Exercice 1

In the course, we have shown 2 versions of the parallel computation of the maximum of n values.
Q1) Explain why the proposed CRCW PRAM algorithm that is using n^2 processors can indeed compare all needed values together in just one single phase. Remember that to find the maximum of a set of n values, each of them must at some point be compared to all the others.

Indeed, to extract the maximum of n values, each value, let us say, x, must be checked against all the n-1 other; in order to eliminate that value x, if one of the other n-1 values , eg y, is greater than this x. . Eg, (1,3,5,7); 1 gets eliminated if compared to 3,5,then 7. Still, we just have eliminated 1, and we do not know yet the maximum. The same has been done for 3, it gets compared to 1,5,7, so gets eliminated. Same for 5. The only value that, once compared to all the others has not been eliminated is indeed the maximum (here 7). The proposed CRCW algo implements this approach. The total number of comparisons done is (n^2-n).

Why in the sequential algorithm that you can easily write down, the time complexity ends up being in O(n) (and not O(n^2)).

In the sequential algorithm below, an optimisation of the brute force above approach is incorporated:

Max=-infinity;

For (int i=1, i<=n,i++) if T[i]>Max, Max=T[i]

=> at the end, Max contains the maximum, and the number of comparisons that have been done is simply (n-1). This may look strange that time is not O(n^2). This is thanks to the fact that after each comparison, the result of that test is kept in Max. So, at each loop turn, whenever Max is updated, it automatically eliminates the need to compare the previous Max values with any remaining (n-1) other values with which it may not have been compared yet. So the complexity of solving the problem in sequential is O(n). Obviously, we would aim for a PRAM algorithm whose total work (parallel time * number of procs) is also O(n). This is not the case for the version 1, as the work is O(n^2), so this is not optimal.

Q2) Explain why the proposed CRCW PRAM algorithm has to allow CR ?

Concurrent read operations are needed, because if eg, i=3, and j=4, one proc will have to read T[3] and T[4]. But in parallel, eg for i=4, and j=6, another proc will read also T[4] (and T[6]). So, at worst, a given value, T[k] is simultaneously read by n-1 processes.

Q3) Explain why the proposed CRCW PRAM algorithm has to allow Arbitrary CW ?

Some of these n-1 processes read T[k], 1<=k<n comparing T[k] with another value, T[l] 2<l<=n in parallel, can find that T[k]<T[l], so, need to write FALSE in m[k]. As all that need to write something, in parallel, must write the same value (FALSE), an ARBITRARY PRAM is sufficient. And we can see that the CW mode is needed.

