15–210: Parallel and Sequential Data Structures and Algorithms

Exam II

7 April 2017

- **Verify** There are 17 pages in this examination, comprising 7 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 8½ × 11in 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><strong>A</strong> 9:30am - 10:20am</td>
</tr>
<tr>
<td><strong>B</strong> 10:30am - 11:20am</td>
</tr>
<tr>
<td><strong>C</strong> 12:30pm - 1:20pm</td>
</tr>
<tr>
<td><strong>D</strong> 12:30pm - 1:20pm</td>
</tr>
<tr>
<td><strong>E</strong> 1:30pm - 2:20pm</td>
</tr>
<tr>
<td><strong>F</strong> 1:30pm - 4:20pm</td>
</tr>
<tr>
<td><strong>G</strong> 3:30pm - 5:20pm</td>
</tr>
<tr>
<td>Question</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>BSTs and Treaps</td>
</tr>
<tr>
<td>A Sparse Problem</td>
</tr>
<tr>
<td>(BFS) Multisource Shortest Paths</td>
</tr>
<tr>
<td>DFS</td>
</tr>
<tr>
<td>Shortest Paths</td>
</tr>
<tr>
<td>Contraction Action, what’s Your Reaction?</td>
</tr>
<tr>
<td>Parallel Bridges</td>
</tr>
<tr>
<td>Total:</td>
</tr>
</tbody>
</table>
Question 1: BSTs and Treaps  (22 points)

(a) (6 points) Suppose we have the keys 1, 2, 3, 4, 5, 6 with priorities $p$ shown below:

<table>
<thead>
<tr>
<th>key</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>p(key)</td>
<td>2</td>
<td>5</td>
<td>4</td>
<td>1</td>
<td>6</td>
<td>3</td>
</tr>
</tbody>
</table>

Draw the max-treap (requires that priority at a node is greater than the priority of its two children) associated with these keys and priorities.

(b) (6 points) What is the probability that a treap with $n$ nodes has depth $n$? You can assume that the priorities are picked uniformly at random, and that there are no duplicates.
(c) (10 points) In lecture you saw an implementation of the union operation on sets represented as binary search trees. This problem involves the set intersect operation.

To put this code in context, we’re using a binary search tree of the following type to represent these sets.

```plaintext
datatype BST = Leaf | Node of (BST * BST * key)
```

Also, recall the following type signatures of `split` and `joinMid`:

- `split(T,k) : BST × key → BST × bool × BST`
- `joinMid(L,m,R) : BST × key × BST → BST`

Fill in the blanks in the SML code below that implements `intersect`.

```plaintext
intersect (T1,T2) = 
case T1 of
  Leaf ⇒ _______________________________
  | Node (L1,R1,k1) ⇒
  let (L2,x2,R2) = split(T2,k1)
  (L,R) = par( ____________________________,
              ____________________________ )
  in
  case x2 of
    NONE ⇒ ____________________________
    | SOME(a) ⇒ __________________________
  end
```

Question 2: A Sparse Problem  (10 points)

A vector is basically a sequence whose values are real numbers, for example:

\( \langle 2.1, 5.3, 1.2, -2.55, 11.0 \rangle \)

Recall that the dot-product of two vectors is an elementwise multiplication followed by a sum. With sequences this can be implemented as:

\[
\text{dotProduct}(A, B) = \text{reduce } + 0 \langle A[i] \times B[i] : 0 \leq i < |A| \rangle
\]

A sparse vector is one in which the value is 0 at most indices. Sparse vectors are common in practice and can be represented more efficiently as a mapping from the non-zero indices to their values. For example the vector

\( \langle 0.0, 2.1, 0.0, 1.2, 0.0, 0.0, 0.0, 11.0, 0.0, 0.0 \rangle \)

can be represented as

\{1 \mapsto 2.1, 3 \mapsto 1.2, 7 \mapsto 11.0\}

Your job is to implement \( \text{dotProduct}(A,B) \) based on this sparse representation. You can use any of the functions on tables, but the implementation must take \( O(m \log(1 + n/m)) \) work and \( O(\log n) \) span, where \( m = \min(|A|, |B|) \) and \( n = \max(|A|, |B|) \), and \( |A| \) and \( |B| \) indicates the number of non-zero elements (i.e. the size of the mapping). Please note that the signature for the Table interface is at the end of the exam. Our solution is one line.

