Exercices

24.1 (**) La configuration matérielle du réseau informatique intranet d'une entreprise nationale est enregistrée dans un fichier texte. Le fichier contient :

- Le nombre n de noeuds du réseau (chaque noeud correspondant plus ou moins à une localisation géographique) ;
- Le nombre m de connexions entre noeuds (une connexion correspondant à un câble). On supposera qu'entre deux noeuds il y a au plus 1 câble.
- La liste des n noms des noeuds ;
- La liste des m connexions, chaque connexion étant représentée par les numéros des deux noeuds extrémités.

a) Proposez une structure de données pour représenter le réseau en mémoire centrale. On rassemblera toutes les données nécessaires dans une structure `Reseau`.

b) Écrire une fonction de chargement du réseau en mémoire.

c) Écrire une fonction de sauvegarde du réseau stocké en mémoire.

d) Écrire une fonction qui détermine si deux machines données sont connectées par un câble.

e) Qu'est-ce qu'un chemin dans le réseau ?

f) Écrire une fonction qui prend en paramètre un tableau de numéros de noeuds du réseau et qui détermine si chaque noeud du tableau est connecté au suivant par un câble.

24.2 (*) (graphes non orientés) On appelle *graphe non orienté* un graphe dont toutes les arêtes vont dans les deux sens. Autrement dit, dans un graphe non orienté, s'il y a un arc d'un sommet s_i vers le sommet s_j, alors il y a aussi un arc du sommet s_j vers le sommet s_j. La paire $\{i, j\}$ est alors appelée une *arête* de G.

Soit un graphe donné par une matrice d'adjacence. Donner une propriété de la matrice qui soit caractéristique d'un graphe non orienté. Donner un algorithme qui prend en paramètre un graphe donné sous forme de matrice d'adjacence, et qui renvoie 1 si le graphe est non orienté, et 0 sinon.

24.3 (*) Soit un graphe G à n sommets. Un *coloriage de G* est une fonction qui à chaque sommet de G associe un entier appelé couleur du sommet. On représente un coloriage de G par un tableau de n entiers.

Un coloriage de G est dit *correct* si pour tout arc (i, j) de G la couleur de s_i est différente de la couleur de s_j.

Écrire une fonction qui prend en paramètre un graphe G représenté sous forme de matrice d'adjacence et un coloriage, et qui renvoie 1 si le coloriage de G est correct et 0 sinon.

25.1 (*) Appliquer le parcours en profondeur au graphe de la figure 25.1 en prenant pour sommet de départ le sommet 2, puis en prenant pour sommet de départ le sommet 7. On donnera la liste des sommets visités dans l'ordre où ils sont visités.

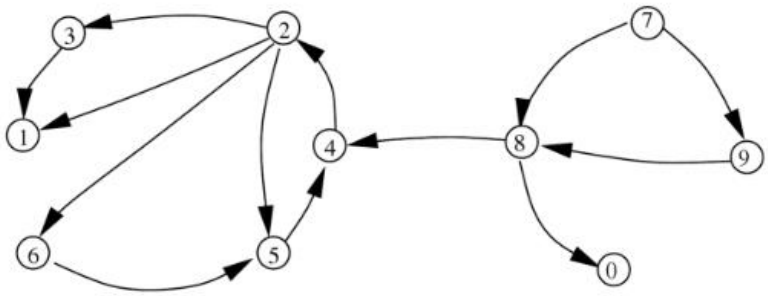

Figure 25.1– **Exemple de graphe**

25.2 (**)

a) Implémenter le parcours en profondeur récursif avec une représentation du graphe sous forme de matrice d'adjacence.

b) Écrire une fonction qui prend en paramètre deux sommets s_1 et s_2 dans un graphe donné sous forme de martrice d'adjacence, et qui renvoie 1 s'il existe un chemin de s_1 à s_2 et renvoie 0 sinon.

25.3 (*) Appliquer le parcours en largeur au graphe de la figure 25.2 en prenant pour sommet de départ le sommet 4, puis en prenant pour sommet de départ le sommet 7. On donnera la liste des sommets visités dans l'ordre où ils sont visités et on maintiendra à jour une file.

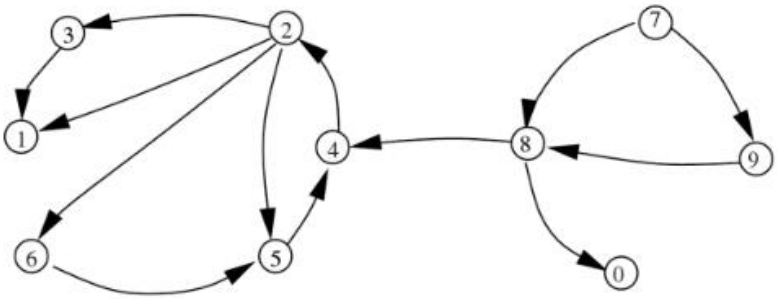

Figure 25.2 – **Exemple de graphe**

25.4 (*) Donner une implémentation en *C* du parcours en largeur.

