STRUCTURES DE DONNÉES ET ALGORITHMES · CHAPITRE 19

Explorer une branche avant de revenir avec DFS

Tracez les appels récursifs d'un DFS, marquez les sommets visités et distinguez accessibilité, ordre de parcours et détection de cycle.

Par Ali ZAINOUL · Cours gratuit

Ce que vous allez apprendre

- Explorer les sommets accessibles sans relancer un sommet marqué.
- DFS poursuit une branche avant de revenir aux choix précédents.
- Marquer avant les appels empêche les cycles de relancer indéfiniment une exploration.

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

Partir d'un besoin

Pour explorer toutes les relations accessibles, on peut suivre une branche aussi loin que possible puis revenir aux choix précédents. Quel état empêche de tourner sans fin sur un cycle ?

Définir la structure et les conventions

Le **parcours en profondeur**, ou DFS (Depth-First Search), termine l'exploration d'un voisin avant de passer au suivant. La récursivité mémorise les sommets en attente. Un tableau visited marque un sommet dès l'entrée de son appel, avant de poursuivre vers ses voisins.

Le DFS présenté répond à une question d'accessibilité : quels sommets peut-on atteindre depuis la source ? Il ne calcule pas nécessairement les plus courtes distances. L'ordre des visites dépend de l'ordre des listes de voisins. Dans le réseau A-B, A-C, B-D, C-D, il visite A, B, D, C avec les voisinages choisis.

Python utilise une fonction interne visit : elle partage la liste visited de l'appel dfs. MATLAB transmet puis renvoie le vecteur de booléens à chaque retour ; la valeur renvoyée conserve les marques acquises pendant les sous-appels. true et false représentent les états logiques. Le contrat exige une source et des voisinages valides.

Suivre les états

Descente et retours du DFS

Appel	Action
A	Marquer A puis appeler B
B	Marquer B ; ignorer A marqué ; appeler D
D	Marquer D ; ignorer B ; appeler C
C	Marquer C ; ses voisins sont marqués ; remonter

Comparer les réalisations

Les quatre sommets sont marqués. La sortie indique l'ensemble atteint, pas l'ordre de visite de la trace. Enregistrez depth_first_demo.m avec sa fonction locale. Le marquage garantit qu'aucun sommet ne déclenche deux explorations complètes.

Pseudo-code

```
// example.pseudo
// depth_first
PROCEDURE VISIT(node)
    visited[node] <- TRUE
    FOR EACH neighbor IN neighbors[node] DO
        IF NOT visited[neighbor] THEN
            VISIT(neighbor)
        END IF
    END FOR
END PROCEDURE
```

Python

```
# depth_first.py
def dfs(neighbors, source):
    visited = [False] * len(neighbors)
    def visit(node):
        visited[node] = True
        for neighbor in neighbors[node]:
            if not visited[neighbor]:
                visit(neighbor)
    visit(source)
    return visited
neighbors = [[1, 2], [0, 3], [0, 3], [1, 2]]
visited = dfs(neighbors, 0)
print(visited)
```

MATLAB

```
% depth_first_demo.m
neighbors = {[2, 3], [1, 4], [1, 4], [2, 3]};
visited = false(1, numel(neighbors));
visited = visit(1, neighbors, visited);
disp(visited);
function visited = visit(node, neighbors, visited)
    visited(node) = true;
    for neighbor = neighbors{node}
        if ~visited(neighbor)
            visited = visit(neighbor, neighbors, visited);
        end
    end
end
```

Justifier et compter

Le parcours abstrait et la version Python examinent O(n + m) entrées, avec O(n) mémoire de marques et d'appels au pire. La profondeur peut atteindre n : les limites de récursion doivent être prises en compte. La version MATLAB qui transmet visited par valeur peut engendrer des copies supplémentaires, jusqu'à O(n²) en temps et mémoire retenue dans un cas en chaîne ; ne lui attribuez pas automatiquement les mêmes garanties. Une implémentation itérative avec stockage maîtrisé peut éviter ces copies récursives.

Prévoir puis expérimenter

Ajoutez un sommet isolé : il reste non marqué. Pour parcourir toutes les composantes, démarrez un DFS depuis chaque sommet encore non marqué, en conservant le même état de visites.

Exercice corrigé

▼ Un voisin déjà visité prouve-t-il à lui seul la présence d'un cycle dans un graphe non orienté ?

Non : l'arête qui ramène au parent apparaît aussi dans sa liste de voisins. Détecter un cycle demande de distinguer cette arête des autres retours ; dans un graphe orienté, on distingue notamment les sommets encore dans la pile des appels.

À retenir

- DFS poursuit une branche avant de revenir aux choix précédents.
- Marquer avant les appels empêche les cycles de relancer indéfiniment une exploration.
- L'accessibilité ne donne pas une distance minimale ; les copies de marques et la profondeur des appels restent à compter.

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