\[
\text{dotProduct}(A, B) =
\]
Question 3: (BFS) Multisource Shortest Paths (12 points)
You are starting a company that supplies maps indicating the proximity of every intersection in Pittsburgh to the nearest coffee shop. This would be useful if you’re buying coffee in a hurry. You should assume that all road segments are the same length, but some are one-way streets. This problem can be defined as follows:

Definition (The multisource unweighted shortest pathlength (MUSP) problem): Given a directed graph $G = (V, E)$ and a set of sources $U \subseteq V$ determine for every vertex in $v \in V$ its shortest path length from any vertex in $U$, denoted as $\delta(U, v)$.

Please answer the following:

(a) (6 points) Describe an algorithm that solves this problem in $O(m \log n)$ work and $O(d \log^2 n)$ span, where $n = |V|$, $m = |E|$, and $d = \max_{v \in V} \delta(U, v)$. A couple sentences should suffice—you may justify the cost by relating it to the cost of some algorithm we covered in class.

(b) (6 points) Suppose now you would like to indicate the shortest path length from every intersection to the nearest coffee shop. That is to say, you want for every vertex in $V$ its shortest path length to any vertex in $U$. How would you modify your algorithm above so that it has the same cost bounds, except that $d = \max_{v \in V} \delta(v, U)$? Give a short justification.
Question 4: DFS  (18 points)

(a) A DFS order is a sequence of vertices of a graph in the order in which they are first visited by depth-first search (DFS). Consider the following graph:

One DFS order of this graph is \( \langle A, B, C, X, H, M \rangle \).

i. (6 points) List ALL other possible DFS orders for this graph, given source vertex \( A \).

ii. (6 points) For the order we gave you, list the forward, the back, and the cross edges.

(b) (6 points) Suppose you perform DFS on an undirected graph \( G \) where \( G \) is a clique (there is an edge between every two vertices). In the resulting DFS tree, how many nodes will have degree greater than 2?
Question 5: Shortest Paths  (13 points)

(a) (2 points) **TRUE** or **FALSE**: If you add a positive constant to the weight on every edge in a weighted graph, it does not change the shortest path tree.

(b) (2 points) **TRUE** or **FALSE**: If a graph has negative edge weights, Dijkstra’s algorithm for shortest path may loop forever.

(c) (2 points) **TRUE** or **FALSE**: We can use the Bellman-Ford algorithm to detect negative cycles in a graph.

(d) (7 points) Given the following graph:

Show the order that the vertices are visited in by Dijkstra’s algorithm, the distance computed from the source A, and the contents of the priority queue after the vertex is visited (order in priority queue does not matter).

<table>
<thead>
<tr>
<th>Vertex</th>
<th>Visit order</th>
<th>Distance</th>
<th>Priority Queue</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>0</td>
<td>(2,C),(1,B)</td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Question 6: Contraction Action, what’s Your Reaction? (15 points)

For each graph property below, state whether or not edge contraction preserves this property. That is, if a graph has the property before performing an edge contraction, will the graph necessarily have the property after the contraction? You can assume that duplicate edges are removed.

Answer Yes or No. Also, if your answer is No, then give a counterexample. That is, give a graph $G$ and highlight an edge $e$ in $G$ such that $G$ has the given property initially but not after contracting $e$.

(a) (3 points) All nodes in the graph have degree $\leq 2$.

(b) (3 points) All nodes in the graph have degree $\leq 3$.

(c) (3 points) The graph is connected.

(d) (3 points) The graph is a tree (single nodes count as trees)

(e) (3 points) The graph has more edges than vertices.
Question 7: Parallel Bridges  (10 points)
In BridgeLab you found bridges of a graph using DFS, sequentially. Recall that a bridge in an undirected connected graph is any edge that does not belong to any cycles. It is also possible to do it in parallel in linear work using the following approach—each step can be parallelized.

