ISEP
Algorithmics and Advanced programming

Tutorial course 4 : Graphs, an introduction

1 Objectives

— Get familiar with graph data structures.
— Get familiar with structural properties of graphs.

2 Manipulating graph data

This tutorial will discuss implementing a graph data structure in Java.

As discussed in the lecture, there are two types of graph representation :
— Adjacency list representation.
— Adjacency matrix representation.

2.1 Adjacency list representation of an undirected graph

Define a class GraphAdjList containing three attributes : the order of the graph (number of nodes), the size of the graph (the number of edges) and an adjacency list representation of the graph.

```
public class GraphAdjList {
    private int N;
    private int M;
    private List<List<Integer>> adj;
}
```

It is also possible (an even sometimes very useful necessary) to create a class *node* and a class *edge*. Then each entry of the adjacency list becomes a list of objects of class *node* (instead of a list of integer numbers).

The file *graph.txt* contains a small graph of an undirected and weighted graph. Open the file in a text editor and see how the file format looks like.

Each line in the file represents and edge. For example, the second line indicates that there is an edge between vertices 1 and 2.

Add the following functions to the class graph :

1. Initialize an empty graph of order *N* (input parameter) and 0 edges.
2. Initializes a graph from a specified input stream. You will test this function by reading the graph from the *graph.txt* file.
3. return the total order and the size of the graph.
4. a function called addEdge(int u, int v) that takes as input parameters two integers representing two vertex labels, the endpoints of the new edge. This function will add an edge between two existing nodes to the graph.

5. create a function Neighbors(int v) that takes as an input a given vertex and prints all the neighbors of that vertex.
6. Add a function that prints the Adjacency list representation of the graph.

Test all these functions with the graph contained the text file.

Now, you are going to study some structural properties of the graph. Create a function called Degree(int v) that returns for a given vertex *v*, its degree. Answer the following questions :

1. What is the average, minimal and maximal degree of the graph ? What is the edge-density ? Do you consider this graph dense or sparse ?
2. Are there any isolated nodes ? If yes, which ones ?
3. Are there any loops ?
4. Verify your answers by using the Neighbors(int v) function.

Finally, write a function that allows to read data graph from the keyboard input. The program will ask the user the total number of vertices, the total number of edges and user will have to type each edge as in the following example :

Number of vertices ? 4
Number of edges ? 4
Edge 1 ? 1 2
Edge 2 ? 2 3
Edge 3 ? 3 4
Edge 4 ? 4 3

2.2 Adjacency matrix representation of an undirected graph

Define a class GraphAdjMatrix containing three attributes : the order of the graph (number of nodes), the size of the graph (the number of edges) and an adjacency matrix.

```
public class GraphAdjMatrix {
private int N;
private int M;
private boolean[][] adj;
}
```

Create the following functions :

1. Initialize an empty graph of order *N* (input parameter) and 0 edges.
2. Initializes a graph from a specified input stream. All the entries of the matrix adj must be initialized.
   You will test this function by reading the graph from the *graph.txt* file.
3. Add a function that calculates the distance between the nodes using multiplication of the adajcency matrix.
4. Why is it preferable to use an adjacency list representation in practical contexts ?

3 Bigger Graph

Now, you should test your previous code on large datasets that contain several thousands of edges. You are provided with two large graphs.

a.Wiki-vote contains voting data from wikipedia, each node represents a user and an edge between two users represents a vote from user a to user
b. Facebook combined contains information about users and who is friends with whom.

Previously, you calculated the number of edges, as well as the average, minimal and maximum degree.
For these two datasets :
1- use the two previous data structures and compute the previous metrics.
2- Which structure is more efficient ?

4 Object Oriented Programming

Classes GraphAdjList and GraphAdjMatrix have a lot in common. Check if they are only behaviors in common or they are some behaviors with implementation.

— What OOP technique and tools (ie., interface, abstract class, ..) can you use to re-use a maximum of code between these two classes ?

5 Bonus 1: Use java Libraries

JGraphT,Download it from https://jgrapht.org/
Make a project and add the lib folder to your project
Copy the hello world class in your class : https://jgrapht.org/guide/HelloJGraphT
Change the name of the package, class and constructor to yours
If you work with python, you may use Networkx. And if your language is c++ you may find the corresponding lib.