Bouquin 925

DEV 1 : Tri topologique

Pb : Soit V un ensemble fini et $<$ une relation binaire sur V^2, bien fondée.
L'algorithme tri topologique permet d'obtenir une numérotation des éléments de V, en temps $O(|V|+|R|)$, où $|R|$ est le nombre de couples d'éléments en relation, vérifiant :
$\forall (x_i, x_j) \in V^2$, $x_i \, R \, x_j \Rightarrow i \leq j$.

Modélisation :
par un graphe $G=(V,E)$ avec $E = \{(u,v) \in V^2, \, u \, R \, v\}$.

Lemme 1 : G est acyclique :
Supposons que G a un cycle $x_1 \to x_2 \to \dots \to x_m \to x_1$, avec $m \geq 2$.
Alors $\{x_1, \dots, x_m\} \subset V$ et fini.
R étant bien fondée, $\exists i \in \{1, \dots, m\}$ x_i n'a pas d'antécédant dans $\{x_1, \dots, x_m\}$
or (en notant $x_0 = x_m$), $x_{i-1} \, R \, x_i$ car $x_{i-1} \to x_i \in E$
$\to$ absurde.

Algorithme : Tri topologique :
Entrée : graphe $G=(V,E)$ orienté acyclique
Sortie : liste des sommets de G, tq si u apparaît avant v dans cette liste alors $v \not R u$

$l = []$
Appeler PP(G)
A chaque fois qu'un sommet est marqué en noir, l'ajouter à l
Renvoyer l.

Lemme 2 : $G=(V,E)$ graphe acyclique, $(u,v) \in V^2$, $u \neq v$.
les intervalles $[d(u), f(u)]$ et $[d(v), f(v)]$ sont disjoints ou inclus l'un dans l'autre.
On peut supposer $d(u) < d(v)$ par symétrie, car à chaque attribution, date augmente.
* si $d(v) < f(u)$
$\Rightarrow$ v a été découvert quand u était gris
$\Rightarrow$ v est un descendant de u
vu qu'on termine PP-visiter(G,v) avant de pouvoir terminer PP-visiter(G,u)
$\Rightarrow$ $f(v) < f(u)$
$\Rightarrow$ $[d(v), f(v)] \subset [d(u), f(u)]$
* sinon : $d(u) < f(u) < d(v) < f(v)$ car $\forall w$, $d(w) < f(w)$ toujours par croissance de date
$\Rightarrow$ $[d(u), f(u)]$ et $[d(v), f(v)]$ sont disjoints.

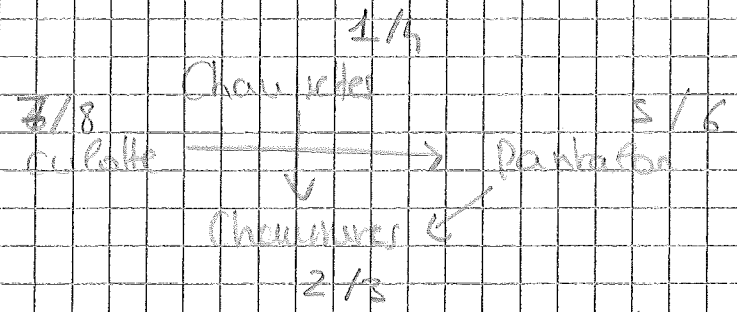
Lemme 3 : Tri topologique est correct
(1) Si u apparaît avant v dans l alors $f(u) > f(v)$ et réciproquement.
$\to$ ok car l'ajout dans la liste et le choix de la valeur de f se font au même moment.
(2) Si $u \, R \, v$ alors u apparaît avant v cad si $u \, R \, v$ alors $f(u) > f(v)$
Toutes les arêtes sont parcourues dans l'algorithme.
Au moment du parcours de (u,v) :
* v est blanc, donc u est gris $\Rightarrow$ $d(v) \in [d(u), f(u)] \Rightarrow [d(v), f(v)] \subset [d(u), f(u)]$
$\Rightarrow$ $f(v) < f(u)$
* v est gris, avec u gris $\Rightarrow$ v était gris avant de commencer u $\Rightarrow$ v est un ancêtre de u $\Rightarrow$ on a un cycle $v \rightsquigarrow u \to v$, or G est acyclique.
* v est noir, avec u gris $\Rightarrow$ $f(v)$ est déjà déterminé, mais pas $f(u)$.

Par croissance de date, $f(v) < f(u)$.

Complexité : exactement celle de parcours en profondeur.
PP-Visiter est appelé exactement une fois pour chaque sommet
car appelé uniquement si le sommet est blanc, puis le marque en gris.
La boucle de PP-Visiter est exécutée $|Adj(u)|$ fois si on est dans PP-Visiter(u).
Les autres opérations sont en $O(1)$.
vu que $\sum_{v \in V} |Adj(v)| = |E|$

$\Rightarrow O(|V| + |E|)$.

