STRUCTURES DE DONNÉES ET ALGORITHMES · CHAPITRE 18

Explorer par distance croissante avec BFS

Utilisez une file pour visiter un graphe non pondéré par couches. Calculez les distances, marquez à la découverte et traitez les sommets inaccessibles.

Par Ali ZAINOUL · Cours gratuit

Ce que vous allez apprendre

- Calculer des distances non pondérées avec une file FIFO.
- BFS explore avec une file FIFO et découvre les sommets par distance en nombre d'arêtes.
- Marquer à la découverte évite plusieurs insertions du même sommet.

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

Partir d'un besoin

Dans le réseau A-B, A-C, B-D, C-D, combien d'arêtes faut-il traverser au minimum pour aller de A à D ? Une exploration par couches trouve d'abord les sommets à distance un, puis ceux à distance deux.

Définir la structure et les conventions

Le **parcours en largeur**, ou BFS (Breadth-First Search), utilise une file FIFO. La distance est ici le nombre d'arêtes d'un plus court chemin depuis la source, pas une longueur géographique. Les graphes peuvent être orientés : seuls les voisins sortants sont alors parcourus.

distance vaut -1 pour un sommet non découvert. La source reçoit 0 avant son ajout à la file. Lorsqu'un voisin est découvert, il reçoit la distance du sommet courant plus un et est marqué avant d'être enfilé. Chaque sommet entre ainsi au plus une fois dans la file, même si plusieurs chemins y mènent.

Les listes d'adjacence reprennent les conventions du chapitre précédent. La file Python est une deque. MATLAB préalloue n cases : chaque sommet n'étant enfilé qu'une fois, cette capacité suffit. head et tail délimitent sa partie active ; aucune suppression en début de vecteur n'est nécessaire.

Suivre les états

BFS depuis A

Sommet retiré	Nouveaux sommets	File ensuite
A	B et C à distance 1	[B, C]
B	D à distance 2	[C, D]
C	Aucun (D déjà découvert)	[D]
D	Aucun	[]

Comparer les réalisations

Les distances de A, B, C et D sont [0, 1, 1, 2] dans les deux langages. Les identifiants des sommets diffèrent, pas ces distances. Enregistrez breadth_first_demo.m avec la fonction locale. Le contrat exige une source valide et des indices de voisins valides ; le graphe lui-même peut être vide seulement si aucun appel de BFS ne demande une source.

Pseudo-code

```
// example.pseudo
// breadth_first
// Mark the source at distance 0 and enqueue it.
// While the queue is not empty, dequeue node.
// For each undiscovered neighbor:
// set distance to distance[node] + 1, then enqueue it.
// Mark at discovery, not at removal from the queue.
```

Python

```python
# breadth_first.py
from collections import deque
def bfs(neighbors, source):
    distance = [-1] * len(neighbors)
    distance[source] = 0
    queue = deque([source])
    while queue:
        node = queue.popleft()
        for neighbor in neighbors[node]:
            if distance[neighbor] == -1:
                distance[neighbor] = distance[node] + 1
                queue.append(neighbor)
    return distance
neighbors = [[1, 2], [0, 3], [0, 3], [1, 2]]
distance = bfs(neighbors, 0)
print(distance)
```

MATLAB

```
% breadth_first_demo.m
neighbors = {[2, 3], [1, 4], [1, 4], [2, 3]};
distance = bfs(neighbors, 1);
disp(distance);
function distance = bfs(neighbors, source)
    n = numel(neighbors);
    distance = -ones(1, n);
    distance(source) = 0;
    queue = zeros(1, n);
    head = 1;
    tail = 1;
    queue(tail) = source;
    while head <= tail
        node = queue(head);
        head = head + 1;
        for neighbor = neighbors{node}
            if distance(neighbor) == -1
                distance(neighbor) = distance(node) + 1;
                tail = tail + 1;
                queue(tail) = neighbor;
            end
        end
    end
end
```

Justifier et compter

Avec listes d’adjacence, initialisation comprise, le temps est O(n + m) : chaque sommet est enfilé au plus une fois et chaque entrée de voisinage est examinée au plus une fois depuis les sommets atteints. La mémoire auxiliaire vaut O(n). Avec une matrice, examiner tous les voisins de chaque sommet peut coûter O(n²). Le plus court chemin trouvé minimise les arêtes uniquement dans un graphe non pondéré, ou lorsque tous les poids sont égaux et positifs.

Prévoir puis expérimenter

Ajoutez un sommet isolé : sa distance reste -1. Inversez l'ordre des voisins de A : les distances restent identiques, mais les chemins choisis en cas d'égalité peuvent changer.

Exercice corrigé

▼ Pourquoi marquer un voisin avant de l'enfiler ?

B et C peuvent tous deux découvrir D. Le marquage immédiat empêche d'enfiler D deux fois ; le premier chemin découvert par couches a déjà une distance minimale.

À retenir

- BFS explore avec une file FIFO et découvre les sommets par distance en nombre d'arêtes.
- Marquer à la découverte évite plusieurs insertions du même sommet.
- La garantie O(n + m) dépend des listes d'adjacence ; elle ne décrit pas un parcours de tous les voisins par matrice.

Version web : https://www.clearcode.fr/courses/programming/dsa/graphs/breadth-first