Example Graph Programs

Often the easiest way to write a program is to *reengineer*, i.e. copy and modify, a similar program. More example programs are available in the `testsrc` directory

```
cd ../jgrapht/jgrapht-0.7.1/testsrc/org/jgrapht/
```

**Demonstration:** Part A.4 must be demonstrated and signed off during **week 8**, and you should aim to complete Part B.

[0]

### Part B: Representing Direct Flights

- i) Write a program to represent the following direct flights with associated costs as a graph. For the purpose of this exercise assume that flights operate in both directions with the same cost, e.g. Edinburgh  $\leftrightarrow$  Heathrow denotes a pair of flights, one from Edinburgh to Heathrow, and another from Heathrow to Edinburgh.  
**Hint:** Flights are directed, i.e. from one airport to another, and weighted by the ticket cost, hence use the `jgrapht.SimpleDirectedWeightedGraph` class.

**Hint:** There is a method  
`Graphs.addEdge(Graph, SourceVertex, DestVertex, Weight)`  
You should display the contents of the graph, and may omit the weights.

Edinburgh ⇔ Heathrow, Cost £100  
Heathrow ⇔ Amsterdam, Cost £120  
Heathrow ⇔ Boston, Cost £230  
Boston ⇔ Pittsburgh, Cost £80  
Boston ⇔ Montreal, Cost £110  
Montreal ⇔ Toronto, Cost £70  
Edinburgh ⇔ Pittsburgh, Cost £560  
New Delhi ⇔ Bombay, Cost £130  
New Delhi ⇔ Hong Kong, Cost £230

[10]

### Part C: Least Cost Connections

Extend your program to search the flights graph to find the least cost route between two cities consisting of one or more direct flights. **Hint:** use methods from the `DijkstraShortestPath` class to find the route. A possible interface for your program might be:

The following airports are used:

```
Edinburgh
Heathrow
Toronto
Boston
Pittsburgh
Montreal
Amsterdam
New Delhi
Bombay
1Hong Kong
```

```
Please enter number of start city
Edinburgh
Please enter number of destination city
Toronto
Airports on shortest (i.e. cheapest) path:
(Edinburgh : Heathrow)
(Heathrow : Boston)
(Boston : Montreal)
(Montreal : Toronto)
Cost of shortest (i.e. cheapest) path = 510.0
```

[15]

**Demonstration:** Parts B and C must be demonstrated and signed off during **week 9**, and you should aim to complete at least part D.

### Part D: Additional Flight Information

Start a new program operating on a graph containing the same flights as in part A, but including the following information about each flight. The flight number, e.g. BA345; the

departure time, the arrival time, the flight duration and the ticket price, e.g. £100. All times should be recorded in 24 hour hhmm format, e.g. 1830.

Use your imagination to populate your graph with sensible flights.

[7]

### Part E: Itinerary

Use the additional flight information to print itineraries for least cost journeys in a format similar to the following example. The key aspects are

- i) A sequence of connecting flights (with least cost)
- ii) A total cost for the route

An example itinerary for parts E & F might resemble:

```
Itinerary for Edinburgh to Toronto
Leg Leave      At      On      Arrive    At
1  Edinburgh 1030    BA345 Heathrow 1130
2  Heathrow  1400    BA657 Boston  1530
3  Boston    1800    AA652 Montreal 1930
4  Montreal  2200    AA216 Toronto 2330
```

```
Total Journey Cost = £910
Total Time in the Air = 930hrssee Part F
```

[15]

**Demonstration:** Parts D and E must be demonstrated and signed off during **week 10**, and you should aim to complete at least part F.

### Part F: Itinerary Duration

Extend your program to calculate the total time in the air, i.e. the sum of the durations of all flights in the itinerary. **Hint:** you'll need to write functions to perform arithmetic on 24 hour clock times.

[8]

### Part G: Alternative Extensions

Attempt at least three of the following extensions to your second program. **Your report must clearly indicate which extensions you implemented and demonstrate them.**

- i) Airline itineraries record arrival and departure times in local time for the destination airport, also in 24 hour format. Calculate the total journey time as the sum of the flight durations plus the sum of the changeovers.
- ii) Extend your program to calculate journey time for journeys that span more than one day, e.g. takeoff at 21:30 and arrive at 04:00.
- iii) Extend your program to allow connections between two flights only if the arrival time of the first flight is between 1 and 5 hours of the departure time of the second flight.
- iv) Extend your program to locate routes with the fewest number of changeovers. **Hint:** use a standard graph traversal algorithm, available in `jgrapht`.
- v) Extend your program in some other way. Carefully describe the extension in your report.

[15]

## General

|   |      |
|---|------|
| Program structure, comments and indentation:  | [10] |
| Demonstration   | [15] |
| Report with appropriate length, correct structure, lack of spelling and grammatical mistakes. | [5]  |

**Total Marks: 100**

## Submission Requirements

Electronically submit all of the files of code you've written using the instructions on the F28DA1 Vision page.

Your paper submission should include a **brief** report (max. 7 pages, excluding the appendices), prepared using the word processor of your choice, and containing the following sections.

### Section 1 Testing

This section *presents a description of test data and testing outcomes, and includes reasons why test data was chosen.*

- 1.1 Program 1 Part B: Representing Direct Flights**
- 1.2 Program 1 Part C: Least Cost Connections**
- 1.3 Program 2 Part D: Additional Flight Information**
- 1.4 Program 2 Part E: Itinerary**
- 1.5 Program 2 Part F: Itinerary Duration**
- 1.6 Program 3 Part G: Alternative Extensions**

In this subsection you should identify which extensions you have implemented, and give a demonstration of each.

### Section 2 Evaluation

As part of your submission you should include a reflection on your coursework (max. 1 page). Please make it clear if parts of your code don't fully work. If some of your work falls into this category then include some analysis of the problem and how you would tackle it given more time.

### Appendices

You should include clearly labeled listings of your Java code, e.g. labeled as

- Appendix A Program 1: Flights**
- Appendix B Program 2: AdvancedFlights**
- Appendix C Program 2: Flight Class**

...

Your code should contain comments that clearly identify parts A, B etc. For example:

```
*****  
* Part C: Least Cost Connections  
*****
```