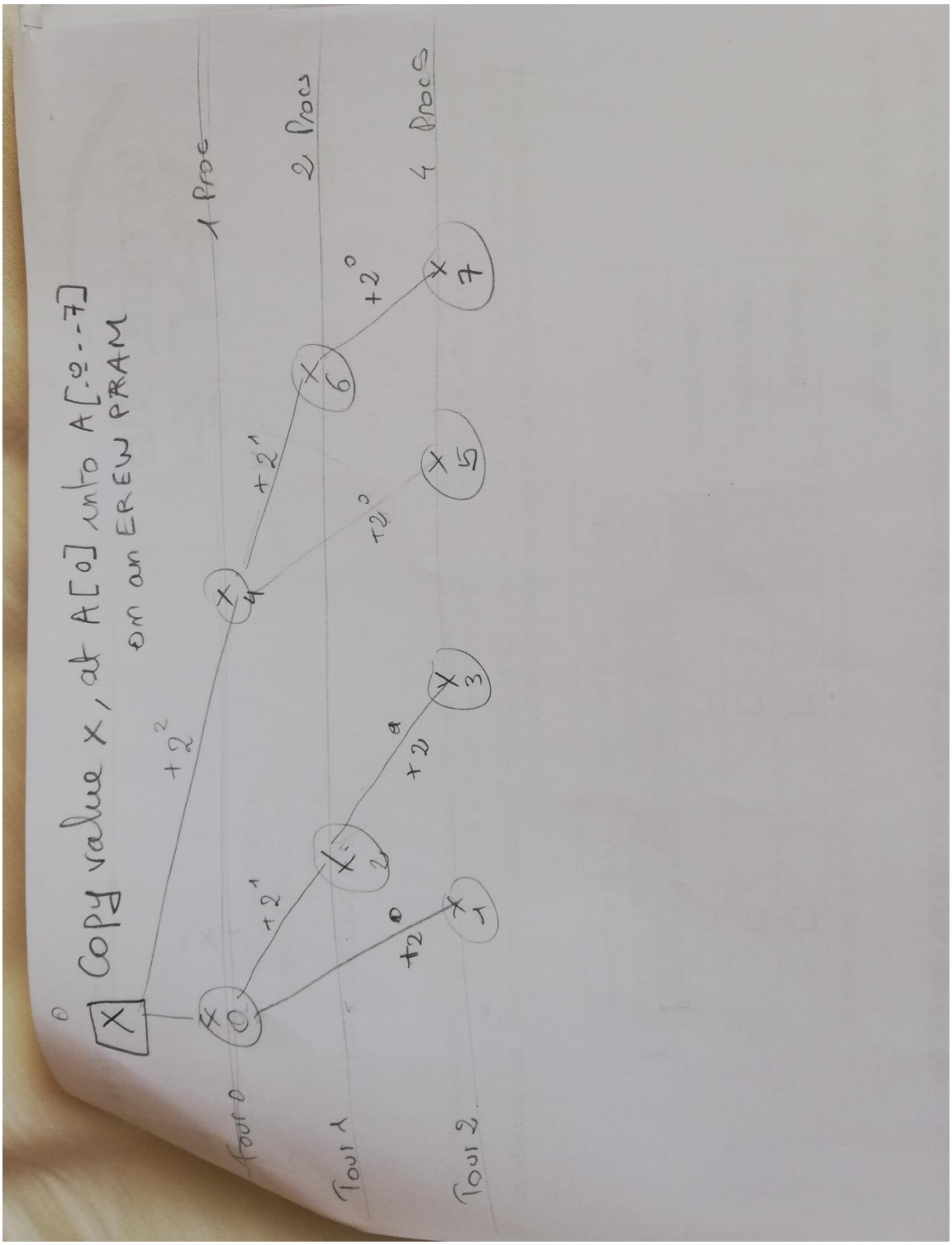
Q4) Sketch how to simulate a C.R. of this algorithm, on an E.R PRAM.

For the C.R simulation, write down the complete algorithm, assuming the value to be copied to the n processors, is stored in an array of size n, at index 0. Assume that n=2^m, so, you can iterate in O(m) parallel steps. Highlight why the simulation has only O(log n) parallel time complexity, given the number of data is n.

