

Algorithmique Avancée - TD 3 : corrigé - 25 octobre 2010

Université Paris 7 — BioInformatique — M1

Exercice 1 Au tableau...

Exercice 2 On note $dist(a, b)$ la “vraie” distance de a à b dans le graphe. La correction découle de :
Lemme : si

1. il y a un arc xy , et
2. $d[x] = dist(r, x)$, et
3. x est situé juste avant y sur au moins un plus court chemin de r à y

alors après $relax(x, y)$ $d[y] = dist(r, y)$.

Démonstration : Le point 3 (qui au passage implique le point 1...) implique aussi que $dist(r, y) = dist(r, x) + l(x, y)$. Or avant $relax$ $d[y]$ est la longueur d’un chemin entre r et y donc est supérieur ou égal à $dist(r, y)$. Si strictement supérieure, $relax$ la met à jour correctement, CQFD

Invariant : S’il existe un plus court chemin de r à y faisant au plus k arcs, alors à la fin du k ème tour de la boucle “Pour i ” $d[y] = dist(r, y)$.

Démonstration : par récurrence. Avant le premier tour c’est vrai pour r qui est relié à lui-même par un chemin de 0 arcs. Plaçons-nous à la fin du tour k . On considère un plus court chemin de r à y d’au plus k arcs. S’il en fait moins que k alors il est déjà correct car on a depuis le tour précédent $d[y] = dist(r, y)$ et $d[y]$ ne pourra plus bouger (il ne pourrait que diminuer, mais c’est la longueur d’un chemin : ne deviendra pas plus court que la longueur du plus court chemin!). S’il en fait k exactement, on considère le prédécesseur x de y sur ce chemin. Il est relié à r par un plus court chemin de $k - 1$ arcs (puisque tout sous-chemin d’un plus court chemin est un plus court chemin). Donc $d[x] = dist(r, x)$. On applique le lemme.

Un plus court chemin est sans boucle : il fait donc au plus $n - 1$ arcs. L’invariant au tour $n - 1$ donne donc la correction.

1 Dijkstra

Exercice 3 aussi au tableau

Exercice 4 Dans l’arbre \iff a un père. Mais le père peut changer pour les sommets atteints et est définitif pour les parcourus

- parcouru : a été extrait de A
- atteint : a été mis dans A mais pas encore extrait
- non-atteint : pas encore passé par A

Rappels sur les tas

- *À quoi sert un tas ?* C’est une structure de données qui permet d’avoir le minimum d’un ensemble géré de façon dynamique. On appelle aussi cela une file de priorité.
- *Comment on l’implémente en machine ?* C’est un arbre binaire tel qu’un nœud est plus petit que ses fils. On le stocke dans un tableau de taille n . Les fils de $T[i]$ sont $T[2i]$ et $T[2i + 1]$

- Comment on modifie la valeur d'une clé? Par une descente (échange avec un fils) ou une remontée (échange avec le père) **diminuer_cle(A,x,c** modifie dans A la clé c de l'élément x .
- Comment on ajoute une nouvelle clé On crée une nouvelle feuille (au seul endroit possible dans un arbre binaire complet) puis remontée. **insérer(A,x,c)** ajoute à A un élément x de clé c
- Comment on extrait le minimum du tas C'est la racine. On la remplace par la dernière feuille, puis descente de celle-ci.

Exercice 5 .

```

A = {r}
pour tout x ≠ r faire
  d[x] = +∞
d[r] = 0
A=creer_tas_vide()
insérer(A,r,0)
tant que non est_vide(A) faire
  x=extraire_min(A)
  pour tout voisin y de x faire
    si d[x] + l(x,y) < d[y] alors
      d[y] = d[x] + l(x,y)
      si y ∉ A, c'est-à-dire si pere[y] = null alors
        insérer(A,y,d[y])
      sinon
        diminuer_cle(A,y,d[y])
        pere[y] = x

```

2 Longueurs négatives

Exercice 6 Pas de cycle de longueur négative!

