JAFFALI Hamza - AG44 Project 2 - 18 november 2014

Abstract

This document constitutes the report of the second project as a part of the UV AG44, a project named "Ski Resort". We consider a ski station, composed of differents points, that can be reached from other points using station's routes. We chose to represent this station by means of a graph, in which every point will be a vertex, and every route an edge. The graph must be connected, because we suppose that one can always reach every point in the station from a specific starting point. The distance between two points in the station is given by the differene of their altitude.

1 Data architecture

We coded this program in Java. So 5 classes have been defined accordingly : Map, Route, Point, FileRead and Main. The diagram class is present in appendix.

A plan is the map of the ski-station, and it is modelized as a graph, so it contains an ArrayList of Point and Route. A point represents a specific location in the station, so it's a vertex of the graph, and it is characterized by his name and number, and his altitude. A route represents the road to take in order to reach a point from another, so it's an edge of the graph, and it is characterized by a name and a number, by the two points that it connects and by the distance betwwen these points. So we don't have any adjacency structure to list all the points reachable from another : we store all points (vertices) and all routes (edges), and so we can traverse all the graph.

2 Fastest path

We have modelized the plan as a graph, where each route is an edge. If we introduce weights for each edge, considering that the edge's weight will be the time spent between two vertices, the problem of searching the fastest path between two points becomes a shortest path problem.

For resolving this problem, we start by determining the time spent in each route. In order to do this, the caracteristics of each route given in the subject permit us to determine the time cost, knowing the exact distance of each route also.

Then we chose to apply the Bellman-Ford algorithm, to find the shortest path between the two specific given points. We chose this algorithm because we only need to know all the edges of the graph, and in this case we have stored all the routes in the plan, so it was the simpliest to implement. The complexity order of this algorithm is of $O(nm)$ with n the number of points, and m the number of routes.

3 Reachable points

The problem of determining all points reachable from a specific starting point can also be transposed in a classical problem of graph theory : graph traversal. Besides, it's a kind of particular traversal, because we chose to restrict the access to few skying routes, according to the skying level of the program user.

But we have chosed to improve and to add some usefull features to this solution, by inform the user what are exactly the fastest paths to all reachable points. And this lead us to reuse, or to adapt the fastest path algorithm to respond to this problem, with the wanted features.

So we run once the Bellman-Ford algorithm in order to fill all the time costs in each vertex. But instead of apply n times the main loop of the algorithm for each edge, we only update time costs when we encounter a route that can be reached by the user. At the end of the Bellman-Ford algorithm, all vertices with time cost at infiny cannot be reached by the starting point. Then we display all other points, with the corresponding fastest path. This is a way to know all reachable points without using DFS of BFS, and with additional informations such a fastest path to these points, with paths and corresponding time costs.

4 Appendix

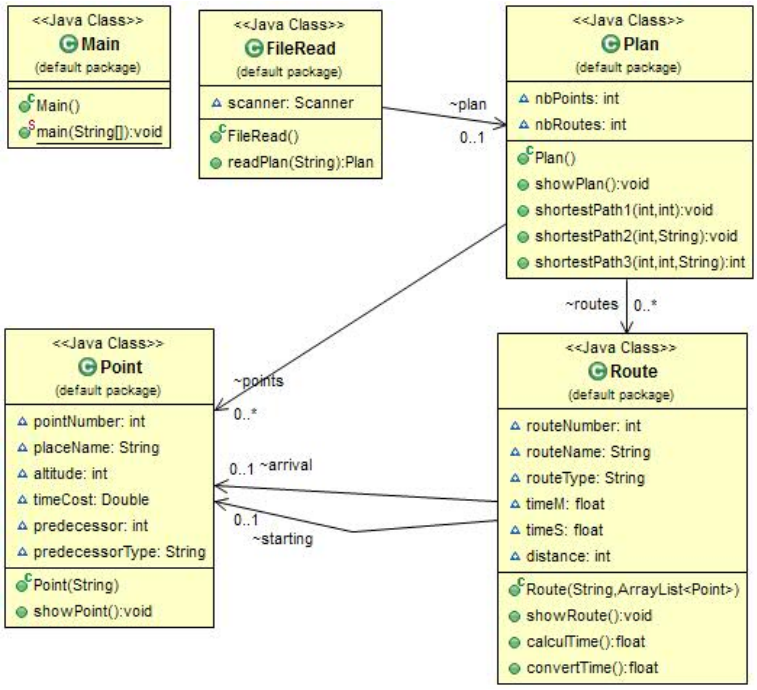

Figure 1: Diagram class of the program

5 Examples

```
   Select your mode :
1. Fastest trip between two Points
2. All reachable Points from one, with specific level of skying
Your choice :
2
Can you use all the pistes of the station ? (yes/no)
no
Write in the same line all the type of pistes you can't use with your level :
N R
What is the starting point ?
12

The fastest path between Point 12 and Point 5 is [12 -> 5] and it costs 907.2 second(s), ie 0 hour(s) and 15 mi
You should take these types of route : TPH.

The fastest path between Point 12 and Point 6 is [12 -> 6] and it costs 51.2 second(s), ie 0 hour(s) and 0 minu
You should take these types of route : KL.

The fastest path between Point 12 and Point 12 is [12] and it costs 0.0 second(s), ie 0 hour(s) and 0 minute(s)
You should take these types of route : null.
```