Université de Caen **Année 2010-2011**
UFR de Sciences **1ère session**

EI61T Approfondissement en algorithmique
durée 2h00

Les notes de cours et TD sont autorisées. L'utilisation d'appareils électroniques est limitée à la fonction qui fournit l'heure.

Les deux parties correspondent aux parties du cours "algorithmique répartie" et "algorithmique générale" et doivent être rendues sur des copies séparées.
Elles sont liées par l'idée mais peuvent être traitées indépendamment l'une de l'autre.

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

On se place dans un réseau de sites communiquant par messages, en utilisant des lignes bidirectionnelles fiables. Les messages sont transmis en un temps fini, les sites ont des identités deux à deux distinctes et, au départ, chaque site ne connaît que sa propre identité et celles de ses voisins immédiats.

Un des sites, appelé initiateur, a un rôle particulier dans l'algorithme étudié ci-dessous.

1.2 Principes de l'algorithme étudié

L'objectif est, pour l'initiateur, de récupérer suffisamment d'informations pour pouvoir, au moyen d'un algorithme centralisé (cf. partie 2), calculer un arbre de recouvrement de poids (ou de coût) minimal dans le réseau. On suppose que le poids d'une arête e est donné par la fonction $w(e)$.

L'algorithme lancé par l'initiateur va donc chercher à récupérer, pour tous les sites du réseau l'information que chacun possède. On utilisera pour cela un algorithme réparti de type parcours parallèle.

Question 1.1 *Rappelez quelles propriétés de ce type d'algorithme, démontrées en cours ou en TD, vont nous assurer que l'initiateur peut effectuer sa demande d'informations auprès de chacun des sites du réseau (on ne demande pas de re-démontrer ces propriétés, simplement de les citer).*

Les sites sollicités vont donc devoir envoyer de l'information vers l'initiateur. Ce sera évidemment leur nom et les noms de leurs voisins immédiats et les poids (ou encore coûts) des arêtes entre eux et leurs voisins immédiats.

Question 1.2 *Indiquez dans quel type de messages ces informations seront véhiculées, et précisez une structure de données permettant d'inclure ces informations dans ces messages.*

N.B. : on ne demande pas dans cette partie de prévoir la structure de données destinée à mémoriser ces informations dans le site initiateur. On supposera simplement que cette structure pourra être utilisée par l'algoritme étudié dans la partie 2.

1.3 Algorithme complet

Question 1.3 *Écrivez le texte de l'algorithme complet permettant à l'initiateur de récupérer ces informations. Pensez notamment à :*

- *bien préciser les types de messages utilisés et leurs paramètres éventuels, ainsi que toutes les variables locales à chaque site et nécessaires à l'algorithme ;*
- *distinguer le texte concernant l'initiateur de celui des autres sites ;*
- *bien récupérer la totalité des informations du réseau, et signaler à l'initiateur la fin de l'opération.*

1.4 Exploitation des résultats

On suppose que l'initiateur a donc pu récupérer ces informations et calculer un arbre couvrant de poids minimal au moyen d'un algorithme centralisé, étudié dans la parte 2. Il peut donc utiliser ces informations mais il est le seul à les posséder.

Question 1.4 *Imaginez une méthode pour diffuser aux sites du réseau le résultat du calcul effectué par l'initiateur, de manière à ce que l'arbre de poids minimal calculé par l'algorithme de Kruskal étudié en partie 2 puisse être diffusé dans le réseau et mémorisé de manière répartie (dans des variables Pere et Fils) par les différents sites.*

N.B. : dans cette dernière question, on ne demande pas d'algorithme complet mais simplement l'idée d'une méthode.

2 Algorithmique générale (à rédiger sur une copie séparée)

Rappel 1 : Soit $G = (V, E, w)$ un graphe connexe, pondéré, non orienté, de sommets V et d'arêtes $E \subset V^2$ de poids $w : E \to R$. Un arbre de recouvrement (ou un arbre couvrant) minimal est un sous–ensemble E' de E tel que :

- Chaque sommet de V est un extrémité d'une arête de E'.
- Le graphe $G' = (V, E')$ est connexe et sans cycle (c'est un arbre).
- le poids (ou le coût) total de l'arbre $\sum_{e \in E'} w(e)$ est minimal.

