Programming Project I

Individual Route Planning Tool

DA 2024/2025 Instructors Team

Departamento de Engenharia Informática (DEI)/Departamento de Ciências de Computadores (DCC)
Faculdade de Eng. da Univ. do Porto (FEUP)/Faculdade de Ciências da Univ. do Porto (FCUP)

Spring 2025

***Due Date:** March 30, 2025, at 23:59 (PT time)*

1. Objectives

This first programming assignment aims at exposing you to realistic implementation of algorithmic solutions presented in class and in particular to the **greedy algorithmic approach** in the context of shortest-path problems. Key objectives include:

- Developing teamwork skills by working in groups of **2-3 students (3 preferred)**, fostering interpersonal and project management skills.
- Conceive and implement a simple path-planning tool.
- Preparing and delivering a **concise demo presentation** to showcase the functionalities and interfaces developed, while enhancing your ability to prioritize and communicate essential information effectively. This will require you to be **succinct**, **time-conscious**, and focused on what is essential (and what is not).

This document describes the motivation of this project, the expected interface followed by a description of the problem statement and description of the demo you are expected to present. The problem statement includes a description of each task (alongside the corresponding grading). Lastly, we provide specific turn-in instructions you need to follow. **Recall, the deadline is March 30, 2025 at 23:59.**

2. Problem Motivation

In this project, you will create a path-planning tool to assist with urban navigation, similar to GPS-based systems like Google Maps. However, your tool will also **emphasize environmentally sustainable and resilient mobility alternatives**. The tool should support various functionalities, including identifying the fastest and a second-fastest independent route, planning routes that exclude specific nodes or segments (restricted routes), and creating environmentally-friendly routes that combine driving and walking with parking options. While addressing these challenges, the tool must accommodate potential constraints, such as the unavailability of certain routes, ensuring practical and versatile solutions for diverse mobility needs.

3. Problem Data and Interface

To make this problem more realistic, you are provided with an idealized dataset representing the **Urban Map of a City**. The dataset describes the layout of streets (assumed to be two-way) divided into segments. Each segment includes the average time required for both **driving** and **walking**.

The urban map can thus be structured as a **weighted undirected graph** where edges represent the street

segments that connect nodes that represent street intersections. Each edge is associated with **two integer values** representing the driving and walking time required to traverse them, respectively. Each node is also associated with a value indicating the existence of a parking lot at that particular place. You can assume that the traveling time between nodes is independent of the time of day and day of the week and that there is always available parking capacity at the nodes where there is a parking lot. Also, there are segments where you can only walk, but assume that it is always possible to walk in a segment where you can also drive. A dataset that represents an instance of this problem is available in the ***Project1Data.zip*** file attached to this project description. Figure 1 shows a simplified example of a possible urban map.

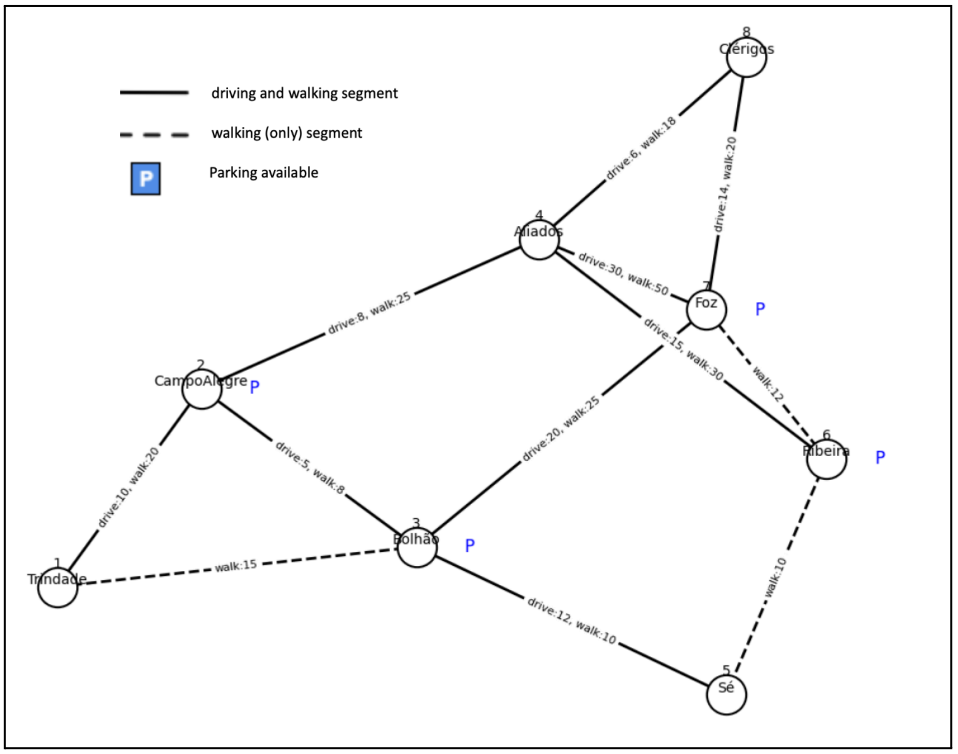


Figure 1. Simplistic example of *Project1Data.zip*.

