STRUCTURES DE DONNÉES ET ALGORITHMES · CHAPITRE 17

Passer des relations aux listes d’adjacence

Distinguez sommet, arête, orientation, poids et connexité. Comparez matrice et listes d’adjacence avec une convention explicite de numérotation.

Par Ali ZAINOUL · Cours gratuit

Ce que vous allez apprendre

- Choisir une représentation des sommets et de leurs relations.
- Orientation, poids et convention d’absence font partie de la définition des données.
- Les listes d’adjacence utilisent Θ(n + m) emplacements ; la matrice utilise Θ(n²).

Prérequis. Avoir étudié les chapitres DSA précédents ; connaître files, piles et structures indexées.

Partir d’un besoin

Un réseau de routes ne possède pas nécessairement une racine ni un parent unique : plusieurs chemins peuvent relier les mêmes lieux. Un arbre ne suffit plus à représenter toutes ces relations.

Définir la structure et les conventions

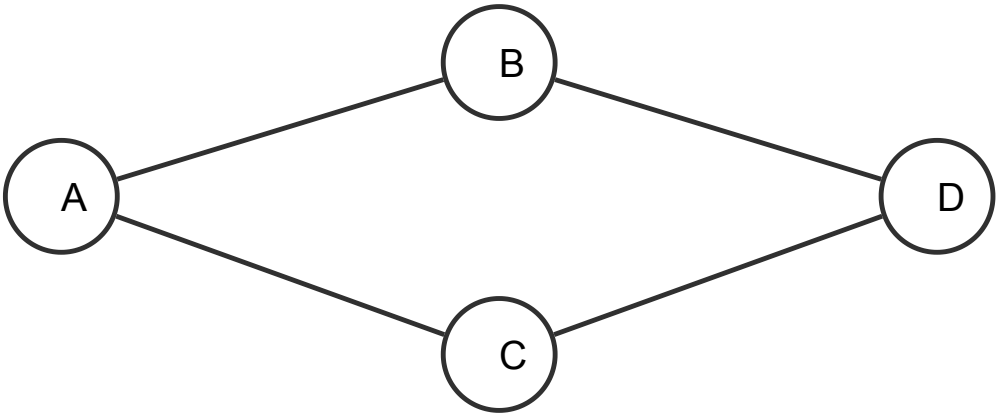
Un **graphe** G = (V, E) contient des sommets V et des arêtes E. Nous notons n = |V| leur nombre et m = |E| le nombre d’arêtes. Un graphe **orienté** distingue u vers v de v vers u. Une arête peut porter un **poids**, par exemple une durée. Les premiers parcours utiliseront des arêtes non pondérées.

Un **chemin** est ici une suite de sommets reliés par des arêtes successives. Un cycle revient à son départ. Dans un graphe non orienté, une composante connexe rassemble les sommets accessibles les uns depuis les autres. Nous étudions d'abord un graphe simple, sans boucle d'un sommet vers lui-même ni arête répétée.

Une **liste d'adjacence** donne, pour chaque sommet, ses voisins. Une **matrice d'adjacence** stocke n × n entrées : l'entrée de ligne u et colonne v indique la présence d'une arête. Pour le graphe non orienté A-B, A-C, B-D, C-D, chaque arête apparaît dans deux listes de voisins.

Python numérote A, B, C, D par 0, 1, 2, 3 ; MATLAB par 1, 2, 3, 4. Python utilise une liste de listes. MATLAB utilise un tableau de cellules, créé avec des accolades ; neighbors{node} extrait le vecteur de voisins de node. Cette convention sera conservée dans les parcours.

Suivre les états



Graphe non orienté : les arêtes relient A à B et C, puis B et C à D. Deux chemins de deux arêtes relient A à D.

Voisins du réseau non orienté

Sommet	Python	MATLAB
A	[1, 2]	[2, 3]
B	[0, 3]	[1, 4]

Sommet	Python	MATLAB
C	[0, 3]	[1, 4]
D	[1, 2]	[2, 3]

Comparer les réalisations

Le graphe contient quatre sommets et quatre arêtes. Le degré d’un sommet non orienté est son nombre de voisins. La somme des degrés vaut deux fois le nombre d’arêtes dans cette représentation sans boucle. Cette division par deux ne s’applique pas telle quelle à un graphe orienté.

Pseudo-code

```
// example.pseudo
// adjacency_lists
// A: B, C. B: A, D. C: A, D. D: B, C.
// Undirected graph: each edge is listed from both endpoints.
// Sum of degrees = 8; number of edges = 8 / 2 = 4.
```

Python

```python
# adjacency_lists.py
neighbors = [[1, 2], [0, 3], [0, 3], [1, 2]]
n = len(neighbors)
degree_sum = sum(len(row) for row in neighbors)
edges = degree_sum // 2
print(n, edges)
```

MATLAB

```
% adjacency_lists.m
neighbors = {[2, 3], [1, 4], [1, 4], [2, 3]};
n = numel(neighbors);
degree_sum = 0;
for node = 1:n
    degree_sum = degree_sum + numel(neighbors{node});
end
edges = degree_sum / 2;
disp([n, edges]);
```

Justifier et compter

Les listes d'adjacence occupent Θ(n + m) emplacements et permettent de parcourir les voisins de u en Θ(degré(u)). Une matrice utilise Θ(n²) emplacements et permet un test direct d'arête en Θ(1). Chercher un voisin dans une liste non triée coûte O(degré(u)) au pire. Un graphe dense peut rendre la matrice pertinente.

Prévoir puis expérimenter

Ajoutez un sommet isolé : une liste vide supplémentaire. n augmente mais m reste identique. Représentez ensuite seulement A vers B : la liste inverse B vers A ne doit pas être ajoutée si le graphe est orienté.

Exercice corrigé

▼ **Pourquoi une matrice de zéros est-elle ambiguë pour un graphe pondéré ?**

Si zéro signifie absence d'arête, on ne peut plus représenter sans autre convention une arête de poids nul. Il faut séparer absence et valeur du poids.

À retenir

- Orientation, poids et convention d'absence font partie de la définition des données.
- Les listes d'adjacence utilisent $\Theta(n + m)$ emplacements ; la matrice utilise $\Theta(n^2)$.
- Dans notre graphe non orienté simple, chaque arête apparaît deux fois dans les voisinages.

Version web : https://www.clearcode.fr/courses/programming/dsa/graphs/representations