Binary Search Trees

Data Structure

Dictionary

S is an ordered set.

1) Search \((k, S) \equiv k \in S?\)

2) Insert \((k, S) \equiv \)

3) Delete \((k, S)\)

Note: If 1, 2, 3) are the design requirements then use a hash table.

4) Range \((k, k', S) \equiv |\{k'' \in S \mid k \leq k'' \leq k'\}|\)

If 1), ..., 4) use BST
Def 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

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

$T$ is balanced if $\max_{\text{depth}(T)} = O(\log n)$
Types of Balanced BSTs

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

\[ T_1 \quad T_2 \quad T_x \]

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

Ideal 2 store only differences \( \text{Insert}(k) \)

\( O(\log n) \) space per new tree.
Vanilla - Insert \((K, T)\)

1) Find empty leaf node
2) add \(k\) to leaf

\[ QS(A) \text{ Bad Quick Sort} \]

1) Pick first \(a\) in \(A\)
2) Split \(S\) into \(S < a, a, S < L\)
3) Return \(QS(S), a, QS(L)\)

\[ \text{Eager} \]

\[ \text{Lazy} \]

Vanilla - BST\((A, T)\) \(\text{VBST}\)

1) Extract first \(a\) from \(A\)
2) Return \(\text{VBST}(A, \text{VI}(a, T))\)

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(U) \text{ for } K \neq U

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

Def: T is in heap order if \forall x \in T \text{ } x \neq root

\quad p(\text{parent}(x)) < p(x)

Lemma: A heap order BST exists and is unique.

Proof: Vanilla-insert in priority order.
Random Treap

$\text{Insert}'(k) = \begin{align*}
1) \text{ insert } & k \text{ into a leaf } (VI(k,T)) \\
2) \text{ pick random } & p(k) \\
3) \text{ rotate } & k \text{ up until in heap order}
\end{align*}$

$\text{Delete}(k) = \begin{align*}
1) \text{ rotate } & k \text{ to a leaf by picking highest priority child} \\
2) \text{ remove } & k
\end{align*}$

Correctness:
**Expected Cost for Treaps**

**Goal:** Determine expected number of comparisons to search \( S(m.s, K) = S(m.s) \) over all treaps.

eg \( K = \{1, 2, 3\} \) & search \( \{2.5, K\} \)

Treaps insert orders \( n! \) 4\(^n\) trees

\[
\begin{align*}
(123) & \quad (132) & \quad (213) & \quad (231) & \quad (312) & \quad (321) \\
\end{align*}
\]

\[
\begin{align*}
1 & \quad 2 & \quad 3 & \quad 1 & \quad 3 & \quad 1 & \quad 3 & \quad 2 & \quad 1 \\
2 & \quad 3 & \quad 1 & \quad 1 & \quad 3 & \quad 1 & \quad 1 & \quad 2 \\
3 & \quad 2 & \quad 1 & \quad 2 & \quad 1 & \quad 2 & \quad 1 & \quad 3 \\
\end{align*}
\]

\[
S(2.5) = \frac{15}{6} = 2.5
\]

Note for random BST \( \frac{13}{5} = 2.3 \)
Keys \{i_1, \ldots, n\} in Treap

\[ C(i, m) = \text{Event } [i \text{ is compared to } m \text{ in } \text{Search}(m)] \]

**Claim**  \[ \text{Prob}[C(i, m,s)] = \frac{1}{m-i+1} \text{ if } i \leq m \]

\[ = \frac{1}{i-m} \text{ if } i > m \]

---

**Treap construction as death game**

---

In informal argument, WLOG branch is

\[ \text{Prob}[C(i, m,s)] = \frac{1}{m-i+1} \]

---

\[ i \]  \[ m \]  \[ m+i \]  \[ n \]

---

maybe  yes  no

maybe  yes
Formal Argument (Using Law of Total Probability)

\[ B_j = \text{Event} \left[ \text{ith object first to land in } [i, \ldots, m] \right] \]

\[ B_k \cap B_j = \emptyset \quad \text{for } k \neq j \quad \text{and} \quad \Pr \left( \bigcup_{j=1}^{m} B_j \right) = 1 \quad \text{(*)} \]

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

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

\[ \Pr \left[ C(i, m, \delta) \right] = \sum_{j=1}^{\infty} \Pr \left[ B_j \right] \Pr \left[ A \mid B_j \right] \quad \text{for } m \text{ by (*)} \]

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

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

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

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

\[ = 2 \, \log n + O(1) \]

---

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 nodes in left subtree of m
   2) Left "right" "left"

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

Proof (Informal)

Board for \( D_i \)

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

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

\[
E(R_m) = \sum_{i < m} E(D_i) + \sum_{i > m} E(D_i')
\]

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

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

**Note**: \( \frac{1}{(i+1)i} = \frac{1}{i} - \frac{1}{i+1} \)

**L.H. Term**: \[
= \sum_{i=2}^{m-1} \frac{1}{i} - \sum_{i=2}^{m} \frac{1}{i} = 1 - \frac{1}{m} < 1
\]

\[
E(R_m) < 2
\]