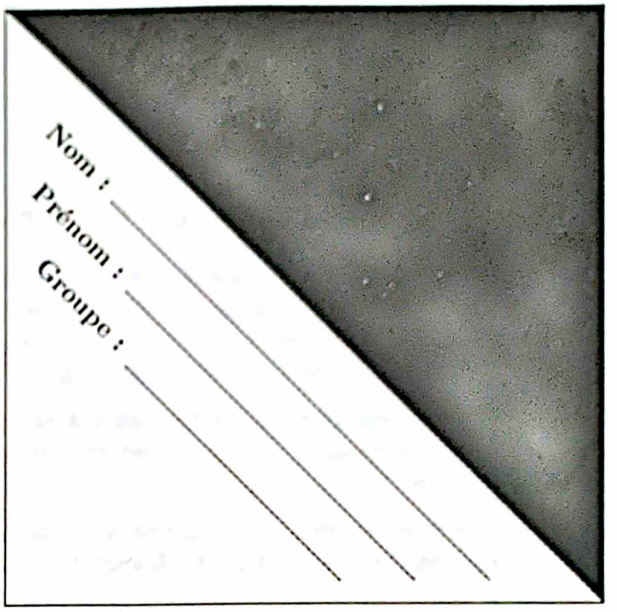
Université Côte d'Azur
Polytech Nice Sophia



Département : SI (4ème année)
Cours : Algorithmique Parallèle
Date : 15 mars 2023

Durée : 1h10
Documents : Aucun

L'utilisation des documents et de n'importe quel dispositif électronique sont interdits. Vos réponses doivent être rédigées de manière claire. Le poids de chaque exercice est donné à titre indicatif et peut-être modifié lors de la correction finale des copies.

1 Tri (5pts)

Cet exo du TD2 a été sciemment laissé sans correction. Le voici à nouveau !

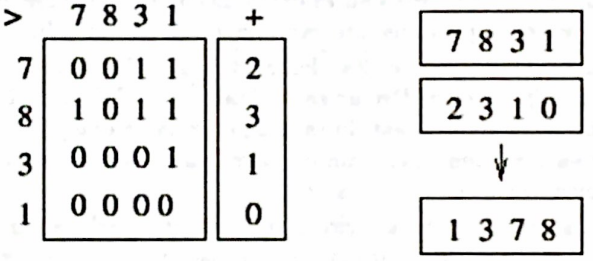


Figure: Exemple de tri sur la séquence de valeurs 7:8:3:1

FIGURE 1 – Tri

L'opération effectuée sur la matrice montrée sur la figure 1 est un test '>' entre la valeur en ligne et la valeur en colonne. Ensuite, on voit que chaque ligne est sommée de la matrice, pour obtenir un vecteur. Puis, les déplacements à réaliser pour obtenir une nouvelle séquence, mais cette fois triée, sont obtenus grâce au dessin à droite de la figure (flèche descendante).

1. Le but de cette première question est d'expliquer avec vos propres mots quel est le principe de cet algorithme de tri (qu'il soit séquentiel ou parallèle). On suppose qu'on nous a fourni en entrée la séquence en ligne (7,8,3,1) dans un tableau L, et sa copie en colonne dans un tableau C (7,8,3,1).
2. Pour l'étape la plus gourmande en nombre de processeurs (celle qui remplit la matrice), donner ce nombre de processeurs qui permettra d'aller le plus vite possible en parallèle. Expliquer quelle variante de PRAM il faut utiliser pour cette étape.
3. Quel est le temps de calcul PRAM total, incluant cette étape, et les autres (celle schématisée par le +, et celle schématisée par la flèche descendante) ?
4. Calculer le travail effectué par cet algorithme.
5. Par rapport à un tri en séquentiel dont la complexité en temps serait de O(NxlogN), est-ce que cet algorithme PRAM est optimal ?

