Recitation 14

Priority Queues

14.1 Announcements

- *PASLLab* has been released, and is due **next Friday** (April 29 – or is that next *next* Friday?). PASLLab worth 175 points.

14.2 Leftist Heaps

Task 14.1. *Identify the defining properties of a leftist heap.*

Task 14.2. *What is an upper bound on the rank of the root of a leftist heap?*

14.2.1 Building A Leftist Heap

Consider the following pseudo-SML code implementing leftist heaps.

Data Structure 14.3. *Leftist Heap*

```
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 (_,$k_a$,$L_a$,$R_a$), Node (_,$k_b$,$L_b$,$R_b$)) ⇒
      if $k_a < k_b$
      then makeLeftistNode ($k_a$, $L_a$, meld ($R_a$, B))
      else makeLeftistNode ($k_b$, $L_b$, meld (A, $R_b$))

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 14.4. *Diagram the process of executing the code*

fromSeq $\langle 3, 5, 2, 1, 4, 6, 7, 8 \rangle$

Task 14.5. *What are the work and span of* (fromSeq S) *in terms of* $|S| = n$*?*

14.2.2 Dynamic Median

Task 14.6. *Design a data structure which supports the following operations:*

	Work	*Span*	*Description*
`fromSeq` S	$O(\lvert S\rvert)$	$O(\log^2 \lvert S\rvert)$	*Constructs a dynamic median data structure from the collection of keys in* S
`median` M	$O(1)$	$O(1)$	*Returns the median of all keys stored in* M
`insert` (M, k)	$O(\log \lvert M\rvert)$	$O(\log \lvert M\rvert)$	*Inserts* k *into* M

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

14.3 Additional Exercises

Exercise 14.7. *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.*