Institut Superieur
De'Electronique D'Paris

PROJECT ON ADVANCED ALGORITHMIC AND PROGRAMMING

1. INSTRUCTIONS TOPIC 1

This project has to be made by groups of 2 or 3 students. There is no possibility to work alone (if you could not make a group send me an email).

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

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.
The cover page must contain the first name, last name and the student identification number of all the authors.

1.4 THE DEFENSE

An oral defense list will be published three days before presentation day. The defense will last about 20 minutes per group and it will consist in about 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 THE NIGHT before your defense (means by 13th of Jan at 23:59). 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. Note: Just one deliver per group must be done.

LastNameStudent1_LastNameStudent2_ LastNameStudent3.pdf.

2. COLLECTION OF DATA and CONSTRUCTION OF THE GRAPH

2.1 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 GTFS data in this website and you fill the file in Moodle with your groups member and city selected. The same city cannot be selected twice and you cannot select Paris:

https://code.google.com/archive/p/googletransitdatafeed/wikis/PublicFeeds.wiki
More information about GTFS :
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5952869/
https://github.com/rombdn/ratp-gtfs-to-json

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.

2.2 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.

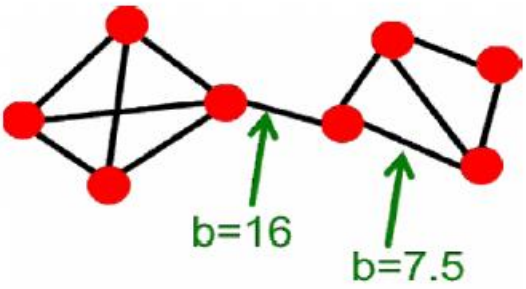
3. CALCULATION OF SHORTEST PATHS

For finding the shortest paths between all pairs of nodes:

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

4. SHORTEST PATHS FOR GRAPH CLUSTERING

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. That is why the edge-betweenness of the edge on the right is 7.5.

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. The method should take requested maximum number of clusters as an input. The algorithm stops when there are no more edges to remove or if the algorithm has reached the requested maximum number of clusters.

Institut Superieur
De'Electronique D'Paris

5. Ranking *K*-quickest (loopless) paths

5.1 Introduction:

Digraph *D* = (*V;A;T*) composed of a set *V* of *n* nodes, a set *A* of *m* directed arcs and *T* set of *T* time steps. t_{ij} *denotes* travel time/delay of arc (*i; j*) and *cij* represents capacity of arc (*i; j*).
An *o-d* path is an ordered set of arcs *{*(*i*0*; i*1*);* (*i*1*; i*2*);* (*i*2*; i*3*); : : : ;* (*ik*−1*; ik*)*}* such that *i*0 = *o*, *ik* = *d* and $(il;\ il+1)\ \in A;\forall\ l\ =\ 0;\ldots;\ k\ -1.$

Part1:

5.2 lazy version of the Chen's algorithm

The so-called lazy version of the Chen's algorithm is used for solving the *K*-quickest loopless paths problem as presented in the works of Pascoal et al. [1-2]. The problem requires the identification of best *K* elementary quickest paths provided in an increasing order of their transmission time. This is the time required to transship an item from source to sink.
The idea underlying the method is to reduce the problem to several computations of a state-of-the-art *K*-shortest loopless path algorithm and to alternate path identification steps to path selection steps till a total of *K* paths are identified. A pseudocode of the algorithm is provided in Algorithm 1.

Algorithm 1 Lazy version of Chen's algorithm

Input: $\mathcal{D} = (\mathcal{V}, \mathcal{A}, \mathcal{T})$, (o, d);
Output: the K quickest paths in $\mathcal{D}$;

1: **Initialization**
a. Order arc capacities: $c_1, \ldots, c_r$;
b. $\mathcal{A}' \leftarrow \mathcal{A}$;
for $l = 1, \ldots, r$:
$\mathcal{A}' \leftarrow \{(i, j) \in \mathcal{A}' \mid c_{ij} \geq c_l\}$;
$L[l] \leftarrow$ the o-d shortest loopless path in $\mathcal{D}' = (\mathcal{V}, \mathcal{A}')$, $\emptyset$ otherwise;

