Project : Optimizing the cost of installation of charging stations in a community of agglomerations

The program solves and displays, using different algorithms, what could be the minimum number of charging stations needed for a community of agglomerations to minimize the cost of electrification of the road network

This project, fully functional, was carried out as a part of our third licence year in CS for the Advanced Programming and Applications course. The program solves and displays, using different algorithms, what could be the minimum number of charging stations needed for a community of agglomerations, for each city to have one or to be directly connected to a city possessing one. it does the following:

*takes a user's file as an argument that contains a CA with or without a solution (if the file doesn't contain any, using a standard one to begin with) for the placement of charging stations.

*the progam proposes to the user an interface with multiple choices:

-the first one permits to solve the problem manually.

-the second to solve it automatically through implemented algorithms. *refer to the 'Understand how the algorithm work' for more details

-the third to save the solution. he then can enter the filename he wants to save the solution to, and if it already exists, he can decide either to create a new version while keeping the old one, or replace the old one

-the fourth to represent it graphically(using javafx).

-the fifth to stop the program.

the two proposed algorithms to solve the problem are those:

Understand how the algorithms work

The `resolutionAlgo2` method works by randomly picking towns and deciding whether to add or remove a charging station there. After each change, it checks if the overall setup gets better. It keeps doing this until it can't make things any better, suggesting the best arrangement of charging stations has been found.

The `resolutionAutomatique` method is an algorithm that strategically removes charging stations from a network of towns based on their connectivity. First, it creates a list of towns and a matrix representing the roads between them. Then, it counts the number of neighboring towns for each town. Using this information, the algorithm starts removing charging stations from towns with the fewest neighbors and gradually moves to those with more neighbors. This process is done under the condition that removing a charging station still keeps the network efficient. The method continues this way, prioritizing the removal of stations from less connected towns, thereby aiming to maintain network efficiency while reducing the number of charging stations.

This is what our program looks like

during the execution:

.

.

.

.

.

.

.

.

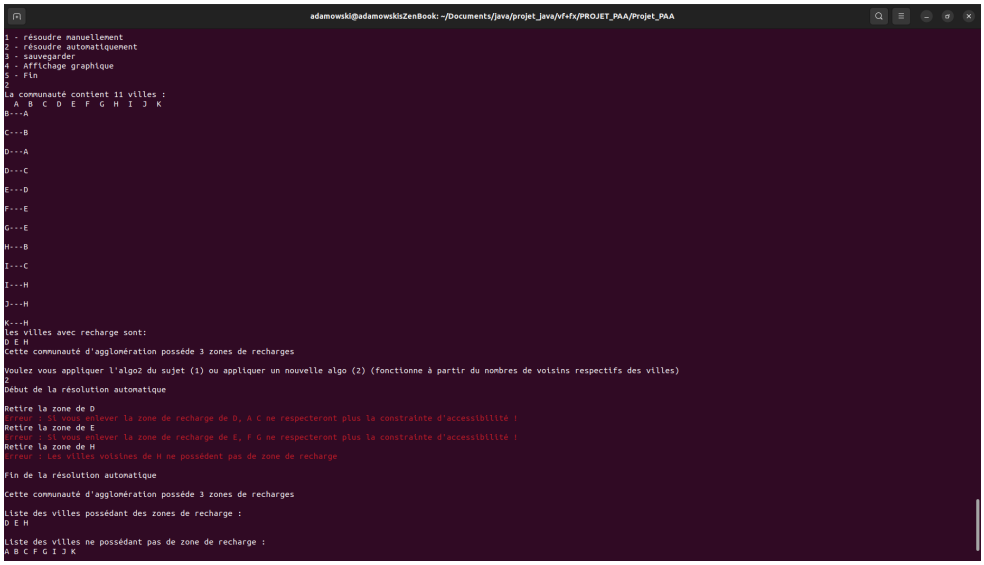
.

.

.

.

.



