P-RAM et anneaux de processeurs

1 Composantes connexes sur une P-RAM

On souhaite concevoir un algorithme CREW qui permette de calculer les composantes connexes d'un graphe $G = (V, E)$ dont les sommets sont numérotés de 1 à n. Plus précisément, on cherche un algorithme qui renvoie un tableau C de taille n tel que $C(i) = C(j) = k$ si et seulement si i et j sont dans la même composante connexe et k est le plus petit indice des sommets de cette composante.

Définition 1. À toute étape de l'algorithme, on appellera pseudo-sommet étiqueté par i l'ensemble de sommets $j, k, l, \dots \in V$ tels que $C(j) = C(k) = C(l) = \dots = i$. On assimilera le pseudo-sommet i étiqueté par i au sommet étiqueté par i.

Un des invariants de l'algorithme est que le plus petit indice des sommets constituant un pseudo-sommet étiqueté par i est i et que les sommets appartenant à un pseudo-sommet sont dans la même composante connexe. Cette assertion est donc vraie si on initialise C par : pour tout $i \in V = [\![1, n]\!]$: $C(i) = i$. Ceci signifie que chaque processeur se considère au départ comme sommet de référence de sa composante connexe. L'objectif de l'algorithme est de modifier ce point de vue égocentrique.

Définition 2. Une arborescence k-cyclique ($k \geqslant 0$) est un graphe orienté faiblement connexe (c'est-à-dire tel que le graphe non orienté sous-jacent est connexe) tel que :
- tout sommet a un degré sortant égal à 1 et
- il existe exactement un circuit de longueur $k + 1$.

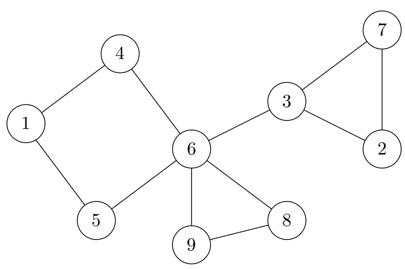
On appelle étoile une arborescence 0-cyclique dans laquelle toutes les arêtes sont incidentes à la racine et l'indice de la racine est le plus petit indice dans l'étoile.

L'invariant précédent est donc que le graphe orienté $(V, \{(i, C(i)) \mid i \in V\})$ est constitué d'étoiles. On peut donc identifier pseudo-sommets et étoiles, le centre de l'étoile étant l'indice du pseudo-sommet. Le calcul des composantes connexes s'effectue en enchaînant plusieurs fois de suite les deux fonctions suivantes :

Gather()
Pour tout $i \in S$ **en parallèle:**
    $T(i) \leftarrow \min\{C(j) \mid \{i, j\} \in E, C(j) \neq C(i)\}$
    {si l'ensemble est vide, on associe $C(i)$}
Pour tout $i \in S$ **en parallèle:**
    $T(i) \leftarrow \min\{T(j) \mid C(j) = i, T(j) \neq i\}$
    {si l'ensemble est vide, on associe $C(i)$}

Jump()
Pour tout $i \in S$ **en parallèle:**
    $B(i) \leftarrow T(i)$
Répéter $\log n$ **fois**
    Pour tout $i \in S$ **en parallèle:**
        $T(i) \leftarrow T(T(i))$
Pour tout $i \in S$ **en parallèle:**
    $C(i) \leftarrow \min\{B(T(i)), T(i)\}$

▷ **Question 1** *On considère le graphe suivant.*



Appliquer la fonction Gather *sur ce graphe, puis la fonction* Jump*, puis la fonction* Gather*, et ainsi de suite. Il sera instructif d'observer l'effet des opérations sur les graphes orientés* $(V, \{(i, T(i)) \mid i \in V\})$ *et* $(V, \{(i, C(i)) \mid i \in V\})$.

▷ **Question 2** *Montrer qu'après l'application de la fonction* Gather*, les composantes connexes contenant plusieurs pseudo-sommets induisent des arborescences* 1*-cycliques dans le graphe orienté* $(V, \{(i, T(i)) \mid i \in V\})$. *On notera également que le plus petit pseudo-sommet d'une arborescence* 1*-cyclique appartient au cycle.*

▷ **Question 3** *Montrer que la fonction* Jump *transforme une arborescence* 1*-cyclique en étoile (ou pseudo-sommet).*

▷ **Question 4** *Montrer qu'après* $\lceil \log n \rceil$ *enchaînements des fonctions* Gather *et* Jump*, les composantes connexes du graphe sont représentées par les pseudo-sommets induits par* C.

▷ **Question 5** *Quelle est la complexité de l'algorithme ? Combien de processeurs sont utilisés ?*

2 Anneaux - Rotations de Givens

Pour triangulariser une matrice A d'ordre n de façon numériquement stable, on peut utiliser les rotations de Givens. L'opération de base $\textsf{Rot}(i, j, k)$ consiste à combiner les deux lignes i et j, qui doivent toutes deux commencer par $k-1$ zéros, pour annuler l'élément en position (j, k) :

$$\begin{pmatrix} 0 & \dots & 0 & \mathbf{a'_{i,k}} & a'_{i,k+1} & \dots & a'_{i,n} \\ 0 & \dots & 0 & \mathbf{0} & a'_{j,k+1} & \dots & a'_{j,n} \end{pmatrix} \leftarrow \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} 0 & \dots & 0 & \mathbf{a_{i,k}} & a_{i,k+1} & \dots & a_{i,n} \\ 0 & \dots & 0 & \mathbf{a_{j,k}} & a_{j,k+1} & \dots & a_{j,n} \end{pmatrix}$$

3p Nous laissons au lecteur le soin de déterminer l'angle θ permettant d'effectuer cette opération. :-)

L'algorithme séquentiel peut s'écrire :

```
Givens(A)
1:  Pour k = 1 to n − 1 :
2:     Pour i = n downto k + 1 step −1 :
3:        Rot(i − 1, i, k)
```

On considère qu'une rotation $\textsf{Rot}(i, j, k)$ s'exécute en temps unité, indépendamment de k.

▷ **Question 6** *Mettre en œuvre cet algorithme sur un réseau linéaire de* n *processeurs.*

▷ **Question 7** *Mettre en œuvre cet algorithme sur un réseau linéaire comportant seulement* $\lfloor \frac{n}{2} \rfloor$ *processeurs.*

3 Facteur d'accélération

On discute diverses ■ lois ■ (Amdahl, Gustafson) sur le facteur d'accélération et l'efficacité. Ces lois font désormais partie du bagage de tout honnête paralléliseur !

▷ **Question 8** *Soit un problème à résoudre, qui comporte un pourcentage* f *d'opérations intrinsèquement séquentielles. Montrer que le facteur d'accélération est limité par* $1/f$*, quel que soit le nombre de processeurs utilisés. Quelle leçon en tirer pour la parallélisation d'un problème de taille fixe ?*

▷ **Question 9** *Pour un problème matriciel de taille* $n \times n$*, on suppose que :*

- *le nombre d'opérations arithmétiques à exécuter est* n^α*, où* α *est une constante ;*
- *le nombre d'éléments à stocker en mémoire est* $w_1 n^2$*, où* w_1 *est une constante ;*
- *le nombre d'opérations d'entrées-sorties (purement séquentielles) est* $w_2 n^2$*, où* w_2 *est une constante.*

Comment estimer l'accélération obtenue avec p *processeurs sur un problème de grande taille ? Quelle leçon en tirer pour la parallélisation d'un problème de taille variable ?*

▷ **Question 10** *Donner des exemples de facteur d'accélération superlinéaire, c'est-à-dire d'efficacité strictement supérieure à* 1.