Recitation 3 – *Scan*

Parallel and Sequential Data Structures and Algorithms, 15-210 (Fall 2014)

September 9th, 2014

1 Announcements

- How was *ParenLab*?
- Lab 2 – *SkylineLab* has been released and is due next Monday, September 15th. This lab is conceptually much more difficult than the previous one, so start early!
- Questions from lecture or homework?

2 Scan

`scan` takes a function as one of its arguments. All of the text below makes the assumption that this function is *associative*. Recall the mathematical definition that a function f is said to be associative if and only if

$$\forall a \forall b \forall c. f(f(a,b),c) = f(a,f(b,c))$$

We also make the assumption that the initial value is a *left-identity* of the functional argument. Recall the mathematical definition that I is a left-identity of f if and only if

$$\forall a. f(I,a) = a$$

We don't need these assumptions in general, and we'll come back to a version of `scan` later that doesn't have them, but it's a cleaner way to start thinking about `scan` with these properties.

With the assumption that `f` is associative, `(scan f b)` is logically equivalent to `(iterh f b)` in the same way that `(reduce f b)` is logically equivalent to `(iter f b)`, but these functions differ in their span. Specifically, if `f` is a function that takes no more than a constant number of steps on all input, `(iterh f)` and `(iter f)` have both work and span in $O(n)$, whereas `reduce` and `scan` both have work in $O(n)$ and span in $O(\lg n)$.

It's worth noting that while `reduce` and `scan` are highly parallel, unlike `iter` and `iterh`, they pay the price by having slightly less general types.

2.1 Note on Terminology

If f is a function and I is a relevant identity for f, we'll often say "f-scan" to mean `scan f I` For example, a "+-scan" is `scan (op +) 0`

2.2 Recap

If $\texttt{s} = \langle 1, 6, 3, -2, 9, 0, -4 \rangle$, then

`(scan Int.min Int.maxInt s)` yields the following:

$(\langle \texttt{Int.maxInt}, 1, 1, 1, -2, -2, -2 \rangle, -4)$

Remember that in the result, location i stores the "sum" of the values at locations **before** i in the original sequence. There is a variant of `scan` called `scanI` which sums the values at locations before and including i.

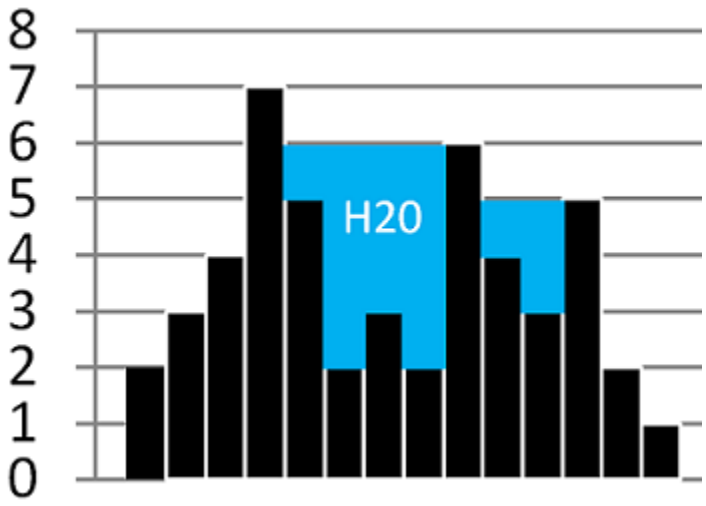
2.3 Example Uses of Scan

At first glance, `scan` seems to offer not much that isn't already available through `reduce`. With clever choices of associative functions, though, `scan` can be used to compute some surprising things efficiently in parallel.

2.3.1 Histogram

Consider the following problem:

Given a histogram, if we were to pour water over it, how much water (in terms of area) would it hold? For simplicity we will represent a histogram as a sequence of non-negative integers. For example the histogram shown below is represenented by the sequence $\texttt{s} = \langle 2, 3, 4, 7, 5, 2, 3, 2, 6, 4, 3, 5, 2, 1 \rangle$, and holds 15 units of water.



Any ideas on how we might solve this problem?

2.3.2 Matching Parentheses

Let's revisit the parentheses matching (parentheses closure may actually be more appropriate) question from the last recitation. Remember, this question just asks: "Given a sequence of parentheses, is the sequence closed?". We can solve this very cleanly using scan, so give it a try!

3 Whose bonus is it anyways? (Where the points don't matter!)

3.1 MCSS

The Stock Market Problem asks "Given a sequence of integers, representing the value of a stock each day, what would have been the best time to buy the stock, and then sell the stock?" Remember that to make the most money off of the stock, you want to buy the stock when its price is low, and sell the stock when its price is very high, and you need to sell the stock after you buy it (no short-selling!). How can we use scan to solve the Stock Market Problem?

Now that we've solved the Stock Market Problem, we want to think about the related "Maximum Contiguous Subsequence Sum" problem. This question asks "Given a sequence of integers, find the subsequence with the maximum sum". Remember that there could be negative numbers in there!

3.2 Matching Parentheses 2, Parenthetical Boogaloo

We'd like to solve a more general form of the parentheses matching problem. Given a sequence of parentheses, we'd like a sequence of integers, where the value of the sequence at position i is the index of the parenthesis that the parenthesis at position i matches with. For example, for the parenthesis sequence "(()(())())"

We'd return the sequence $\langle 9, 2, 1, 6, 5, 4, 3, 8, 7, 0 \rangle$

Like everything else in this recitation, this can be quite cleanly done by using scan in your solution! Note that there may be some pre and post processing for this solution.