Lec 12 Balanced Binary Tree II (BT ADT, Treaps)

Today: split, union, filter, splitAt

Generic interface

```
struct
  type T                                        — tree with augmentation, etc.
  type E                                        — elem
  type N = Leaf | Node of T×E×T                 — exposed form
  size : T → Z                 O(1)
  expose : T → N               O(1)
  empty : T
  joinM : T×E×T → T            usually O(|height L - height R|)
end
```

helpers:

```
singleton = λ x ⇒ joinM (empty, x, empty)
append    = λ A B ⇒ case expose A of            ← only preserves BST if L < R
                Leaf ⇒ B
              | Node (L, x, R) ⇒ joinM (L, x, append R B)
```

Impls filter ← works on both BST and tree seq

```
filter p A = case expose A of
    Leaf ⇒ empty
  | Node (L, x, R) ⇒
      let (L', R') = (filter L || filter R) in
      if p x then joinM (L', x, R')
             else append (L', R')
```

Assume for now:

joinM, append (← both sequential) W, S $O(\lg n)$ $(n = |L'| + |R'|$, assume $|L| = |R|)$

$$W_{filter}(n = |L| + |R|) = 2W(\tfrac{n}{2}) + O(\lg n) \in O(n)$$

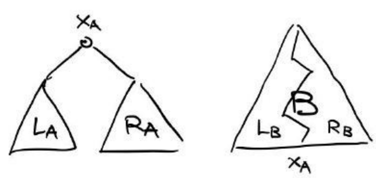
$$S_{filter}(n) = S(\tfrac{n}{2}) + O(\lg n) \in O(\lg^2 n)$$

Impl split (BST only)

```
| REQ BST A                                                    : T × E
| ENS return ({x ∈ A | x < k}, k ∈ A, {x ∈ A | x > k})         : T × B × T
split A k = case expose A of                    W = O(lg n)  w.h.p.
  Leaf ⇒ (empty, false, empty)
 | Node (L, x, R) ⇒ case cmp x k of
     EQUAL ⇒ (L, true, R)
    | LESS ⇒ let
         (L1, b, R1) = split L k
       in
         (L2, b, joinM (R1, x, R))
       end
    | GREATER ⇒ [symmetry]
```

Impl union

```
| REQ A, B BSTs
| ENS BST with set of all keys in A, B
union A B = case (expose A, expose B) of
   (Leaf, _) ⇒ B
 | (_, Leaf) ⇒ A
 | (Node (LA, xA, RA), _) ⇒ let
        (LB, _, RB) = split B xA
        (L', R') = (union LA LB || union RA RB)
     in
        joinM (L', xA, R')
     end
```



cost analysis (assume $|L'| = |R'|$, $|L_A| = |L_A|$, $|L_B| = |R_B|$, WLOG $\underbrace{|A|}_{m} \leq \underbrace{|B|}_{n}$)

$W(n, m) = 2W(\frac{m}{2}, \frac{n}{2}) + O(\lg n)$ m-n ratio is same

$W(1, n_1) = \lg n_1$ ← Leaf-dominated but $n_1 = \frac{n}{m}$

$= \lg \frac{n}{m} \in O(\lg(\frac{n}{m} + 1))$ ← in case $\frac{n}{m} = 1$

$\#$ leafs $= 2^{\lg m} = m$

cost (base) $= m \lg(\frac{n}{m} + 1)$

So $O(m \lg(\frac{n}{m} + 1))$ ← in fact this is also lower bound.

(Span ... $\in O(\lg n \lg m)$)

Treaps aka Tree-Heap (with randomisation !)

Basically bin tree + heap ordering on priority

Priority $p : \mathbb{E} \to \mathbb{Z}$ can assume for large enough co-domain

"random hash" elem $\mapsto$ unique int priority

Tree priority $\quad pr\ A =$ case expose A of
$$\begin{aligned} &\text{Leaf} \Rightarrow -\infty \\ &\mid \text{Node}(\_, x, \_) \Rightarrow p(x) \end{aligned}$$

__Def__ Treap A satisfies: A is bin tree s.t. $\forall \text{Node}(L, x, R) \in A$,
$p(x) > pr(L) \qquad p(x) > pr(R)$

__Thm__ Treap has $O(\lg n)$ depth whp.

Proof sketch similar to quicksort

$$\text{IRV} \quad A_{i,j} = \begin{cases} 1 & \text{if rank } i \text{ is ancestor of } j \\ 0 & \text{else} \end{cases}$$

rank in tree

$$\mathbb{E}[A_{i,j}] = P[\,i \text{ ancestor of } j\,] = \frac{1}{|i-j|+1}$$