Université d'Ottawa
Faculté de génie

École d'ingénierie et de technologie de l'information

uOttawa

L'Université canadienne
Canada's university

University of Ottawa
Faculty of Engineering

School of Information Technology and Engineering

Assignment 4 (7.5%) – revised with changes in red

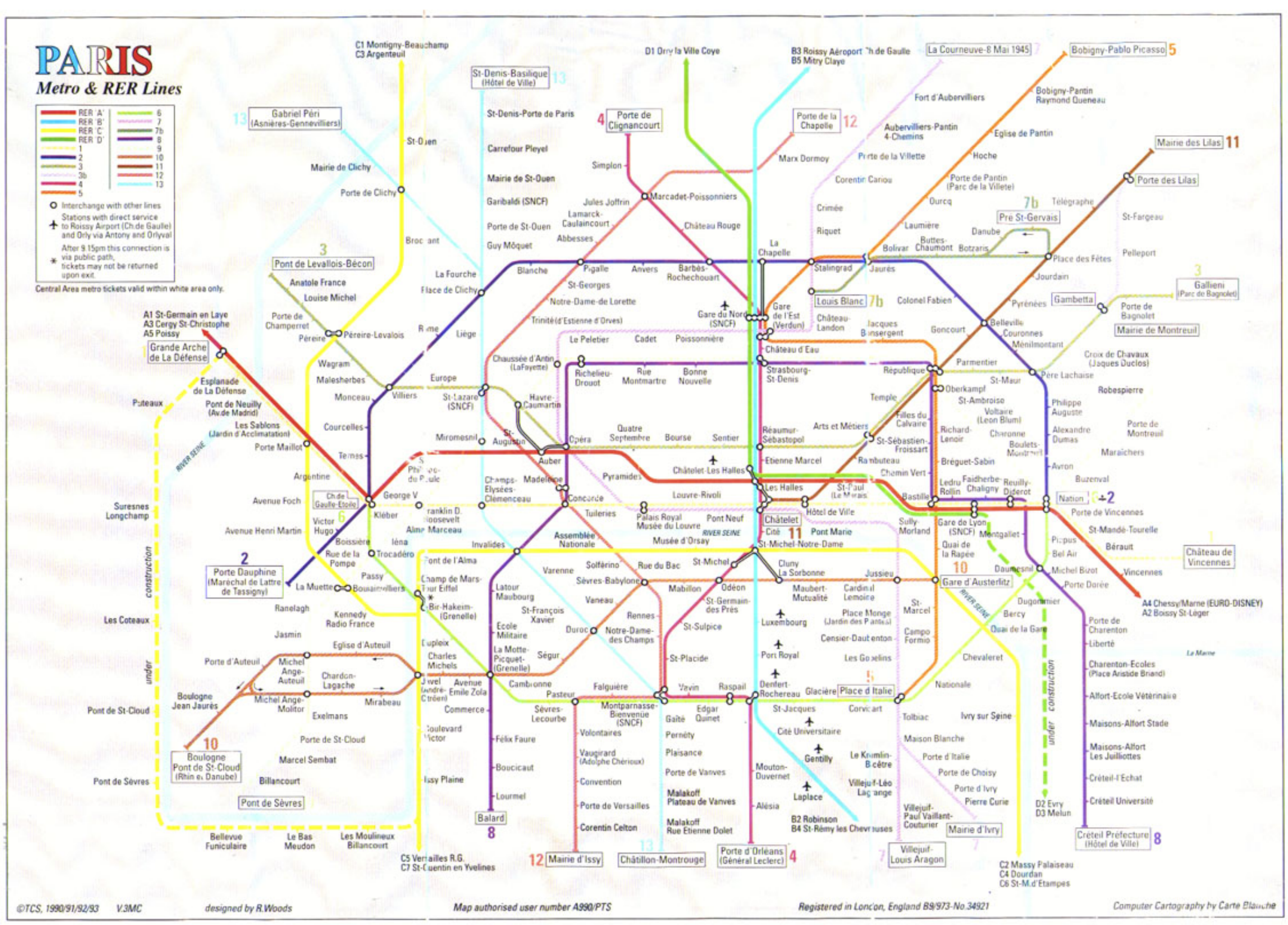
CSI2110

Submit to virtual campus by 23h55 December 5, 2017

(from 1 min to 24hs late receives 30% discount)

Paris Metro: Riding Paris Subway Network

The objective of this assignment is simply for you to find a shortest path in a real graph. For this, we will use the metro (subway) of the City of Paris as an example of a relatively complex network.
For the purpose of this handout the word "lines" is used to refer to subway lines, which in the picture below are lines of different colours.