Exple



$\rightarrow$ culotte, Pantalon, chaussettes, chaussures

Applica° : ordonnancement de tâche

DEV2 : Algorithme de Bellman-Ford

Pb : Pour $G=(V,E)$ un graphe pondéré par $w : E \to \mathbb{R}$, l'algorithme de Bellman-Ford détermine si le graphe G a un cycle de poids négatif, et pour $s \in S$, renvoie le tableau d des distances minimales de tout sommet à s, en temps $O(|V||E|)$.

Principe : l'algorithme maintient un tableau d de distance minimale entre tout sommet u et s et considère chaque arête, l'une après l'autre dans un ordre quelconque, pour faire diminuer les distances. Il le fait $|V|-1$ fois, c'est à dire la longueur maximale d'un chemin de $u \in V$ à s, sans cycle. Enfin, il teste si une distance peut encore être raccourcie pour savoir s'il y a un cycle.

Algorithme :

Entrée : $G=(V,E)$ un graphe pondéré par w et $s \in V$.

Sortie : False si G contient un cycle de poids négatif accessible depuis s.
True sinon, et il renvoie alors le tableau d des distances et π des prédécesseurs.

```
BF(G,w,s)
    pour tout u ∈ V
        d(u) = +∞
        π(u) = NIL
    d(s) = 0
    pour i = 1 à |V|-1
        pour (u,v) ∈ E
            si d(v) > d(u) + w(u,v)
                d(v) = d(u) + w(u,v)
                π(v) = u
    pour (u,v) ∈ E
        si d(v) > d(u) + w(u,v)
            renvoyer Faux
    renvoyer Vrai
```

Reconstruction :

Entrée : π tableau des prédécesseurs, $s \in V$ et $t \in V$ pour lequel $d(t) \neq +\infty$

Sortie : liste l des sommets d'un chemin de poids minimal de s à t

```
CheminBF(π,s,t)
    l = []
    u = t
    tant que u ≠ s
        Ajouter(l,u)
        u = π(u)
    Ajouter(l,u)
    renvoyer l
```

Complexité :

1e boucle pour : $O(|V|)$
4e boucle pour : $O(|E|)$
3e boucle pour : $O(|E|)$
2e boucle pour : $O(|V||E|)$

$\}$ $\to O(|V||E|)$ pour BF

pour CheminBF : $O(|V|)$.

Rq $\forall i$, $\delta(u_i, u_{i+1}) = \omega(u_i, u_{i+1})$

Sinon ($\exists i$ $u_i \to v_1 \to \dots \to v_k \to u_{i+1}$, $u_i \to u_{i-1} \to v_1 \to \dots \to v_k \to u_{i+1} \to \dots \to u_1$)

Correction : on note $\delta(u,v)$ la distance minimale de u à v.

* si G n'a pas de cycle négatif

(1) : $\forall u \in V$, $d(u) = \delta(s,u)$: soit $s = u_0 \to u_1 \to \dots \to u_n = u$ un chemin de poids minimal.

Par récurrence, après k itérations, $d(u_k) = \delta(s, u_k) = \sum_{i=1}^{k} \omega(u_{i-1}, u_i)$ et $u_0 \to \dots \to u_k$ de poids min.

$k=1$: $s \to u_1$ est un chemin de poids minimal de s à u_1,

et vu que $s \to u_1 \in E$, on l'a considéré dans la 1e boucle,

donc $d(u_1) = \delta(s, u_1) = \sum_{i=1}^{1} \omega(u_{i-1}, u_i)$.

$1 \leq k \leq n-1$, $u_0 \to \dots \to u_n$ est un chemin de poids min de s à u_n donc

$u_0 \to \dots \to u_k \to u_{k+1}$ est un chemin de poids min de s à u_{k+1}.

Après k itérations, $d(u_k) = \delta(s, u_k) = \sum_{i=1}^{k} \omega(u_{i-1}, u_i)$

et $d(u_{k+1}) \geq d(u_k) + \omega(u_k, u_{k+1})$ (sinon $u_0 \rightsquigarrow u_{k+1}$ n'est pas de poids min)

$\Rightarrow$ Après $k+1$ itérations :

$d(u_{k+1}) = d(u_k) + \omega(u_k, u_{k+1})$ (sinon $u_0 \rightsquigarrow u_{k+1}$ n'est pas de poids min)

$= \delta(s, u_{k+1}) = \sum_{i=1}^{k+1} \omega(u_{i-1}, u_i)$

Donc $d(u)$ vaut exactement $\delta(s,u)$ après n itérations.

A la fin, $d(u) = \delta(s,u)$: d ne peut pas augmenter et

$d(u) < \delta(s,u)$ est impossible car $\delta(s,u)$ poids min.

