Algorithmique II
Année 2007-2008

Jean-François Pineau
Jean-Baptiste Rouquier

TD 5
Drôle de graphe

1 Animaux

Exercice 1 Un graphe $G = (X, E)$ à n sommets, non orienté, est un graphe *scorpion* s'il existe trois sommets x, y, et z, tels que
- $d(x) = 1$ (x est le dard)
- $d(y) = 2$ et $xy \in E$ (y est la queue)
- $d(z) = n - 2$ et $yz \in E$ (z est la tête)

Les autres sommets constituent le corps.

1. Quels sont les nombres minimum et maximum d'arêtes d'un graphe scorpion ?
2. Montrer que s'il existe trois sommets a, b, et c tels que $ab \in E$, $bc \in E$, $d(a) = 1$, $d(b) = 2$ et $d(c) \neq n - 2$ alors le graphe n'est pas scorpion.
3. En déduire un algorithme en $O(n)$ pour reconnaître les graphes scorpions.

Solution :

1. *Le dard et la tête on un degré fixé, les autres sommets sont de degré au plus $n - 2$. On a donc un nombre maximum d'arêtes si le corps forme une clique de taille $n - 2$. Le nombre maximum d'arêtes est $1 + 1 + \frac{(n-2)(n-3)}{2}$. Au minimum, seuls x, y et z ont des arêtes, ce qui donne $n - 1$ arêtes.*
2. *Montrons le par l'absurde. Supposons qu'il existe un graphe $G = (X, E)$ ayant de tels sommets a, b, et c, qui soit scorpion.*
   - *Si n est 3 ou 4, le seul graphe scorpion à n sommets est la chaîne. a est nécessairement une extrémité de la chaîne, b son voisin, c le voisin de b, donc $d(c) = n - 2$, contradiction.*
   - *Si $n > 4$, G admet un sommet z tel que $d(z) = n - 2$. Or,*
     - *$z \notin \{a, b, c\}$ d'après les degrés de ces 4 sommets.*
     - *$d(a) = 1$ et $ab \in E$ impliquent que $za \notin E$.*
     - *$d(b) = 2$ et $ab \in E, bc \in E$ impliquent que $zb \notin E$.*

     Donc $d(z) < n - 2$, contradiction.
3.

---

if $n < 3$ **then return** Faux
foreach $x \in X, d(x) = 1$ //test en temps $O(1)$ **do**
  $y \leftarrow \Gamma(x)$
  if $d(y) = 2$ //test en temps $O(1)$ **then**
    $z \leftarrow \Gamma(y) \setminus \{x\}$
    if $d(z) = n - 2$ //test en temps $O(n)$, mais ne se fait qu'une fois
    then return Vrai
    else return Faux
return Faux

□

2 Des voyages fous, fous, fous

Soit un espace imaginaire de dimension vachement supérieure à ce qu'on connaît (mais quand même fini). Dans ce monde, on peut prendre des portes magiques pour aller d'un endroit à l'autre. Bien entendu ces portes sont unidirectionelles ; pire que ça, il faut payer (des sousous, également notés $\widehat{c}$) pour les emprunter et toutes n'ont pas le même coût. On note E l'ensemble des endroits qui existent, $P \subseteq E \times E$ l'ensemble des portes menant d'un endroit à un autre, et $\widehat{c} : P \mapsto \mathbb{R}$ le prix à payer pour emprunter une porte (dans ce monde bizarre, on peut vous demander de payer $\pi\ \widehat{c}$).

Si $\mu = (e_1, e_2, \ldots, e_n)$ est un chemin, son coût $\widehat{c}(\mu)$ est $\Sigma_{i=1}^{n-1}\widehat{c}(e_i, e_{i+1})$. Le prix pour aller d'un endroit e à un autre est le prix minimal d'un chemin de e à cet endroit.

Nous sommes une agence de voyage et nous souhaitons calculer, pour un client difficile situé à l'endroit s, le nombre de $\widehat{c}$ minimum qu'il devrait débourser pour aller à n'importe quelle destination.

Exercice 2

On suppose dans cette partie que tous les prix sont positifs.

