925 : Graphes : représentations et algorithmes

Bibliographie :
FOURNIER - Théorie des graphes et applications
CORMEN

Motivations : permet de modéliser ≠ systèmes, ...
exs : cartes, ...

Notations : Soit $G=(S,A)$ un graphe

I - Représentation de graphes [FOURNIER p33]

1 - Matrice d'adjacence

Définition : matrice carrée d'ordre $n=|S|$.
$m_{ij} = \begin{cases} 1 & \text{si } (i,j) \in A \\ 0 & \text{sinon} \end{cases}$

Place mémoire : $O(|S|^2)$

Intérêt : lorsque G graphe dense

EXEMPLE :

$$\begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

2 - Tableau d'adjacence

Définition : Tableau de taille $|S|$ T, tel que $T[i]$ = Adj(i) l'ensemble des voisins de i.

Place mémoire : $O(|S|+|A|)$ EX : 10 → 20 → 30
20 → 10 → 30 → 40
30 → 20 → 10
40 → 20
50

Intérêt : lorsque G peu dense

3 - Liste d'arêtes

Définition : liste d'arêtes implémentée par un tableau ou une liste chaînée.

Place mémoire : $O(|A|)$

Intérêt : place mémoire minimale.

EXEMPLE :

arêtes	extremités
1	2 1
2	3 1
3	2 3
4	2 4

II - Parcours de graphes

1 - Parcours en largeur [CORMEN 549]

Entrée : $G=(S,A)$ et un sommet origine s.

Principe : Visiter tous les sommets situés à une distance k de s avant de découvrir les sommets situés à la distance $k+1$.

Outil : Utilisation d'une file.

PL(G,s)

pour chaque sommet $u \in S \setminus \{s\}$ } Initialisation des sommets de $S \setminus \{s\}$
  $u.couleur = BLANC$
  $u.d = \infty$
  $u.\pi = NIL$
$s.couleur = GRIS$ } Initialisation du sommet s
$s.d = 0$
$s.\pi = NIL$
$F = \emptyset$
ENFILER(F,s)
tant que $F \neq \emptyset$ } Etude du sommet de la file.
  $u \leftarrow$ DEFILER(F)
  pour chaque $v \in Adj[u]$
    si $v.couleur == BLANC$
      $v.couleur = GRIS$
      $v.d = u.d + 1$
      $v.\pi = u$
      ENFILER(F,v)
  $u.couleur = NOIR$

EXEMPLE : cf ANNEXE

Complexité : implémentation par liste d'adjacence : $O(|S|+|A|)$
par matrice d'adjacence : $O(|S|^2)$

Application : Calcul du plus court chemin dans un graphe non pondéré.

plein d'applications ?

2- Parcours en profondeur [COR]

Entrée : $G=(S,A)$

Principe : Descendre plus "profondément" dans le graphe chaque fois que c'est possible.

Outil : utilisation d'une pile

```
PP(G)
    pour chaque sommet u∈S                    } Initialisation
        u.couleur = BLANC
        u.π = NIL
    date = 0
    pour chaque sommet u∈S                    } Etude de chaque
        si u.couleur == BLANC                 } sommet BLANC.
            VISITER-PP(G,u)

VISITER-PP(G,u)
    date = date + 1                           } Initialisation
    u.d = date                                } de l'étude
    u.couleur = GRIS
    pour chaque v∈Adj[u]                      } Etude
        si v.couleur == BLANC
            v.π = u
            VISITER-PP(G,v)
    u.couleur = NOIR                          } Clôture de l'étude
    date = date + 1                           } du sommet u.
    u.f = date
```

EXEMPLE : Annexe

Complexité : liste d'adjacence $O(|S|+|A|)$
matrice d'adjacence $O(|S|^2)$

Applications :
- Tri topologique
- Composantes fortement connexes [DEV]
- Théorie des jeux [FOU]

3- Autres types de parcours

* Circuit eulérien

Définition Un graphe connexe $G=(S,A)$ admet un circuit eulérien s'il existe un circuit qui traverse chaque arête une fois exactement.

Théorème (EULER) Un graphe connexe G tq $m \geq 1$ est eulérien (ssi) tous ses sommets sont de degrés pairs.