1. Find a spanning tree $T$ of $G$.
2. Select one vertex $r$ to be the root of the tree $T$.
3. Give a preorder numbering ($\text{preNum}$) to the nodes in the $T$ starting at the root $r$. Recall that a pre-order numbering first numbers the root, then recursively numbers each of the subtrees. Here is an example:

   ![Diagram of a graph with preorder numbering](image)

   - Edges of $T$ (directed from the root $r$)
   - Other edges of $G$

4. For every vertex, determine the minimum and maximum pre-order number of its neighbors in $G$, not including its parent in $T$. In the example, 3's min is 2 and max is 4.
5. For each vertex, determine
   - $\text{minSubNgh}$: the minimum neighbor in its subtree (the minimum in the subtree of the minimum values calculated in the previous step),
   - $\text{maxSubNgh}$: the maximum neighbor in its subtree (the maximum in the subtree of the maximum values calculated in the previous step),
   - $\text{maxPreNum}$: the maximum pre-order number of its subtree

For the example graph:
- $\text{minSubNgh}[3] = 2$, $\text{maxSubNgh}[3] = 7$, $\text{maxPreNum}[3] = 4$,

(a) (4 points) Argue in one or two sentences that if an edge $(u, v)$ is a bridge then it must be in the spanning tree found in step 1.

(b) (6 points) Given an edge $(u, v) \in T$, with $u$ the parent, write an equation for determining if it is a bridge based on the values of $\text{preNum}[u]$, $\text{preNum}[v]$, $\text{minSubNgh}[u]$, $\text{minSubNgh}[v]$, $\text{maxSubNgh}[u]$, $\text{maxSubNgh}[v]$, $\text{maxPreNum}[u]$, and $\text{maxPreNum}[v]$. 


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 : ('b * 'a -> 'b) -> 'b -> 'a seq -> 'b seq
val scanIncl : ('b * 'a -> 'b) -> 'b -> 'a seq -> 'b 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
<table>
<thead>
<tr>
<th>Function</th>
<th>Work</th>
<th>Span</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>collate</code> : <code>a ord -&gt; 'a seq ord</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>argmax</code> : <code>a ord -&gt; 'a seq -&gt; int</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>$</code> : <code>a -&gt; 'a seq</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>%</code> : <code>a list -&gt; 'a seq</code></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### ArraySequence

<table>
<thead>
<tr>
<th>Function</th>
<th>Work</th>
<th>Span</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>empty ()</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>singleton a</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>length s</code></td>
<td>$O(1)$</td>
<td>$O(1)$</td>
</tr>
<tr>
<td><code>nth s i</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>subseq s (i, len)</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>tabulate f n</code></td>
<td>$O\left(\sum_{i=0}^{n-1} W_i\right)$</td>
<td>$O\left(\max_{i=0}^{n-1} S_i\right)$</td>
</tr>
<tr>
<td><code>map f s</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>zipWith f (s, t)</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>reduce f b s</code></td>
<td>$O(n)$</td>
<td>$O(\lg n)$</td>
</tr>
<tr>
<td><code>scan f b s</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>filter p s</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>flatten s</code></td>
<td>$O\left(\sum_{i=0}^{n-1} (1 +</td>
<td>s[i]</td>
</tr>
<tr>
<td><code>sort cmp s</code></td>
<td>$O(n \lg n)$</td>
<td>$O(\lg^2 n)$</td>
</tr>
<tr>
<td><code>merge cmp (s, t)</code></td>
<td>$O(m + n)$</td>
<td>$O(\lg(m + n))$</td>
</tr>
<tr>
<td><code>append (s, t)</code></td>
<td>$O(m + n)$</td>
<td>$O(1)$</td>
</tr>
</tbody>
</table>

15–210 Exam II 14 of 17 7 April 2017
signature TABLE =
  sig
    structure Key : EQKEY
    structure Seq : SEQUENCE

    type 'a t
    type 'a table = 'a t

    structure Set : SET where Key = Key and Seq = Seq

    val size : 'a table -> int
    val domain : 'a table -> Set.t
    val range : 'a table -> 'a Seq.t
    val toString : ('a -> string) -> 'a table -> string
    val toSeq : 'a table -> (Key.t * 'a) Seq.t

    val find : 'a table -> Key.t -> 'a option
    val insert : 'a table * (Key.t * 'a) -> 'a table
    val insertWith : ('a * 'a -> 'a) -> 'a table * (Key.t * 'a) -> 'a table
    val delete : 'a table * Key.t -> 'a table