The information required to build the graph corresponding to the Paris subway network is contained in the text file included here. This file is organized as follows:

- *The file starts by specifying two integers the number of vertices and the number of edges in the graph.*
- *The name of all metro stations is then listed and associated with a unique number. Note that some stations appear more than once when they are associated with more than one metro line*.
- *The symbol $ denotes the end of the subway station list*.
- *The reminder of the file lists the edges of our graph, i.e. the existing connections between the subway stations, and their weight which the time required (in seconds) to go from one station to the other*.
- *This graph is* **directed** *in order to take into account for some cases where the link is only one-way*.
- *In some cases, it is possible to walk in order to switch from one line to another. These cases are identified by the number -1 in place of their weight. For these cases, the weight must be substituted by the estimated walk time which will be a constant chosen by the program.*

Questions :

1. *You must read the file and create a graph using the representation of your choice.*

   i) You should create a static method `readMetro(filename)` which will read the data.
   ii) You should create a class `ParisMetro` which will contain graph of the Paris Subway.

2. Using this graph, we want you to produce a software tool that will allow to automatically:

Note: when we mention station in items below we mean the station number.

i) Identify all the stations belonging to the same line of a given station. ~~These stations must be listed in travel order (including the specified station) from one end of the line to the other end of the line (the direction they are listed is not important).~~

ii) Find the shortest path between any two stations, printing all the stations of the path in order, and printing the total travel time.

iii) Find the shortest path between two stations when we have the information that one given line is not functioning (the line is identified by one of its endpoints). The same type of information will be printed as in ii).

To specify a station, you must use its number. You must use 90 secs as the constant estimated time to transfer from one line to another (time to walk between locations labeled with weight -1) .

Your program:

In order to run your program from the command line, the call should be as follows:

```
java ParisMetro N1 N2 N3
```

with `N1`, `N2` and `N3` being numbers identifying subway stations.

- If only `N1` is specified the program must answer question 2-i) where `N1` is the given station identifying the line to be printed.
- If `N1`and `N2`are specified the program must answer question 2-ii) where `N1` is the departure station and `N2` is the arrival station.
- If `N1, N2` and `N3` are specified the program must answer question 2-iii) where `N1` is the departure station, `N2` is the arrival station and `N3` is the endpoint of a broken line (line that is not functioning).

You must submit your Java code plus a document/report (pdf file) describing:
- Your chosen data structure (classes, attributes and methods).
- High level description of the algorithms you use to answer the questions in part 2.
- Examples of outputs for answering the questions of part 2.

Marking criteria:
30% Description of the algorithms (clarity of explanation, good level of abstraction)
30% Correctness of the solution
40% Programming and design
(comments, headers, clarity and code readability, conciseness of methods, did you design/use a good algorithm? did you define the right methods? did your algorithms solve the right problem? Is your solution unnecessarily complex? etc)

Important remarks:

For this assignment, you can use any code you find (online, textbook or from our labs) that can help you solving this problem. However you must give a reference to any external resource you used (e.g. provide the links where you download the code, or other references used). Even if you modify a code after, you have to specify that you used certain code or certain library as a starting point, even if you just consult it to guide you in your design. Failure to provide adequate references will be considered academic fraud with serious consequences.

In the same way, since this assignment is individual, you must make sure that nobody uses your code to write their solution. You can discuss general ideas with friends, but do not share or show specific parts of your code: exchanging source codes also constitutes academic fraud. Note that all submitted software will be subject to a plagiarism detection software that measures similarities in the program structure.

Reference

Assignment inspired by :
Michel Couprie, Gilles Bertrand, ‘Graphes et algorithmes - TP 3’,
https://perso.esiee.fr/~coupriem/Graphestp3/graphestp3.html, accessed Nov 14, 2017.