EE 382V: Parallel Algorithms **Summer 2017**

Lecture 2: June 9

Lecturer: Vijay Garg *Scribe: Eric Addison*

Problem 1.3

Give a parallel algorithm on a CREW PRAM to determine the largest odd number in a given array of positive integers. Assume that the array has size n and the number of processors available is also n. The size n may not be a power of 2. Your algorithm should not take more than O(log n) time.

Assume 0-based indexing. This algorithm follows the standard binary tree work-depth model, similar to the canonical `reduce`-sum algorithm. The key expression is on line 8.

```
1  i = get_my_id();
2  B[i] = A[i];
3  hmax = ceil(log(n))
4  for h = 1 to hmax do
5    if (i < n/(2^h)) then
6      x = B[2*i];
7      y = B[2*i+1];
8      B[i] = max( (x%2)*x, (y%2)*y );
9  print B[0];
```

This algorithm is asymptotically equivalent in complexity to the binary tree `reduce`-sum problem. It has complexities $\boxed{T(n) = O(\log n)}$ and

Problem 1.5

Given an integer array A and two numbers x and y, give a parallel algorithm on a CREW PRAM to compute an array D such that D consists only of entries in A that are greater than or equal to x and less than or equal to y. The order of entries in D should be same as that in A.

I use a four step approach here:

1. Compute indicator array `B` determining which elements meet the critieria
2. Compute the correct index locations in array `C` using parallel prefix-sum
3. Allocate new properly sized array `D`
4. Populate `D` with appropriate elements