Recitation 4 – *Scan Reloaded and Reductions*

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

February 4th, 2014

1 Announcements

- How did Skyline go?
- Bignum is out—get an early start!
- Questions about homework or lecture?

2 Scan Implementation

Scan is a complex operation, so we're going to work through one level of recursion (not a whole trace like you did for `iterh` on Minilab).

Let $S = \langle 1, 2, 3, 4, 5, 6, 7, 8 \rangle$. We'll look at `scan op+ 0 S`.

First, `scan` contracts the sequence to a new S' by contracting every other pair of elements, giving us

$S' = \langle 3, 7, 11, 15 \rangle$.

Then, a recursive call to `scan op+ 0 S'` will return:

$(\langle 0, 3, 10, 21 \rangle, 36)$

We then interleave values from S into S', giving us the final scan of

$(\langle 0, 1, 3, 6, 10, 15, 21, 28 \rangle, 36)$

Note that this particular sequence is much easier to scan than certain other sequences—why?

3 Scanning the Stock Market

You're working as a consultant for the QADSAN stock market, and to maximize your profits you want to determine the optimal times to buy and sell stocks. Instead of making predictions, however, you're going to look at all the opportunities you didn't take in the past to make money on the market. Your task is: given a sequence of stock prices over time $S = \langle p_1, \ldots, p_n \rangle$, find the largest increase in price, or $\max_{i=1}^{n} \left(\max_{j=i+1}^{n} \left(p_j - p_i \right) \right)$ in $O(n)$ work and $O(\log n)$ span.

```
fun stockMax (S : int seq) : int =
```

Solution 3.0

```
fun stockMarket (S : int seq) : int =
  let val mins = scani Int.min (valOf Int.maxInt) S
      val maxs = rev(scani Int.max 0 (rev S))
  in reduce Int.max 0 (map2 op- maxs mins)
  end
```

4 Reduction

1. Write a function `rev` which reverses the input sequence. Here's the twist: you can only use the following functions: `map`, `reduce`, `empty`, `singleton`, `append`, `length`, `filter`.

   ```
   fun rev (S : 'a seq) : 'a seq =
   ```

   Solution 4.0

   ```
   reduce (fn (x, y) => append(y, x)) (empty()) (map singleton S)
   ```

2. Give a closed form for the work and span of `rev` under both the `ArraySequence` and `TreeSequence` implementations. Given two sequences S and T of size n and m, the cost bounds in `ArraySequence` for the above functions are:

Function	Work	Span
`map`	$\sum_{e \in S} \mathcal{W}(f(e))$	$\max_{e \in S} \mathcal{S}(f(e))$
`reduce`	$O(n) + \sum_{f(x,y) \in \mathcal{O}_r(f,b,s)} \mathcal{W}(f(x,y))$	$O(\log n \max_{f(x,y) \in \mathcal{O}_r(f,b,s)} \mathcal{S}(f(x,y)))$
`empty`	$O(1)$	$O(1)$
`singleton`	$O(1)$	$O(1)$
`append`	$O(n+m)$	$O(1)$
`length`	$O(1)$	$O(1)$
`filter`	$\sum_{e \in S} \mathcal{W}(p(e))$	$O(\log n) + \max_{e \in S} \mathcal{S}(p(e))$

`TreeSequence` has identical bounds except `append` has $O(\log(n+m))$ work and span, and the span of `map` is $\log n$ plus the max term.

Solution 4.0 ArraySequence: $W(n) \in O(n \log n)$, $S(n) \in O(\log n)$
TreeSequence: $W(n) \in O(n)$, $S(n) \in O(\log^2 n)$