Travaux Pratiques de Structures de Données n°7

Cours d'Informatique de Seconde Année

— Licence 2.2 —

Pour représenter un graphe on utilisera la structure suivante.

```
#define MAX_S 100
typedef int Graphe[MAX_S][MAX_S];
```

La macro `MAX_S` nous donne le nombre maximum de sommets que peut posséder le graphe. Le type `Graphe` est un tableau à deux dimension de `MAX_S` lignes et colonnes qui ne contient que les valeurs 1 ou 0. Une case non nulle de ce tableau représente l'arrête qui relie les deux sommets donnés par la ligne et la colonne de la case. Si par contre une case est nulle alors cela signifie qu'il n'existe pas d'arrête reliant les deux sommets donnés par la ligne et la colonne de la case.

Par exemple plaçons-nous dans le cas où `T` est une variable de type `Graphe`. Si la valeur de `T[4][2]` est égale à 1 alors il existe une arrête qui relie le sommet `4` et le sommet `2`. On remarque que comme les arrêtes ne sont pas orientées, la case `T[2][4]` est aussi égale à 1.

▸ **Exercice 1.**

- *Écrire une fonction* `void Deconnecter(Graphe A)` *qui retire toutes les arrêtes du graphe* `A`.
- *Écrire une fonction* `void Ajouter(Graphe A,int sommet_a,int sommet_b)` *qui ajoute au graphe* `A` *une arête reliant les sommets* `sommet_a` *et* `sommet_b`. *Si l'arête existait dèjà la fonction ne fait rien.*
- *Écrire une fonction* `void Retirer(Graphe A, int sommet_a,int sommet_b)` *qui retire au graphe* `A` *l'arête reliant les sommets* `sommet_a` *et* `sommet_b` *si elle existe.*

▸ **Exercice 2.** *Écrire une fonction* `void Fabriquer(Graphe A, FILE* fichier)` *qui fabrique un graphe à partir d'un fichier. La description du graphe se fera en donnant ses arêtes. Une arête est donnée par un couple de nombre séparé par une virgule. Les arêtes doivent être séparées par un point virgule.*
Par exemple `1,2 ; 3,4;3,7;` *est la description d' un graphe qui possède trois arêtes : une qui relie les sommets* 1 *et* 2, *une autre qui relie* 3 *et* 4 *et une dernière qui relie* 3 *et* 7. *Les autres sommets ne sont par reliés.*

La composante connexe d'un sommet s faisant parti d'un graphe A est l'ensemble des sommets qui sont reliés à lui par un chemin passant par zéro, une ou plusieurs arrêtes du graphe. La composante connexe d'un sommet s contient toujours s.

▸ **Exercice 3.** *Écrire une fonction* `void ComposanteConnexe(Graphe A, int sommet, int T[MAX_S])` *qui remplit le tableau* `T` *de la façon suivante. Si le sommet i appartient à la composante connexe du sommet* `sommet` *alors après appel de la fonction la case* `T[i]` *doit contenir* 1 *sinon aprés appel de la fonction elle doit contenir* 0. *On pourra écrire et utiliser une fonction récursive* `void ComposanteConnexeT(Graphe A, int sommet, int T[MAX_S])` *qui suppose que* `T` *ne contient que des valeurs nulles.*

La longueur d'un chemin est le nombre d'arêtes que le chemin traverse. La distance entre deux sommets d'un graphe reliés par un chemin est la longueur du chemin de plus petite longueur qui relie ces deux sommets.

▸ **Exercice 4.** *Écrire une fonction* `void Distance(Graphe A, int sommet, int T[MAX_S])` *qui remplit le tableau* `T`. *Après appel de la fonction la case i du tableau doit contenir la distance entre le sommet i et le sommet* `sommet` *s'ils sont reliés par un chemin sinon la case i doit contenir* -1.