Remarks Today, I let you work on your own. You may discuss about the algorithm or the implementation, but you should write your own code. Try to do as many exercises as you could, then please send your codes and the analysis to my email `ni-luh-dewi.sintiari@ens-lyon.fr` before Tuesday, October 13th, 11.59 pm.

As in the previous session, we assume that we work with a graph $G = (V, E)$ that has n vertices denoted $V := \{0, 1, \dots, n-1\}$. If the graph is a tree, we will denote it as $T = (V, E)$.

Graph representations

During the last session, we represented a graph G by its list of edges E. There are two other common representations. First, a graph can be represented by its *adjacency matrix*. This is a symmetric matrix $A \in \{0,1\}^{n \times n}$ such that $A_{ij} = 1$ if $(i, j) \in E$ and $A_{ij} = 0$ otherwise. Second representation is the *adjacency list*. In this representation, we are given a list L of length n such that $L[i]$ is the list of neighbors of the vertex i.

Exercise 1.

(a) Write a function `edges_to_matrix(n,E)` that takes a list of edges E and outputs the adjacency matrix of the graph G. Similarly, write a function `edges_to_list(n,E)` that outputs the adjacency list of G. Write functions that do the opposite conversions. Can you tell what the advantages and drawbacks of each of these representations?

Distances in trees

Suppose that we have a tree $T = (V, E)$ with n vertices. Observe that if $(u, v) \in V$ are two vertices, then there are joined by exactly one path in T. We define the *distance* $d(u, v)$ between u and v as the length (number of edges) of this path. We put $d(u, u) := 0$. The *diameter* of the tree is then defined as the length of the longest path in T, $\operatorname{diam}(T) := \max_{u,v \in V} d(u, v)$.

Exercise 2.

(a) Write a function `is_tree(n,E)` that verifies if a given graph is a tree. To do so, you may use an algorithm from the previous lab session. (Hint: a tree is a connected graph with $n - 1$ edges.)

(b) Suppose that $T = (V, E)$ is a tree. Write a function `distances(u,E)` that, for a given vertex $u \in V$, outputs a list D of length n such that $D[v]$ is the distance $d(u, v)$. To make the algorithm efficient, you may want to create the adjacency list of T and use a recurrence. (Hint: note that the neighbors of u are at distance 1, their neighbors different than u are at distance 2 and so on. If you are stuck, read about the Depth-first search algorithm and adapt it to this exercise.)

(c) Write a function `diameter(E)` that finds the diameter of the tree T. To do so, use the following algorithm. Take any vertex $u \in V$ and find a vertex $v \in V$ such that $d(u, v)$ is maximal. Then, repeat the procedure for the vertex v, i.e., find a vertex $w \in V$ such that $d(v, w)$ is maximal. It can be proven that $d(v, w)$ is the diameter of T. What is the complexity of the function `diameter(E)`?

Laplacian matrices

In this section, we want to find out what is the exact number of (labelled) trees on $n \geq 2$ vertices. We recall that a *Laplacian matrix* of a graph $G = (V, E)$ is a matrix $L \in \mathbb{R}^{n \times n}$ defined as $L := D - A$, where $A \in \{0,1\}^{n \times n}$ is the adjacency matrix of G, and $D \in \mathbb{R}^{n \times n}$ is the *degree matrix*, i.e., a diagonal matrix defined as $D_{ii} = \deg(i)$ for all $i \in V$ and $D_{ij} = 0$ if $i \neq j$.

Exercise 3.

(a) Write a function `laplacian(n,E)` that finds the Laplacian matrix of G.

(b) Write a function `count_spanning_trees(n,E)` that counts the number of spanning trees of G. Use the fact that this number is equal to $\det(M)$, where $M \in \mathbb{R}^{(n-1)\times(n-1)}$ is the matrix obtained from the Laplacian of G by removing its first column and first row.

(c) Write a function `clique_laplacian(n)` that creates the Laplacian matrix of a complete graph (clique) on n vertices. Use this function to create a function `number_trees(n)` that finds the number of trees on n vertices. Test it on some values of n. What did you find? (Additional question: can you prove that your result is correct using elementary operations on matrices?)