Conséquence : L'algorithme de Rosenstiehl a une complexité $O(|A|)$ avec une implémentation par listes d'adjacence.

* Circuit hamiltonien

Définition Un cycle hamiltonien dans un graphe G est un cycle qui passe par tous les sommets du graphe.

Définition Le problème du voyageur de commerce (VC) Trouver un cycle hamiltonien de valeur totale minimale dans un graphe G valué par $v : A \to \mathbb{R}^+$.

Théorème Le problème d'existence d'un circuit hamiltonien et le problème VC sont NP-complets

III - Arbre couvrant minimal

1. Présentation du problème

Description du problème $G=(S,A)$ graphe connexe muni d'une fonction de pondération $w : A \to \mathbb{R}^+$
On souhaite trouver un sous-ensemble acyclique $T \subseteq A$ qui connecte tous les sommets et dont le poids total $w(T) = \sum_{(u,v)\in T} w(u,v)$ est minimal.
T est appelé arbre couvrant minimal de G

Outil : approche gloutonne qui respecte l'invariant de boucle suivant :

I = « Avant chaque itération, E est un sous-ensemble d'un certain arbre couvrant minimal »

Application : Connecter toutes les villes d'une région avec des routes pour un coût minimal (proportionnel à la distance des routes).

2. Algorithme de Kruskal

Principe E est une forêt contenant tous les sommets du graphe.

```
KRUSKAL(G, w)
    E = ∅
    pour chaque sommet v ∈ S
        CREER_ENSEMBLE(v)
    trier les arêtes de A par ordre croissant de w
    pour chaque (u,v) ∈ A
        si TROUVER_ENS(u) ≠ TROUVER_ENS(v)
            E = E ∪ {(u,v)}
            UNION(u,v)
    retourner E
```

Complexité (admis) : $O(|A| \log(|S|))$

EXEMPLE : Annexe

3. Algorithme de Prim

Principe E est un arbre couvrant minimal pour un sous-graphe de G.

DEVELOPPEMENT :
- présentation de l'algorithme de Prim
- **Application** : Approximation du problème du voyageur de commerce euclidien.

IV - Plus courts chemins

Définition : poids de plus court chemin entre u et v ∈ S :

$$\delta(u,v) = \begin{cases} \min\{w(p) : u \rightsquigarrow v\} \\ +\infty \end{cases}$$

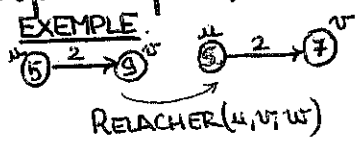
1 - A Origine unique

Problème : On souhaite trouver un plus court chemin dans un graphe G depuis un sommet d'origine $s \in S$ vers chaque sommet $v \in S$

Outil : processus de relâchement d'un arc (u,v) : peut-on améliorer le PCC de s à v en passant par u ?

```
RELACHER(u, v, w)
    si v.d > u.d + w(u,v)
        v.d = u.d + w(u,v)
        v.π = u
```

EXEMPLE



* Algorithme de DIJKSTRA

Soit G = (S, A) un graphe orienté pondéré où tous les arcs ont des poids positifs.

[DEVELOPPEMENT] Présentation de l'algo

Complexité : $O(S \log S + A)$

Remarque : Lorsqu'on a des poids négatifs, on peut utiliser l'algorithme de Bellman-Ford → de complexité $O(SA)$

2 - Entre toutes paires de sommets

Outil : programmation dynamique = Floyd-Warshall

[DEV] Présentation de l'algo Cf leçon ?

Complexité : $O(n^3)$

Application : Fermeture transitive d'un graphe orienté.

On n'a plus de place mais il faudrait aussi parler de :
- flots
- problèmes NP-complet sur les graphes
- graphe planaire, graphe d'intervalle
- ...

la liste est si longue...

