Graphes (Graphs)
Représentations et parcours

1 Représentations

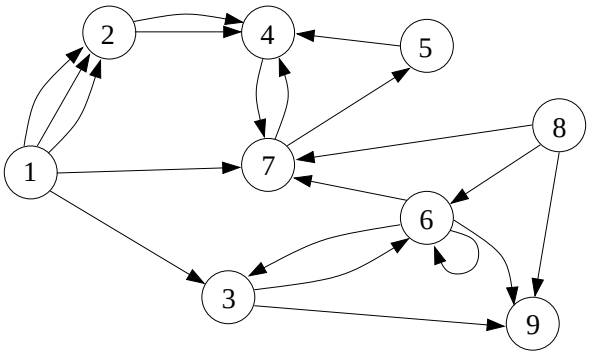

FIGURE 1 – Graphe G'_1

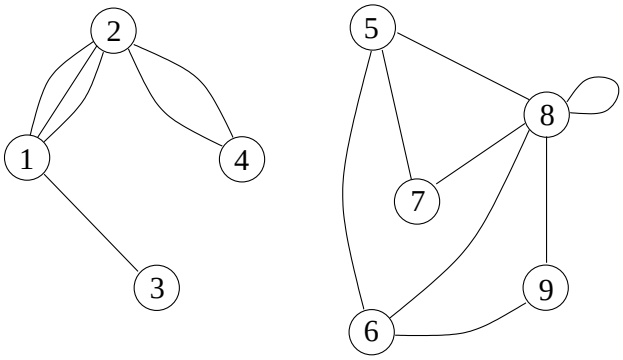

FIGURE 2 – Graphe G'_2

Exercice 1.1 (Représentation statique)

1. Comment s'appelle la représentation statique d'un graphe ?
2. Décrire cette représentation.
3. Donner les représentations statiques des graphes des figures 1 et 2.
4. Quelles sont les différences, pour la représentation, lorsque le graphe est orienté ou non orienté, valué ou non, possède des liaisons simples ou multiples ?
5. Sachant que l'on veut pouvoir utiliser le même type pour représenter un graphe qu'il soit orienté ou non, simple ou 1-graphe et valué ou non, ou encore un multigraphe ou p-graphe (non valué dans ce cas), écrire le type de données statique `t_graph_stat` correspondant.

Exercice 1.2 (Représentation dynamique)

1. Quelle est l'autre manière de représenter un graphe ?
2. Décrire les différentes représentations selon que tel ou tel élément est statique ou dynamique.

   Nous choisissons ici d'avoir une représentation complètement dynamique. De plus, lors du parcours, nous voulons avoir un accès "immédiat" à l'extrémité d'un arc (ou d'une arête) emprunté !
3. Quelles sont les différences pour la représentation lorsque le graphe est orienté ou non orienté, valué ou non, possède des liaisons multiples ?
4. Donner la représentation des sous-graphes issus de $S' = \{1, 2, 4, 5, 7\}$ des figures 1 et 2.
5. Sachant que l'on veut pouvoir utiliser le même type pour représenter n'importe quel type de graphe : orienté ou non, simple (ou 1-graphe) ou multigraphe (ou p-graphe), valué ou non ; écrire le type dynamique `t_graph_dyn` correspondant.
6. Écrire une fonction qui recherche le sommet s dans un graphe G représenté dynamiquement. On pourra supposer que la recherche est toujours positive ($1 \leq s \leq ordre(G)$).

Exercice 1.3 (De la représentation dynamique à la représentation statique – *Partiel jan. 2012*)

Écrire un algorithme qui construit la représentation statique d'un graphe quelconque (orienté ou non) non valué à partir de sa représentation dynamique.

Notes :
Sauf indications particulières, les graphes utilisés dans les exercices suivants seront des graphes simples pour les non orientés, et des 1-graphes sans boucles pour les orientés. Les exemples utilisés ici seront les graphes G_1 (1-graphe sans boucle issu de G'_1) et G_2 (graphe partiel simple issu de G'_2).

---

2 Parcours

Exercice 2.1 (Parcours en largeur)

1. Donner, en précisant les forêts couvrantes obtenues, les parcours en largeur complets des graphes G_1 et G_2 à partir du sommet 8 (les sommets sont choisis en ordre croissant).
2. Donner le principe de l'algorithme de parcours en largeur. Comparer avec le parcours en largeur d'un arbre général.
3. Comment stocker la forêt couvrante de manière linéaire ?
4. Écrire les algorithmes de parcours en largeur, avec construction de la forêt couvrante, pour les deux représentations.