(2) BF renvoie Vrai :

$\forall (u,v) \in E$, $d(v) = \delta(s,v) \leq \delta(s,u) + \omega(u,v)$ inégalité triangulaire.

$\leq d(u) + \omega(u,v)$.

* si G a un cycle de poids négatif, accessible depuis s, BF renvoie Faux.

Soit $(u_1, \dots, u_n)$ un cycle accessible depuis s de poids négatif avec $u_0 = u_n$.

$\sum_{i=1}^{n} \omega(u_{i-1}, u_i) < 0$

Si BF renvoie vrai $\Rightarrow \forall i \in \{1, \dots, n\}$, $d(u_i) \leq d(u_{i-1}) + \omega(u_{i-1}, u_i)$

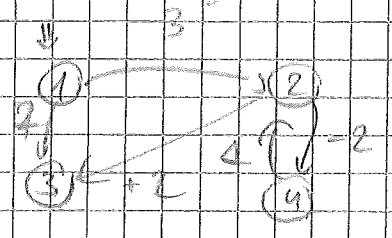
$\Rightarrow \sum_{i=1}^{n} d(u_i) \leq \sum_{i=1}^{n} d(u_{i-1}) + \sum_{i=1}^{n} \omega(u_{i-1}, u_i)$

$\Rightarrow 0 \leq \sum_{i=1}^{n} \omega(u_{i-1}, u_i) < 0$ $\to$ absurde

$\Rightarrow$ BF renvoie Faux.

Terminaison : boucles for

Exple



d	1	2	3	4
0	0	~~+~~	~~+~~	~~+~~
1	0	3	7	~~+~~
2	0	3	5	1
3	0	2	5	1
4	0	2	5	0

$\Rightarrow$ FAUX

Cormen (925)

DEV 3 - Algorithme de Ford-Fulkerson

Pb : Soit $G=(V,E)$ un réseau de capacité c, et $s \in V$ une source et $t \in V$ un puits.
On cherche à trouver un flot maximum dans ce réseau.

Principe : On commence par le flot nul. Puis on cherche un chemin p de s à t dans le réseau résiduel (chemin augmentant) dont la capacité minimale est $c_f(p)$. On rajoute alors cette augmentation suivant p dans le flot, et on recommence.

Méthode de Ford-Fulkerson :

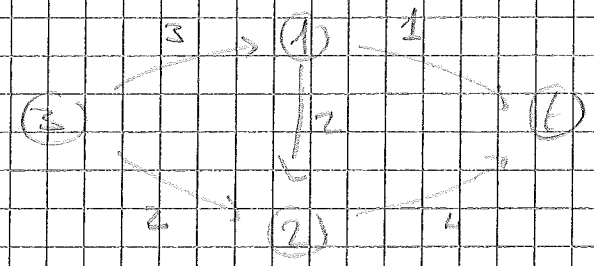
Entrée : réseau $G=(V,E)$ de capacité c, $(s,t) \in V^2$
Sortie flot maximum dans G de s à t : f
FF(G,s,t)
pour tout $(u,v) \in E$
  $f(u,v)=0$
  $f(v,u)=0$
tant qu'il existe un chemin p de s à t dans G_f
  $c_f(p) = \min\{c_f(u,v),\ (u,v) \text{ dans } p\}$
  pour toute (u,v) de p
    $f(u,v) = f(u,v) + c_f(p)$
    $f(v,u) = -f(u,v)$
renvoyer f

→ exple ?

Rq : ne termine pas forcément si les capacités sont irrationnels.
⇒ se ramener aux rationnels, puis aux entiers
→ chaque augmentation augmente au moins de 1.
et si $m = \sum_{u \in V} f(s,u)$ alors on peut augmenter m fois au plus.

Principe : chercher le chemin p avec un parcours en largeur pour un plus court chemin.
⇒ Algorithme d'Edmonds-Karp

Exple



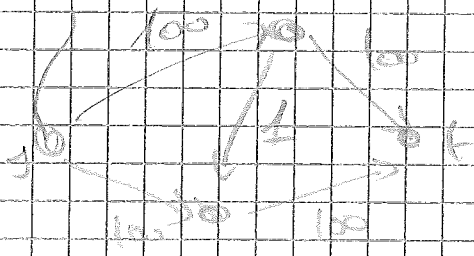
+ court chemin en nbre d'arrêtes

Problème de retournement des arrêtes

de G_f → G_f : $\downarrow 5$ + flot $\downarrow 2$

G_f $\uparrow 2$ $\downarrow 3$

Edmond-Karp → compl en $O(|V||E|^2)$

De base → compl en $O(|V||F^*|)$

Exple (non Edmond Karp) où c'est lent à cuger