Recitation 15

Priority Queues

15.1 Announcements

- *DPLab* has been released, and is due **this Friday**.

15.2 Leftist Heaps

Task 15.1. *Identify the defining properties of a leftist heap.*

Task 15.2. *What is an upper bound on the rank of the root of a leftist heap?*

15.2.1 Building A Leftist Heap

Consider the following pseudo-SML code implementing leftist heaps.

Data Structure 15.3. *Leftist Heap*

```
1  datatype PQ = Leaf | Node of int × key × PQ × PQ
2
3  fun rank Q =
4    case Q of
5      Leaf ⇒ 0
6    | Node (r,_,_,_) ⇒ r
7
8  fun makeLeftistNode (k, A, B) =
9    if rank A < rank B
10   then Node (1 + rank A, k, B, A)
11   else Node (1 + rank B, k, A, B)
12
13 fun meld (A, B) =
14   case (A, B) of
15     (_, Leaf) ⇒ A
16   | (Leaf, _) ⇒ B
17   | (Node (_,k_a,L_a,R_a), Node (_,k_b,L_b,R_b)) ⇒
18       if k_a < k_b
19       then makeLeftistNode (k_a, L_a, meld (R_a, B))
20       else makeLeftistNode (k_b, L_b, meld (A, R_b))
21
22 fun singleton k = Node (1,k,Leaf,Leaf)
23
24 fun insert (Q,k) = meld (Q, singleton k)
25
26 fun fromSeq S = Seq.reduce meld Leaf (Seq.map singleton S)
27
28 fun deleteMin Q =
29   case Q of
30     Leaf ⇒ (NONE, Q)
31   | Node (_,k,L,R) ⇒ (SOME k, meld (L, R))
```

Task 15.4. *Diagram the process of executing the code*

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

Task 15.5. *What are the work and span of* (`fromSeq` S) *in terms of* $|S| = n$*?*

15.2.2 Dynamic Median

Task 15.6. *Design a data structure which supports the following operations:*

	Work	*Span*	*Description*
`fromSeq` S	$O(\|S\|)$	$O(\log^2 \|S\|)$	*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)	$O(\log \|M\|)$	$O(\log \|M\|)$	*Inserts* k *into* M

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

15.3 Additional Exercises

Exercise 15.7. *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.*