VNIVERSITAT DE VALÈNCIA

STRUCTURES Laboratory

DATA AND ALGORITHMS

Degree in Data Science (1st)

Academic year 2023-24

Practice No. 7: Graphs
Completion period: Week of**05/13 to 05/17/2024**

Work methodology

Phases	Guy	Dedication
1. Solve the tasks set out in this scriptbefore to start the face-to-face session in the laboratory.	No presential	Max. 4.5 hrs.
2. Solve a problem in the laboratorynew exercise based on the resolution of previous tasks.	In person	Max. 3 hrs.

Problem

Graphs allow you to represent complex relationships between data sets. In this practice, the use of a graph will be shown in a hypothetical case in which relationships are established between cities, specifically, between Spanish provincial capitals. This example could represent a transportation network between these cities (road, bus, train) or any other type of relationship between them.

The objective of the first phase of the practice is the construction of the graph of cities to become familiar with the Graph class, which is provided already implemented. To do this, the Virtual Classroom has the following files that must be used to solve the exercise:

- graph.py
- queue_class.py
- cities.dat
- city_arches.csv

Exercise (Phase 1)

Homework 1

The file "graph.py"It completely implements the Graph class and includes a test.It is very important to read and properly interpret the content of this file to understand**how they are done**operations, but also to understand**how can they be used**these operations. The way in which the graph is created in the test, information is entered into the nodes or graph explorations are performed serve as an example to know how to solve the main task of this exercise.

The first task is to run this file (test) and read the code it contains (and the comments) to correctly interpret how the different actions are performed.

Task 2

Taking as reference the test of the Graph class that has been provided, a program must be written "pr7_v1.py"that builds the graph of cities by reading the information contained in the files "cities.dat"and "city_arcs.csv".

The file "cities.dat"It contains information on the provincial capitals in Spain. Each line of the file contains the name of a city. You must start the program by creating a function that reads this data file and returns a list (ofstr)with information on all cities.

A graph must be created with as many nodes as cities have been read and the information with the name of each city must be assigned to these nodes. To do this, you must use the operationAssignNodeInfo contained in the Graph class and whose use has been shown in the class test.

Now the graph will not have arcs, only nodes. You can print the graph to check it.

Next, implement a function that creates the arcs of the graph, reading the information contained in the file "city_arcs.csv",which indicates what relationships are established between cities and with what weight. Each line of this file contains information for an arc in the format:

origin; destination; weight

Example:

Line content:

Albacete;Granada;5

Interpretation : There is a directed arc betweenAlbaceteandGrenadewith weight 5.

To implement this function it is important to understand that the nodes are identified in the graph by an index in the list of nodes, not by the name, and that, therefore, the operation exists NodeIndexin the Graph class. Given a city name, one must determine what its index is in the list of nodes in order to incorporate the arc. In all arc operations, both the source and destination are numerical identifiers, not names.

Once the nodes and arcs of the graph have been created, this graph must be printed to check its correctness. Below is the expected output. For each arc, the destination identifier and the weight of the arc are indicated:

```
Node 0 (Albacete): (16.5) (45.4) Node 1 (Alicante):
(33.7) (35.8) (41.1)
Node 2 (Almería): (17.9) (26.10) (29.8) (42.6) (46.6) Node 3 (Ávila): (0.4)
(26.5)
Node 4 (Badajoz): (0.2) (51.9)
Node 5 (Barcelona): (51.3) (7.9) (45.8) Node 6
(Bilbao): (12.9) (37.2)
Node 7 (Burgos): (19.7) (24.9) Node 8
(Cáceres): (35.5)
Node 9 (Cádiz): (20.1) (33.5) (34.9) (36.10) Node 10
(Castellón de la Plana):
Node 11 (Ceuta): (22.7) (45.9)
Node 12 (Ciudad Real): (33.10) (34.8) (39.2) Node 13
(Córdoba): (38.4)
Node 14 (Cuenca): (17.6) (21.9) (25.3)
```

Node 15 (Gerona): (20.10) (39.3) Node 16
(Granada): (45.2)
Node 17 (Guadalajara): (9.10) (29.2) Node 18
(Huelva):
Node 19 (Huesca): (10.7) Node 20
(Jaén): (15.5) (50.1)
Node 21 (La Coruña): (11.5) (19.3) (30.8)
Node 22 (Las Palmas de Gran Canaria): (20.2) (21.3) Node 23
(León): (49.10)
Node 24 (Lérida): (11.10) (18.5) (19.9) (27.6) (38.3) (42.7) Node 25 (Logroño):
(32.4)
Node 26 (Lugo): (4.5) (37.4) Node 27
(Madrid): (4.7)
Node 28 (Málaga): (10.1) (34.4) (42.6) Node 29
(Melilla): (42.1)
Node 30 (Murcia): Node 31 (Orense): (10.10)
Node 32 (Oviedo): (34.5) (45.5) Node 33
(Palencia): (10.9) (45.3) Node 34 (Palma de
Mallorca): (0.7)

Node 35 (Pamplona): (1.2) (7.8) (8.7) (12.5) (15.10) Node 36
(Pontevedra): (42.8) (46.6)
Node 37 (Salamanca): (11.10) (35.6) (36.1) Node 38 (San
Sebastián):
Node 39 (Santa Cruz de Tenerife): Node 40
(Santander):
Node 41 (Segovia): (1.8) (25.4) (47.9) Node 42
(Seville): (3.3) (28.4)
Node 43 (Soria): (21.4) (49.7) Node 44
(Tarragona):
Node 45 (Teruel): (13.3) (43.9) (48.2) Node 46
(Toledo): (9.5) (41.6)
Node 47 (Valencia): (25.9) Node 48
(Valladolid):
Node 49 (Vitoria): (19.3) (28.5) (49.4) Node 50
(Zamora): (19.3)
Node 51 (Zaragoza): (32.7) (36.10) (41.6)

Phase 2: Final task

Resolution of a new exercise proposed during the face-to-face session in the laboratory.