Cf leçon 901

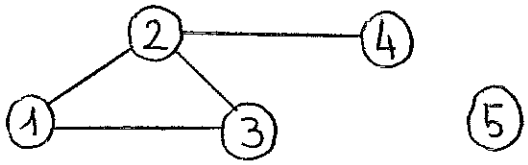
Cf prog dyna

Cf 926

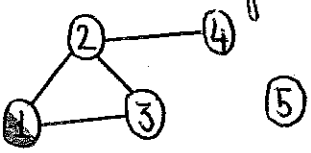
Couvert

Cf 915

Annexe

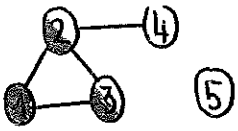


Parcours en largeur

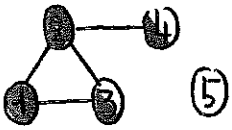


Evolution de la file :

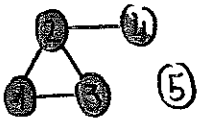
1



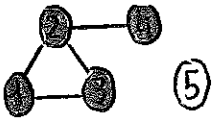
3	2



4	3

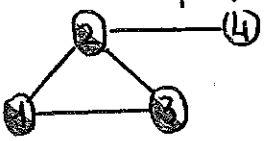


4

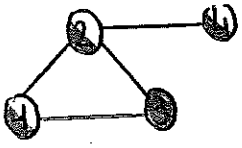


ϕ

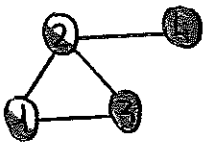
Parcours en profondeur



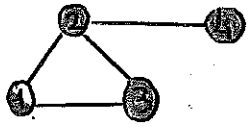
3
2
1



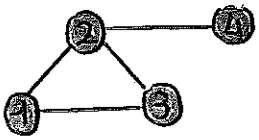
4
2
1



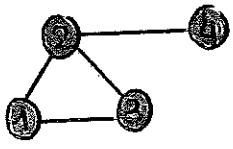
2
1



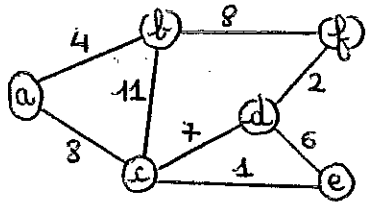
1



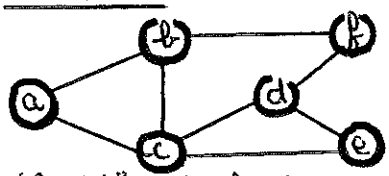
5

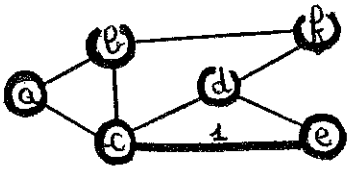


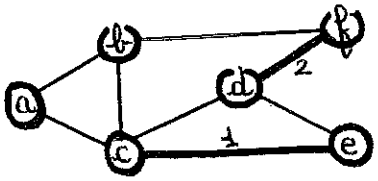
ϕ



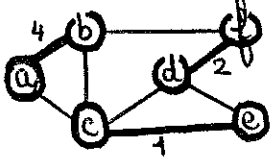
Kruskal

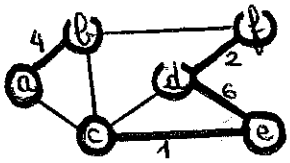




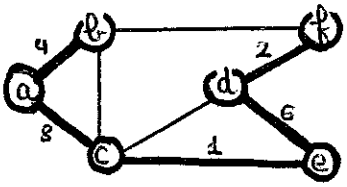


Création des forêts





si on ajoute (c,d) on crée un cycle, on étudie donc l'arête suivante



= algo de Kosaraju

Composantes fortement connexes

But : Calculer les composantes fortement connexes dans un graphe orienté.

CFC (G)
    Calculer PP(G)
    Calculer G^T
    Calculer PP(G^T) en traitant les sommets par date de fin du premier parcours décroissantes
↳ les arbres de la forêt calculés par PP(G^T) sont alors les composantes fortement connexes de G.

Complexité : $O(S+A)$

Correction :

Lemme 1 : v est un descendant de u dans une forêt de parcours en profondeur ssi et seulement si à la date de début de visite de u il existe un chemin de u à v composé de sommets blancs.

preuve :
⇒] Si $v = u$, c'est immédiat
Si $v \neq u$, on remarque à la ligne 5-6 de Visiter PP que le père d'un sommet dans l'arbre est toujours un sommet ayant une date de

Début strictement inférieur.

