TD Algo2 – session 9 – Plus court chemin

28 mai 2025

Objectifs d'apprentissage :
- trouver des exemples de graphes présentant des propriétés particulières relatives aux plus courts chemins ;
- analyser formellement les propriétés des plus courts chemins ;
- adapter et concevoir des algorithmes pour le problème du plus court chemin ;

> Les 6 premiers exercices sont essentiels. Plus d'exercices sur `https://algs4.cs.princeton.edu/44sp/`.

Exercice 1 : Constante

Donner un exemple simple qui montre qu'ajouter une constante à chaque arc change la solution du problème du plus court chemin.

> Trois sommets a, b et c avec $w(a, b) = w(b, c) = 1$ et $w(a, c) = 3$. Si on ajoute une constante 2 à chaque arc, le plus court chemin de a à c change.

Exercice 2 : Dijkstra et poids négatifs

Donner un exemple simple de graphe orienté comportant des arcs de poids négatif, sans circuit négatif, pour lequel l'algorithme de Dijkstra donne des réponses incorrectes. Pourquoi la démonstration n'est-elle plus valable quand on autorise des arcs de poids négatif ?

> Quatre sommets a, b, c et d avec $w(a, b) = w(b, d) = 1$ et $w(a, c) = -w(c, b) = 2$.
> On suppose à chaque étape que le sommet extrait est le sommet non traité le plus proche de l'origine. Cela permet de garantir que sa distance est bien celle du plus court chemin et d'éviter de reparcourir ses arcs. Avec des poids négatifs, ce n'est plus le cas.

Exercice 3 : Sous-structure optimale

Montrer que chaque sous-chemin sur un plus court chemin de v à w est aussi un plus court chemin entre ses deux extrémités.

> Supposons qu'un sous-chemin entre a et b sur un plus court chemin de v à w ne soit pas un plus court chemin. Il existe donc un plus court chemin entre a et b de plus petit poids. On peut donc réduire la longueur du plus court chemin de v à w en passant par celui-ci, ce qui est une contradiction car un plus court chemin ne peut pas être réduit.

Exercice 4 : Fiabilité

Soit un graphe orienté $G = (V, E)$ pour lequel chaque arc $(u, v) \in E$ possède une valeur $w(u, v)$ réelle vérifiant $0 \leq w(u, v) \leq 1$. Cette valeur représente la fiabilité du canal de communication entre le sommet u et le sommet v. On interprète $w(u, v)$ comme étant la probabilité pour que le canal entre u et v ne soit pas interrompu, et on suppose que ces probabilités sont indépendantes. Donner un algorithme efficace permettant de trouver le chemin le plus fiable entre deux sommets donnés.

On peut représenter les fiabilités avec $w(u, v) = \log F$ où F est la fiabilité de l'arc, puis utiliser la procédure DIJKSTRA.

On peut aussi ré-écrire la procédure en considérant que $w(u, v) = F$. Dans ce cas, on initialise $v.d$ à 0 (1 pour l'origine). Le relâchement devient :

RELÂCHER(u, v, w)

si $v.d < u.d \times w(u, v)$ **alors**
  $v.d \leftarrow u.d \times w(u, v)$
  $v.\pi \leftarrow u$

Et il faut utiliser une file de priorités max plutôt que min.

Exercice 5 : Chemin négatif

Modifier l'algorithme de Bellman-Ford pour qu'il donne à $v.d$ la valeur $-\infty$ pour tous les sommets v pour lesquels il existe un circuit de longueur strictement négative sur un certain chemin entre l'origine et v.

On rajoute $|V|$ passages supplémentaires sur les arcs avec ce relâchement :

RELÂCHER(u, v, w)

si $v.d > u.d + w(u, v)$ **alors**
  $v.d \leftarrow -\infty$

Ou alors, trouver un sommet sur un circuit négatif et lancer un parcours en largeur/profondeur dessus.

