Recitation 6

Treaps

6.1 Announcements

- Midterm 1 is on **Friday**. You are allowed a single, double-sided, 8.5×11in sheet of paper for notes.
- *FingerLab* is due **next Friday, Mar 3**.

6.2 Example

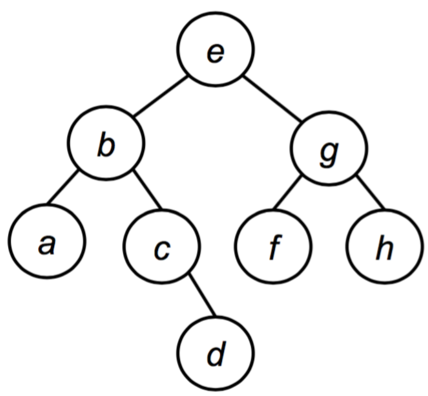
Recall that a treap is a BST with a priority function $p : U \to \mathbb{Z}$, where U is the universe of keys. You should think of p as a random number generator: for each key, it returns a random integer. A treap has two structural properties:

1. **BST invariant**: For every `Node`(L, k, R), we have $\ell < k$ for every ℓ in L, and symmetrically $k < r$ for every r in R.
2. **Heap invariant**: For every `Node`(L, k, R), we have that $p(k) > p(x)$ for every x in either L or R.

Task 6.1. *Build a treap from the following keys and priorities using two different strategies, and observe that the resulting treap is the same in both cases.*

1. *Run quicksort, creating a new node every time a pivot is chosen.*
2. *Beginning with an empty tree, sequentially insert keys in priority-order. Each newly inserted key should be placed at a leaf.*

k	a	b	c	d	e	f	g	h
$p(k)$	5	7	3	2	8	4	6	1

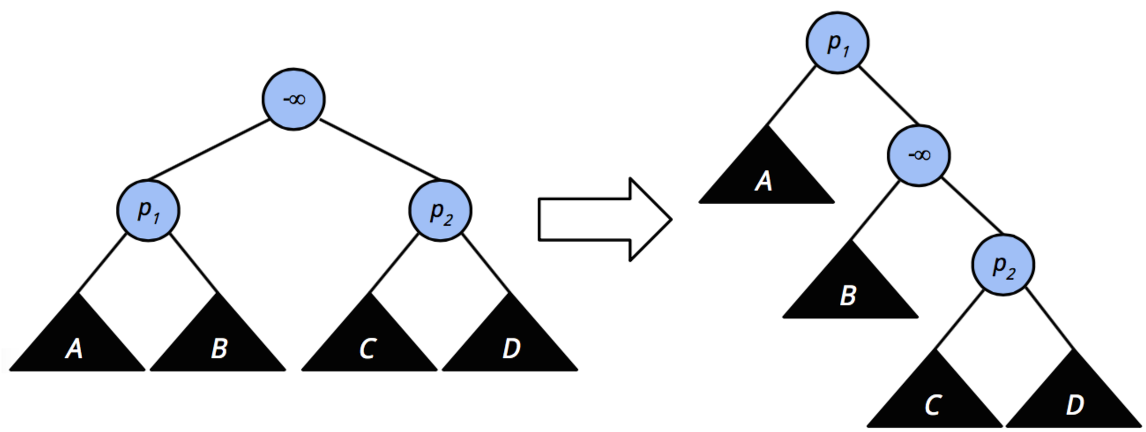


6.3 Deletion

Consider the following strategy for deleting a key k from a treap:

1. Locate the node containing k,
2. Set the priority of k to be $-\infty$ (note that if k has children, then this breaks the heap invariant of the treap),
3. Restore the heap invariant by rotating k downwards until it has only leaves for children,
4. Delete k by replacing its node with a leaf.

A "rotation" in this case refers to the process of making one of k's children the root, depending on their relative priorities. For example, if k has two children with priorities p_1 and p_2 where $p_1 > p_2$, we rotate like so:



The case of $p_1 < p_2$ is symmetric. In turns out that this process is equivalent to calling `join` on the children of k. You should convince yourself of this.

We're interested in the following: in expectation, *how many rotations must we perform before we can delete k?*

Let's set up the specifics: we have a treap T formed from the sorted sequence of keys S, $|S| = n$. We're interested in deleting the key $S[d]$. Let T' be the same treap, except that the priority of $S[d]$ is now $-\infty$.