The provided dataset is given in the form of csv (comma separated values) files, as described below.

- **File Locations.csv:** contains the information regarding the various locations, or points, in the urban environment (L). The first line includes the headers: **`Location`** (name of the urban location), **`Id`** (unique identifier of that location), **`Code`** (simple code that could be a key in another table) and **`Parking`** indicating the existence (value 1) or absence (value 0) of a parking facility at the location.

```
Location,       Id,    Code,  Parking
Trindade,       1,     P1,    0
CampoAlegre,    2,     P2,    1
```

- **File Distances.csv:** contains the information regarding the travelling time (or distance) between two locations, in the two modes of mobility, in this case, driving and walking (a future version could also include bike riding). The first line includes the headers: **`Location1`** (the **`Code`** of the start location of a segment), **`Location2`** (the **`Code`** of the end location of a segment), **`Driving`**

(the duration in minutes required to cover the distance of the segment when driving), **`Walking`** (the duration in minutes required to cover the distance of the segment when walking).

```
Location1,      Location2,   Driving,     Walking
1,              2,           1,           10
4,              5,           2,           30
```

Whenever you cannot drive on a specific segment, the corresponding driving time is indicated by an `X` character, which you then need to internally convert into a suitable representation in your data structure to be used by your algorithm.

4. Problem Statement and Organization of Work

To facilitate your work, we have structured the functionalities that you are expected to develop as follows.

4.1. Basic Application Definition and Set-Up

[T1.1: 1.0 point] Develop a Route Planning Analysis Tool. The first task will be to create a simple command-line menu that exposes all implemented functionalities in a user-friendly manner. This menu will also serve as a key component for showcasing the work you have developed in a short demo at the end of the project. When displaying the results, show all relevant details, including the full sequence path points from origin to destination. The program should handle exceptional cases/errors properly.

[T1.2: 1.0 point] Read and Parse the Input Data. You will need to develop basic functionality (accessible through your menu) to read and parse the provided data set files. This functionality will enable you (and the eventual user) to request information such as the start and end points of your trip, the alternative independent route, the locations and/or segments to exclude, and the environmentally-friendly routing parameters.

Behind the menu interface (T1.1), you are also required to implement batch mode functionality. While the menu can be configured freely to showcase all features during the presentation, the batch mode must adhere to a fixed input format (via **input.txt**) and output format (via **output.txt**) as specified for each task in the project. Input/Output files with an invalid structure are to be ignored.

With the extracted information, you should create one (or more) appropriate data structures upon which you will carry out the requested tasks. **This data structure must be based on the data structure described in the TP lectures for graphs, for which it may make a small number of modest additions.** You must use the provided graph structure as the primary representation of the urban environment, and you may use other auxiliary structures as needed to facilitate your implementation. Any alterations made to the original graph should be clearly justified in your presentation, ensuring that your modifications are sensible and enhance the application of the required algorithms.

[T1.3: 2.0 points] Documentation and Time Complexity Analysis. Include documentation for all implemented code using Doxygen. This should indicate the time complexity for each of the most relevant implemented algorithms.

4.2. Route Planning Functionalities (driving only)

For this part of the project, consider the following assumptions:

- The origin and destination can be adjacent or non-adjacent nodes.
- The origin and destination can be parking or non-parking nodes.

- The routes calculated will be for driving only.

[T2.1: 3.0 points] Independent Route Planning:

- Determine the best (fastest) route between a source and destination.
- Identify a best alternative independent route, ensuring the two routes share no intermediate nodes or segments, except for the source and destination, and that the alternative route is equal to or greater in travel time than the primary route. This provides a robust "Plan-B" option for navigation.

```
Mode:driving
Source:<id>
Destination:<id>
```

input.txt for T2.1

```
Source:<id>
Destination:<id>
BestDrivingRoute:<id>,<id>,<id>(<int>)
AlternativeDrivingRoute:<id>,<id>,<id>(<int>)
```

output.txt for T2.1 (*the last integer with parenthesis refers to total time*)

In case of an impossible route, the output needs to include the following: "`BestDrivingRoute:none`" and "`AlternativeDrivingRoute:none`".

