Mineure "Informatique Parallèle et Distribuée"

Contrôle (2 heures)

Eric Goubault

19 mars 2002

- l'exercice et le problème sont indépendants,
- les questions 1 à 6 du problème sont indépendantes, mais demandent une lecture séquentielle du sujet.

1 Exercice (5 points)

On considère une « implémentation » d'un lien *duplex* entre deux machines X et Y, qui ne sont reliées que par un seul tampon de communication C, ne fonctionnant qu'en mode *simplex*.

Par duplex on entend que les deux machines doivent pouvoir s'envoyer des messages de X vers Y aussi bien que de Y vers X. Par mode simplex pour C on entend ici que C est capable d'acheminer n'importe quel nombre de messages dans une seule direction à la fois, soit de X vers Y, soit de Y vers X selon les besoins. Des messages allant dans des sens opposés ne peuvent transiter en même temps dans le tampon de communication: on supposera que le tampon bloque toutes communications futures. Donc tout processeur voulant communiquer par C doit s'assurer que son destinataire n'est pas en train d'envoyer un message en sens inverse.

On appelera `send(X,Y)` l'action d'envoyer un message de X vers Y sur C (non-bloquant) et `recv(X)` (respectivement `recv(Y)`) l'action (bloquante) de réception d'un message venant de X (respectivement Y).

Modéliser, en utilisant la syntaxe des sémaphores `P` et `V` et tout nombre de variables qui pourraient vous être utile, les processeurs (vus comme des processus JAVA) X et Y, qui bouclent à l'infini en essayant d'envoyer un message à Y et X respectivement, tout en évitant l'interblocage. On prouvera que X et Y ne créent pas d'interblocage.

2 Problème: composantes connexes d'un graphe (15 points)

Soit $G = (V, E)$ un graphe *non*-orienté: V est un ensemble de noeuds et $E \subseteq V \times V$ est l'ensemble d'arêtes du graphe G (en tant que relation, on supposera E symmétrique). On rappelle qu'un noeud x est *voisin immédiat* de y dans G si et seulement si xEy. Deux noeuds x et y sont *reliés* si et seulement si il existe une suite finie $(x_0, \cdots, x_n)$ de noeuds de G, avec $x_0 = x$, $x_n = y$ et (x_i, x_{i+1}) immédiatement voisins (pour $i = 0, \cdots, n-1$). La relation "être relié à" est une relation d'équivalence sur les noeuds de G. On appelle *ensemble de composantes connexes* de G l'ensemble des classes d'équivalence de noeuds de G pour cette relation d'équivalence. On dira

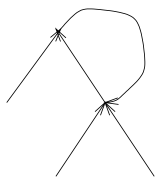

FIG. 1 – *Une pseudo-forêt.*

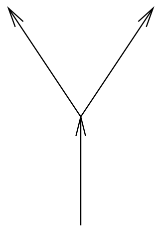

FIG. 2 – *Un graphe qui n'est pas une pseudo-forêt.*

qu'un noeud v de G appartient à la composante n de G si sa classe d'équivalence est la $n$ième parmi les composantes connexes de G.

Le principe de l'algorithme séquentiel est de faire un parcours en profondeur d'abord, et de marquer chaque noeud rencontré en cours d'un parcours provenant initialement d'un sommet donné, par le numéro de ce sommet.

Question 1

Soit u le nombre de noeuds d'un graphe G et v son nombre d'arêtes. Quelle est la complexité asymptotique, au pire, de l'algorithme séquentiel décrit précédemment (on ne cherchera pas à prouver ce résultat)?

On appelle pseudo-forêt tout graphe *orienté* (c'est-à-dire que les arêtes ont maintenant une direction: la relation de voisinage n'est plus forcément symétrique) tel que tout noeud ait au plus un voisin immédiat. Dit de façon plus formelle, pour tout noeud n de la pseudo-forêt $G = (V, E)$, le cardinal de $\{n' \mid (n, n') \in E\}$ est inférieur ou égal à un: le nombre d'arcs sortant de tout noeud est au plus un. A la figure 1 est donné un exemple de pseudo-forêt, à la figure 2 on trouvera un graphe qui n'est pas une pseudo-forêt.

On appelle arbre tout graphe *orienté* avec une seule racine (c'est-à-dire un seul point qui n'a pas de voisin immédiat), et un unique chemin entre tout noeud du graphe et cette racine. Remarquez bien que l'on définit les arbres avec une convention d'orientation des arcs exactement à l'envers de ce que l'on a fait par exemple en tronc-commun. On dit qu'un arbre est étoilé si chacun de ses noeuds est *directement* relié à la racine.