Exercice 6 : Vérification

Le professeur Gaedel a écrit un programme qui, selon lui, implémente l'algorithme de Dijkstra. Le programme produit $v.d$ et $v.\pi$ pour chaque sommet $v \in V$. Donner un algorithme en temps $O(V + E)$ pour vérifier la sortie du programme du professeur. Il doit déterminer si les attributs d et π correspondent à ceux d'un arbre de plus courts chemins. Vous pouvez supposer que tous les poids des arcs sont non négatifs.

On parcours tous les arcs et on vérifie si un relâchement est possible. Si c'est le cas, c'est que l'on n'a pas un plus court chemin. Il faut aussi vérifier que le parent est cohérent.

Exercice 7 : Arbre de plus courts chemins

Donner un exemple de graphe orienté pondéré $G = (V, E)$, de fonction de pondération $w : A \to R$ et d'origine s, qui satisfasse à la propriété suivante : pour tout arc $(u, v) \in E$, il existe un arbre de plus courts chemins de racine s qui contient (u, v) et un autre qui ne contient pas (u, v).

Exercice 8 : Circuit négatif

Soit $G = (V, E)$ un graphe orienté pondéré de sommet origine s, initialisé par SOURCE-UNIQUE-INITIALISATION(G, s). Démontrer que si une séquence d'étapes de relâchement donne à $s.\pi$ une valeur autre que NIL, alors G contient un circuit de longueur strictement négative.

Exercice 9 : Optimisation de Dijkstra

Supposons que la boucle principale de l'algorithme de Dijkstra s'exécute $|V| - 1$ fois au lieu de $|V|$ fois. L'algorithme ainsi modifié est-il correct ?

Oui, car le dernier sommet traité est le plus éloigné et aucune relaxation ne peut donc être réalisée à partir de celui-ci.

Exercice 10 : Optimisation Dijkstra (bis)

Modifier la procédure DIJKSTRA pour que la file de priorités Q fonctionne davantage comme la file dans un parcours en largeur en ne contenant que les sommets qui ont été atteints depuis l'origine à ce stade : $Q \subseteq V \setminus S$ et $v \in Q$ implique $v.d \neq \infty$.

Exercice 11 : Optimisation Bellman-Ford

On considère un graphe orienté pondéré $G = (V, E)$ sans circuit négatif. Pour tous les couples de sommets $u, v \in V$, on compte le nombre d'arcs minimal de tous les plus courts chemin de u à v. (Ici, "plus court" signifie de poids minimal et ne concerne pas le nombre d'arcs). Soit m le maximum de ces nombres d'arcs. Suggérer une modification simple à l'algorithme de Bellman-Ford, lui permettant de se terminer après $m + 1$ passages.

Vérifier que chaque itération réalise au moins un changement et s'arrêter sinon. Si l'on peut atteindre chaque sommet avec un plus court chemin contenant au maximum m arcs, alors les valeurs $v.d$ ne changeront pas après m itérations.

```
BELLMAN-FORD-OPTIMISÉ(G, w, s)
SOURCE-UNIQUE-INITIALISATION(G, s)
répéter
   modif ← FAUX
   pour chaque arc (u, v) ∈ G.E faire
      si RELÂCHER'(u, v, w) alors
         modif ← VRAI
jusqu'à modif = FAUX
```

```
RELÂCHER'(u, v, w)
si v.d > u.d + w(u, v) alors
   v.d ← u.d + w(u, v)
   v.π ← u
   retourner VRAI
sinon
   retourner FAUX
```

Comme il n'y a pas de circuit négatif, la partie détection a été retirée.

Exercice 12 : Circuit négatif (bis)

Supposons qu'un graphe orienté pondéré $G = (V, E)$ comporte un circuit de longueur strictement négative. Donner un algorithme efficace permettant de lister les sommets d'un tel circuit.

On peut utiliser Bellman-Ford, marquer les sommets qui sont dans un circuit négatif, puis explorer de parent en parent jusqu'à ce qu'un circuite se forme.

Exercice 13 : Plus long chemin

Concevoir un algorithme pour le problème du plus long chemin dans un graphe orienté sans circuit.