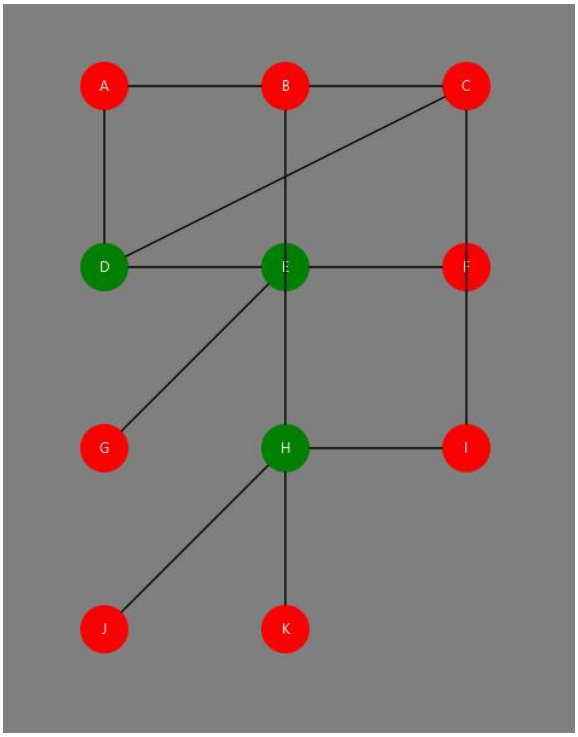
for option number 4, how it represents the solution graphically:

.

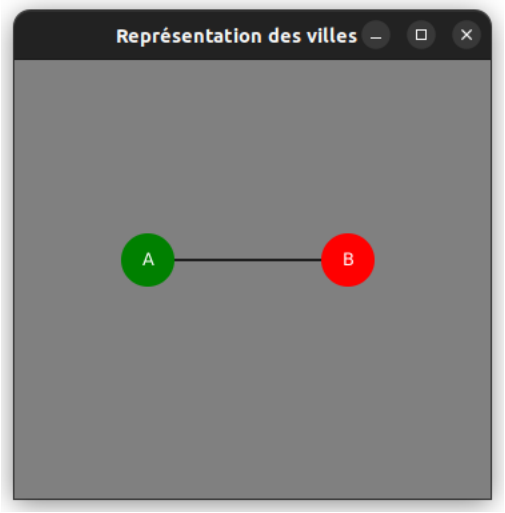
.

.

.



Legend for the representation



How to install and run this project

-first, make sure you have a javaFX installed on your computer, and that it's compatible with the java version you're running. Link for javaFx -Then, clone the repository or download it and unzip it at the location of your choice

To run on Eclipse:

-In Eclipse, add a library to the project classpath

-In the user library, create a new library that you can call JFX 11 for exemple

-Add the jar archives present in the project directory in the Lib directory of the SDK

-You can confirm. The new library should be present in the classpath of the project

-To run the program, go to the class Menu.java:

Run --> Run Configurations --> Arguments and add those arguments to the VM : "--module-path /path/of/the/sdk/lib --add-modules=javafx.controls." make sure the "Use the -XstartOnFirstThrea" box is unchecked

Run --> Run Configurations -> Program Arguments --> add the path of the file containing the data of the community of agglomerations.

-You can now run the class Menu.java.

To run directly from the bash terminal

-After that, place yourself at the same level of the src and bin (from the repertory where you extract the project, the path is "PROJET_PAA/Projet_PAA")

```
For linux and Macos users:

* give the permission to run the script with the"chmod +x run.sh" command.
* run the script through "./run.sh"
* give the path to your javafx SDK. for example:"/home/user/javafx-sdk/lib"


For Windows users:
*run the script with : "run_windows.bat"
or
* just double click on "run_windows.bat"
    then
*give the path to your javafx SDK. for example: "C:\path\to\javafx-sdk\lib"
```

*notice that as for any java program, you need to have java and the sdk downloaded correctly. So if during the execution of the script you run into this type of error "the operation couldn't be completed. unable to locate a java Runtime that supports javac." you may need to install the SDK correctly. You can do that from the oracle official website website and by choosing the plateform corresponding to yours

**To run the program, you can use just the name of the file (without putting it's path first) if you place it at the same level of src and bin folders. **for testing purposes, you can use the "ville.txt" already present in the project to try with a predefined CA and solution

*feel free to try the different options of your program and enjoy !

Some small known issues

in the description of our project, all the roads between two cities are standard, of lenght 1. but in a more practical definition, taking it count the real world, roads are far from being the same length and this would add complexity to our project

to save or to represent our project, the user needs first to try to solve the problem manually our with the help of one of our algorithms.

Even tho this isn't really an issue, we could change this for it to be solved automatically if the user wants to save or represent the solution without applying a solution, it's more of a developpment choice

Another problem we faced is the fact that when you run the graphical representation the first time, and then close the window, you can't re-run the representation without re-running the program with a new start, so don't forget to save your solution

Tools used during development

- VsCode - code editor
- Eclipse - code editor

Auteurs

Zaidi Rayan

Mougamadou Muzzammil

Guelai Adam