2 Parcours de Graphes

2.1 Applications des algo (*)

On considère le graphe non orienté G_2 de la figure 5

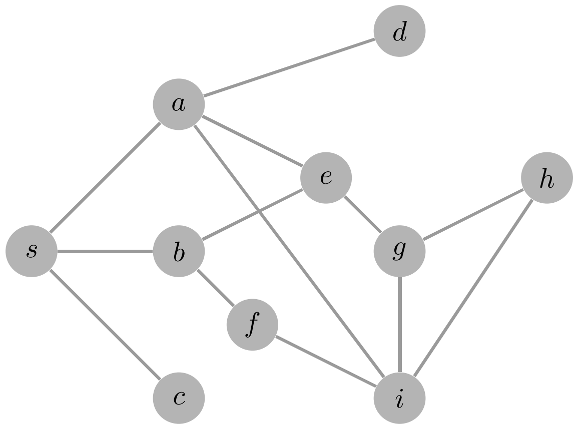

Figure 5 – Graphe G_2

Question 1. Parcours en profondeur. Appliquez l'algorithme de parcours en profondeur sur le graphe G_2 à partir du sommet s. Lorsque plusieurs sommets peuvent être sélectionnés, choisissez l'ordre alphabétique. Représentez **l'arborescence obtenue** et **la numérotation des sommets** (marqués comme visités).

Arborescence du parcours DFS

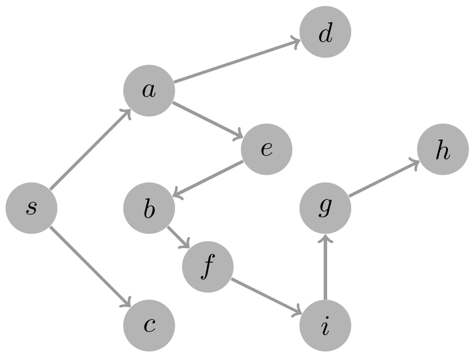

Figure 6 – Parcours DFS du graphe G_2

Numérotation des sommets (visités)

1	2	3	4	5	6	7	8	9	10
d	h	g	i	f	b	e	a	c	s

Table 3 – ordre visité DFS

Question 2. Parcours en largeur. Appliquez l'algorithme de parcours en largeur sur le graphe G_2 à partir du sommet s. Lorsque plusieurs sommets peuvent être sélectionnés, choisissez l'ordre alphabétique. Représentez **l'arborescence obtenue** et **la numérotation des sommets** (marqués comme visités).

Arborescence du parcours BFS

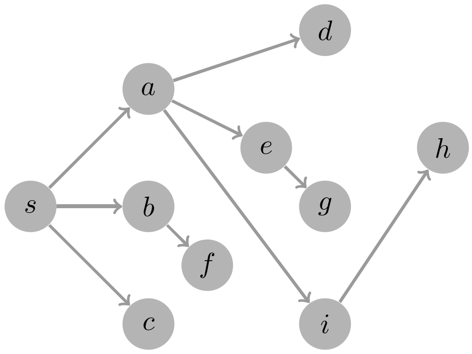

Figure 7 – Parcours BFS du graphe G_2

Numérotation des sommets (visités)

1	2	3	4	5	6	7	8	9	10
s	a	b	c	d	e	i	f	g	h

Table 4 – ordre visité BFS

Question 3 (optionnelle). Parcours dans des graphes orientés. Reprenez le graphe G_1 de la section 1 pour appliquer un parcours en profondeur et un parcours en largeur à partir du sommet a.

2.2 Cycle - Graphe non orienté

Soit un graphe non orienté G, il existe une propriété aussurant l'existence d'un cycle dans un tel graphe. Cette propriété est : si tout sommet de G est de dégré supérieur ou égal à 2 alors G possède au moins un cycle.
Nous n'allons pas chercher à montrer cette propriété (voir les livres ou sites internet sur le sujet).
Remarque : une conséquence de cette propriété est qu'un graphe non orienté sans cycle possède au moins un sommet de degré 0 ou 1.

Question 1. Graphe acyclique et nombre d'arêtes. Montrez, par induction sur le nombre de sommets, qu'un graphe acyclique d'ordre n (ie. ayant n sommets) possède au plus $n-1$ arêtes.
Si le graphe acyclique G est d'ordre 1, il possède 0 arête, la propriété est vérifiée.
Supposons la propriété vraie pour un graphe acyclique d'ordre n, et cherchons à la vérifier pour l'ordre $n+1$.
Soit le graphe acyclique $G = (X, E)$ ayant $n+1$ sommets, ce graphe comporte un sommet x de degré vallant au plus 1 (propriété énoncée en début d'exercice). Si on considère le graphe $G' = (X', E')$ ayant comme sommets $X' = X\backslash\{x\}$, et les arêtes reliant les sommets de X', ce graphe est sans cycle. Il comporte donc au plus $n-1$ arêtes par hypothèse d'induction. Or le degré de x, $d(x)$ est tel que $d(x) \leq 1$, donc l'ensemble E' comporte au maximum $n-1$ arêtes plus une éventuelle arête reliant x à un autre sommet du graphe. Donc G a au plus n arêtes.