Question 1 On se donne un ensemble $S \subseteq E$ dont le prix depuis s est connu $(\widehat{c}_s)$, et tel que $\forall x \notin S, \forall y \in S, \widehat{c}_s(x) \geqslant \widehat{c}_s(y)$.

Montrer que l'on peut calculer le prix depuis s d'au moins un endroit situé à l'extérieur de S.

Solution : *Tout chemin sortant de S prend à un moment une arête xx', avec $x \in S, x' \notin S$. Il existe donc un chemin de s à x' de prix $\widehat{c}_s(x) + \widehat{c}(x, x')$.*

Soit x' tel que ce coût soit minimal parmi tous les voisins de S. Alors $\widehat{c}_s(x') = \widehat{c}_s(x) + \widehat{c}(x, x')$. En effet, tout autre chemin de s à x' passe par une arête yy' avec $y \in S$ et $y' \notin S$, et $\widehat{c}_s(y) + \widehat{c}(y, y') \geqslant \widehat{c}_s(x) + \widehat{c}(x, x')$. □

Question 2 En déduire un algorithme de calcul des prix depuis un endroit.

Solution : *C'est l'algorithme de Dijkstra.*

Algorithm 1: Dijkstra

```
S ← {s}
while G \ S ≠ ∅ do
    trouver x le plus proche de S parmi Γ⁺(S)
    calculer ĉ(x)
    S ← S ∪ {x}
```

□

Question 3 Quelle complexité peut-on obtenir pour cet algorithme ?

Solution : *La version naïve est $O(n^2)$.*

Il est parfois mieux d'utiliser un tas T pour stocker S, ce qui permet les opérations suivantes :
– insertion ou modification d'un élément en temps $O(\log |T|)$. Cette opération sera faite m fois.
– « pop » (recherche et suppression) de l'élément minimal en temps $O(\log |T|)$. Cette opération sera faite n fois. Remarquons que $O(\log |T|) = O(\log n)$. La complexité de l'algorithme est alors $O((m+n)\log n)$. □

Exercice 3 On suppose dans cette partie que certaines portes sont tellement pourries qu'on vous paye pour que vous les empruntiez, ce qui se traduit par un coût négatif. De plus, on suppose que si vous quittez un endroit, il est impossible d'y revenir en utilisant des portes (donc bien réfléchir avant de dépenser ses soussous $\widehat{c}$).

Soit s un endroit quelconque (mais est-il vraiment quelconque ? ! ?) et $V \subseteq E$ l'ensemble des destinations de voyage où l'on peut aller depuis s.

Question 1 Montrer que si on ne peut pas revenir à un endroit qu'on quitte (du moins pas avec les portes magiques), alors il existe au moins un endroit e où aucune porte ne mène. Montrer de plus que, quel que soit un tel endroit e, et bien même s'il n'existait pas on ne pourrait toujours pas revenir à endroit qu'on quitte.

Solution : *En d'autre termes, montrer que s'il n'y a pas de circuits alors il existe un sommet de degré entrant nul. Montrons la contraposée. Soit $x_0 \in E$ quelconque, on définit récursivement une suite infinie de sommets en choisissant, pour tout i, x_i un prédécesseur de x_{i-1}. Il n'y a qu'un nombre fini d'endroits donc il existe $i < j$ tels que $x_i = x_j$. La sous-suite d'indice compris entre i et j est un cycle.*

Deuxième question : un sous-graphe d'un graphe sans circuit est sans circuit. □

On en déduit qu'on ne peut revenir à un endroit qu'on quitte si et seulement si il existe une permutation σ des endroits telle que pour tout endroit $i \leqslant n$, il n'y a pas de nœud menant à i dans l'ensemble $\{e_{\sigma(1)}, \cdots, e_{\sigma(i)}\}$. Une telle permutation est un *tri topologique*.

Question 2 Histoire de mettre un peu d'ordre dans notre monde plein de dimensions, donner un algorithme qui calcule un tri topologique d'un graphe.

