15–210: Parallel and Sequential Data Structures and Algorithms

Exam I (Solutions)

24 February 2017

- **Verify** There are 15 pages in this examination, comprising 6 questions worth a total of 100 points. The last 2 pages are an appendix with costs of sequence, set and table operations.

- **Write** the name (e.g., “J. Snow”) of the persons sitting to your left and to your right below your andrew id (in left to right order).

- **Time**: You have 80 minutes to complete this examination.

- **Goes without saying**: Please answer all questions in the space provided with the question. Clearly indicate your answers.

- **Beware**: You may refer to your one double-sided $8\frac{1}{2} \times 11$ in sheet of paper with notes, but to no other person or source, during the examination.

- **Primitives**: In your algorithms you can use any of the primitives that we have covered in the lecture, unless otherwise states. A reasonably comprehensive list is provided at the end and sometimes in the body of the question itself.

- **Code**: When writing your algorithms, you can use ML syntax or the pseudocode notation used in the notes or in class. In the questions, we use pseudocode.

- **Good luck!**

---

<table>
<thead>
<tr>
<th>Sections</th>
</tr>
</thead>
<tbody>
<tr>
<td>A 9:30am - 10:20am</td>
</tr>
<tr>
<td>B 10:30am - 11:20am</td>
</tr>
<tr>
<td>C 12:30pm - 1:20pm</td>
</tr>
<tr>
<td>D 12:30pm - 1:20pm</td>
</tr>
<tr>
<td>E 1:30pm - 2:20pm</td>
</tr>
<tr>
<td>F 1:30pm - 4:20pm</td>
</tr>
<tr>
<td>G 3:30pm - 5:20pm</td>
</tr>
<tr>
<td>Question</td>
</tr>
<tr>
<td>----------------------------</td>
</tr>
<tr>
<td>True/False</td>
</tr>
<tr>
<td>Recurrences</td>
</tr>
<tr>
<td>Short Answer Problems</td>
</tr>
<tr>
<td>Exclamations</td>
</tr>
<tr>
<td>Iterate and Reduce</td>
</tr>
<tr>
<td>Random Other Questions</td>
</tr>
<tr>
<td><strong>Total:</strong></td>
</tr>
</tbody>
</table>
Question 1: True/False  (16 points)
Please Circle your choice.

(a) (2 points) TRUE or FALSE: If you can reduce comparison-based sorting to your problem in $O(n)$ work, then you can solve your problem in $\Theta(n \log n)$ work.

Solution: FALSE

(b) (2 points) TRUE or FALSE: Since we can reduce the shortest superstring (SS) problem to the Traveling Salesperson Problem (TSP) using polynomial work, the SS problem can be solved in polynomial work.

Solution: FALSE

(c) (2 points) TRUE or FALSE: Parallelism is proportional to Work divided by Span.

Solution: TRUE

(d) (2 points) TRUE or FALSE: The union bound implies that if $n$ people each have probability $1/n^2$ of having a twin sister, then the probability that any has a twin sister is exactly $1/n$.

Solution: FALSE

(e) (2 points) TRUE or FALSE: The expressions $(\text{Seq.scan } f \ I \ A)$ always takes $O(\log |A|)$ span.

Solution: FALSE

(f) (2 points) TRUE or FALSE: $f(n) = \Theta(g(n))$ implies $f(n) = O(g(n))$.

Solution: TRUE

(g) (2 points) TRUE or FALSE: $f(n) = O(g(n))$ implies $g(n) = \Omega(f(n))$.

Solution: TRUE

(h) (2 points) TRUE or FALSE: For independent random variables $X$ and $Y$:


Solution: TRUE
Question 2: Recurrences  (12 points)
Give a closed-form solution in terms of $\Theta$ for the following recurrences. Also, state whether the recurrence is root dominated, leaf dominated, or approximately balanced in the recurrence tree (as defined in class). You do not have to show your work, but it might help you get partial credit.

