Machines P-RAM

Résumé: Dans ce TD, nous regardons divers algorithmes sur P-RAM. Si le saut de pointeurs est une technique fondamentale, il est intéressant de remarquer que la plupart des algorithmes n'ont rien à voir avec leur équivalent séquentiel et se résument rarement à un bête saut de pointeurs.

1 Sélection dans une liste

▷ **Question 1.** Soit L une liste contenant n objet coloriés soit en bleu, soit en rouge. Concevoir un algorithme EREW efficace qui sépare les éléments bleus des éléments rouges.

2 Composantes connexes

On souhaite concevoir un algorithme CRCW qui permet de calculer les composantes connexes d'un graphe dont les sommets sont numérotés. 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 $j, k, l, \cdots \in V$ tels que $C(j) = C(k) = C(l) = \cdots = i$. 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 tel que pour tout $i \in V = [\![1, n]\!] : C(i) = i$. Ceci signifie que chaque processeur se considère comme sommet de référence de sa composante connexe. L'objectif de l'algorithme est de modifier leur point de vue...

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

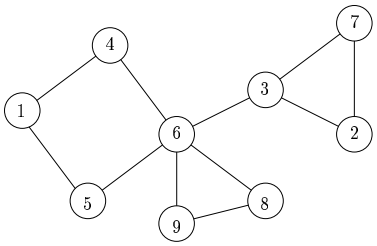
On appelle étoile une arborescence 0-cyclique

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-sommet 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
1: **Pour tout** $i \in S$ **en parallèle**
2: $T(i) \leftarrow \min \{C(j) \mid \{i, j\} \in A, C(j) \neq C(i)\}$
{si l'ensemble est vide, on associe $C(i)$}
3: **Pour tout** $i \in S$ **en parallèle**
4: $T(i) \leftarrow \min \{T(j) \mid C(j) = i, T(j) \neq i\}$
{si l'ensemble est vide, on associe $C(i)$}

JUMP
5: **Pour tout** $i \in S$ **en parallèle**
6: $B(i) \leftarrow T(i)$
7: **Répéter** $\log n$ **fois**
8: **Pour tout** $i \in S$ **en parallèle**
9: $T(i) \leftarrow T(T(i))$
10: **Pour tout** $i \in S$ **en parallèle**
11: $C(i) \leftarrow \min \{B(T(i)), T(i)\}$

▷ **Question 2.** Appliquer la fonction GATHER au graphe suivant.



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 3.** 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 arborescences 1-cyclique appartient au cycle.

▷ **Question 4.** Montrez que la la fonction JUMP transforme une arborescence 1-cyclique en étoile (ou pseudo-sommet).

▷ **Question 5.** Montrez qu'après $\log n$ enchaînements des fonctions GATHER et JUMP, les composantes connexes du graphe sont représentés par les pseudo-sommets induits par C.

▷ **Question 6.** Quelle est la complexité de l'algorithme ? Combien de processeurs avez-vous utilisé ?

3 Rupture de symétrie déterministe

On veut concevoir un algorithme EREW qui permet de sélectionner «beaucoup» d'objets dans une liste chaînée sans jamais choisir deux objets adjacents dans la liste. Chaque objet est associé à un processeur mais le numéro du processeur n'est pas relié à l'ordre des éléments dans la liste (ça serait trop facile...).

▷ **Question 7.** Concevoir un algorithme qui permet sélectionner un nombre optimal d'objets en temps $O(\log n)$ sur une EREW.

Dans la suite, on va montrer qu'il est possible d'extraire «beaucoup» d'éléments en un temps $O(\log^* n)$ où

$$\log^* n = \min \left\{ i \mid \log^i n \leqslant n \right\} .$$

Dans l'expression précédente, $\log^i$ représente la composition de i fois la fonction log. La fonction $\log^*$ a une croissance très lente, puisque $\log^*(2^{65536}) = 5$.

Nous allons maintenant préciser le sens de «beaucoup» d'éléments a partir de la notion d'ensemble indépendant maximal.

Définition 3. Un ensemble de sommets V' d'un graphe $G = (V, E)$ est indépendant ssi

$$\forall (a, b) \in E, \text{ au plus un élément de } \{a, b\} \text{ est dans } V'.$$

Un ensemble de sommets V' d'un graphe $G = (V, E)$ est indépendant et maximal (attention pas maximum) ssi l'ajout de tout sommet à V' en fait un ensemble non indépendant.

Dans ce qui suit, notre objectif est d'arriver à extraire un ensemble indépendant maximal de la liste en temps $O(\log^* n)$. Pour cela, on va commencer par colorier la liste (avec 6 couleurs). On va construire un algorithme qui part d'une n-coloration (où la couleur de chaque objet est déterminée par la couleur du processeur qui lui est associe) et qui diminue à chaque étape le nombre de couleurs utilisées.

▷ **Question 8.** Donner un algorithme astucieux pour diminuer le nombre de couleurs utilisées tout en gardant une coloration (on pourra raisonner sur le codage binaire des couleurs).

▷ **Question 9.** Pourquoi obtient-on 6 couleurs ? Montrer qu'on obtient 6 couleurs après $O(\log^* n)$ étapes.