[T2.2: 5.0 points] Restricted Route Planning: Implement a feature to compute the fastest route with specific routing restrictions, namely:

- Excluding specific nodes from the graph, enabling users to avoid undesirable areas.
- Excluding specific segments from the graph, enabling users to avoid undesirable road segments.
- Simultaneously excluding the combination of nodes and segments of the graph.
- Including a **single** specific node (or stop) that the route must include while ensuring that the calculated route remains the fastest possible.

For the functionalities, the input file (named "input.txt") must comply with the following generic structure in the order of elements provided (**`Mode`** first, followed by **`Source`**, **`Destination`**, **`AvoidNodes`**, **`AvoidSegments`** and **`IncludeNode`**). The last three elements can be left with "empty" values. For example, if **AvoidNodes** is not used, the content should be left blank: "**AvoidNodes**:".

```
Mode:driving
Source:<id>
Destination:<id>
AvoidNodes:<id>,<id>,...
AvoidSegments:(id,id),(id,id),..
IncludeNode:<id>
```

input.txt for T2.2

```
Source:<id>
Destination:<id>
RestrictedDrivingRoute:<id>,<id>,<id>(<int>)
```

output.txt for T2.2 (*the last integer with parenthesis refers to total time*)

In case of an impossible route, the output needs to include the following: "`RestrictedDrivingRoute:none`".

4.3. Environmentally-Friendly Route Planning (driving and walking)

Develop functionality to plan routes that combine **driving and walking**, allowing users to:

1. Drive the first section of the route and then park the vehicle.
2. Walk the remaining distance to the destination.

For this part of the project, consider the following assumptions:

- The origin and destination cannot be adjacent nodes.
- The origin and destination cannot be parking nodes.
- All explored routes must include at least one driving segment and one walking segment.

In addition to the origin and destination, the user provides the following parameter:

- **Max. Walking Time**: Time the user is willing to walk after parking.

[T3.1: 4.0 points] Best route for driving and walking: After setting the parameters, you should:

- Implement the best (shortest overall) route for driving and walking, ensuring that it meets user-defined constraints. The best route must include at least one driving segment and one walking segment. The goal is to minimize both driving and walking time. Should there be two or more feasible routes with the same overall minimum travel time, you should select the one with the longest walking section (that obviously meets the maximum walking time restriction). See Appendix 1 for one illustrative example. Should there be two or more, pick any of them as the selected route.
- Note that, while a single intermediate parking location (and therefore stop) is required, no additional intermediate stop points are allowed in this mode. Still, the user may choose to exclude selected nodes and segments from either the driving or walking sections of travel.
- If no suitable route is found that satisfies the requirements, indicate which requirements cannot be met, i.e., walking time exceeds predefined maximum limit or absence of parking, or both.

For the functionalities, the input file (named "input.txt") must comply with the following generic structure in the order of elements provided (**Mode** first, followed by **Source**, **Destination**, **MaxWalkTime**, **AvoidNodes** and **AvoidSegments**). The last two elements can be left with "empty" values. For example, if **AvoidNodes** is not used, the content should be left blank: "**AvoidNodes**:".

```
Mode:driving-walking
Source:<id>
Destination:<id>
MaxWalkTime:<int>
AvoidNodes:<id>,<id>,...
AvoidSegments:(id,id),(id,id),...
```

input.txt for T3.1

```
Source:<id>
Destination:<id>
DrivingRoute:<id>,<id>,<id>(<int>)
ParkingNode:<id>
WalkingRoute:<id>,<id>,<id>(<int>)
TotalTime:<int>
```

output.txt for T3.1 (*the last integer with parenthesis refers to total time*)

In case of an impossible route, the output needs to include the following: "`WalkingRoute:none`", "`ParkingNode:none`", and "`DrivingRoute:none`".

