Algorithmique II
Année 2006-2007

Florent Bouchez
Jean-François Pineau

TD n°7

Agences de voyages magiques et autres bizarreries
Version avec solutions

1 Des voyages fous, fous, fous

Soit un espace imaginaire de dimension vachement supérieure à ce qu'on connait. 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 ¢) pour les emprunter et toutes n'ont pas le même coût. On note E l'ensemble des endroits qui existent, $P = E \times E$ l'ensemble des portes menant d'un endroit à un autre, et $¢ : P \mapsto \mathbb{R}$ le prix à payer pour emprunter une porte (dans ce monde bizarre, on peut vous demander de payer π ¢).

Si $\mu = (e_1, e_2, \ldots, e_n)$ est un chemin, son coût $¢(\mu)$ est $\Sigma_{i=1}^{n-1}(e_i, e_{i+1})$. 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 ¢ minimum qu'il devrait débourser pour aller à n'importe quelle destination.

Exercice 1

Question 1.1. On se donne un ensemble $S \subset E$ dont le prix depuis s est connu ($¢_s$), et tel que $\forall x \notin S, \forall y \in S, ¢_s(x) \geq ¢_s(y)$.

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

Solution: *Tout chemin qui va à un tel x minimal quitte à un moment S pour ne plus y revenir. À cet endroit le premier $x_i \notin S$ peut être calculé. Comme on ne connait pas les chemins minimaux on ne sait pas quel voisin de S prendre. Il suffit de prendre celui le moins loin.*

Question 1.2. En déduire un algorithme de calcul des prix depuis un endroit.

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

Algorithm 1: Dijkstra

$S \leftarrow \emptyset$
tant que $G \backslash S \neq \emptyset$ **faire**
  trouver x le plus proche de S parmi $\Gamma^+(S)$
  calculer $¢(x)$
  $S \leftarrow$ S $\cup \{x\}$
fin

Question 1.3. Quelle complexité peut-on obtenir pour cet algorithme ?

Solution: n^2

Exercice 2 On suppose 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, vous ne pourrez plus y revenir (donc bien réfléchir avant de dépenser ses soussous ¢).

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

Question 2.1. Montrer que si ne peut revenir à un endroit qu'on quitte, alors il existe au moins un endroit e où aucune porte ne mène et réciproquement. De plus, quel que soit un tel endroit e, et bien même s'il n'existait pas on ne pourrait toujours jamais revenir d'un endroit qu'on quitte.

Solution: $\Rightarrow$, *supposons que pas de circuit mais tous sommets on un degré entrant non nul. Soit μ une chaine de longueur maximale, et x le premier élément. Comme x a un prédécesseur, pas dans μ car pas de circuit, $x\mu$ est une chaine de longueur plus grande que μ. Contradiction. De plus, sous-graphe d'un graphe sans circuit est sans circuit.*

$\Leftarrow$, *si $G\backslash\{x\}$ sans circuit et degré entrant de x nul, alors $G\backslash\{x\} \cup \{x\}$ 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 $i \leq n$, si l'on supprime tous les endroits $e_{\sigma(1)}, e_{\sigma(2)}, \cdots, e_{\sigma(i)}$, on ne peut toujours pas revenir à un endroit qu'on quitte. Une telle permutation est un *tri topologique*.

Question 2.2. Histoire de mettre un peu d'ordre dans notre monde plein de dimensions, calculer un tri topologique de V.

Solution:

Question 2.3. 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.

Question 2.4. Peut-on obtenir une complexité linéaire ?

Exercice 3 On appelle *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 3.1. Montrer que le prix ¢ entre deux endroits quelconques est bien défini si et seulement si il n'y a pas de croisière vicieuse.

On considère que le prix du point de départ est nul : $¢_s(s) = 0$. 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`

pour chaque *porte* $p = (x, y)$ **faire**
  si $¢_s(x) + ¢(p) < ¢_s(y)$ **alors**
    $¢_s(y) \leftarrow ¢_s(x) + ¢(p)$
  fin
  si $¢_s(y) + ¢(p) < ¢_s(x)$ **alors**
    $¢_s(x) \leftarrow ¢_s(y) + ¢(p)$
  fin
fin

Question 3.2. 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.

Question 3.3. 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 des valeurs, c'est qu'il y a une croisière vicieuse.*

2 Maintenance de station spatiale

La station spatiale Ajax© est composée de pièces reliées par des couloirs. Pour minimiser les risques en cas de problème dans un couloir, ceux-ci relient exactement deux pièces, et en plus sont fermés hermétiquement par une porte à chaque extrémité. On place un robot dans une des pièces ; le but est de lui écrire un algorithme qui lui permette d'explorer les pièces et de s'arrêter avec la garantie qu'il a alors visité toutes les pièces (sauf si la station n'est plus connexe, mais on considérera qu'alors il y a d'autres problèmes plus urgents que de s'amuser à promener un robot).

Note : le robot a deux bras avec des pistolets à peinture au bout, un rouge et un bleu. Il a le droit de faire des marques sur les portes qu'il traverse, mais étant affublé d'un système de reconnaissance des couleurs un peu archaïque, il risque de griller sur place s'il ne sait pas décider si la porte est plutôt avec plus de bleu ou plutôt avec plus de rouge (ou rien du tout). Comme il vise très mal, on n'est pas certain que le robot réussira à bien recouvrir totalement une vieille peinture avec une nouvelle couleur et donc il vaut mieux ne l'autoriser à marquer qu'un seule fois chaque porte.

Solution:

1. *en entrant dans un couloir, le traverser jusqu'à l'autre bout,*
2. *en entrant dans un pièce où toutes les portes sont sans marques, marquer 1 sur la porte qu'il vient de franchir,*
3. *dans une pièce où il y a une porte sans marque, en choisir une, la marquer avec 0 et la franchir,*
4. *dans une pièce où toutes les portes sont marquées, sortir par une porte sans la marque 0 s'il y en a une, et sinon s'arrêter (on a la garantie qu'il a alors tout visité, à prouver pour les plus tatillons :-)*