Exercice 1

Do the following former exam. See annex.

Exercice 2

Goal is to find what is the algorithm used below to sort using a PRAM. The expected behaviour is summarized by the figure below.

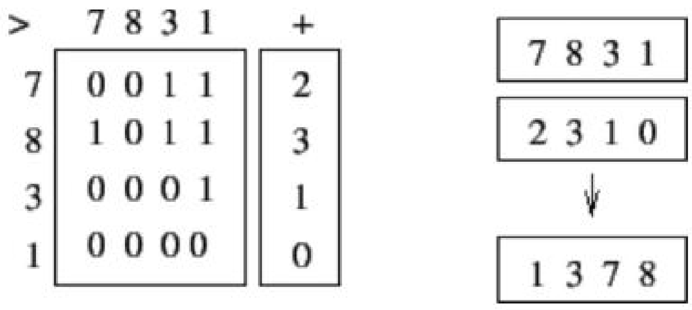

Figure: Exemple de tri sur la séquence de valeurs 7;8;3;1

Sketch the algorithm, with the aim it runs as quick as possible, without constraint on the number of required processors of the PRAM. Give precisely which PRAM variant is needed, again, do not constraint yourself. Under the necessary PRAM variant to use, give precisely what is the parallel time, and the work. To which sort of sequential sorting method does that algorithm belong to ?

Université Nice Sophia Antipolis
UFR Sciences

Parallelism

Duration : 90 minutes
Documents : A single A4 hand-written paper
All answers must be **justified.**

Exercise 1: Find the position of the first occurrence of element X in an unsorted array

The goal of this exercise is to design a parallel algorithm using a minimal number of processors to find the position of the first occurrence of an element X in an array. For the remaining of the exercise, we well consider only an array containing either 0 or 1 and we look for the first 1. For example, given the array $[0, 0, 0, 1, 0, 1, 0, 1]$, the algorithm will return 4 (because indexes start at one).
Here is a summary of the algorithms and their various steps that will be designed in this exercise.

Algorithm	steps
firstOne	prelim modif final
firstOne-V2	there-is-a-1 firstOne

Preliminary steps

1. Give a sequential and iterative version of an algorithm finding the position of the first 1 in an array.
2. What is the time complexity of this algorithm ?

First PRAM Version (`firstOne`)

For the PRAM version, we will use an array B with the same size as the initial array (A) and initially filled with the same values. This first step is called `prelim`.

1. Write a parallel algorithm which creates B from A (step `prelim`)

The rest of the algorithm is composed of 2 parts which will be studied separately. The first one (called `modif`) modifies B as follows :

```
For all i,j such as i<j
If B[i]=1 AND B[j]=1 Then B[j]=0
```

1. Show that at the end of `modif`, array B contains a single 1 at the position we are looking for.
2. Write the algorithm for step `modif` using N^2 processors.

The second part (called `final`) takes B as an input and writes the correct position in a variable called $POSITION$.

1. Write a parallel version of `final`
2. How many processors are required ?

Now that we have all the steps (`prelim, modif, final`) we can study the whole algorithm (called `firstOne`)

1. For each step, indicate the time complexity, the required number of processors, and the needed PRAM.
2. Is this algorithm work optimal ?

Reducing the number of processors

The goal of this part is to reduce the number of processors of the previous algorithms to have a more efficient version and create a `firstOne-V2`. We will create an algorithm, called `there-is-a-1` which store '1' in a variable if the array given as a parameter contains at least one '1'. It is equivalent to a logical OR.

1. Propose a parallel algorithm for `there-is-a-1` which takes $O(1)$ on a CRCW PRAM.

The new version of our algorithm now works like this. Array A is divided into sub-arrays of size X. Each sub-array is processed in parallel using `there-is-a-1` and the returned values are stored in C. We then execute `firstOne` on C which return the position of the first '1' in C. Using this information, we use `firstOne` on the corresponding sub-array of A which gives us the final value.

1. Show the execution of the different steps of the *firstOne-V2* algorithm on $[0,0,0,0,1,1,1,0]$ with $X = 2$.
2. Give the index of the first '1' in C, how can you find the position of the first '1' in A ?
3. What is the time complexity of this algorithm ?
4. What could be an optimal value for X *(difficult)* ?