Binary Search Trees

Data Structure Dictionary

$S$ is an ordered set.

1) Search $(K, S)$ ≡ $x \in S$?
2) Insert $(K, S)$ ≡
3) Delete $(K, S)$

Note: If 1, 2, 3) are the design requirement then use a Hash Table

4) Range $(K, K', S) \equiv \left| \{k^* \in S \mid K \leq k^* \leq K' \} \right|

If 1), ..., 4) use BST
Define a tree $T$ in BST for keys $S$ if:

1) $T$ is an ordered binary tree with 15 nodes.
2) Each node stores a key.
3) Keys are in inorder.

Example: $S = \{a, b, c, d, e\}$

$T = \quad \Rightarrow \quad$ BST

$T$ is balanced if $\text{max depth}(T) = O(\log n)$
Always Balanced - AVL, 2-3-4, RB, B-Trees
Randomized - Skip-list, Treaps ≈ tree-heaps
Amortized - Splay Trees

All these use the Rotation

To show: inorder is preserved
Note: Subtrees α, β, γ unchanged!
Applications

Persistence (Add 5) undo last op

First Idea keep a tree for each time

\[ T_1 \quad T_2 \quad \cdots \quad T_k \]

\( O(n^2) \) space.

Second store only changes Insert(\( k \))

\[ \text{time}_i \quad \text{time}_{i+1} \]

\( O(\log n) \) space per new tree.
Vanilla-Insert (K, T)
1) Find empty leaf node
2) add K to leaf

QS(A)  Bad Quick Sort
1) Pick first a in A
2) Split A into <S< a>, a, a <L>
3) Return <QS(S), a, QS(L)>

Vanilla-BST(A, T) VBST
1) Extract first a from A
2) Return VBST(A, VI(a, T))

Note. QS & V-BST do exactly the same comparisons but in different order!
Treaps (Tree-Heaps)

Keys \in \{1, \ldots, n\}

Priorities \equiv p(k) \quad p(k) \neq p(l) \text{ for } k \neq l

T = \text{tree with a key at each node.}

**Def:** T is in heap order if \( \forall x \in T \) x not root

\[ p(\text{parent}(x)) < p(x) \]

**Lemma** A heap order BST exists and is unique.

**Pf:** Vanilla-insert in priority order.
Random Treap

\(\text{Insert}(K) = \)
1) insert \(k\) into a leaf \((VI(k, T))\)
2) pick random \(p(k)\)
3) rotate \(k\) up until in heap order

\(\text{Delete}(k) = \)
1) rotate \(k\) to a leaf
   by picking highest priority child
2) remove \(k\).

Correctness:
Expected Cost for Treaps

**Goal:** Determine expected # of comparisons to 
\[ \text{search}(m, s, K) = S(m, s) \] over all treaps.

**Example:** \( K = \{1, 2, 3\} \) & search \( \{2.5, K\} \)

Treaps: Insert orders \( n! \), \( 4^n \) trees

\[(123), (132), (213), (231), (312), (321)\]

\[
\begin{array}{cccccc}
\text{#} & \text{Compare} & 3 & 3 & 2 & 2 & 3 & 2 \\
\text{S}(2.5) &=& \frac{15}{6} &=& 2^{3/2} \\
\text{Note for random BST} &=& \frac{13}{5} &=& 2^{3/5}
\end{array}
\]
Assume keys = \{1, \ldots, n\} in our Treap.

**Definitions:**

Event \( E \) is \( C(i, m) = \{i \text{ is compared with } m \text{ on } \text{Search}(m)\} \)

**Claim:**

\[
\text{Prob} \left[ C(i, m, s) \right] = \begin{cases} 
\frac{1}{m-i+1} & \text{if } i \leq m \\
\frac{1}{i-m} & \text{if } i > m 
\end{cases}
\]

Treap construction as a dart game:

- **Board:** 1 \( \ldots \) n

Game = Throw darts at empty sqs till full.

