Explorattion des graphes et applications

GRE

3 - Exploration des graphes et application

Abstract

Definition

Table des matières

1. BFS (exploration en largeur)

L'exploration en largeur est un algorithme permettant de parcourir un graphe.

- utilise une liste FIFO
- permet de calculer les plus courtes chaînes (ou chemin pour les graphes orientés) d'un sommet à tous les autres sommets de sa composante

1.1. Idée

1. On commence par le sommet de départ
2. Tous les voisins directs sont ajoutés à la liste (souvent dans l'ordre alphabétique)
3. On traite successivement les sommets de la liste en y ajoutant à chaque fois les voisins directs non découverts
4. On continue jusqu'à ce que la liste soit vide

Note
Dans un graphe non pondéré la distance entre deux sommets est le nombre d'arêtes du plus court chemin entre ces deux sommets.

1.2. Complexité

1.2.1. Exécution

$$O(n+m)$$

1.2.2. Mémoire

$$O(n) + \\ O(n+m) \quad \text{pour stocker le graphe}$$

1.3. Applications

LE BFS est utilisé pour :

- Calcul des plus courtes chaînes (nombre d'arêtes) depuis un sommet vers tous les autres sommets d'un graphe non orienté et non pondéré
- Calcul des plus courts chemins (nombre d'arcs) depuis un sommet vers tous les autres sommets d'un graphe orienté et non pondéré
- Calcul des composantes (faiblement) connexes
  - Si le graphe est non orienté, toute exploration depuis s (en largeur, en profondeur ou autre) visite tous les sommets de la composante connexe de s (et uniquement eux). On détermine toutes les composantes connexes en multipliant les explorations tant qu'il existe des sommets non découverts.
  - Si le graphe est orienté, la méthode est la même sauf que lors du traitement d'un sommet on doit parcourir sa liste de successeurs et sa liste de prédécesseurs. Souvent il faudra commencer par construire les listes de prédécesseurs à partir des listes de successeurs !
- Recherche d'un cycle, circuit, le plus court possible passant par un sommet donné

2. DFS (exploration en profondeur)

L'exploration en profondeur est un algorithme permettant de parcourir un graphe. Dans cet algorithme, on explore le plus loin possible le long de chaque branche avant de revenir en arrière. De manière générale il s'agit d'un algorithme récursif. Dans un cas non récursif, on utilise une liste LIFO (Last In First Out) pour stocker les sommets à explorer.

2.1. Idée

1. On commence par le sommet de départ
2. On traite le sommet le plus proche (si graphe pondéré) ou le sommet suivant dans l'ordre alphabétique
3. On continue jusqu'à ce que l'on ne puisse plus avancer
4. On revient en arrière et on traite le sommet suivant
5. On continue jusqu'à ce que la liste soit vide

2.2. Complexité

$$O(n+m)$$

2.3. Applications

- Calcul des composantes (faiblement) connexes d'un graphe
- Recherche d'un cycle ou circuit passant par un sommet d'un graphe
- Tri topologique d'un graphe orienté sans circuits
- Calcul des composantes fortement connexes d'un graphe orienté

3. Kosaraju

L'algorithme de Kosaraju est un algorithme permettant de calculer les composantes fortement connexes d'un graphe orienté.

3.1. Idée

1. Construire le graphe transposé (inversion des arcs)
2. Effectuer une exploration en profondeur du graphe transposé
3. Utiliser les dates de fin de traitement de l'expoloration pour ordonner les sommets dans l'ordre décroissant de leur date de fin
4. Effectuer une exploration du graphe initial en utilisant la liste précédente pour le choix des racines de chaque exploration
5. Les sommets de chacune des arborescences construitent lors de ce second parcours définissent les composantes fortement connexes du graphe initial

3.2. Applications

- Calcul des composantes fortement connexes d'un graphe orienté

4. Tarjan

L'algorithme de Tarjan est un algorithme permettant de calculer les composantes fortement connexes d'un graphe orienté en une seule exploration.

4.1. Idée

On utilise 3 tableaux :

- *dfsnum* : numéros de découverte des sommets
- *low* : stocke le plus petit numéro que l'on peut attenindre depuis *u* **pour l'instant**
- *scc* : numéro de la composante fortement connexe du sommet *u*

1. On numérote les sommets au fur et à mesure de leur visite (*dfsnum* correspond à la date de début de traitement)

4.2. Applications

- Calcul des composantes fortement connexes d'un graphe orienté