2: **Find K-Quickest Paths**
$k \leftarrow 0$;
while $k < K$ **and** $\exists\, l$ s.t. $L[l] \neq \emptyset$ **do**:
a. $p \leftarrow L[l]$ s.t. $L[l] = \text{argmin}_{i=1,\ldots,r}\, T(L[i])$;
b. $\mathcal{A}' \leftarrow \{(i, j) \in \mathcal{A} \mid c_{ij} \geq c_l\}$;
$L[l] \leftarrow$ the next o-d shortest loopless path in $\mathcal{D}' = (\mathcal{V}, \mathcal{A}')$, $\emptyset$ otherwise;
c. **if** $k = 0$ **or** $p \notin \{\bar{p}_1, \ldots, \bar{p}_k\}$ **then** $k \leftarrow k + 1$, $\bar{p}_k \leftarrow p$;

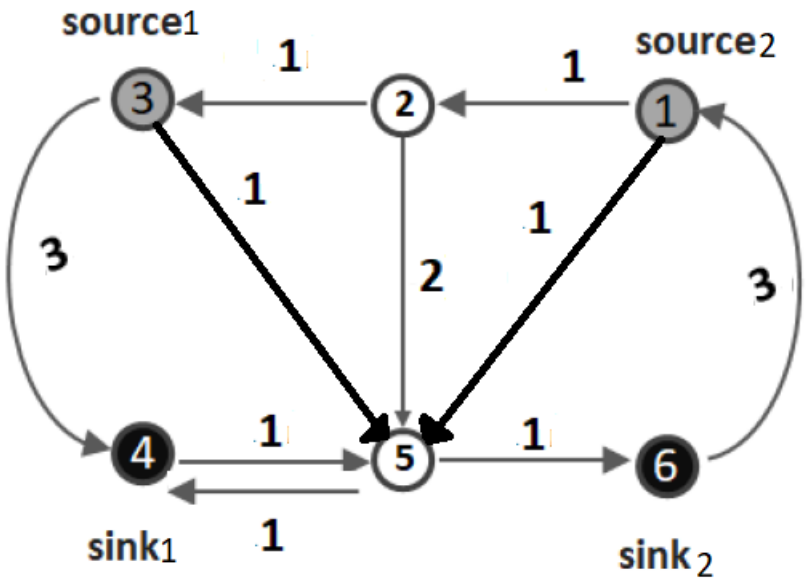
3: **Return** $\{\bar{p}_1, \ldots, \bar{p}_K\}$;

Institut Superieur
De'Electronique D'Paris

It requires as an input the dynamic digraph and the given o-d (source-sink) to be transshipped. In the initialization phase, Step 1, it lists all the arc capacities in an increasing order excluding potential repetitions, hence obtaining *r* different values with $r \leq m$. Then, for each of these values it constructs a subgraph by considering only arcs whose capacities exceed or are equal to the given value set as a lower bound and in the so-constructed digraph it identifies the shortest loopless path w.r.t. arc travel times, Step 1*b*. The generated shortest paths are stored in the array *L*.

In Step 2, the algorithm iteratively selects the best shortest path in the array w.r.t. the transmission time and replaces it by the next shortest loopless path found in the related subgraph. At this stage any algorithm for solving the *K* shortest loopless path problem can be employed, such as the Yen's algorithm [3] or Katoh et al. [4]. The procedure stops whenever *K* different paths have been selected or no more shortest paths are available.

- **implement and test the algorithm on the following graph (K=2)**



Part 2:
Apply the algorithm 1 on the chosen city graph and ask for k (from user) and suggest K passible path.

References

[1]Pascoal, M., Captivo, M., Cl__maco, J., 2006. A comprehensive survey on the quickest path problem. Annals of Operations Research 147 (1), 5{21.
[2] Pascoal, M., Captivo, M., Cl__maco, J., 2007. Computational experiments with a lazy version of a k quickest simple path ranking algorithm. TOP 15 (2), 372{382
[3] Yen, J., 1971. Finding the k shortest loopless paths in a network. Management Science 17 (11), 712-716.
[4] Kleinberg, J. M., 1996. Approximation algorithms for disjoint paths problems. Ph.D. thesis, Massachusetts Institute of Technology.