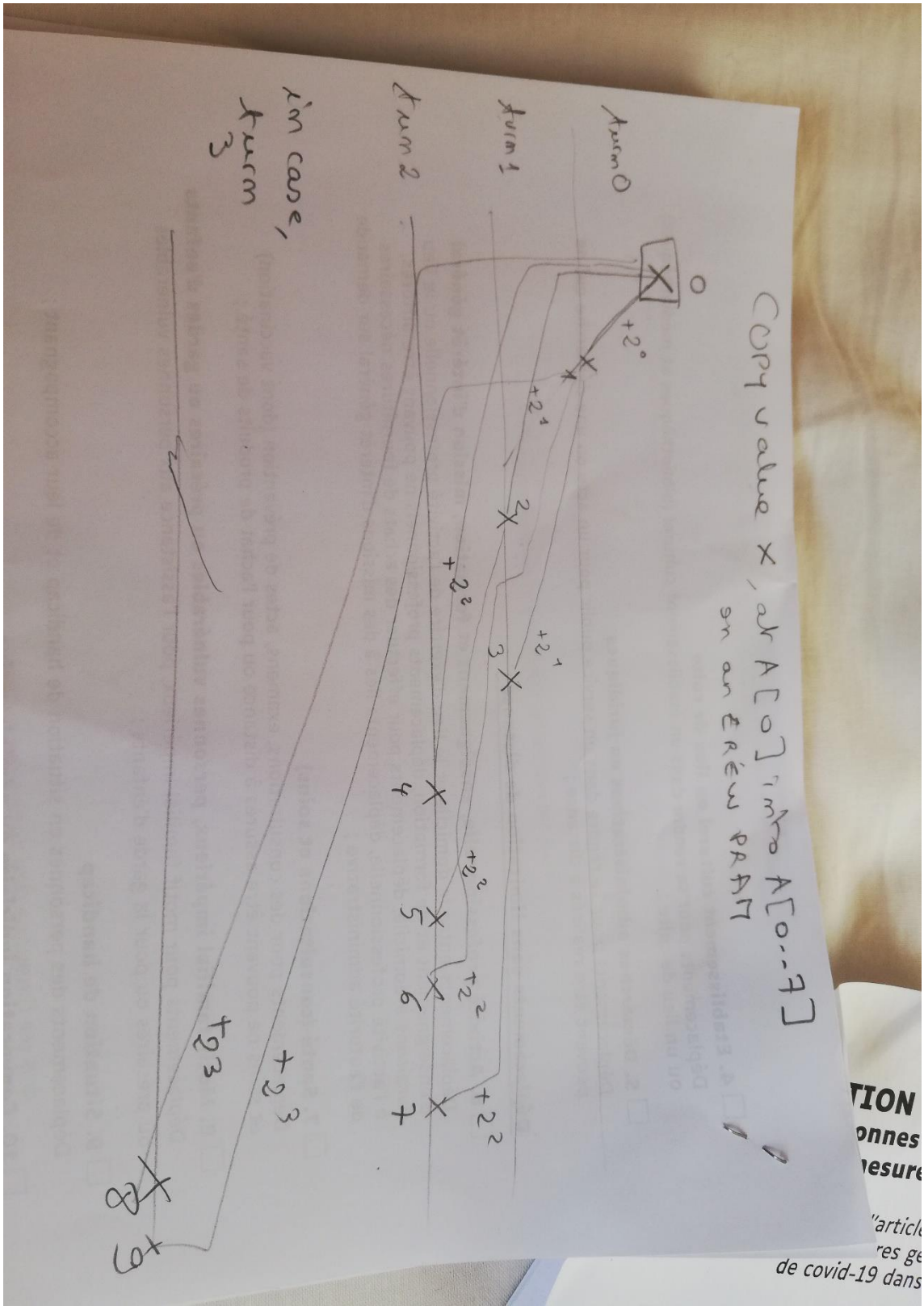
The simulation of a CR operation will be done in this specific context, but, consider that it is a general method to simulate a CR operation. (same will happen for the CW arbitrary operation).

To simulate a concurrent read operation of T[k] by O(n) procs : one proc only reads T[k] , it duplicates it in an auxiliary array A, at addresses eg A[1], A[2]. Then, two proc in parallel read A[1] and A[2]. Proc 1 duplicates A[1] in A[3] and A[4], while Proc 2 duplicates A[2] in A[5] and A[6], etc… Let us write that clearly: In order to do so, we will assume we start at A[0] and not A[1], and, more precisely, procs will not fill up A in order as you will see.

We may think of this sort of “pseudo tree” that depicts data movement, on which one can see which procs ids should be active at each step, etc…



Or, because writing the resulting algorithm of that data movement schema is not easy, we can think of another schema for data to be copied and by which proc, at each parallel step/turn.



Now, writing the algorithm, and “testing” that it makes the job is more easy

Q5) Sketch how to do the same for the C.W arbitrary mode, on an E.W. PRAM.