- nécessaire car sinon à chaque tour on diminue la longueur. Infinité de tours = longueur de $-\infty$!
- suffisant : si chaque cycle a longueur positive un plus court chemin est sans cycle à l'intérieur. Il y a un nombre fini de tels chemin (moins que factorielle de n) donc la distance, qui est la longueur du plus petit, est bien définie.

Exercice 7 Considérer l'exemple suivant : $V = \{a,b,c,d,e\}$ et $w(a,b) = 10, w(a,c) = 1, w(b,d) = -15, w(d,c) = 1, w(c,e) = 1$. On voit alors que Dijkstra s'arrête avant d'avoir propagé une information ...

Pour la preuve de l'algo on a vraiment besoin que les sommets soient insérés par distances croissantes, ce qui n'est plus possibles (les cercles de distance ne sont plus imbriqués les uns dans les autres)

Exercice 8 On a vu plus haut qu'un plus court chemin, s'il existe, est sans boucle. Il fait donc moins de n arcs. l'invariant vu en cours reste vrai : après le tour k tout plus court chemin de $\leq k$ arcs est correct. Donc en $n - 1$ tours on aura trouvé tous les plus courts chemins. Ce qui veut dire que si d baisse encore, il y a un chemin de longueur infinie. Inversement si d converge (relax ne change plus rien) alors $[x] = dist(x,y)$ puisqu'en cas de cycle négatif, à chaque tour au moins un d baisse dans le cycle.

D'où l'algorithme :

```
pour tout  $x \neq r$  faire
└─  $d[x] = +\infty$ 
 $d[r] = 0$ 
pour  $i$  de 1 à  $n$  faire
└─ pour tout arc  $(x, y) \in E$  faire
    └─  $\text{relax}(x, y)$ 
pour tout arc  $(x, y) \in E$  faire
└─ si  $d[x] + l(x, y) < d[y]$  alors
    └─ Échec : cycle négatif quelque part
Succes : renvoyer  $d$ 
```

3 Taux de change optimal

Exercice 9 simplement $c(A_1, A_2) * c(A_2, A_3) * \dots * c(A_{k-1}, A_k)$.

Exercice 10 S'il y a un cycle dans le graphe donc le produit des taux est strictement plus que 1 : on se retrouve avec plus d'argent qu'au départ ; il n'y a plus qu'à recommencer encore et encore !

Exercice 11 On pose $\text{taux}(M)$ le meilleur taux de change connu pour changer M en A (0 si inconnu ; $\text{taux}(A) = 1$)

Il faut juste comparer

- $\text{Taux}(B)$
 - $\text{taux}(C) * c(C, B)$
- et garder le max.

Exercice 12 Bellman-Ford-Change(G: graphe, w: fonction de pondération, s:sommet)
soit n le nombre de sommets de G
soit $\text{taux}[]$ un tableau de n entiers initialise a 0

```
 $\text{taux}[A]=1$ 
pour  $i$  de 1 a  $n$  faire
  pour chaque arc (B,C)
    // relaxation
    si  $\text{taux}[C] < \text{taux}[B] * c(B,C)$ 
       $\text{taux}[C] = \text{taux}[B] * c(B,C)$ 

pour chaque arc (B,C)
  si  $\text{taux}[C] < \text{taux}[B] * c(B,C)$ 
    retourner Faux
retourner vrai
```

C'est correct car en posant $d(u, v) = -\log(c(u, v))$ on se ramène a un problème de plus courts chemins ! (minimiser et + au lieu de maximisez et *)

Exercice 13 Ca ne marche pas des que des taux sont > 1 , d'après ce qui précède.

Exemple : $c(A, B) = 2$ $c(B, C) = 2$ $c(A, C) = 3$ $c(C, D) = 1$

Dijkstra calcule faussement un taux de 3 (au lieu de 4) pour changer A en D .