P-RAM

1 Sélection dans une liste

▷ **Question 1** *Soit L une liste contenant n objets coloriés soit en bleu, soit en rouge. Concevoir un algorithme EREW efficace qui sépare les éléments bleus des éléments rouges (c'est-à-dire qui construit une nouvelle liste ne contenant que les éléments bleus).*

2 Recherche des racines dans une forêt

On donne ici un autre exemple de problème pour la séparation des modèles EREW et CREW. Soit $\mathcal{F}$ une forêt d'arbres binaires. Chaque nœud i d'un arbre est associé à un processeur $P(i)$ et possède un pointeur vers son père $pere(i)$. On va chercher des algorithmes EREW et CREW pour que chaque nœud connaisse la racine de son arbre (notée $racine(i)$), et ainsi prouver l'intérêt des lectures concurrentes.

▷ **Question 2** *Donner un algorithme P-RAM CREW pour que chaque nœud détermine $racine(i)$. Démontrer que l'algorithme proposé n'utilise que des lectures concurrentes et déterminer sa complexité.*

3 Procédure mystère

On définit les deux opérateurs suivants pour un tableau $A = [a_0, a_1, \ldots, a_{n-1}]$ de n entiers :
– PRESCAN(A) renvoie le tableau $[0, a_0, a_0 + a_1, a_0 + a_1 + a_2, \ldots, a_0 + a_1 + \ldots + a_{n-2}]$
– SCAN(A) renvoie le tableau $[a_0, a_0 + a_1, a_0 + a_1 + a_2, \ldots, a_0 + a_1 + \ldots + a_{n-1}]$
Nous avons vu en cours comment réaliser ces opérateurs en temps $O(\log n)$ sur une P-RAM EREW.

▷ **Question 3** *Étant donné un tableau de booléens Flags, que fait la procédure suivante ?*

```
SPLIT(A,Flags)
    Idown ← PRESCAN(not(Flags))
    Iup ← n - REVERSE(SCAN(REVERSE(Flags)))
 Pour i = 1 to n en parallèle Si Flags(i)
    Index[i] ← Iup[i] Sinon
       Index[i] ← Idown[i]
       Result ← PERMUTE(A,Index)
       Renvoyer Result
```

Les noms des différentes fonctions sont relativement intuitifs ; en particulier, REVERSE *renverse le tableau, et* PERMUTE(A,*Index*) *réordonne le tableau A selon la permutation Index. L'horrible expression* REVERSE(SCAN(REVERSE(*Flags*))) *effectue simplement un* SCAN *à partir de la fin du tableau Flags, dont les éléments sont considérés comme des entiers.*

Voici un exemple d'utilisation :

A	=	[5	7	3	1	4	2	7	2]
Flags	=	[1	1	1	1	0	0	1	0]
Idown	=	[0	0	0	0	*0*	*1*	2	*2*]
Iup	=	[*3*	*4*	*5*	*6*	7	7	*7*	8]
Index	=	[3	4	5	6	0	1	7	2]
Result	=	[4	2	2	5	7	3	1	7]

Quel est le coût de la fonction SPLIT *?*

On considère la procédure MYSTÈRE suivante :

```
MYSTÈRE(A, Number_Of_Bits) Pour i = 0 to Number_Of_bits − 1
    bit(i) ← tableau indiquant si le i-ème bit des éléments de A est à 1
    A ← SPLIT(A,bit(i))
```

▷ **Question 4** *Faire tourner la procédure sur* $A = [5, 7, 3, 1, 4, 2, 7, 2]$ *avec* $Number\_Of\_Bits = 3$.

▷ **Question 5** *Que fait la procédure* MYSTÈRE *?*

▷ **Question 6** *Avec des entrées de taille* $O(\log n)$ *bits, quelle est la complexité avec* n *processeurs ? Et avec seulement* p *processeurs ? Quelles sont les valeurs de* p *les plus intéressantes ?*