Question 2. Recherche d'un cycle. La propriété précédente ne permet pas d'obtenir un cycle dans le graphe non orienté. Vous allez déterminer une méthode pour cela.

a Proposez une méthode, basée sur un algorithme de parcours en profondeur, pour obtenir un cycle.

b Appliquez votre méthode sur le graphe de la figure 5, puis sur le même graphe dans lequel vous supprimez les arêtes (a, e) et (a, i).

Idée générale : quand on effectue un parcours en profondeur, si à partir d'un sommet x on trouve un sommet adjacent y dans l'état « en traitement »(gris) et que ce sommet est différent de celui d'où on vient (ie. le précédent dans la pile), un cycle est détecté. Pour trouver les sommets composant ce cycle, il suffit de vider la pile jusqu'à retrouver le sommet y.
Variante d'algo de parcours en profondeur (voir support de cours). On suppose que $Lire-Preced(Pile)$ renvoie l'avant dernier élément de la pile.
Attention : on détecte un cycle, mais pas forcément le plus petit cycle.

```
Require: G un graphe, s sommet de départ
Ensure: L liste des sommets du cycle détecté
  {Initialisation}
  for tous les sommet x de G do
    Etat(x) ← inexplore
  end for
  Creer_Pile(P)
  Empiler(P, s)
  L ← Liste_Vide
  {Parcours modifié}
  while not Est_Vide(P) do
    x ← Lire_Pile(P)
    if ∃y/Etat(y) == Blanc or Etat(y) == Gris then
      if Etat(y) == Blanc then
        Etat(y) ← Gris
      end if
      if Etat(y) == Gris and Lire_Preced(Pile) ≠ x then
        {Cycle detecte, on le memorise dans une liste L}
        Inserer(L, x)
        L ← Lire_Pile_jusqu_a(P, y)
      end if
    else
      Depiler(P)
      Etat(x) ← visite
    end if
  end while
```

2.3 Tri topologique

L'ensemble des composants (ie. chapitres) d'une formation peut être représenté par le graphe orienté de la figure 8. Dans ce graphe, les sommets représentent les enseignements et il y a un arc du sommet x vers le sommet y si le chapitre x est un pré-requis du chapitre y.

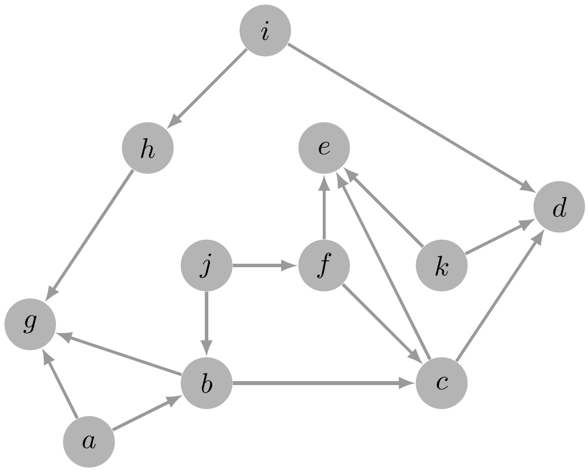

Figure 8 – Maquette d'une formation

Pour une meilleure lisibilité on souhaite ré-organiser la représentation du graphe de telle sorte que tous les arcs soient orientés de la gauche vers la droite.

Question 1. Schéma.

a Proposez un schéma avec cette ré-organisation Essayez mais ne pas y passer trop de temps ....

b Est-ce toujours possible ? Non, ce n'est pas possible si le graphe comporte un cycle

Tri topologique. Une telle représentation peut s'appuyer sur une relation d'ordre entre les sommets, appelée ordre topologique. Avec cette relation d'ordre, un numéro est associé à chaque sommet, tel que le numéro du sommet x est inférieur au numéro du sommet y s'il y a un arc de x vers y.

Question 2.

a Proposez une utilisation de la numérotation des sommets avec un parcours en profondeur pour obtenir un ordre topologique.

b Appliquez votre méthode à l'exemple et donnez l'ordre topologique obtenu (représentez la solution dans le tableau ci-dessous).

c Pour un graphe donné, l'ordre topologique des sommets est-il unique ? Justifiez.

d Peut-on obtenir un ordre topologique des sommets lorsque le graphe comporte un cycle ?

e Pourriez-vous utiliser la numérotation d'un parcours en largeur ?

1	2	3	4	5	6	7	8	9	10	11

Corrigé - Tri topologique. On peut appliquer un parcours en profondeur pour obtenir un tri topologique des sommets d'un graphe. En effet, dans un parcours en profondeur, l'ordre dans lequel les sommets sont marqués dans l'état `visité` est l'ordre inverse de la numérotation pour le tri topologique.