We need a couple indicator random variables:

$$X_j^i = \begin{cases} 1, & \text{if } S[i] \text{ is an ancestor of } S[j] \text{ in } T \\ 0, & \text{otherwise} \end{cases}$$
$$(X')_j^i = \begin{cases} 1, & \text{if } S[i] \text{ is an ancestor of } S[j] \text{ in } T' \\ 0, & \text{otherwise} \end{cases}$$

Task 6.2. *Write R_d, the number of rotations necessary to delete $S[d]$, in terms of the given random variables.*

The number of rotations is equal to the **number of nodes which aren't an ancestor of $S[d]$ in T, but are in T'**. Therefore we have

$$R_d = \sum_{i=0}^{n-1} (X')_d^i - \sum_{i=0}^{n-1} X_d^i$$

Task 6.3. *Give $\mathbf{E}\left[X_d^i\right]$ and $\mathbf{E}\left[(X')_d^i\right]$ in terms of i and d.*

We have both $X_d^i = 1$ and $(X')_d^i = 1$ if $S[i]$ has the largest priority among the $|d-i|+1$ keys between $S[i]$ and $S[d]$. However, notice that in the latter case, we already know that the priority of $S[i]$ is larger than that of $S[d]$, unless $i = d$. So we only need that $S[i]$ is the largest among the $|d-i|$ significant keys in this range. Therefore:

$$\mathbf{E}\left[X_d^i\right] = \begin{cases} 1, & \text{if } i = d \\ \frac{1}{|d-i|+1}, & \text{otherwise} \end{cases}$$
$$\mathbf{E}\left[(X')_d^i\right] = \begin{cases} 1, & \text{if } i = d \\ \frac{1}{|d-i|}, & \text{otherwise} \end{cases}$$

Task 6.4. *Compute* $\mathbf{E}[R_d]$. *For simplicity, you may assume* $1 \leq d \leq n-2$.

$$
\begin{aligned}
\mathbf{E}[R_d] &= \sum_{i=0}^{n-1} \mathbf{E}\left[(X')_d^i\right] - \sum_{i=0}^{n-1} \mathbf{E}\left[X_d^i\right] \\
&= \left(\sum_{i=0}^{d-1} \mathbf{E}\left[(X')_d^i\right] + 1 + \sum_{i=d+1}^{n-1} \mathbf{E}\left[(X')_d^i\right] \right) - \left(\sum_{i=0}^{d-1} \mathbf{E}\left[X_d^i\right] + 1 + \sum_{i=d+1}^{n-1} \mathbf{E}\left[X_d^i\right] \right) \\
&= \left(\sum_{i=0}^{d-1} \frac{1}{d-i} + \sum_{i=d+1}^{n-1} \frac{1}{i-d} \right) - \left(\sum_{i=0}^{d-1} \frac{1}{d-i+1} + \sum_{i=d+1}^{n-1} \frac{1}{i-d+1} \right) \\
&= \big(H_d + H_{n-d-1}\big) - \big((H_{d+1} - 1) + (H_{n-d} - 1) \big) \\
&= 2 + \big(H_d - H_{d+1}\big) + \big(H_{n-d-1} - H_{n-d}\big) \\
&= 2 - \frac{1}{d+1} - \frac{1}{n-d} \\
&\leq 2
\end{aligned}
$$

6.4 Additional Exercises

Exercise 6.5. *Describe an algorithm for inserting an element into a treap by "undoing" the deletion process described in Section 6.3.*

Exercise 6.6. *For treaps, suppose you are given implementations of* `find`*,* `insert`*, and* `delete`*. Implement* `split` *and* `joinMid` *in terms of these functions. You'll need to "hack" the keys and priorities; i.e., assume you can do funky things like insert a key with a* specific *priority.*

Exercise 6.7. *Given a set of key-priority pairs* $(k_i, p_i) : 0 \leq i < n$ *where all of the* k_i*'s are distinct and all of the* p_i*'s are distinct, prove that there is a unique corresponding treap* T.

6.4.1 Selected Solutions

Exercise 6.6.

- Implement $\texttt{split}(T, k)$ as follows. First, determine if k is present in T via `find`. Then, insert k with priority ∞ into T. The resulting treap will have the form $\texttt{Node}(L, k, R)$. We then return (L, m, R), where m was the result of the `find`.
- Implement $\texttt{joinMid}(L, k, R)$ as follows. Set $p(k) = \infty$, and then let $T = \texttt{delete}(\texttt{Node}(L, k, R), k)$. Finally, restore $p(k)$ to its correct value, and finish with $\texttt{insert}(T, k)$.