Recitation 1

Priority Queues and Hashing

1.1 Announcements

- PASLLab is due this Friday, December 9.
- The final exam is Tuesday, December 13.
- A review session for the final is on Sunday, December 11, 4-6pm, in Rashid.
- A practice final and its solutions are up on the website and on Piazza.
1.2 Leftist Heaps

Task 1.1. Identify the defining properties of a leftist heap.

Task 1.2. What is an upper bound on the rank of the root of a leftist heap?
1.2. LEFTIST HEAPS

1.2.1 Building A Leftist Heap

Consider the following pseudo-SML code implementing leftist heaps.

Data Structure 1.3. Leftist Heap

```sml
datatype PQ = Leaf | Node of int × key × PQ × PQ

fun rank Q =
  case Q of
  Leaf ⇒ 0
  | Node (r,_,_,_) ⇒ r

fun makeLeftistNode (k,A,B) =
  if rank A < rank B
  then Node (1 + rank A, k, B, A)
  else Node (1 + rank B, k, A, B)

fun meld (A,B) =
  case (A,B) of
  (_, Leaf) ⇒ A
  | (Leaf, _) ⇒ B
  | (Node (_,ka,La,Ra), Node (_,kb,Lb,Rb)) ⇒
    if ka < kb
    then makeLeftistNode (ka, La, meld (Ra,B))
    else makeLeftistNode (kb, Lb, meld (A,Rb))

fun singleton k = Node (1,k,Leaf,Leaf)

fun insert (Q,k) = meld (Q, singleton k)

fun fromSeq S = Seq.reduce meld Leaf (Seq.map singleton S)

fun deleteMin Q =
  case Q of
  Leaf ⇒ (NONE, Q)
  | Node (_,k,L,R) ⇒ (SOME k, meld (L,R))
```

Task 1.4. Diagram the process of executing the code

```sml```
```
fromSeq ⟨3,5,2,1,4,6,7,8⟩
```

Task 1.5. What are the work and span of (fromSeq S) in terms of |S| = n?
1.3 Removing Duplicates

Removing duplicates is a crucial substep of many interesting algorithms. For example, in BFS, consider the step where we construct a new frontier. One viable method would be to generate the sequence of all out-neighbors, and then remove duplicates:

\[ F' = \text{removeDuplicates} \{ v : u \in F, v \in N_G^+(u) \} \]

So, how fast is it to remove duplicates? Can we do it in parallel?

1.3.1 Sequential

Before we think about parallelism, we should acquaint ourselves with a good sequential algorithm solving the same problem. This way, we know what to shoot for in terms of work bounds, since we want our parallel algorithm to be asymptotically work-efficient.

**Task 1.6.** Describe a sequential algorithm which performs expected \( O(n) \) work to remove duplicates from a sequence of length \( n \). Also argue that \( \Omega(n) \) work is necessary in order to solve this problem, and conclude that your algorithm is asymptotically optimal.

*Hint: try hashing elements one at a time.*

1.3.2 Parallel

**Task 1.7.** Implement a function

\[
\text{val removeDuplicates} : (\alpha \times \text{int} \rightarrow \text{int}) \rightarrow \alpha \text{ Seq.t} \rightarrow \alpha \text{ Seq.t}
\]

\[ \text{where (removeDuplicates } h \ S) \text{ returns a sequence of all unique elements of } S, \]

\[ \text{given that } h(e, m) \text{ hashes the element } e \text{ to a uniform random integer in the range } [0, m) \]

(\( \text{thus the probability of collision for any two distinct elements is } 1/m \)).

*Hint: as a first attempt, try simultaneously hashing as many elements as possible all at the same time. What do you do when elements collide?*
### Exercise 1.8.

**Task 1.9.** Design a data structure which supports the following operations:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Work</th>
<th>Span</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>fromSeq S</code></td>
<td>$O(</td>
<td>S</td>
<td>)$</td>
</tr>
<tr>
<td><code>median M</code></td>
<td>$O(1)$</td>
<td>$O(1)$</td>
<td>Returns the median of all keys stored in $M$</td>
</tr>
<tr>
<td><code>insert (M,k)</code></td>
<td>$O(</td>
<td>M</td>
<td>)$</td>
</tr>
</tbody>
</table>

For simplicity, you may assume that all elements inserted into such a structure are distinct.

### Exercise 1.10. Prove a lower bound of $\Omega(\log n)$ for `deleteMin` in comparison-based meldable priority queues. That is, prove that any meldable priority queue implementation which has a logarithmic `meld` cannot support `deleteMin` in faster than logarithmic time.