

## - TP 3. Arbre en largeur et en profondeur. -

Le but de ce TP est de calculer un arbre en largeur sur un graphe  $G$ . L'ensemble des sommets de  $G$  est  $\{0, \dots, n - 1\}$ , les arêtes étant codées par listes de voisinages. Ainsi, à chaque sommet  $i$  est associée la pile voisins[ $i$ ] des voisins de  $i$ . Votre programme pourra contenir :

```
#include <cstdlib>
#include <iostream>
#include <vector>
#include <fstream>

using namespace std;

int main(){
int n;           //Le nombre de sommets.
int m;           // Le nombre d'aretes.
cout << "Entrer le nombre de sommets: ";
cin >> n;
cout << "Entrer le nombre d'aretes: ";
cin >> m;
vector<int> voisins[n]; // Les listes des voisins.
int pere[n];           // L'arbre en largeur.
int ordre[n];          // L'ordre de parcours.
int niveau[n];         // Le niveau du point.

return 0;
}
```

Ce début de code est récupérable là: <http://www.lirmm.fr/~bessy/GLIN501/TP/tp3.cc>

!!!! Pensez à tester chaque code produit sur de petits exemples !!!!

### - Exercice 1 - Création d'un graphe aléatoire.

Ecrire une fonction void *voisinstrandom*(*int n*, *int m*, *vector<int> voisins[ ]*) qui engendre aléatoirement les listes de voisins d'un graphe aléatoire de  $n$  sommets et  $m$  arêtes. On prendra garde à :

- la symétrie: si  $x$  est voisin de  $y$ , alors  $y$  est voisin de  $x$ .
- ne pas créer de boucle.
- ne pas créer d'arête multiple.

### - Exercice 2 - Parcours en largeur.

Implémenter l'algorithme du cours void *parcourslargeur*(*int n*, *vector<int> voisins[ ]*, *int niveau[ ]*, *int ordre[ ]*, *int pere[ ]*) qui effectue un parcours en largeur de racine 0 et calcule pour tout  $i$  :

- **pere**[ $i$ ], représentant le père de  $i$  dans l'arbre en largeur, lorsque  $i \neq 0$ .
- **ordre**[ $i$ ], représentant la date à laquelle  $i$  a été lu en premier.
- **niveau**[ $i$ ], représentant le niveau de  $i$  dans l'arbre.

**- Exercice 3 - Ecriture des niveaux.**

Ecrire une fonction `void ecritureniveaux(int n, int niveau[ ])` qui écrit le nombre de sommets dans chaque niveau, et le nombre de sommets qui ne sont pas joignable à partir de 0. Votre résultat sera de la forme (ici pour  $n = 40$  et  $m = 80$ ):

Il y a 1 sommets au niveau 0.  
Il y a 4 sommets au niveau 1.  
Il y a 12 sommets au niveau 2.  
Il y a 17 sommets au niveau 3.  
Il y a 4 sommets au niveau 4.  
Il y a 2 sommets qui ne sont pas dans la composante de 0.

**- Exercice 4 - Parcours en profondeur.**

Modifier votre algorithme afin de le transformer en parcours en profondeur. Enumérer de même le nombre de sommets sur chaque niveau de l'arbre en profondeur.

**- Exercice 5 - Pour aller plus loin.**

- Lorsque  $m = 2n$ , c'est à dire lorsque le degré moyen des sommets du graphe est égal à 4, quelle est à votre avis le nombre de niveaux, en fonction de  $n$ , de l'arbre en largeur et de l'arbre en profondeur. Faire pour cela des essais avec  $n$  de plus en plus grand.
- Au lieu d'un graphe aléatoire, tirer les sommets du graphe au hasard parmi les points de la grille 612x792 et ne garder que les arêtes de longueur inférieure à un certain seuil. Afficher ce graphe en utilisant la fonction d'affichage du TP2. Calculer ensuite un arbre en largeur dans ce graphe et afficher celui-ci. Voir les figures ci-dessous.
- Implémenter le calcul des arêtes séparatrices vu en TD, et tester sur un graphe avec  $\frac{3}{2}n$  arêtes.

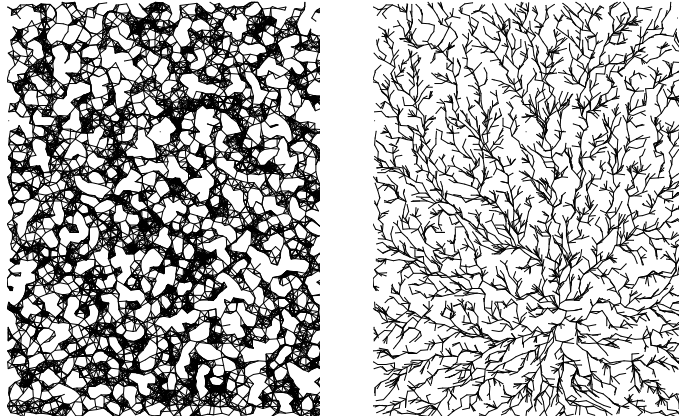


Figure 1: Un bien bel exemple d'arbre en largeur.