If no suitable route is found that satisfies the requirements, include a "Message" (free text) in the output that explicitly describes which specific requirements could not be met. Consider the following output:

```
Source:<id>
Destination:<id>
DrivingRoute:none
ParkingNode:none
WalkingRoute:none
TotalTime:
Message:<string>
```

output.txt for T3.1 (*no suitable route is found*)

[T3.2: 2.0 points] Approximate Solution: If no suitable route is found, display a list of suggestions representing the best feasible alternative routes that approximate user requirements. For example, if the suggestions increase the maximum walking time, indicate the new time. Present 2 alternatives, sorted by overall travel time, but always including a driving and a walking segment.

```
Mode:driving-walking
Source:<id>
Destination:<id>
MaxWalkTime:<int>
AvoidNodes:<id>,<id>,...
AvoidSegments:(id,id),(id,id),...
```

input.txt for T3.2

```
Source<id>
Destination:<id>
DrivingRoute1:<id>,<id>,<id>(<int>)
ParkingNode1:<id>
WalkingRoute1:<id>,<id>,<id>(<int>)
TotalTime1:<int>
DrivingRoute2:<id>,<id>,<id>(<int>)
ParkingNode2:<id>
WalkingRoute2:<id>,<id>,<id>(<int>)
TotalTime2:<int>
```

output.txt for T3.2 (*the last integer with parenthesis refers to total time*)

5. Demo & Presentation

[T4.1: 2.0 points] Giving a presentation that is "short-and-to-the point" is increasingly important. As such, you are required to structure a short 10-minute demo of your work, highlighting key aspects of your implementation, specifically:

- Present your solution for the provided input datasets demonstrating the results requested in this project;
- Highlight your graph class, explaining the conceptual decisions that were made, including alterations to the original structure provided;
- Highlight the most important and challenging aspects of your implementations.

You should also elaborate a PowerPoint presentation to support your presentation. Instructions for preparing the PowerPoint presentation are available on Moodle.

6. Turn-In Instructions & Deadline

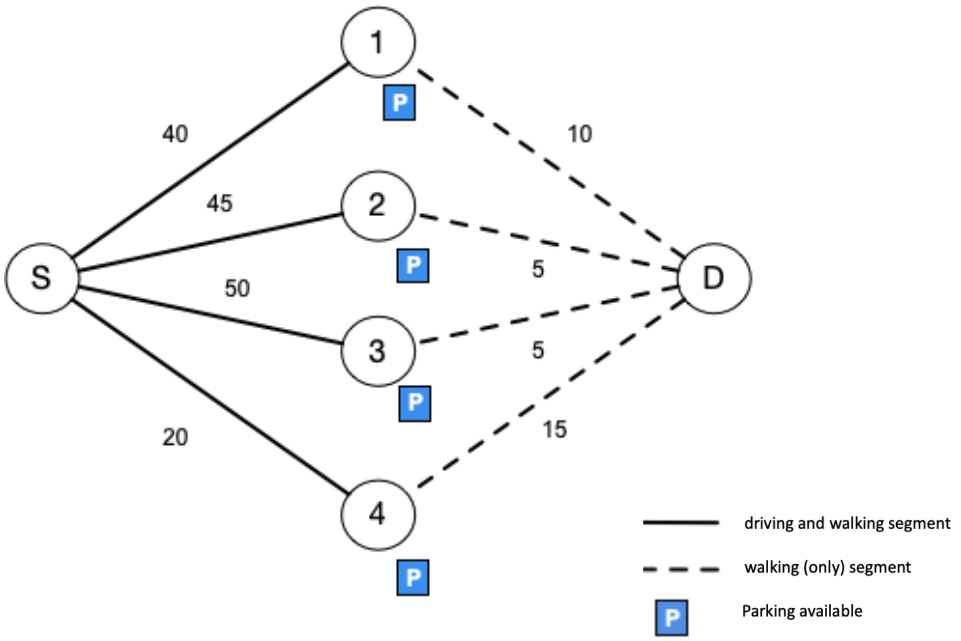
Submit a zip file named **DA2025_PRJ1_G<GN>.zip** on Moodle, where **GN** stands for your group number, with the following content:

- **Code** folder (contains program source code)
- Documentation folder (contains html documentation, generated using **Doxygen**)
- Presentation file (**PDF format**) that will serve as a basis for the demonstration.

Late submissions, up to 24 hours and 48 hours, will incur a penalty of 10% and 25% of the grade, respectively. No submissions will be accepted 48 hours after the deadline. Exceptions apply for justified and documented technical submissions issues.

Recall, the deadline is March 30, 2025, at 23:59 (PT time).

Appendix 1



In the simplified graph below, the best (and only) route from S to D with the maximum walking time of 10 minutes would be the route S-1-D with a total time of 50 minutes. Route S-4-D does not meet the restriction of maximum walking time. Route S-3-D is longer than the best route. Also, the route S-2-D has the same total travel time as S-1-D but the latter one has a higher walking time, and hence is more environmentally-friendly. So, for routes with the same total time, we want to select the one that has the longest walking time. If more than one exists we pick any of them.