Algorithmes de base de théorie de graphes

N. TSOPZE

Département d'Informatique - Université de Yaoundé I

Définition

graphe

$G = (V, E)$ où V est l'ensemble fini des sommets et E l'ensemble des arêtes $E \subset V \times V$

G est non orienté si $\forall (x, y) \in E$, $(y, x) \in E$ et non orienté sinon

degré d'un sommet $v \in V$

nombre d'arêtes de E incidentes en v

Le degré du graphe G est le maximum des degrés d'un sommet.

chemin

Soient u, v deux sommets de V, un chemin de u à v est un ensemble d'arêtes e_1, e_2, . . . ,e_k de E telles que $e_i = (v_{i-1}, v_i), v_0 = u$ et $v_k = v$

Si tous les v_i sont distincts, le chemin est élémentaire.
Si $u = v$, le chemin (élémentaire) est un cycle (élémentaire).

composant connexe

sous ensemble $V_1 \subset V$ t.q. $\forall u, v \in V_1$ il existe toujours un chemin de u à v

définitions

arbre

graphe connexe sans cycle

théorème

Soit G un graphe à n sommets et m arêtes.
On a l'équivalence :

1. G est connexe sans cycle (ie G est un arbre)
2. G est connexe et minimal pour cette propriété (la suppression d'une arête implique G n'est plus connexe)
3. G est connexe et $m = n - 1$
4. G est sans cycle et maximal pour cette propriété (l'ajout d'une arête crée un cycle dans G)
5. Toute paire de sommets est reliée par un chemin élémentaire et un seul

Représentation

Deux approches:

1. Liste de successeurs ie $Succ(u) = \{v \ (u, v) \in E\}$
2. Matrice d'adjacence:

$$M_{i,j} = \begin{cases} 0 & (i,j) \notin E \\ 1 & (i,j) \in E \end{cases}$$

3. Matrice d'incidence

problèmes

1. Accessibilité
2. plus court chemin:
   1. plus court chemin entre une source u et une destination v;
   2. tous les plus courts chemins entre une source u et tous les autres sommets du graphe
   3. tous les plus courts chemins entre les sommets du graphe pris deux à deux
3. composants connexes

accessiblité

accès

y est dit accessible à partir du sommet x s'il existe un chemin de x à y

Parcours en largeur

découvrir dans le graphe tous les sommets accessibles depuis s.
découvrir d'abord tous les sommets situés à une distance k de s avant de découvrir tout sommet situé à la distance $k+1$.

1. calcul la plus petite distance (en nombre d'arcs) entre s et tous les sommets accessibles.
2. construction d'une arborescence de parcours en largeur de racine s, ayant tous les sommets accessibles depuis s.
3. Pour tout sommet v accessible depuis s, le chemin reliant s à v dans l'arborescence de parcours en largeur correspond à un plus court chemin de s vers v dans G.

Algorithme de PL

1. pour chaque sommet $u \in V - \{s\}$ faire
   1. $couleur(u) \leftarrow BLANC$,
   2. $d(u) \leftarrow \infty$; $parent(u) \leftarrow NIL$
2. $couleur(s) \leftarrow GRIS$; $d(s) \leftarrow 0$; $parent(s) \leftarrow NIL$; $F \leftarrow \{s\}$;
3. tant que $F \neq \emptyset$ faire
   1. $u \leftarrow tete(F)$
   2. pour chaque $v \in Adj(u)$ faire
      si $couleur(v) = BLANC$ alors
      1. $couleur(v) \leftarrow GRIS$
      2. $d(v) \leftarrow d(u) + 1$
      3. $parent(v) \leftarrow u$
      4. $ENFILE(F, v)$
      5. $DEFILE(F)$
   3. $couleur(u) \leftarrow NOIR$

Exemple: livre de Cormen Page 519

Soit M la matrice d'adjacence de $G = (V, E)$. On considère ses coefficients comme des entiers. soit M^p la puissance p-ème de M. Le coefficient M^p_{ij} est égal au nombre de chemins de longueur p de v_i à v_j
$G^+ = (V, A^+)$ défini par $(i, j) \in A^+ \Leftrightarrow$ il y a un chemin non vide de v_i à v_j dans G

- $M^+ = M + M^2 + ... + M^n$ est la fermeture transitive
- $M^* = I + M^+$ est la fermeture réflexive et transitive
- Coût de calcul de M^+ en $O(n^4)$ car on doit faire n multiplications de matrices
- Coût de calcul de M^* en $O(n^3 logn)$ car on peut calculer $logn$ élévations successives au carré de $I + M$

Algo. de Warshall

Proposition

La récurrence $A_{ij}^{p} = A_{ij}^{p-1} + A_{ip}^{p-1} \times A_{pj}^{p-1}$ avec $A^0 = M$, calcule $M^+ = A^n$

Algorithme:

1. $A \leftarrow M$
2. pour p allant de 1 à n faire
   pour i allant de 1 à n faire
   pour j allant de 1 à n faire
   $A_{ij} \leftarrow A_{ij} + A_{ip}A_{pj}$

Plus court chemin - Algorithme de Dijkstra

Calcul du plus court chemin à partir d'une source donnée dans un graphe pondéré ayant les poids positifs
On note $\Gamma(i) = \{j/(i,j) \in E\}$.
Algorithme de Dijkstra

1. $S \leftarrow \{1\}$
2. $d(i) = w(1,i)$ si $i \in \Gamma(1)$, et $+\infty$ sinon
3. tant que $S \neq V$ faire
   1. Soit j tel que $j \notin S$ et $d(j) = min_{i \in S} d(i)$
   2. $S \leftarrow S \cup \{j\}$
   3. pour tout $k \in \Gamma(j)$ vérifiant $k \notin S$
      $d(i) \leftarrow min\{d(i), d(k) + w(k,i)\}$

algo. de Bellman

$d_{t,k}$ = distance du sommet source s à t avec un chemin qui contient au plus k arcs
W est la matrice des poids
Par programmation dynamique on a:

1. $d_{t,0} = +\infty$ pour $t \neq s$ et $d_{s,0} = 0$
2. $d_{t,k} = min(d_{t,k-1}, min_{arc_{u,t}}(d_{u,k-1} + w_{u,t}))$

arbre couvrant de poids minimal

But

permet de transformer le graphe en arbre afin d'éviter les cycles lors du parcours

Algorithme de Kruskal pour les graphes valués:

1. Lire la liste des arêtes commençant par l'arête de plus faible poids (trier dans l'ordre croissant).
   $T = \{u_1\}$
2. A l'étape k lire l'arête u_k (de poids faible), si elle ne forme pas de cycle dans T alors ajouter u_k à T sinon passer à l'arrête suivante.
   Si T a $N-1$ arrêtes alors STOP.

graphe exemple