2 Permutation de valeurs d'un tableau (5pts)

On a étudié en TD, un exercice où on déplace toutes les valeurs qui ne sont pas nulles, vers la gauche du tableau résultat. Et on se moquait du reste du tableau ! On avait eu recours à un tableau de Flags, contenant pour chaque élément un 1, si la valeur était non nulle (donc intéressante), et donc à déplacer vers la gauche. Les indices de tous nos tableaux commencent à 1.

Dans cet exercice, on désire réaliser à peu près la même opération, mais, on va aussi s'intéresser à déplacer les éléments dont la valeur n'est pas intéressante, mais vers la droite, en respectant leur position initiale relative.

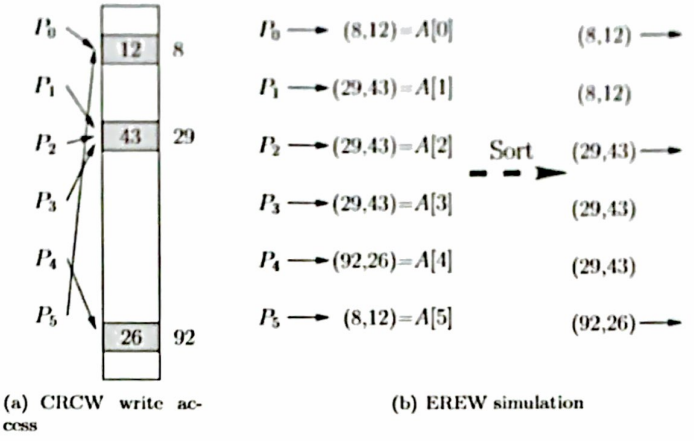
Exemple : supposons qu'on veuille déplacer les valeurs impaires à gauche, et les valeurs paires à droite, du tableau A=[3 4 2 1 4 2 5 6]. L'algorithme devra déplacer les éléments obtenant ainsi A=[3 1 5 4 2 4 2 6]

Le but est de proposer un algorithme PRAM qui permette de réaliser ceci, pas à pas pour un tableau de taille connue.

1. Commencez par rappeler comment on peut aisément, à l'aide d'une opération Prefix avec l'opérateur binaire Somme, savoir quelle sera la position finale des éléments intéressants (comme sur l'ex, la position finale des éléments impairs). On utilise pour cela un tableau de Flags et un tableau auxiliaire nommé sum_prefix. Illustrez votre explication avec le tableau A.
2. On a besoin de déterminer combien il y a de tels éléments intéressants au total (dans l'exemple, le nombre d'éléments impairs). A-t-on forcément besoin d'une variable globale accessible en lecture concurrente pour connaitre ce nombre total ? Si on veut absolument une ER PRAM, expliquer comment faire ? A la fin de cette étape, vous aurez expliqué comment chaque processeur fait pour disposer de cette information, par exemple dans une variable locale nommée "nb_interessants".
3. On veut connaitre la position relative des éléments non intéressants -dans l'ex, les pairs - dans le tableau d'origine. Pour cela, on considère un second tableau de Flags, ou bien la négation du précédent tableau de flags. Montrez en vous aidant du tableau d'exemple que pour un processeur i en charge de l'indice i, si la valeur en i est non intéressante (paire), alors sa position relative au sein de toutes les valeurs non intéressantes est i - sum_prefixe(i)
4. Terminez d'élaborer l'algorithme. Vous avez toutes les informations sur chaque processeur : en une seule opération PRAM if-then-else, écrire le code permettant de déplacer chaque élément à sa bonne position.

