Parallel Algorithms Spring 2022

Problem Set 2

Prof. Nodari Sitchinava *Due: Wednesday, February 2, 2022 at 9am*

You may discuss the problems with your classmates, however **you must write up the solutions on your own** and **list the names** of every person with whom you discussed each problem.

Start **every** problem on a separate page. ***Any problem submitted by 11:59pm Friday January 28, 2022 will receive an additional 10% of the score you receive on that problem.***

1 Reducing Loop Steps (30 pts)

In lecture, when computing the minimum iteratively, we used the following pseudocode:

```
MIN(a[1..n])
  for ℓ = 1 to log n
    for i = 1; i ≤ n; i = i + 2^ℓ in parallel
      if i + 2^(ℓ−1) ≤ n
        a[i] = min(a[i], a[i + 2^(ℓ−1)])
  return a[1]
```

The problem with this code is that it is not trivial to allocate processors to the inner parallel **for** loop. One way to do this is to use n processors, and have each processor check whether its index satisfies a condition to execute the body of the loop, or whether it should do nothing. As the outer loop progresses, more and more processors would stay idle. However, they cannot be utilized for other tasks, as they still need to check the condition. This is a waste of processor resources.

Instead, rewrite the above code, so the inner for loop increments only by 1 step in each iteration, i.e.,

```
MIN(a[1..n])
  for ℓ = 1 to log n
    for i = 1 to k in parallel        ▷ For an appropriately-defined k as a function of ℓ
      ...
  return a[1]
```

Note how much easier it is to determine how to allocate only the processors that will be doing the work of the the inner loop iterations: always allocate processors $p_1, p_2, \ldots, p_k$ and they all will be busy executing the body of the inner loop.

Make sure that you replace k with the appropriate function of ℓ and that your modified code still computes the minimum correctly.

2 Prefix Maxima (30 pts)

Given an array $a[1..n]$ of real numbers, the problem of ***prefix maxima*** asks to compute an array $b[1..n]$, such that each entry $b[i]$ is equal to the maximum element among the first i elements of a, i.e. $b[i] = \max_{1 \le k \le i} a[i]$.

Design an EREW PRAM algorithm for computing prefix maxima in time $T(n) = O(\log n)$ and work $W(n) = O(n)$. As always, don't forget to prove its correctness and analyze its time and work complexity.

Hint: modify the recursive algorithm for Prefix Sums shown in lecture.

3 Ocean View (40 pts)

Waikiki has many tall buildings, but only some of them have a clear view of the ocean. Suppose we are given an array $A[1..n]$ that stores the height of n buildings on a city block, indexed south to north (from the ocean to the Ala Wai Canal). Building i has a good ocean view if and only if every building to the south of i is shorter than i.

Design a CREW PRAM algorithm that identifies which buildings have a good view. If m buildings have a good view, the output of your algorithm should be an array B of size m, that contains the indices of the buildings that have good views, i.e., for every $1 \leq k \leq m$, the $B[k]$-th building in A has a good view.

Your algorithm should run in $O(\log n)$ time and $O(n)$ work.

Hint: Think about the structure of the solution before starting to write the pseudocode. What identifies a building that has a good view?