Exercice 2.2 (Parcours en profondeur)

1. Donner, en précisant les forêts couvrantes obtenues, les parcours en profondeur du graphe G_1 à partir du sommet 1, et du graphe G_2 à partir du sommet 5 (les sommets sont choisis en ordre croissant).
2. Donner le principe de l'algorithme récursif du parcours en profondeur. Comparer avec le parcours en profondeur d'un arbre général.
3. Quels sont les différents types d'arcs rencontrés lors d'un parcours en profondeur ?
   Classer les arcs des parcours en profondeur effectués en question 1 et ajouter aux forêts couvrantes les arcs manquants.
   Comment peut-on reconnaître le type d'un arc ?
4. On utilise la notion d'ordre préfixe de visite (première rencontre) et ordre suffixe de visite (dernière rencontre). En numérotant les sommets suivants ces deux ordres, écrire les conditions de classification des arcs.
5. Écrire l'algorithme récursif de parcours en profondeur, en indiquant au cours du parcours, quel est le type de l'arc rencontré, pour les cas suivants :
   (a) Le graphe est non orienté et représenté par une matrice d'adjacence.
   (b) Le graphe est orienté et représenté par listes d'adjacence.

(6.) Le parcours en profondeur peut être fait en itératif.
Donner le principe d'un tel parcours et écrire l'algorithme correspondant pour un graphe orienté en représentation dynamique.

3 Applications

Exercice 3.1 (Graphes bipartis (Bipartite graph) – *Partiel Dec. 2012*)

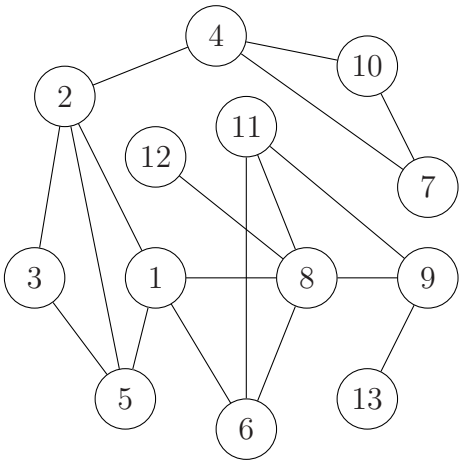

FIGURE 3 – Graphe G_3

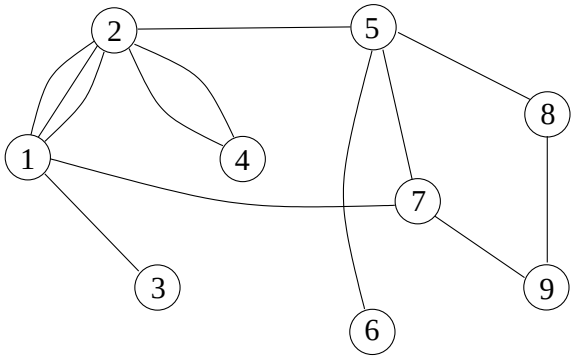

FIGURE 4 – Graphe G_4

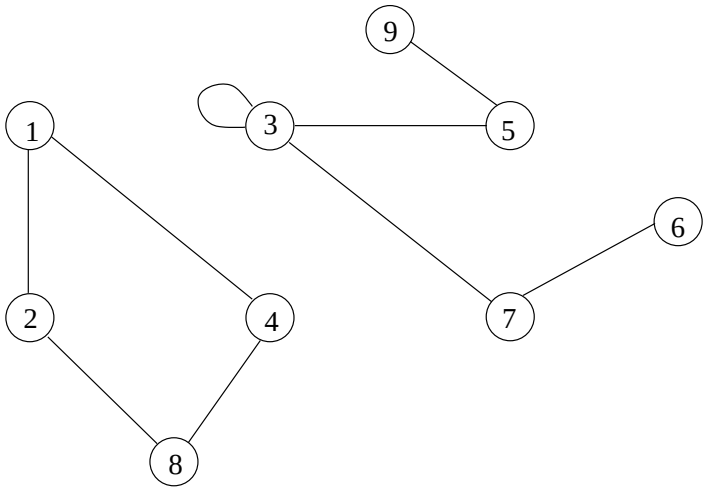

FIGURE 5 – Graphe G_5

Un graphe biparti est un graphe (un multigraphe) non orienté $G =< S, A >$, dans lequel S peut être partitionné en deux ensembles S_1 et S_2 tels que $(u, v) \in A$ implique soit que $u \in S_1$ et $v \in S_2$, soit que $u \in S_2$ et $v \in S_1$. Aucune arête ne doit relier deux sommets d'un même ensemble.

