Université de Caen **Année 2008-2009**
UFR de Sciences **1ère session**

EI61T Approfondissement en algorithmique
durée 2h00

Les notes de cours et TD sont autorisées.
Les deux parties sont indépendantes, correspondent aux parties du cours "algorithmique des graphes" et "algorithmique répartie" et doivent être rendues sur des copies séparées.

Chaque candidat doit, au début de l'épreuve, porter son nom dans le coin de la copie qu'il cachera par collage après avoir été pointé. Il devra en outre porter son numéro de place sur chacune des copies, intercalaires, ou pièces annexées.

1 Algorithmique répartie (à rédiger sur une copie séparée)

1.1 Introduction

Un algorithme réparti A fonctionne sur un système constitué d'un réseau de processus ou sites communiquant par messages. Les lignes de communication entre sites sont supposées bi-directionnelles et fiables (tout message émis à une extrémité est reçu par l'autre extrémité en un temps fini). Les sites sont supposés avoir des identités deux à deux distinctes, sans toutefois que celles-ci soient connues des autres sites du réseau.

Relativement à cet algorithme A, les différents sites peuvent être actifs ou passifs. Ceci est mémorisé, sur chaque site i, dans la variable $Etat_i$.

- Un site i ne peut envoyer de message relatif à l'algorithme A que si $Etat_i$ = actif.
- Un site qui a terminé son travail relatif à l'algorithme A met sa variable $Etat_i$ à passif.
- Un site qui reçoit un message relatif à l'algorithme A (re)met sa variable $Etat_i$ à actif (si elle ne l'était pas déjà).

N.B. : les questions ci-dessous ne sont pas indépendantes, mais il est possible d'admettre le résultat d'une section pour traiter la section suivante.

1.2 Site collecteur d'information

Un des sites (nommons-le collecteur) à pour but de récupérer des informations en provenance des autres sites (pour simplifier, on supposera qu'il ne participe jamais à l'algorithme A).

Pour cela, il devra tout d'abord établir une structure sur le réseau afin de permettre aux autres sites de pouvoir lui envoyer facilement des informations.

Question 1.1 *Quel algorithme peut lancer le site collecteur afin d'atteindre ce but ? Écrivez le texte de cet algorithme et précisez comment, une fois cet algorithme terminé, les différents sites du réseau pourront envoyer une information au site collecteur lorsqu'ils devront le faire.*

Dans la suite, on supposera donc que cet algorithme a été exécuté.

1.3 Synchronisation

Dans la section 1.4, on va imposer aux différents sites d'envoyer des messages au site collecteur, lorsque certains événements de l'algorithme A ont lieu. Afin que le collecteur puisse éventuellement trier ces messages, on désire que les événements de l'algorithme A soient "datés", non pas avec des dates réelles qui risqueraient de ne pas être compatibles entre tous les sites, mais avec un système de datation compatible avec la relation de précédence des événements de l'algorithme A.

Question 1.2 *Indiquez comment on peut concrétiser ce système de datation des événements de l'algorithme A, de manière à assurer que, si deux événements e et f de cet algorithme sont tels que e précède f, alors la date de e soit strictement inférieure à celle de f.*
Précisez les variables à mettre en place sur les sites et les instructions nécessaires à incorporer aux traitements relatifs à l'algorithme A permettant la gestion de ces dates.

Dans la suite, on supposera que l'algorithme A incorpore cette gestion des dates.

1.4 Envoi des changements d'état

On impose à présent à chaque site envoie au site collecteur (en utilisant la structure mise en place à la section 1.2), un message à chaque changement d'état actif/passif relativement à l'algorithme A. Ces messages devront comporter :

- la date de l'événement ayant conduit à ce changement d'état
- l'identité du site qui a subi ce changement et qui envoie le message au collecteur
- la nature du changement (passage à actif ou à passif)

Le site collecteur va donc recevoir ces messages, et les mémoriser.

Question 1.3 *Proposez une structure de données permettant au site collecteur de mémoriser, pour chaque site, les changements d'état avec les dates auxquels ils ont eu lieu, et de retrouver facilement le dernier état connu pour chaque site.*
Précisez comment le collecteur mémorise les informations dans cette structure à la réception d'un message venant d'un autre site.
N.B. : pour cette question, on peut supposer que le site collecteur connaît les identités des sites du réseau, car on pourrait par exemple faire en sorte qu'il les apprenne lors de l'algorithme décrit à la section 1.2.

1.5 Détection de terminaison

Question 1.4 *Pensez-vous que les informations ainsi mémorisées par le site collecteur puissent lui permettre de détecter la terminaison de l'algorithme A ? Si oui, indiquez comment, si non une information complémentaire sur les caractéristiques des transmissions de message le permettrait-il ? Précisez éventuellement d'autres conditions nécessaires pour que cette détection puisse avoir lieu (en utilisant toutefois les informations collectées, donc sans écrire un autre algorithme réparti de détection de terminaison tels que ceux vus en cours ou TD).*

2 Algorithmique des graphes : Algorithme de Johnson (à rédiger sur une copie séparée)

Les questions (à part la question 2.6) de cette partie sont indépendantes.

L'idée principale consiste à changer les poids des arcs d'un graphe orienté $G = (V, E)$ afin de pouvoir utiliser l'algorithme de Dijkstra. Au départ, on a une fonction de poids w sur G, et on va en considérer une autre $\hat{w}$ qui vérifie les deux propriétés suivantes :

(1) Pour toute paire (u, v) de sommets de V, un plus court chemin d'origine u et d'extrémité v utilisant la fonction de poids w est aussi un plus court chemin d'origine u et d'extrémité v utilisant la fonction $\hat{w}$.

