Fundamental Algorithms 9 - Solution Examples

Exercise 1 (Parallel sort)

BucketSortPRAM is a proposed parallel implementation of the BucketSort algorithm in a PRAM, \( n \) processor model. \texttt{copy} is a sequential algorithm which takes an array \( A \) and an array of arrays \( B \) as arguments and copies the elements of \( B \) into \( A \) in order. The function \texttt{index} distributes the array elements evenly into buckets, i.e. elements are assigned to buckets with (roughly) the same probability. Moreover, larger elements are assigned to larger buckets, i.e. \( \texttt{index}(a) < \texttt{index}(b) \) implies \( a < b \).

Algorithm 1: BucketSortPRAM

\begin{verbatim}
Input: A: Array[1..n]
Result: A is sorted
B ← Array[1..nb];
for i = 1 to n in parallel do insert(B[index(A[i])], A[i]);
for i = 1 to nb in parallel do BubbleSort(B[i]);
copy(A, B);
\end{verbatim}

1. For the two parallel loops in BucketSortPRAM, state for both arrays \( A \) and \( B \) whether there is concurrent or exclusive read / write access to their elements.

2. Implement \texttt{CopyPRAM}, i.e. a parallel version of \texttt{copy} You can use a constant-time \texttt{len} function to determine the length of an array. What is the parallel complexity (depending on \( nb \)), and how many processors can your algorithm use?

Solution:

1. In the first parallel loop, there is exclusive read access to \( A \), but concurrent read and write access to the buckets \( B \) (as soon as two or more elements are assigned to the same bucket). In the second parallel loop, there is exclusive read and write to the bucket elements (each processor sorts exactly one bucket assigned to it exclusively).

2. The proposed algorithm requires \( O(\log nb + \log l) \) steps, where \( l \) is the largest bucket, on \( p = n \) processors.
Algorithm 2: CopyPRAM

**Input:** A: Array[1..n]
\hspace{1cm} B: Array[1..nb]

**Result:** A contains all elements of B in order

\hspace{1cm} Len ← Array[1..nb];

\hspace{1cm} for \( i = 1 \) to \( nb \) in parallel do \( \text{Len}[i] \leftarrow \text{len}(B[i]) \);

\hspace{1cm} Pos ← Array[1..nb]; // Parallel Prefix-Addition of Len into Pos

\hspace{1cm} for \( i = 1 \) to \( nb \) in parallel do

\hspace{2cm} len ← \text{Len}[i], \ pos ← \text{Pos}[i], \ arr ← B[i];

\hspace{2cm} PosI ← Array[1..len]; // Copy pos into PosI with binary-fan out

\hspace{2cm} for \( j = 1 \) to \( len \) in parallel do \( A[\text{PosI}[j] + j] \leftarrow arr[j] \);

end