Institut Superieur
De'Electronique D'Paris

PROJECT ON ADVANCED ALGORITHMIC AND PROGRAMMING

1. INSTRUCTIONS

This project has to be made by groups of 2 or 3 students. There is no possibility to work alone.

1.1 OBJECTIVE

The main objective is to use the algorithmic and programming tools treated in the lectures and tutorial courses in a real graph.

The specific objectives of the project are:

1. Initiate the students to the manipulation of graph data, from the data collection to the graph construction.
2. Implement algorithms studied in the lectures and tutorial courses for the calculation of shortest paths in real data.

1.2 EVALUATION CRITERIA

The project notation is 30-40% of the final mark. The presentation and quality of the final report as well as the oral defense will be taken into account.

1.3 THE FINAL REPORT

Particular attention will be paid to the presentation of the final report. The language of the report should be English. In any case, the quality of writing will be appreciated. The sentences must be clear, explicit and well understandable.Thereshould be an abstract and introduction (which explains the structure of the report) inthe report.

The final report must also contain a cover page, a table of contents, the body of the report explained in the following sections (results, figures, tables, etc.), the conclusion and the appendix for the code. The methods of using the tools, and libraries should be explained as well (No need of explanations about the libraries and tools).

The number of pages should not exceed 25 pages and 10 pages for the appendix (in total 35 pages).
The cover page must contain the first name, last name and the student identification number of all the authors.

Institut Superieur De'Electronique D'Paris

1.4 THE DEFENSE

An oral defense will be held.
The defense will last about 15 minutes per group and it will consist in about 10 minutes of presentation plus 5 minutes of questions.

1.5 DELIVERY OF THE REPORT

In moodle, you will find a deposit box named Project-Reports-box. The final report must be uploaded in this deposit box. If you want to upload more than 2 files compress all of them in only one file in zip format. The final report must be sent in pdf format. The file names must be as follows:

LastNameStudent1_LastNameStudent2_ LastNameStudent3.pdf.

Just one deliver per group must be done.

2. COLLECTION OF DATA and CONSTRUCTION OF THE GRAPH

The purpose of the project is to calculate the diameter (the longest of the shortest paths) of the Parisian subway graph. You will calculate the shortest paths in two types of graphs:

1.6 Unweighted graph

First of all, you will consider the subway as an unweighted graph. You will get the data to build the graph from the RATP site web:

https://data.ratp.fr/explore/

Check for the data available in the link : *Offre transport de la RATP - format GTFS read more about GTFS in the other document.*

Given this data, build the unweighted graph where:

- A vertex is a subway station and
- there is an edge between two vertices if there is subway path between them.

The instructions given in the tutorial course Number 3: Bonus questions, will be very helpful for this part.

On http://vasyenmetro.com/?lang=en
On Chrom => View => Developer Tools => NetworkReload The page.
Filter by Xhr (Ajax Request)- this is what we want:
http://vasyenmetro.com/data/reseau.json

1.7 Weighted graph

Consider the graph built in the previous section, you will add weights to the edges in order to convert it into a weighted graph. This will make this study more realistic.

Each weight will represent the distance between two subway stops. To calculate the distances, you will consider the geographical position of each stop. You will find, for each subway line, a file named "stop.txt" which contains the position of each stop. The columns "stop_lat" and "stop_lon" describe the position coordinates. For example, for the stop Nation of line 1:

stop_name	stop_lat	stop_lon
Nation	48.84811123157566	2.3980040127977436

The formula used to calculate the distances will be the Euclidean distance between two points. Therefore, you will do abstraction of the shape of the paths by considering them as straight lines.

3. CALCULATION OF SHORTEST PATHS

For finding the shortest paths between all pairs of nodes:

- For the unweighted graph, you will use the BFS for shortest paths algorithm
- For the weighted graph, you will use the Dijkstra algorithm.

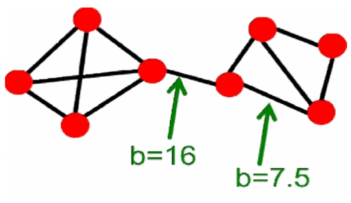
For both cases, the purpose is to find the diameter of the subway. Once you calculated the diameter, give the path (containing subway stations names) as well as the total length of the path and of each sub-path. For example,

Longest path: 1 => 5 => 6
Lengths of sub-paths: 1-5: 3m; 5-6: 2m
Total length of the path: 5 m

4. SHORTEST PATHS FOR GRAPH CLUSTERING

In a connected graph, clusters are group of vertices densely connected. One well known criterion to detect clusters in a network is to cut (remove) edges with high betweenness.

Roughly, the edge betweenness is the number of shortest paths between pairs of nodes passing over that edge. For example, consider the following graph:



There are 16 shortest paths passing through the edge in the middle. By removing this edge, we find the two clusters, two subgraphs densely connected. If there is more than one shortest path between a pair of nodes, each path is assigned equal weight such that the sum of the total weight of all of the paths is equal to unity[1].

In this part, the purpose is to detect the edges that lay "between" clusters. To this end, you will have to identify the edges with the highest betweenness in order to remove them and to reveal the underlying clusters. Finally, once the clusters identified you will interpret them.

5. Some help :

To create your graph you need to parse data from Json. To parse data from Json in java there are many libraries. The highlighted ones are the common ones :

- jackson
- genson
- fastjson
- gson
- org.json
- javax-json (from Oracle)
- json-io
- flexjson

[1] That is why the edge-betweenness of the edge on the right is 7.5

- boon
- json-smart
- johnzon
- logansquare
- dsl-json
- json-simple
- nanojson
- moshi
- tapestry
- jsoniter
- minimal-json
- mjson
- underscore-java

Below are examples for some of these libraries :

For the sake of the example let's assume you have a class Person with just a name.

```
private class Person {
    public String name;

    public Person(String name) {
        this.name = name;
    }
}
```

Google GSON (Maven)

A great JSON serialisation / de-serialisation of objects.

```
Gson g = new Gson();

Person person = g.fromJson("{\"name\": \"John\"}", Person.class);
System.out.println(person.name); //John

System.out.println(g.toJson(person)); // {"name":"John"}
```

Update

If you want to get a single attribute out you can do it easily with the Google library as well:

```
JsonObject jsonObject = new JsonParser().parse("{\"name\": \"John\"}").getAsJsonObject();

System.out.println(jsonObject.get("name").getAsString()); //John
```

Org.JSON (Maven)

If you don't need object de-serialisation but to simply get an attribute, you can try org.json (or look GSON example above!)

```
JSONObject obj = new JSONObject("{\"name\": \"John\"}");
```

```
System.out.println(obj.getString("name")); //John
```

Jackson (Maven)

```
ObjectMapper mapper = new ObjectMapper();
Person user = mapper.readValue("{\"name\": \"John\"}", Person.class);

System.out.println(user.name); //John
```

I suggest to work with this library and watch the following video to learn it.

http://mrbool.com/jackson-2-converting-java-to-and-from-json/34355

You can read about the libraries performance below :
https://blog.overops.com/the-ultimate-json-library-json-simple-vs-gson-vs-jackson-vs-json/

6. BIBLIOGRAPHY

Do not forget to include the bibliography in your report