(a) (2 points) $f(n) = f(n/2) + \log n$

Solution: $\Theta(\log^2 n)$, balanced

(b) (2 points) $f(n) = 2f(n/2) + n^{1.25}$

Solution: $\Theta(n^{1.25})$, root dominated

(c) (2 points) $f(n) = 2f(n/2) + n \log n$

Solution: $\Theta(n \log^2 n)$, balanced

(d) (2 points) $f(n) = 3f(n/2) + n^{1/2}$

Solution: $\Theta(n \log^3 n)$, leaf dominated

(e) (2 points) $f(n) = f(\sqrt{n}) + \log n$

Solution: $\Theta(\log n)$, root dominated

(f) (2 points) $f(n) = \sqrt{n}f(\sqrt{n}) + n$

Solution: $\Theta(n \log \log n)$, balanced
Question 3: Short Answer Problems  (21 points)

(a) (7 points) Scan and reduce require associative functions and an identity. For each of the following binary functions circle it if is associative, and if it is associative, specify what the identity is.

- \(a \oplus b = a + b\)
- \(a \oplus b = a/b\)
- \(a \oplus b = a \times b\)
- \(a \oplus b = a \cup b\)
- \((a, b) \oplus (c, d) = (\max(a, c), \min(b, d))\)
- \(a \oplus b = a + \max(a, b)\)
- \(a \oplus b = \text{case } b \ldots : \text{yes, NONE} \Rightarrow a \mid _- \Rightarrow b\)

Solution:

- \(+\): yes, 0
- \(\times\): yes, 1
- \(/\): no
- \(\cup\): yes, \(\emptyset\)
- \((\max(a, c), \min(b, d))\): yes, \(-\infty, \infty\)
- \(a + \max(a, b)\): no
- \(\text{case } b \ldots\): yes, NONE

(b) (6 points) Write a function to compute the factorial of all numbers from 1 to \(n\). The function should return a sequence of length \(n\), where the index \(i\) (starting at zero) stores the factorial of the number \(i + 1\). Your solution should take \(O(n)\) work and \(O(\log n)\) span. You may assume multiplication is a unit cost operation. You may not simply use the ! operator in your solution. (Our solution is one line.)

```plaintext
factorial (n : int) : int seq =

Solution:

factorial(n) = scanIncl \times 1 \{i : 0 < i \leq n\}
```
(c) (8 points) What is the asymptotic work and span of the following code for finding primes:

```plaintext
primes(n) =
    let sieves = \{(i \times j, \text{false}): 2 \leq i \leq \lceil \sqrt{n} \rceil, 2 \leq j < \lceil n/i \rceil\}
    R = inject(sieves, \{\text{true}: 0 \leq i \leq n\})
    in
    \{i: 2 \leq i \leq n | R[i]\}
end
```

Solution:

\[
W(n) = O\left(\sum_{i=2}^{\sqrt{n}} \frac{n}{i}\right) = O(nH(\sqrt{n})) = O(n \log n)
\]

\[
S(n) = O(\log n)
\]
**Question 4: Exclamations**  (14 points)

Doctor Tooten is tired of seeing long lists of exclamation marks in a row!!!!!! She therefore writes a linear work \((O(n))\) logarithmic span \((O(\log n))\) algorithm based on reduce that takes a sequence of \(n\) characters and reports back the length of the longest contiguous sequence of exclamation marks.

(a) (10 points) Please fill in the code for her algorithm below.

**Solution:**

```haskell
exclamation(S : char seq) : int = let
 (* All(n) indicates all n characters are !
   Some(s, e, m) indicate the first s characters are !, the last e characters are !, and the longest contiguous sequence of ! not reaching the front or end has length m *)
 datatype ex = All of int
   | Some of (int × int × int)

singleton(v) = if (x = '!') then All(1)
               else Some(0,0,0)

I = All(0)
combine(a1a2) =
 case (a1, a2) of
   (All(n1),All(n2)) ⇒ All(n1 + n2)
   | (All(n1),Some(s2,e2,m2)) ⇒ Some(n1 + s2,e2,m2)
   | (Some(s1,e1,m1), All(n2)) ⇒ Some(s1,e1 + n2,m1)
   | (Some(s1,e1,m1), Some(s2,e2,m2)) ⇒ Some(s1,e2,max(m1,m2,e1 + s2))

in case (reduce combine I (singleton(x) : x ∈ S)) of
   All(n) ⇒ n
   | Some(s,e,m) ⇒ max(s,max(e,m))
end
```

To get the following right, you must get (a) right.

