Fundamental Algorithms 6

Exercise 3

(from previous worksheet)

Given is the following parallel algorithm for prefix multiplication (for an EREW-PRAM).

```
PrefixPRAM(A: Array [1..n]) {
  // n assumed to be 2^k
  // Model: EREW PRAM (n-1 processors)

  for l from 0 to k-1 do
     for j from 2^l+1 to n do in parallel {
        tmp[j] := A[j-2^l];
        A[j] := tmp[j]*A[j];
     }
}
```

Assume that the j-loop of the above program is changed to

```
for j from 2^l+1 to n do { ... }
```

(i.e., changed to a sequential loop). State why the resulting algorithm is no longer correct, and suggest how to change the j-loop to obtain a correct sequential implementation. Also, state why the parallel loop works correctly.

Solution:

If the j-loop of the program is changed to

```
for j from 2^l+1 to n do { ... }
```

then A[j−2^l] is already changed to its new value, when A[j] is updated. We obtain a correct implementation, if the j-loop is executed in reverse order, or if the j-loop is split into two loops: the first loop to compute all tmp[j], and the second loop to update the A[j]. The parallel loop works correctly, because all tmp[j] are assigned their value at the same time, i.e., before these values are copied to the A[j].

K-Exercise 1

The following parallel implementation of the BucketSort algorithm is proposed:

```
BucketSortPRAM(A: Array [0..n−1]) {
   // Model: PRAM with n processors

   Create Array B[0..n−1] of Buckets;
   // all Buckets B[i] are created empty

   for i from 0 to n−1 do in parallel {
      insert A[i] into Bucket B[index(A[i])];
   }

   for i from 0 to n−1 do in parallel {
      BubbleSort( Bucket B[i] );
   }

   A := Concat( B[0], B[1], ..., B[n−1] );
}
```

Here, Concat is a sequential algorithm that concatenates all buckets given as parameters in the provided order. For simplicity, we assume that Concat can have any number of parameters.

The function index has to distribute the array elements *evenly* into buckets. Hence, its range of values is $1, \ldots, n$, and elements are assigned to buckets with about the same probability (for the expected distribution of elements).

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

b) Describe a parallel algorithm, for an EREW PRAM, to concatenate the buckets in the last step, using as many processors as possible. What is the parallel complexity to concatenate all buckets (depending on n), and how many processors can your algorithm use?

Solution:

a) 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).

b) The buckets can be combined by a *binary fan-in* algorithm, for example via the following implementation:

```
nb := 1;
while nb < n {
   for i from 0 to n−1 by (2∗nb) do in parallel {
      B[i] := Concat( B[i], B[i+nb] );
   }
   nb := 2∗nb;
```

```
}
```

The algorithm requires $O(\log n)$ steps on $p = \frac{n}{2}$ processors.

There is also an alternative implementation:

```
nb := n/2;
while nb > 1 {
  for i from 0 to nb-1 by n/nb do in parallel {
     B[i] := Concat( B[i], B[i+n/nb] );
  }
  nb := nb/2;
}
```