TD Algo2 – session 12 – Graphes orientés

28 mai 2025

Objectifs d'apprentissage :
— trouver des graphes orientés présentant des propriétés particulières ;
— concevoir des algorithmes relatifs aux graphes orientés.

> Les 5 premiers exercices sont essentiels. Plus d'exercices sur `https://algs4.cs.princeton.edu/42digraph/`.

Exercice 1 : Énumération

Combien y a t-il de graphes orientés avec 2 sommets et sans label sur les sommets ?

> Les graphes qui sont identiques à un repositionnement près ne doivent être comptés qu'une seule fois. Il ne faut pas oublier le graphe sans arc et les arcs qui vont d'un sommet sur lui-même. Cela en fait 10.

Exercice 2 : Mauvais arcs

Lorsqu'on exécute Tri-Topologique(G) sur un graphe orienté G contenant au moins un circuit, on pourrait espérer qu'il s'agit d'un ordre sur les sommets qui minimise le nombre de "mauvais" arcs, c'est-à-dire avec le moins possible d'arcs dans le mauvais sens. Trouver un exemple qui montre que ce n'est pas le cas.

> Avec moins de 4 sommets et un circuit, tous les tris topologiques se valent en terme de nombre de mauvais arcs. Il faut donc 4 sommets, a, b, c et d, avec les arcs (a, b), (b, c), (a, d), (d, c) et (c, a). On considère un parcours en profondeur qui part de c.

Exercice 3 : Nombre de chemins

Concevoir un algorithme en temps $O(V + E)$ qui prend en entrée un graphe orienté sans circuit $G = (V, E)$ et deux sommets s et t, puis qui retourne le nombre de chemins allant de s à t dans G. L'algorithme n'est pas obligé d'énumérer les chemins, il suffit de les compter.

> Compte-Chemin(G, s, t)
>
> **pour** chaque sommet $v \in G.V$ **faire**
> $v.chemin \leftarrow 0$
> $s.chemin \leftarrow 1$
> Tri-Topologique(G)
> **pour** chaque sommet $u \in G.V$ **faire**
> **pour** chaque sommet $v \in G.adj[u]$ **faire**
> $v.chemin \leftarrow v.chemin + u.chemin$
> **retourner** $t.chemin$

Exercice 4 : Chemin critique

Pour la planification de projets, on cherche à déterminer le chemin critique, c'est-à-dire le plus long

chemin d'un graphe orienté sans circuit. Par ailleurs, la représentation des tâches par des arcs va un peu contre l'intuition. Il serait plus naturel que les sommets représentent les tâches et que les arcs représentent les contraintes de séquencement ; autrement dit, l'arc (u, v) indiquerait qu'il faut faire la tâche u avant la tâche v. Les poids w seraient alors affectés aux sommets, et non aux arcs. Modifier la procédure PCC-GSS pour qu'elle trouve en temps $O(V + E)$ la longueur d'un plus long chemin dans un graphe orienté sans circuit à sommets pondérés.

```
PCC-GSS(G, w, s)
TRI-TOPOLOGIQUE(G)
pour chaque sommet v ∈ G.V faire
  v.d ← −∞
  v.π ← NIL
s.d ← w(s)
pour chaque sommet u ∈ G.V faire
  pour chaque sommet v ∈ G.adj[u] faire
    si v.d < u.d + w(u) alors
      v.d ← u.d + w(u)
      v.π ← u
retourner max(v.d)
```

Exercice 5 : Circuit hamiltonien

Concevoir un algorithme en temps $O(V + E)$ qui détermine s'il existe un chemin qui passe par chaque sommet exactement une fois dans un graphe orienté sans circuit.

Calculer un tri topologique et vérifier s'il existe un arc entre chaque paire de sommets successifs dans cet ordre.

Exercice 6 : Tri topologique itératif

Un autre moyen d'effectuer le tri topologique d'un graphe orienté sans circuit $G = (V, E)$ consiste à faire, de manière itérative, les opérations suivantes : trouver un sommet de degré entrant 0, l'imprimer, puis le supprimer du graphe ainsi que tous les arcs qui en partent. Expliquer comment implémenter cette idée par un algorithme à temps $O(V + E)$. Qu'advient-il de cet algorithme si G contient des circuits ?

```
TRI-TOPOLOGIQUE-ITÉRATIF(G)
Source ← ∅
Tri ← ∅
pour chaque sommet v ∈ G.V faire
  v.entrant ← 0
pour chaque arc (u, v) ∈ G.E faire
  v.entrant ← v.entrant + 1
pour chaque sommet v ∈ G.V faire
  si v.entrant = 0 alors
    ENFILE(Source, v)
tant que Source ≠ ∅ faire
  u ← DEFILE(Source)
  ENFILE(Tri, u)
  pour chaque sommet v ∈ G.adj[u] faire
    v.entrant ← v.entrant − 1
    si v.entrant = 0 alors
      ENFILE(Source, v)
retourner Tri
```

S'il y a des circuits, la boucle principale ne s'arrête jamais.

Exercice 7 : Tri topologique unique

Concevoir un algorithme qui détermine si un graphe orienté sans circuit possède un tri topologique unique.

Un graphe orienté sans circuit possède un tri topologique unique si et seulement s'il y a un arc entre chaque paire de sommets successifs dans cet ordre (c'est-à-dire s'il existe un circuit hamiltonien).

Exercice 8 : Plus court circuit

Concevoir un algorithme en temps $O(VE)$ pour trouver un circuit avec le plus petit nombre d'arcs dans un graphe orienté (ou indiquer que le graphe est sans circuit).

On pourrait s'en servir dans la situation suivante : des patients nécessitent une greffe de rein et chacun a un membre de la famille acceptant de donner un rein, mais du mauvais type. Ils sont d'accord pour donner leur rein à un autre patient si leur patient parent obtient un rein. L'hôpital peut ensuite réaliser des opérations en domino où toutes les greffes sont réalisées simultanément.
On lance un parcours en profondeur sur chaque sommet s. Le plus petit cycle en passant par s est obtenu en considérant l'arc de v vers s, puis le plus court chemin de s vers v.

Exercice 9 : Accessibilité

Concevoir un algorithme en temps $O(V + E)$ qui détermine si un graphe orienté sans circuit possède un sommet accessible à partir de tous les autres.

Calculer le degré sortant de chaque sommet. Si le graphe a exactement un seul sommet avec un degré sortant 0, alors celui-ci est accessible à partir de tous les autres sommets.

Exercice 10 : Chemin de longueur donnée

Concevoir un algorithme qui détermine s'il existe un chemin de longueur L entre deux sommets s et t dans un graphe orienté sans circuit.