(b) (2 points) **TRUE** or **FALSE:** Your function `combine` is associative.

**Solution:** TRUE

(c) (2 points) **TRUE** or **FALSE:** If you replace `reduce` with `iterate` it will return the same result and in the same asymptotic work.

**Solution:** TRUE
Question 5: Iterate and Reduce  (12 points)

Let's say we had an implementation of sequences such that \( \text{Seq.append}(A, B) \) takes \( \Theta(\sqrt{|A| + |B|}) \) work and \( \Theta(1) \) span. All other costs are the same as for array sequences. Please determine \( \Theta \) bounds for the work and span of the following functions in terms of \( n = |S| \).

(a) (6 points)

\[
\text{Seq.iterate Seq.append (Seq.empty()) (Seq.map Seq.singleton S)}
\]

Solution:

\[
W(n) = W(n - 1) + \Theta(\sqrt{n}) \in \Theta(n^{3/2}) \\
S(n) = S(n - 1) + \Theta(1) \in \Theta(n)
\]

(b) (6 points)

\[
\text{Seq.reduce Seq.append (Seq.empty()) (Seq.map Seq.singleton S)}
\]

Solution:

\[
W(n) = 2W(n/2) + \Theta(\sqrt{n}) \in \Theta(n) \\
S(n) = S(n/2) + \Theta(1) \in \Theta(\log n)
\]
Question 6: Random Other Questions  (25 points)

(a) (5 points) Consider a random sequence of \( n \) bits. Briefly argue that there is at most a \( 1/n \) chance that this sequence will have a string of \( 2\log_2 n \) or more consecutive zeroes. You should include no more than two sentences.

Solution: For each \( i \), the probability there exists such a sequence beginning at location \( i \) is at most \( 1/n^2 \). Then use the union bound.

For (b) and (c): You independently throw two unbiased 3-sided die (sides 1, 2, 3).

(b) (2 points) What is the expected sum?

Solution: 4 by linearity of expectations

(c) (2 points) What is the expected maximum value?

Solution: \((5 \times 3 + 3 \times 2 + 1)/9 = 22/9\)

(d) (5 points) You have a stash of \( k \) candy bars. Each day, when you get home, you roll a die. If it comes up 6, then you eat a candy bar, otherwise you don’t. In expectation, how many days will it take for you to eat all \( k \) candy bars?

Solution: \(6k\). We can solve this by defining \( X_1 \) to be the number of days until you’ve eaten the first candy bar, \( X_2 \) to be the number of days after that until you’ve eaten the next candy bar, etc. Let \( X \) be the total number of days to eat all \( k \). Then \( X = X_1 + \ldots + X_k \), and \( E[X_i] = 6 \).

One more page.
(e) (5 points) \( n \) people are standing in line to see a rock concert when all of a sudden the lead singer shows up at the ticket window and poses for a photo. You can only take a clear photo of her if you are taller than everyone in line in front of you. Furthermore everyone in line has a different height and they are standing in a random order. Write an expression for the exact expected number of people who can take a clear photo, and give what it approximately equals.

**Solution:** Let \( X_i \) be the random indicator variable that the \( i^{th} \) person in line (starting at 1) can see the star, and \( X \) be the random variable that gives the total number of people who can see the star. Then we have that \( E[X_i] = 1/i \).

\[
E[X] = \sum_{i=1}^{n} E[X_i] = \sum_{i=1}^{n} 1/i \approx \ln(n)
\]

(f) (6 points) In a sequence \( S \), we say that positions \( i \) and \( j \) have an inversion if \( i < j \) but \( S[i] > S[j] \). If elements in \( S \) are all distinct and are in a random order, what is the expected total number of inversions? For instance, if \( S \) was in reverse-sorted order, the total number of inversions would be \( \binom{n}{2} \).

