Le projet est à rendre par courriel (gavoille@labri.fr) au plus tard ***lundi 16 avril*** *2007 minuit. Vous devez m'envoyer un fichier .java pour l'exercice 2, source que je testerai sous ViSiDiA en mode asynchrone, ainsi qu'un rapport dans un format lisible non compressé (.ps, .pdf, ou .txt – pas de .doc, .sdw ou de .zip). Si vous pensez que cet utile, vous pouvez également m'envoyer un fichier de graphe d'exemple sur lequel votre algorithme fonctionne.*

Le rapport devra contenir au plus ***3 pages*** *(sans la page de garde). Il décrira le plus clairement possible la méthode ou algorithme utilisé, les remarques et vos commentaires. Votre programme devra être suffisament commenté.*

Pour les calculs de complexité (donnés avec la notation $O()$), vous veillerez à ce que les algorithmes et les complexités donnés dans votre rapport correspondent à ceux de vos programmes. Il n'y a pas une, mais plusieurs réponses possibles. La notation tiendra compte de la correspondance entre le rapport et votre programme, ainsi que de la qualité de l'argumentation.

Exercice 1

Dans le cours nous avons vu deux algorithmes asynchrones pour la construction d'arbre en largeur d'abord (BFS). L'un est basé sur Dijkstra, l'autre sur Bellman-Ford dont la complexité en temps est proportionelle au diamètre du graphe.

a) Écrire en C-distribué l'algorithme permettant de construire un arbre en largeur d'abord de racine r_0 selon Bellman-Ford avec la détection locale de la terminaison globale. Plus précisément, chaque sommet u doit gérer les trois variables suivantes :

`L`: doit, à la fin de l'algorithme, contenir la distance de u à r_0, c'est-à-dire le nombre minimum d'arêtes d'un chemin connectant u à r_0 dans le graphe.

`père`: doit contenir l'arête incidente à u menant au père dans l'arbre BFS, et `-1` pour la racine.

`terminé`: booléen initialisé à `false` qui doit passer à `true` seulement après que la variable `L` soit correcte pour tous les sommets du graphe, c'est-à-dire contienne la distance du sommet à la racine.

b) Quels sont les complexités en temps et en nombre de messages de votre algorithme.

Exercice 2

Soit G un graphe connexe à n sommets et S un sous-ensemble de sommets de G. On dit que S est un ensemble *dominant* si tout sommet de G est soit dans S, soit possède un voisin dans S.

On souhaite construire un algorithme asynchrone qui permette de construire rapidement pour G un ensemble dominant S de « petite » taille, typiquement avec au plus $n/2$ sommets.

L'ensemble S est représenté de manière distribuée par une variable locale `S`. On notera $\mathtt{S}_u$ le booléen qui doit valoir `true` si et seulement si, à la fin de l'algorithme, $u \in S$. On souhaite donc calculer $\mathtt{S}_u$ pour tout sommet u de G.

L'idée de l'algorithme est la suivante :

A. on construit une forêt couvrante G comprenant beaucoup de composantes, c'est-à-dire d'arbres que l'on supposera enracinés.

B. puis en parallèle sur chaque arbre T de la forêt, calculer l'ensemble P des sommets de T à distance paire (dans T) de r, et l'ensemble I des sommets de T à distance impaire (dans T) de r, où r représente la racine de T. On note M le plus petit des deux ensembles P et I. On pose finalement $\mathtt{S}_u = \mathtt{true}$ si et seulement si $u \in M$.

Pour calculer la forêt de la partie **A**, chaque sommet choisit comme père son voisin ayant la plus petite identité, avec la convention que si deux voisins se choissent mutuellement, alors celui de plus petite identité devient la racine.

Pour la partie **B**, et à chaque sommet u de l'arbre T, on associe un compteur $C(u) = (x, y)$ composé de deux valeurs : x étant le nombre de descendants de u à distance paire de u (u compris) et y le nombre de descendants de u à distance impaire de u. Donc pour toutes les feuilles u de T on a $C(u) = (1, 0)$. On remarque deux choses : 1) le compteur de la racine $C(r)$ peut être calculé à partir des compteurs des feuilles en remontant vers la racine r ; 2) Si $C(r)$ est connu de u, alors $\mathtt{S}_u$ peut calculer correctement.

a) Fournir un programme ViSiDiA implémentant l'algorithme précédent calculant un petit ensemble dominant.

b) Estimer les complexités en temps et en nombre de messages de votre algorithme. Préciser la taille de vos messages. Validez ces complexités par des expériences. Discutez éventuellement des résultats obtenus.

c) Démontrer que l'ensemble S ainsi construit est de taille $|S| \leqslant n/2$.