Donc $u.d < v.d$ et donc v est blanc à l'instant $u.d$

De même tous les sommets entre u et v dans l'arbre de parcours en profondeur ont $u.d < w.d$ et sont donc blancs au moment $u.d$.

Donc le chemin de u à v dans l'arbre est blanc.

$\Leftarrow$:

Supposons qu'il existe un chemin blanc de u à v mais que v ne soit pas un descendant de u dans l'arbre.

Quitte à changer v, on peut supposer que tous les sommets du chemin blanc autres que v sont des descendants de u dans l'arbre.

Notons w le prédécesseur de v dans le chemin blanc.

Comme w est un descendant de u, on finit de visiter w avant de finir de visiter u et donc

$$w.f \leq u.f$$

Comme on a $u.d < v.d$, v est blanc, on a $u.d < v.d$

De plus comme $v \in Adj[w]$, on visite v avant de finir les visites de w et donc $v.d < w.f$

D'où $$u.d < v.d < w.f \leq u.f$$

Donc v est parcouru entre $u.d$ et $u.f$ donc c'est un descendant de u dans l'arbre $\square$

Lemme 2 : Soit $C \neq C'$ deux composantes fortement connexes de G. Soit $u \in C$ et $v \in C'$

(i) Si $(u,v) \in A$, alors $f(C) \geq f(C')$

(ii) Si $(u,v) \in A^T$, alors $f(C) < f(C')$

avec $f(C) = \max_{v \in C} \{v.f\}$

(de même $d(C) = \min_{v \in C} \{v.d\}$)

preuve :

(i) Si $d(C) < d(C')$

Soit x le premier sommet découvert de C.

En $x.d$, tous les sommets de C et C' sont blancs

$\Rightarrow \forall y \in C$, il existe un chemin blanc de x à y

$\forall y \in C'$, il existe ——————— de x à y

car $(u,v) \in A$ et u et v sont blancs en $x.d$

par le lemme 1, tous les sommets de C et C' sont des descendants de x dans le parcours en profondeur et donc x est le dernier sommet de C et C' dont on termine la visite.

$\Rightarrow x.f = f(C) > f(C')$

Si $d(C) > d(C')$

Soit y le premier sommet découvert de C' par le même raisonnement qu'avant tous les sommets de C' sont des descendants de y et à l'instant $y.d$ tous les sommets de C sont blancs.

Comme $C \neq C'$ et $C \neq C'$, il n'y a pas d'arc partant de C' et allant vers C et donc aucun sommet de C n'est accessible depuis y. Et donc au moment $y.f$ les sommets de C sont encore tous blancs

Donc $\forall y \in C$, $y.f > y.f = f(C')$

$\Rightarrow f(C) > f(C')$

(ii) G^T et G ont les mêmes composantes fortement connexes et on applique (i) à G^T

Proposition : CFC calcule bien les composantes fortement connexes.

preuve : Par récurrence sur le nombre d'arbre de parcours découverts.

On suppose que les k premiers arbres construits sont des CFC.

k = 0 : immédiat

Supposons que les k premiers arbres sont des CFC.
Soit v la racine du (k+1) arbre trouvé
Notons C la CFC de v.
Par hypothèse de récurrence, aucun des sommets de C n'a déjà été visité en v.d (car sinon ils seraient présents dans les k premiers arbres)
Donc en v.d, tous les sommets de C sont blancs.
Donc par le lemme 1, tous les sommets de C sont des descendants de v dans l'arbre de parcours.
De plus pour toute composante fortement connexe C' ≠ C restant à découvrir, on a $v.f = f(C) > f(C')$ par choix de l'ordre des sommets dans $L(G^T)$
Donc le lemme 2 implique que tout arc de G^T partant de C mène à une des k premières CFC.
Donc les C' restant à découvrir ne sont pas accessibles depuis C, donc leurs sommets ne sont pas des descendants de v.
Finalement tous les sommets de C sont des descendants de v et uniquement eux.
⇒ le k+1 arbre est une CFC (C' ≠ C)