Solution :

Numérotation des sommets au cours d'un parcours en largeur modifié :

Algorithm 2: tri topologique

```
calculer les degrés entrants (notés d⁻)
i := 1
foreach x tel que d⁻(x) = 0 do a_traiter ∪={x}
while a_traiter ≠ ∅ do
    Soit x ∈ a_traiter
    σ(i) := x; i++
    foreach y ∈ Γ⁺(x) do
        d⁻(y)--
        if d⁻(y) = 0 then a_traiter ∪= {y}
```

Ainsi, on est sûr que lorsqu'un sommet est dans l'ensemble des éléments numérotés, tous ses précédeceurs y sont aussi. □

Question 3 En remarquant qu'un plus court chemin de s à e passe par une porte qui mène à e (!), en déduire un algorithme de calcul des tarifs de voyages depuis s si on ne peut revenir à un endroit qu'on quitte.

Solution : *On réalise un tri topologique. On met à jour les prix des endroits selon l'ordre topologique. Puisqu'aucun chemin menant à $\sigma(i)$ ne peut passer par des sommets $\sigma(j)$ si $j > i$, on est sûr d'avoir calculé les prix des trajets vers tous les prédécesseurs de $\sigma(i)$ avant de calculer le prix de $\sigma(i)$.* □

Question 4 Peut-on obtenir une complexité linéaire ?

Solution : *Oui, on ne parcourt les arêtes qu'une seule fois.* □

Exercice 4 On ne suppose plus qu'il n'y pas de cycle, ni que les prix sont positifs. On appelle donc *croisière vicieuse* un chemin qui revient à son point de départ et dont le prix est strictement négatif. Le danger est qu'un voyageur avare qui tombe dans une croisière vicieuse n'en sortira plus jamais, n'arrivera donc jamais à destination (ce qui nuit à notre réputation d'agence de voyage) et crée rapidement un déficit gigantesque chez les portiers qui,

mécontents, font grève et plus personne ne peut se déplacer avant qu'on envoie la milice et des big blaster atomic guns.

Question 1 Montrer que le prix $\widehat{c}$ entre deux endroits quelconques n'est bien défini que s'il n'y a pas de croisière vicieuse.

Solution : *A l'intérieur d'une croisière vicieuse, les prix des destinations sont égales à $-\infty$. Donc s'il en existe une, $\widehat{c}$ n'est pas toujours définie.* □

On considère que le prix du point de départ est nul : $\widehat{c}_s(s) = 0$. À chaque autre endroit e, on donne un prix éventuellement infini qui surestime le coût d'un voyage de s à e. On considère la procédure `passe` suivante :

Procedure `passe`

foreach *porte* $p = (x, y)$ **do**
  if $\widehat{c}_s(x) + \widehat{c}(p) < \widehat{c}_s(y)$ **then** $\widehat{c}_s(y) \leftarrow \widehat{c}_s(x) + \widehat{c}(p)$

Question 2 Supposons qu'il n'y a pas de croisière vicieuse et que les prix ne sont pas tous justes. Montrer qu'alors, après exécution de la procédure `passe`, au moins un nouvel endroit reçoit le prix juste.

Solution : *Soit S l'ensemble des endroits pour lesquels les prix sont justes, $x \notin S$, μ un chemin optimal de s à x, y le premier sommet de μ hors de S. Puisque μ est optimale, la chaîne μ coupée juste après y (notons la ν) est une chemin optimal de s à y.*

Lors de la procédure `passe`*, on rencontre la porte de μ menant de S à y, on met à jour le prix de y, qui devient juste car on lui affecte le prix de ν.* □

Question 3 En déduire un algorithme qui détecte la présence de croisière vicieuse, qui calcule les tarifs pour aller à n'importe quel endroit s'il n'y en a pas et qui envoie la milice atomique avec des bbags (mieux vaut prévenir que guérir) s'il y en a.

Solution : *Exécuter n fois* `passe()` *sur chaque arc. S'il n'y a pas de croisière vicieuse, alors tout devrait être bon. On fait une dernière* `passe` *sur les arcs, et si on change encore de valeurs, c'est qu'il y a une croisière vicieuse.* □