3 Simulation d'une PRAM CW avec une PRAM EW (5pts)

1. Expliquez en quelques mots ce qu'est une PRAM où l'on autoriserait le mode CW, ainsi que la variante que vous pouvez visualiser en étudiant la figure 2 partie la plus à gauche, qui est la variante connue sous le nom "Consistent mode". Illustrez votre explication en vous servant par exemple de la case grisée 43, et ce qui est à sa droite, 29.
2. Le reste de la figure 2 schématise comment simuler une instruction d'écriture de la PRAM CW si on dispose seulement d'une PRAM où les écritures sont exclusives.
   Vous pouvez remarquer que chaque processeur stocke dans son entrée du tableau A, une paire d'éléments, tableau qui est ensuite trié comme le montre la droite de la figure 2. Quel est le critère de tri (i.e. "sur quoi on trie") ? Quel est le temps parallèle minimum d'un tri qu'on peut espérer mettre en oeuvre sur une PRAM ?
3. Grâce au tri on a un tableau de valeurs A, trié, et on peut donc utiliser un processeur pour chaque entrée du tableau ; certains des processeurs de la EREW PRAM font l'opération schématisée par les flèches noires les plus à droites, alors que les autres ne font rien. Quel code chacun doit-il exécuter pour réaliser les "flèches noires" qui sont les écritures (sans conflit) dans la mémoire de la EW PRAM ? Pour cette dernière opération que vous devez maintenant coder, vous faut-il une CR ou une ER PRAM ?
4. Conclure : quel est le coût total en temps de cette simulation d'une opération d'écriture d'une CW "consistent mode" PRAM.



(a) CRCW write access

(b) EREW simulation

FIGURE 2 – Simulation de CW avec EW

4 Calcul du préfixe ou Scan (5pts)

L'opération binaire qui doit être utilisée dans les 3 phases de montée, descente, finale du calcul du préfixe sur la PRAM, doit être définie comme associative. Le but de l'exercice est d'arriver à expliquer pourquoi. Supposez l'opération binaire notée $\oplus$.

Pour tout i,j,k, si $\oplus$ est associative, alors $(ai \oplus aj) \oplus ak$ doit être égal à $ai \oplus (aj \oplus ak)$. Et la définition de l'opération prefixe avec cet opérateur $\oplus$ est la suivante :

$$x_i = a_i \; if \; i == 0, else \; x_i = \; x_{(i-1)} \oplus a_i \; if \; 0 < i < n \tag{1}$$

Considérez la donnée d'entrée sur laquelle on veut calculer prefix selon l'opérateur $\oplus$ consituée d'un nombre d'éléments, ici juste 8, éléments notés : a0, a1, a2, a3, a4, a5, a6, a7.

1. Rappeler quelle est la sortie complète du calcul de préfix avec l'opérateur $\oplus$, sur cette collection de 8 éléments
2. Il n'est évidemment pas question d'obtenir cette sortie en implémentant l'équation ci-dessus en séquentiel, mais on recourt à l'algorithme parallèle PRAM étudié. Faites dérouler schématiquement, en dessinant des arbres binaires, quelles combinaisons intermédiaires des valeurs ai sont calculées, lors de la montée dans l'arbre, lors de la descente et de la phase finale. Vous mettrez en évidence le fait qu'on installe l'élément neutre de $\oplus$ à la racine de l'arbre B sur lequel s'exécute la phase de descente.
   Rappels :
   — l'instruction PRAM s'exécutant sur chaque processeur j lors de la phase de montée s'écrit ainsi : A (j)=A(2j)+A(2j+1).
   — le bloc d'instructions PRAM s'exécutant sur chaque processeur j lors de la phase de descente s'écrit ainsi : si j est pair (c'est un fils gauche), B(j)=B(j/2) ; si j est impair (c'est un fils droit), B(j)=B((j-1)/2) $\oplus$ A(j-1)
   — l'instruction PRAM s'exécutant sur chaque processeur j lors de la phase finale s'écrit ainsi : B(j) = B(j) $\oplus$ A(j)
3. Conclure qu'en effet l'opérateur $\oplus$ doit être associatif puisque il en est fait usage dans l'exécution de l'algorithme parallèle que vous avez schématiquement déroulé à la question 2. Montrez plus précisément, quels sont les différents endroits de l'arbre B, où l'on constate l'importance que $((x \oplus y) \oplus z)$ soit égal à $(x \oplus (y \oplus z))$, ces x, y, z pouvant désigner des combinaisons de plusieurs valeurs obtenues avec $\oplus$.