Rappel 2 : Soit T un arbre. Si l'on ajoute une arête à T, on y fait apparaître un cycle.

L'algorithme de Kruskal construit un arbre de recouvrement de poids minimal, en maintenant une partition P des sommets. On rappelle qu'une partition de V est la donnée de k sous–ensembles non–vides $V_1, \ldots, V_k$ de V tels que

$$i \neq j \Longrightarrow V_i \cap V_j = \emptyset \quad \text{et} \quad V_1 \cup V_2 \cdots \cup V_k = V.$$

Pour $v \in V$, on appelle $P(v)$ l'unique élément de la partition contenant v.

On commence par trier les arêtes par ordre croissant des poids. On considère alors un graphe G' ayant les mêmes sommets que le graphe de départ et initialement aucune arête ($E' = \emptyset$ au début). Ensuite pour chaque arête de E l'algorithme la rajoute à E' si et seulement si elle ne crée pas de cycle dans G'. Voici le pseudo-code de l'algorithme de Kruskal :

$E' = \emptyset$
$P = \{\{v\} | v \in V\}$.
(Les éléments de la partition initiale sont les singletons : $P(v) = \{v\}, \forall v \in V$.)
Trier E par ordre croissant des poids w.
Pour chaque arête $(u, v) \in E$
  Si $P(u) \neq P(v)$ alors
    $E' = E' \cup \{(u, v)\}$;
    Fusionner les deux éléments de la partition $P(u)$ et $P(v)$;
  FinSi
FinPour

---

1. On suppose ici que $v = \{1, 2, 3, 4\}$. L'ensemble des arêtes E est

$$E = \{(1, 2), (1, 3), (1, 4), (2, 3), (2, 4)\}.$$

   Et les coûts des arêtes sont :

$$w((1, 2)) = 100 \ ; \ w((1, 3)) = 50 \ ; \ w((1, 4)) = 40 \ ; \ w((2, 3)) = 20 \ ; \ w((2, 4)) = 30$$

   – Dessiner le graphe précédent et donner sa représentation par liste d'adjacence.
   – Donner la trace d'exécution et le résultat de l'exécution de l'algorithme de Kruskal sur ce graphe.
   – Donner la trace d'exécution et le résultat de l'exécution de l'algorithme de Prim, vu en TD, sur ce graphe en partant du sommet 1.

2. Preuve de correction de l'algorithme.
   – Montrer qu'une arête ajoutée crée un cycle si et seulement si ses deux extrémités sont dans le même élément de la partition. Indication : Il faut montrer, par récurrence sur le nombre d'étapes, qu'à chaque instant durant l'exécution, les éléments de la partition des sommets sont des arbres.
   – Soit $G' = (V, E')$ le graphe courant à une étape de l'algorithme. On suppose qu'il existe un arbre couvrant de poids minimal contenant G'. Soit e l'arête que l'algorithme s'apprête à considérer. (i.e. celle de poids minimal parmi les arêtes non encore considérées). Si e ne crée pas de cycle, montrer qu'il existe un arbre couvrant de poids minimal contenant $G' = (V, E' \cup \{e\})$.
   – En déduire la correction de l'algorithme : Si le graphe départ est connexe, l'algorithme de Kruskal termine toujours en retournant bien un arbre couvrant de poids minimal.
   – Que se passe–t–il si le graphe de départ n'est pas connexe ?
   – Que peut–on dire si chaque arête du graphe de départ arête est de poids 1 ?

3. On suppose ici que le coût de la comparaison entre $P(v)$ et $P(v')$ est en $O(\log(|E|)$. On suppose également que le coût de la fusion de deux éléments de la partition est $O(\log |E|)$. Quelle est alors la complexité (le coût en nombre d'opérations élémentaires) de l'algorithme de Kruskal ?