Un pseudo-arbre est un arbre auquel on a rajouté un arc allant de la racine à un autre noeud (par exemple à la figure 1). Un pseudo-arbre étoilé est un arbre étoilé auquel on a rajouté une boucle de la racine vers elle-même (on appelera encore ce noeud, la racine du pseudo-arbre étoilé).

On appelle graphe fonctionnel G tout graphe fabriqué à partir d'une fonction $f : V \to V$, où V est un ensemble fini, de la façon suivante: $G = (V, E)$ (le même V) où $E = \{(v, f(v)) \mid v \in V\}$.

Question 2

Est-ce que tous les graphes sont des graphes fonctionnels[1]? Montrer que les graphes fonctionnels sont des pseudo-forêts, et que les pseudo-forêts sont des unions disjointes de pseudo-arbres.

On va maintenant pouvoir définir un algorithme parallèle de recherche de composantes connexes sur une machine *PRAM*, dont on ne spécifie pas pour le moment le type (EREW, CREW ou CRCW). On suppose disposer au début de l'algorithme, d'une représentation d'un graphe $G = (V, E)$ par une matrice d'adjacence A, c'est-à-dire si on suppose $V = \{1, \cdots, n\}$, A est une matrice $n \times n$ telle que $A_{i,j} = 1$ si

1. On pourra soit donner une preuve si oui, soit donner un contrexemple, si non.

$(i,j) \in E$ (et 0 sinon).

On définit maintenant la fonction $C : V \rightarrow V$ avec

$$C(v) = \begin{cases} v & \text{si } v \text{ est un noeud isolé} \\ min\{u \mid A_{u,v} = 1\} & \text{sinon} \end{cases}$$

Question 3

Indiquer, selon le type de la PRAM utilisée, à quelle complexité on peut arriver pour le procédé qui, étant donné `A`, calcule le tableau qui à l'entrée v associe $C(v)$.

Question 4

Soit G_C le graphe fonctionnel de C. On sait par la question 2 que G_C est une union de pseudo-arbres $G_i = (V_i, E_i)$ $(i = 1, \cdots, s)$. Montrer que tous les noeuds de chaque V_i sont dans la même composante connexe du graphe G de départ.

On peut prouver de plus (mais on ne demande pas de le faire) que tout cycle de G_C est soit une boucle triviale (de la forme (v,v) pour un noeud v) soit contient exactement deux arcs. On sait aussi par la question 2 que chaque cycle des pseudo-arbres G_i contient la racine de G_i.

Question 5

On cherche à transformer les pseudo-arbres G_i de la pseudo-forêt G_C de la question 4 en pseudo-arbres étoilés T_i ou en points isolés. Indiquer en pseudo-code (on pourra se contenter de décrire ce que fait l'algorithme en français) un algorithme PRAM en $O(\lceil log(n) \rceil)$ (où n est le nombre de noeuds du pseudo-arbre considéré).

Remarquez que les T_i forment une pseudo-forêt qui est le graphe fonctionnel d'une fonction $D : V \rightarrow V$.

Soient r_i les racines des pseudo-arbres étoilés T_i $(i = 1, \cdots, s)$.

Grâce à la question 4, on sait que les r_i définissent un sur-ensemble des composantes connexes du graphe G de départ. Mais en fait, certains des r_i sont reliés par un chemin dans le graphe G à d'autres r_j. On acceptera sans démonstration le fait que r_i est relié par un chemin à r_j dans G si et seulement si il existe $v \in V_i$, $w \in V_j$ tels que $(v,w) \in E$.

Question 6

En utilisant la fonction D définissant les pseudo-arbres étoilés T_i, indiquer en pseudo-code sur une machine PRAM CRCW (en mode consistant) comment on peut trouver efficacement à calculer le graphe des "super-noeuds" r_i par une matrice d'adjacence A', dans la mémoire partagée, tel que r_i est relié à r_j si il existe un chemin dans G qui relie r_i à r_j.

Question 7

Indiquer *sans donner de pseudo-code* quelle serait *a priori* la complexité du calcul de la question 6 si on avait une machine PRAM CREW, et si on avait une machine PRAM EREW.

On définit un algorithme qui, partant de $k = 0$, $A_0 = A$ de taille $n_0 \times n_0$, fait les étapes suivantes, tant que $n_k > 0$ (n_k étant la taille de A_k):

- calculer G_C et les T_i par la méthode de la question 4 et 5,
- calculer $A_k = (A_{k-1})'$ de taille $n_k \times n_k$ par la méthode de la question 6.

Question 8

Prouver que $n_k \leq n_{k-1}/2$. On pourrait prouver, *mais on ne le demande pas ici*, que l'algorithme sur une machine PRAM CRCW en mode consistant prendrait un temps $O(log^2(n))$ pour un total de $O(n^2)$ opérations, où n est le nombre de noeuds dans le graphe G de départ. Quelle est l'efficacité de l'algorithme?