    val empty : unit -> 'a table
    val singleton : Key.t * 'a -> 'a table
    val tabulate : (Key.t -> 'a) -> Set.t -> 'a table
    val collect : (Key.t * 'a) Seq.t -> 'a Seq.t table
    val fromSeq : (Key.t * 'a) Seq.t -> 'a table

    val map : ('a -> 'b) -> 'a table -> 'b table
    val mapKey : (Key.t * 'a -> 'b) -> 'a table -> 'b table
    val filter : ('a -> bool) -> 'a table -> 'a table
    val filterKey : (Key.t * 'a -> bool) -> 'a table -> 'a table

    val reduce : ('a * 'a -> 'a) -> 'a -> 'a table -> 'a
    val iterate : ('b * 'a -> 'b) -> 'b -> 'a table -> 'b
    val iteratePrefixes : ('b * 'a -> 'b) -> 'b -> 'a table -> ('b table * 'b)

    val union : ('a * 'a -> 'a) -> ('a table * 'a table) -> 'a table
    val intersection : ('a * 'b -> 'c) -> ('a table * 'b table) -> 'c table
    val difference : 'a table * 'b table -> 'a table

    val restrict : 'a table * Set.t -> 'a table
    val subtract : 'a table * Set.t -> 'a table

    val $ : (Key.t * 'a) -> 'a table
  end
signature SET =
sig
  structure Key : EQKEY
  structure Seq : SEQUENCE

  type t
  type set = t

  val size : set -> int
  val toString : set -> string
  val toSeq : set -> Key.t Seq.t

  val empty : unit -> set
  val singleton : Key.t -> set
  val fromSeq : Key.t Seq.t -> set

  val find : set -> Key.t -> bool
  val insert : set * Key.t -> set
  val delete : set * Key.t -> set

  val filter : (Key.t -> bool) -> set -> set

  val reduceKey : (Key.t * Key.t -> Key.t) -> Key.t -> set -> Key.t
  val iterateKey : ('a * Key.t -> 'a) -> 'a -> set -> 'a

  val union : set * set -> set
  val intersection : set * set -> set
  val difference : set * set -> set

  val $ : Key.t -> set
end
<table>
<thead>
<tr>
<th><strong>MkTreapTable</strong></th>
<th><strong>Work</strong></th>
<th><strong>Span</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>size</strong> $T$</td>
<td>$O(1)$</td>
<td>$O(1)$</td>
</tr>
<tr>
<td><strong>filter</strong> $f$ $T$</td>
<td>$\sum_{(k\mapsto v)\in T} W(f(v)) \lg</td>
<td>T</td>
</tr>
<tr>
<td><strong>map</strong> $f$ $T$</td>
<td>$\sum_{k\in X} W(f(k)) \max_{k\in X} S(f(k))$</td>
<td></td>
</tr>
<tr>
<td><strong>reduce</strong> $f$ $b$ $T$ if $f$ does constant work</td>
<td>$O(</td>
<td>T</td>
</tr>
<tr>
<td><strong>insertWith</strong> $f$ $T$ $(k,v)$ if $f$ does constant work</td>
<td>$O(\lg</td>
<td>T</td>
</tr>
<tr>
<td><strong>find</strong> $T$ $k$</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>delete</strong> $(T,k)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>domain</strong> $T$</td>
<td>$O(</td>
<td>T</td>
</tr>
<tr>
<td><strong>range</strong> $T$</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>toSeq</strong> $T$</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>collect</strong> $S$</td>
<td>$O(</td>
<td>S</td>
</tr>
<tr>
<td><strong>fromSeq</strong> $S$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For each argument pair $(A, B)$ below, let $n = \max(|A|, |B|)$ and $m = \min(|A|, |B|)$.

<table>
<thead>
<tr>
<th><strong>MkTreapTable</strong></th>
<th><strong>Work</strong></th>
<th><strong>Span</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>union</strong> $f$ $(X,Y)$</td>
<td>$O(m \lg \left(\frac{n+m}{m}\right))$</td>
<td>$O(\lg(n+m))$</td>
</tr>
<tr>
<td><strong>intersection</strong> $f$ $(X,Y)$</td>
<td>$O(m \lg(n+m))$</td>
<td></td>
</tr>
<tr>
<td><strong>difference</strong> $(X,Y)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>restrict</strong> $(T,X)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>subtract</strong> $(T,X)$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>