1 Algorithms for PRAMs

1.1 Simulating a new type of PRAM

A Concurrent Read Tolerant Write (CRTW) PRAM is a PRAM model that is slightly different from the CRCW PRAM model:

- concurrent reads are authorized;
- several processing units can write the same value in the same memory cell;
- if several processing units attempt to write different values in the same memory cell, the conflict is not resolved and the content of that memory cell is not modified.

Show that any algorithm that runs on a CRTW PRAM with \( n \) processing units in time \( t \) can be simulated on a CRCW PRAM (in arbitrary mode) in time \( O(t) \).

Recall that in the arbitrary mode, processing units can write different values in the same memory cell but only one of the values will be written (this mode does not specify which value is written).

1.2 Array initialization

We want to write a parallel algorithm that takes as input two integers \( n \) and \( x \), and that creates an array \( A \) of size \( n \) such as \( A[i] = x \) for \( 1 \leq i \leq n \).

1. Propose an algorithm running in time \( O(\log n) \) on an EREW PRAM with \( O(n) \) processors.

2. Propose an algorithm running in time \( O(\log n) \) on an EREW PRAM with \( O(n/\log n) \) processors.

3. Propose an algorithm running in time \( O(1) \) on a CREW PRAM with \( O(n) \) processors.
1.3 Linear recursion

We consider the sequence \((u_n)_{n \in \mathbb{N}}\) defined as follows:

\[
\begin{align*}
  u_0 &= b_0 \\
  u_i &= a_i \times u_{i-1} + b_i & \text{for } i > 0
\end{align*}
\]

We want to compute efficiently the terms \(u_i\) with \(0 < i < n = 2^p\).

1. In this question, we assume that \(a_i = 1\) for any \(i\). Propose an algorithm that efficiently computes the \(u_i\)'s on a PRAM. What is its complexity?

2. Propose an algorithm that efficiently computes the \(u_i\)'s on a PRAM in the general case. What is its complexity?

3. Propose an algorithm to efficiently evaluate the value of a polynomial:
   \(P(X) = p_0 + p_1 \times X + \cdots + p_{n-1} \times X^{n-1}\) on a PRAM with \(n\) processors. What is its complexity?

2 Algorithms on rings or grids of processors

2.1 Line of processors

Let us consider a linear, bidirectional network of processors in the message-passing model:

\(P_1 \leftrightarrows P_2 \leftrightarrows \ldots \leftrightarrows P_n\).

Assume that, at the beginning, each processor \(P_i\) holds a piece of data \(d_i\) (typically, a matrix) and that we want to compute the product \(d_1 \otimes d_2 \otimes \ldots \otimes d_n\), assuming \(\otimes\) to be associative. We only want the result to be stored in at least one processor (you should specify which one).

Call \(\tau\) the time needed to send a piece of data through one link and \(w\) the time needed to compute a product \(a \otimes b\).

For simplicity’s sake, you can assume that processors can both send and receive (unrelated) data in a single step to/from both sides.

1. Propose an algorithm for computing the product in a \(P_i\) of your choice, optimizing for the case \(\tau \gg w\). Discuss its precise running time (i.e., do not hide the constants in a \(O()\)).

2. Same question, but try to optimize for \(w \gg \tau\) (you can assume \(n = 2^k - 1\) for some \(k\) to make your life easier).

2.2 Matrix transposition

We want to design a parallel algorithm to transpose an \(n \times n\) matrix. We assume the matrix to be stored in a distributed way on the processors. We also assume that the communication links are bidirectional links and that they can be used simultaneously (to transfer independent data).

1. Propose an algorithm for a ring of \(p\) processors. What is its complexity?

2. Propose an algorithm for a torus grid with \(p = q \times q\) processors. What is its complexity?
3 Sorting networks

This section only considers primitive sorting networks (i.e., crossing of wires are not free and do add to the depth of networks).

1. Recall the statement of the $0 - 1$ principle.

We want to design a sorting network to put $n \times m$ matrices in snakelike order: matrix $A = (a_{i,j})_{i<n, j<m}$ is in snakelike order when the following constraints (illustrated in Figure 1) are satisfied:

$$
\begin{align*}
    a_{i,j} &\leq a_{i,j+1} & \text{if } i \text{ is even and } j < m - 1 \\
    a_{i,j+1} &\leq a_{i,j} & \text{if } i \text{ is odd and } j < m - 1 \\
    a_{i,m-1} &\leq a_{i+1,m-1} & \text{if } i < n - 1 \text{ is even} \\
    a_{i+1,0} &\leq a_{i,0} & \text{if } i < n - 1 \text{ is odd}
\end{align*}
$$

Figure 1: The snakelike order over a $4 \times 4$ grid.

A candidate sorting algorithm, the shearsort, iterates a main loop $\lceil \log(n) \rceil + 1$ times consisting of these two steps:

(i) sort even rows in increasing order and odd rows in decreasing order in parallel;

(ii) sort every column of the matrix in increasing order in parallel.

2. Estimate the number of comparators and the depth of the network needed to implement this algorithm using odd-even sorting networks.

Now, we turn to proving the correctness of this algorithm. Using the $0 - 1$ principle, we shall restrict ourselves to $0 - 1$ matrices.

A row $i$ of $A$ is said to be uniform if $a_{i,j}$ is constantly 0 or 1.

3. Suppose that $A$ is a matrix obtained after sorting the rows as prescribed above. Argue that we get exactly

$$
\min_{j<m} \left( \sum_{i=0}^{n-1} a_{i,j} \right) + \min_{j<m} \left( \sum_{i=0}^{n-1} 1 - a_{i,j} \right)
$$

uniform rows after sorting the columns, in the right position (i.e., rows of 0s at the top and rows of 1s at the bottom).

4. Show that this quantity is greater or equal than $\lfloor \frac{n}{2} \rfloor$ (hint: group rows $2i$ and $2i+1$ in the sums before making the minima commute with the sum in your inequality).

5. Show that the shearsort does sort a grid in snakelike order.