1. Les graphes des figures 3 à 5 sont-ils bipartis ? Pour chaque graphe biparti, donner les deux ensembles S_1 et S_2.

2. Écrire un algorithme qui teste si un graphe en représentation dynamique est biparti.

Exercice 3.2 (Distance au départ – *Contrôle fév. 2013*)

L'objectif dans cet exercice est de connaître les sommets qui se trouvent à une distance maximale donnée d'un sommet de départ. Le résultat se présentera sous forme d'un vecteur de distances. Nous travaillerons dans des graphes orientés et non-valués en représentation statique.

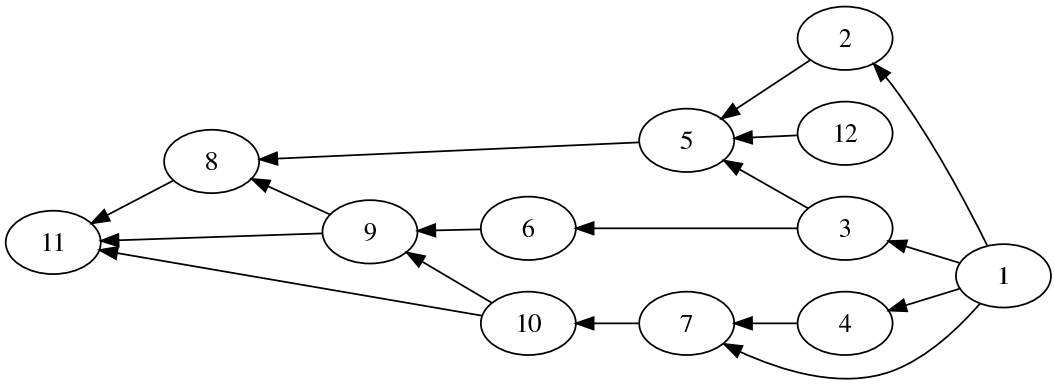

FIGURE 6 – Un graphe

Le remplissage du vecteur de distance est relative simple : à la case i on trouve la distance entre le sommet source et le sommet i. On note en particulier que :
- `dist[src] = 0` avec `src` le sommet source.
- `dist[i]` = $+\infty$ pour tous les sommets qui ne sont pas accessibles depuis `src`.

1. Calculer les distances depuis le sommet numéro 1 dans le graphe de la figure 6 (on veut toutes les distances des sommets atteignables, sans borne).
2. Écrire la procédure `distances (g, src, maxdist, dist)` qui calcule les distances entre le sommet `src` et tous les autres sommets du graphe `g` se trouvant à une distance inférieure ou égale à `maxdist` et stocke le résultat dans le vecteur `dist`.

Exercice 3.3 (Compilation, cuisine...)

1. *Ordonnancement , un exemple simple :*
   Supposons l'ensemble d'instructions suivantes à effectuer par un seul processeur :

   ① lire (a)
   ② b ← a + d
   ③ c ← 2 * a
   ④ d ← e + 1
   ⑤ lire (e)
   ⑥ f ← h + c / e
   ⑦ g ← d * h
   ⑧ h ← e - 5
   ⑨ i ← h - f

   Quels sont les ordres possibles d'exécution ?
   Comment représenter ce problème sous forme de graphe ?

   Chaque solution correspond à un *tri topologique* du graphe.

2. Quelle doit être la propriété du graphe pour qu'une solution de tri topologique existe ?

3. (a) Soit $suffix$ le tableau contenant les dates de dernière visite : l'ordre suffixe de tous les sommets de G lors d'un parcours en profondeur.
   Démontrer que pour une paire quelconque de sommets distincts $u, v \in S$, s'il existe un arc dans G de u à v, et si G a la propriété de la question 2, alors $suffix[v] < suffix[u]$.
   (b) En déduire un algorithme qui trouve une solution de tri topologique pour un graphe en représentation statique, on supposera qu'une solution existe.
   (c) Que faut-il changer à cet algorithme pour vérifier l'existence d'une solution dans un graphe quelconque ?

Et la cuisine dans tout ça ?