(2) Pour tout arc (u, v) de E, le poids $\hat{w}(u, v)$ est positif ou nul.

Question 2.1 Quel problème peut se poser lors de l'exécution de l'algorithme de Dijkstra, si le graphe admet des arcs de poids (strictement) négatif ?

Question 2.2 Soit une fonction h définie sur V et à valeurs réelles. Sur un graphe $G = (V, E)$, on définit la fonction de poids $\hat{w}$ à partir des fonctions w et h par la relation

$$\hat{w}(u, v) = w(u, v) + h(u) - h(v).$$

Soit un chemin $p = (v_0, v_1, \ldots, v_k)$ joignant le sommet v_0 au sommet v_k.

a) Trouver une relation entre les deux poids $w(p)$ et $\hat{w}(p)$ du chemin p.

b) Montrer qu'il y a équivalence entre les deux propriétés (i) et (ii) :

(i) p est un plus court chemin de v_0 à v_k pour le poids w;

(ii) p est un plus court chemin de v_0 à v_k pour le poids $\hat{w}$,

et également entre les deux propriétés (iii) et (iv) :

(iii) p est un cycle absorbant (i.e., de poids négatif) pour le poids w;

(iv) p est un cycle absorbant (i.e., de poids négatif) pour le poids $\hat{w}$.

Question 2.3 On ajoute à G un nouveau sommet s et de nouveaux arcs partant de s et arrivant à tout sommet v de V. On obtient ainsi un nouveau graphe $G' = (V', E')$. On prolonge la fonction w initiale en attribuant à chacun des nouveaux arcs un poids nul.
On définit pour tout sommet u de V, la fonction $h(u) = \delta(s, u)$ avec $\delta(s, u)$ le minimum des poids des chemins de s à u.

a) Pourquoi la fonction h n'est pas définie si le graphe contient un ou des cycles absorbants (cycles de poids négatif) ?

b) Si le graphe ne conient pas de graphe absorbant, montrer qu'avec cette définition de h la fonction $\hat{w}$ satisfait également la condition (2).

Question 2.4 (Bellman-Ford) L'algorithme suivant permet de calculer la fonction h si le graphe ne contient pas de cycle absorbant :

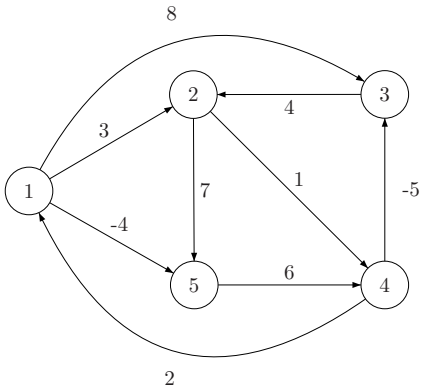
Algorithme Bellman-Ford(G, w, s)
```
Init(G, s)
for i ← 1 to |V| − 1 do
    for each edge (u, v) ∈ E do
        Relax(u, v, w)
    end for
end for
```

où les fonctions Init et Relax sont définies ci-contre :

Fonction Init(G, s)
```
for tout sommet v de V do
    h[v] ← ∞
end for
h[s] ← 0
```

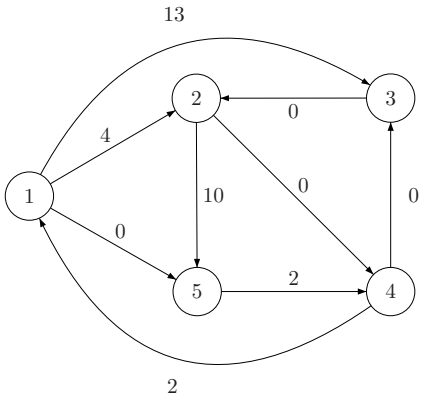
Fonction Relax(x, y, w)
```
if h[x] + w(x, y) < h[y] then
    h[y] ← h[x] + w(x, y)
end if
```

On considère le graphe $G = (V, E)$ (sans cycle absorbant) avec sa fonction de poids w :



a) Appliquer l'algorithme Bellman-Ford au graphe $G' = (V', E')$ construit à partir de G comme dans la question 2.3, et calculer la fonction h en chaque sommet.
b) Redessiner le graphe avec la nouvelle fonction de poids $\hat{w}$ définie à la question 2.2.

Question 2.5 Soit le graphe $\widetilde{G}$ suivant :



Appliquer l'algorithme de Dijkstra sur le graphe $\tilde{G}$ en prenant comme source le sommet 2, et donner la trace d'exécution en donnant un tableau du type vu en cours :

v	1	2	3	4	5
d	∞	0	∞	∞	∞
π	–	–	–	–	–

Remarque. Avec les notations du cours, $\pi(v)$ désigne le prédécesseur de v dans un plus court chemin depuis la source jusqu'à v (avec tous les sommets intermédiaires traités), et $d(v)$ le poids de ce chemin. Le tableau donné correspond à l'étape d'initialisation.

Question 2.6 En justifiant par rapport aux questions précédentes, donner pour chaque sommet v de V du graphe G (et pour la fonction de poids w correspondante) de la question 2.4 les plus courts chemins en prenant comme source le sommet 2.