Dart form insertion order

Consider first dart for case \( i < m \).

\[1 \quad i \quad m \quad m+1 \quad \ldots \quad n\]

maybe \quad no \quad maybe
\[\text{maybe} \quad \text{yes} \quad \text{no} \quad \text{maybe}\]

**Informal argument:** WLOG board is

\[i \quad \ldots \quad m\]

\[\text{yes} \quad \text{no} \quad \text{yes} \quad \text{no}\]

\[
\text{Prob} \left[ C(i, m, s) \right] = \frac{1}{m-i+1} \quad \text{for } i \leq m
\]
Formal Argument (Using Law of Total Probability)

\[ B_j = \text{Event } [\text{ith plant is first to land in } [i, \ldots, m]] \]

\[ B_k \cap B_j = \emptyset \quad k \neq j \quad \text{and} \quad \sum_{j=1}^{\infty} \Pr(B_j) = 1 \quad (\star) \]

Conditional Probability: Let \( A, B \) be events

\[ \Pr[A \mid B] = \frac{\Pr[A \cap B]}{\Pr[B]} \]

\[ \Pr[C(i, m, z)] = \sum_{j=1}^{\infty} \Pr[B_j] \Pr[A \mid B_j] \quad \forall m \text{ by } (\star) \]

Note: \( \Pr[A \mid B_j] = \frac{1}{m-i+1} \)

\[ \Pr[A] = \sum_{j=1}^{\infty} \Pr[B_j] \left( \frac{1}{m-i+1} \right) = \frac{1}{m-i+1} \sum_{j=1}^{\infty} \Pr[B_j] = \frac{1}{m-i+1} \]

QED
\[ S(m, s) = \# \text{comparisons searching m.s} \]
\[ = \sum_{i=1}^{n} C(i, m, s) \]

\[ E(S(m, s)) = \sum E(C(i, m, s)) = \sum \text{Prob}[C(i, m, s)] \]
\[ = \sum_{1 \leq i \leq m} \frac{1}{m-i+1} + \sum_{i=m}^{n} \frac{1}{i-m} \leq 2 \sum_{i=1}^{n} \frac{1}{i} = 2 \ln n \]
\[ = 2 \ln n + O(1) \]

Expect \# of Comparisons for

Insert, Search, Delete = \( O(\log n) \)
Counting Rotations

2 Cases: Insert & Delete

Delete: Move-to-Leaf

Thm: Expect #rotations ≤ 2

Claim: Pivots with m are

1) Right-most node in left subtree of m
2) Left" "right"

Diagram:

- Root
- Node m
- Subtrees
- Induction
Def Random variables $D_i$ & $D'_i$

$D_i = \begin{cases} 1 & \text{if } i \text{ is a right-most node in left subtree of } m \\ 0 & \text{otherwise} \end{cases}$

$D'_i = \begin{cases} 1 & \text{if } i \text{ is a left-most node in right subtree of } m \\ 0 & \text{otherwise} \end{cases}$

Claim $\Pr[D_i = 1] = \left( \frac{1}{m-i+1} \right) \left( \frac{1}{m-i} \right)$

pf Dart Game (Informal)

Board for $D_i$

\[\Pr[D_i = 1] = \left( \frac{1}{m-i+1} \right) \left( \frac{1}{m-i} \right)\]

$\Pr[D'_i = 1] = \left( \frac{1}{i-m+1} \right) \left( \frac{1}{i-m} \right) \quad \text{(similarly)}$
Def: RV $R_m = \# \text{rotations in move-to-leaf of } m$.

By linearity of expectation:

\[
\mathbb{E}[R_m] = \sum_{i < m} \mathbb{E}[D_i] + \sum_{i \geq m} \mathbb{E}[D'_i]
\]

\[
= \sum_{i < m} \Pr[D_i] + \sum_{i \geq m} \Pr[D'_i]
\]

\[
= \sum_{i=1}^{m-1} \frac{1}{(m-i+1)(m-i)} + \sum_{i=m+1}^{\eta} \frac{1}{(i-m+1)(i-m)}
\]

\[
= \sum_{i=1}^{m-1} \frac{1}{(i+m)(i)} + \sum_{i=1}^{\eta-m} \frac{1}{(i+1)i}
\]

Note \( \frac{1}{(i+m)i} = \frac{1}{i} - \frac{1}{i+1} \) Thru

\[
\text{LH Term} = \sum_{i=1}^{m-1} \frac{1}{i} - \sum_{i=2}^{\eta} \frac{1}{i} = 1 - \frac{1}{m} < 1
\]

Thus $\mathbb{E}[R_m] < 2$. 