Attention, il faut démarrer le parcours d'un sommet sans prédécesseur. Ici il y en a 4 sur l'exemple (a, i, j et k). On peut ajouter un sommet fictif représentant l'origine et relié à ces 4 sommets. Appelons S un tel sommet et faisons démarrer le parcours de s.

L'arborescence obtenue avec le parcours DFS est donnée dans la fihure 9, la numérotation des sommets par DFS (et ordre alphabétique) est donnée dans le tableau 5, l'ordre topologique résultant est donné dans le tableau 6.

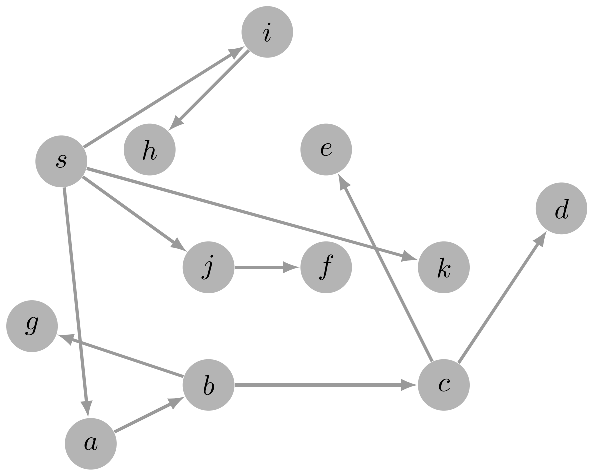

Figure 9 – Parcours DFS

1	2	3	4	5	6	7	8	9	10	11	(12)
d	e	c	g	b	a	h	i	f	j	k	s

Table 5 – ordre visité DFS

On prend les sommets dans l'ordre inverse de DFS pour avoir la numérotation de l'ordre topologique.

(0)	1	2	3	4	5	6	7	8	9	10	11
s	k	j	f	i	h	a	b	g	c	e	d

Table 6 – ordre topologique

Le graphe initial peut alors se représenter en suivant l'ordre topologique. Tous les arcs sont orientés de la gauche vers la droite : $num(x) < num(y)$ s'il existe un arc de x vers y. Le graphe obtenu est donné en figure 10.

Unicité de l'ordre topologique. La numérotation de l'ordre topologique obtenu dépend de la façon dont les sommets "en traitement" sont pris en compte : ici à chaque étape le parcours a exploré les sommets dans l'ordre alphabatique. Une autre exploration des sommets "en traitement" aurait conduit à un autre ordre topologique (correct bien sûr).

Ordre topologique en présence de cycle. On peut obtenir une numérotation des sommets en faisant un parcours dans un graphe comportant un cycle car il est toujours possible d'effectuer un parcours en profondeur

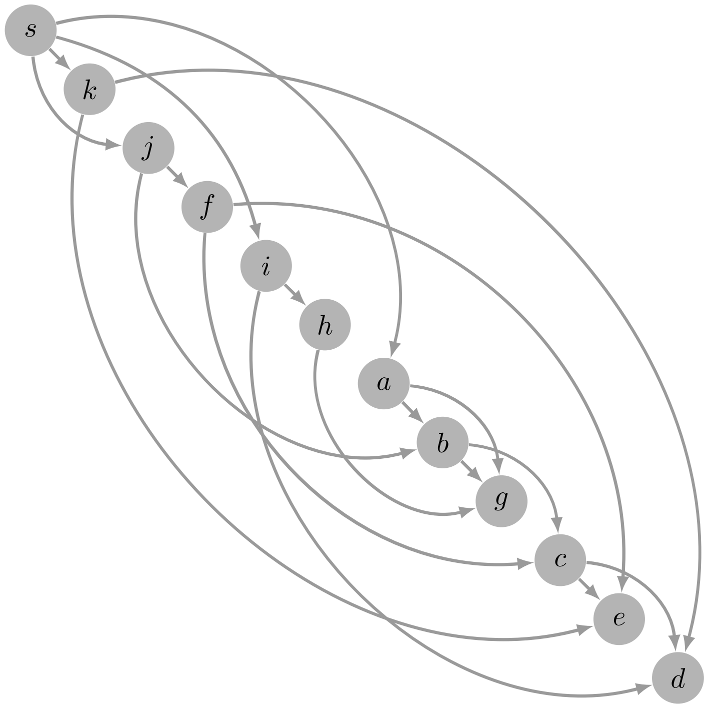

Figure 10 – Schéma ordre topologique

dans un graphe avec un cycle. **Cependant cette numérotation n'est pas un ordre topologique** : il y aurait des arcs orientés de la droite vers la gauche (en plus des arcs orientés de la gauche vers la droite).

La numérotation des sommets avec un parcours en largeur ne permet pas de fournir un ordre topologique (un sommet peut être numéroté avant certains de ses prédécesseurs). Sur l'exemple, essayez et vous verrez que le sommet g et numéroté avant le sommet h alors que h est un pré-requis pour g.

Avec l'exercice suivant (décomposition en niveaux), on va étudier une autre manière de représenter le graphe orienté acyclique des enseignements. Avec la décomposition en niveaux on peut obtenir un ordre topologique.