25.5 (*) (plus court chemin) Appliquer le parcours en largeur au graphe de la figure 25.3 en prenant pour sommet de départ le sommet 4, puis en prenant pour sommet de départ le sommet 8. On donnera la liste des sommets visités dans l'ordre où ils sont visités et on maintiendra à jour une file.

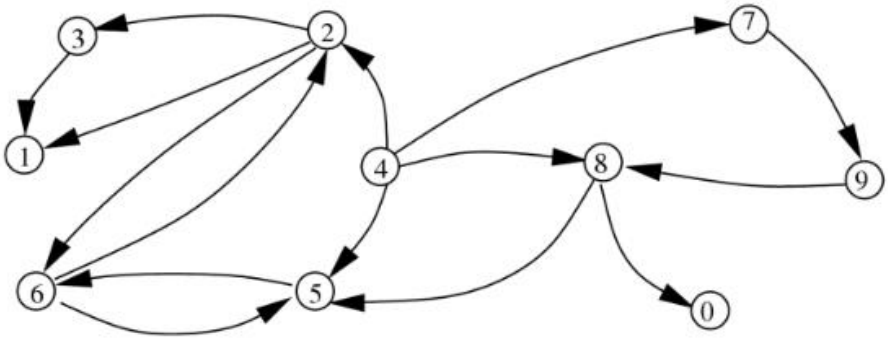

Figure 25.3 – **Exemple de graphe**

Pour chaque sommet S inséré dans la file, on dessinera une flèche de couleur du sommet S vers le dernier sommet V défilé. Que remarque-t'on en suivant les flèches de couleur ?

25.6 (**) (implémentation du plus court chemin) Écrire une fonction *C* qui donne le plus court chemin entre deux sommets s_1 et s_2 d'un graphe donné sous forme de matrices d'adjacence. Pour chaque sommet S inséré dans la file, on mémorisera le numéro `preced` du dernier sommet V défilé avant d'insérer S dans la file. Pour afficher le chemin, on suivra les numéros `preced` à la manière des pointeurs `suivant` dans un parcours de liste chaînée.

Pour les exercices de ce chapitre, on pourra utiliser les fonctions de gestion des listes chaînées du chapitre 19, notamment l'insersion en tête de liste et l'insertion en queue de liste.

26.1 (**) Écrire un programme *C* qui construit la représentation sous forme de matrice d'adjacence d'un graphe donné sous forme de listes d'adjacence.

26.2 (* * *) Écrire un programme *C* qui construit la représentation sous forme de listes d'adjacence d'un graphe donné sous forme de matrice d'adjacence.

26.3 (**) La base de données d'une compagnie d'aviation est stockée dans un fichier texte au format suivant :

- La première ligne du fichier contient le nombre n d'aéroports du réseau.
- La deuxième ligne du fichier contient le nombre m d'avions de la semaine.
- Les n lignes suivantes du fichier contiennent les noms des n aéroports.
- Les m lignes suivantes du fichier contiennent la liste des vols de la semaine. Chaque vol est représenté par :
  - le numéro du vol ;
  - le jour et l'heure du décolage codée sur un `int` ;
  - le numéro de l'aéroport de départ du vol ;
  - le numéro de l'aéroport d'arrivée du vol.

a) Proposer une structure de données pour représenter la base de données en mémoire.

b) Écrire une fonction de chargement de la base de données en mémoire centrale.

c) Écrire une fonction qui affiche tous les vols de la semaine au départ d'un aéroport dont le nom est passé en paramètre.

d) Écrire une fonction qui détermine s'il y a un vol direct entre deux aéroports dont les noms sont passés en paramètre.

e) Écrire une fonction qui affiche le premier vol direct à partir d'une date courante entre deux aéroports passés en paramètre. La fonction doit afficher un message d'erreur s'il n'y a pas de vol entre ces deux villes.

26.4 (**)

a) Implémenter le parcours en profondeur récursif avec une représentation du graphe sous forme de listes d'adjacence.

b) Écrire une fonction qui prend en paramètre deux sommets s_1 et s_2 dans un graphe donné sous forme de listes d'adjacence, et qui renvoie 1 s'il existe un chemin de s_1 à s_2 et renvoie 0 sinon.

26.5 (**) Écrire les fonctions C permettant de faire le parcours en largeur d'un graphe donné sous forme de listes d'adjacence.