**Solution:** There are \( \binom{n}{2} \) pairs and each has probability 1/2 of being an inversion, so the total in expectation is \( \frac{n(n-1)}{4} \).
Scratch Work:
Scratch Work:
Appendix: Library Functions

signature SEQUENCE =
  sig
    type 'a t
    type 'a seq = 'a t
    type 'a ord = 'a * 'a -> order
  datatype 'a listview = NIL | CONS of 'a * 'a seq
  datatype 'a treeview = EMPTY | ONE of 'a | PAIR of 'a seq * 'a seq

exception Range
exception Size

val nth : 'a seq -> int -> 'a
val length : 'a seq -> int
val toList : 'a seq -> 'a list
val toString : ('a -> string) -> 'a seq -> string
val equal : ('a * 'a -> bool) -> 'a seq * 'a seq -> bool

val empty : unit -> 'a seq
val singleton : 'a -> 'a seq
val tabulate : (int -> 'a) -> int -> 'a seq
val fromList : 'a list -> 'a seq

val rev : 'a seq -> 'a seq
val append : 'a seq * 'a seq -> 'a seq
val flatten : 'a seq seq -> 'a seq

val filter : ('a -> bool) -> 'a seq -> 'a seq
val map : ('a -> 'b) -> 'a seq -> 'b seq
val zip : 'a seq * 'b seq -> ('a * 'b) seq
val zipWith : ('a * 'b -> 'c) -> 'a seq * 'b seq -> 'c seq

val enum : 'a seq -> (int * 'a) seq
val filterIdx : (int * 'a -> bool) -> 'a seq -> 'a seq
val mapIdx : (int * 'a -> 'b) -> 'a seq -> 'b seq
val update : 'a seq * (int * 'a) -> 'a seq
val inject : 'a seq * (int * 'a) seq -> 'a seq

val subseq : 'a seq -> int * int -> 'a seq
val take : 'a seq -> int -> 'a seq
val drop : 'a seq -> int -> 'a seq
val splitHead : 'a seq -> 'a listview
val splitMid : 'a seq -> 'a treeview
val iterate : ('b * 'a -> 'b) -> 'b -> 'a seq -> 'b
val iteratePrefixes : ('b * 'a -> 'b) -> 'b -> 'a seq -> 'b seq * 'b
val iteratePrefixesIncl : ('b * 'a -> 'b) -> 'b -> 'a seq -> 'b seq
val reduce : ('a * 'a -> 'a) -> 'a -> 'a seq -> 'a
val scan : ('a * 'a -> 'a) -> 'a -> 'a seq -> 'a seq * 'a
val scanIncl : ('a * 'a -> 'a) -> 'a -> 'a seq -> 'a seq

val sort : 'a ord -> 'a seq -> 'a seq
val merge : 'a ord -> 'a seq * 'a seq -> 'a seq
val collect : 'a ord -> ('a * 'b) seq -> ('a * 'b seq) seq
val collate : 'a ord -> 'a seq ord
val argmax : 'a ord -> 'a seq -> int

val $ : 'a -> 'a seq
val % : 'a list -> 'a seq
end
<table>
<thead>
<tr>
<th><strong>ArraySequence</strong></th>
<th><strong>Work</strong></th>
<th><strong>Span</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>empty ()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>singleton a</td>
<td></td>
<td></td>
</tr>
<tr>
<td>length s</td>
<td></td>
<td></td>
</tr>
<tr>
<td>nth s i</td>
<td></td>
<td></td>
</tr>
<tr>
<td>subseq s (i, len)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **tabulate f n**
  - if \( f(i) \) has \( W_i \) work and \( S_i \) span

- **map f s**
  - if \( f(s[i]) \) has \( W_i \) work and \( S_i \) span, and \( |s| = n \)

- **zipWith f (s, t)**
  - if \( f(s[i], t[i]) \) has \( W_i \) work and \( S_i \) span, and \( \min(|s|, |t|) = n \)

- **reduce f b s**
  - if \( f \) does constant work and \( |s| = n \)

- **scan f b s**
  - if \( f \) does constant work and \( |s| = n \)

- **filter p s**
  - if \( p \) does constant work and \( |s| = n \)

- **flatten s**
  - \( O\left(\sum_{i=0}^{n-1} (1 + |s[i]|)\right) \)
  - \( O(\lg |s|) \)

- **sort cmp s**
  - if \( cmp \) does constant work and \( |s| = n \)

- **merge cmp (s, t)**
  - if \( cmp \) does constant work, \( |s| = n \), and \( |t| = m \)

- **append (s, t)**
  - if \( |s| = n \), and \( |t| = m \)

- **inject (p, a)**
  - if \( |p| = n \), and \( |a| = m \)