COMS E6998: Advanced Data Structures (Spring'19) Jan 31, 2019

Lecture 2: Dynamic Optimality I

Instructor: *Omri Weinstein* Scribes: *Weston Jackson and Justin Wong*

1 Plan

- BST Model + Instance Optimality
- Splay Trees
- Geometric View of Dynamic Optimality Conjecture
- Greedy BST (via Treaps)

2 Last Time

2.1 Predecessor Search

Maintain $S \subseteq [n]$ s.t. under predecessor search queries:

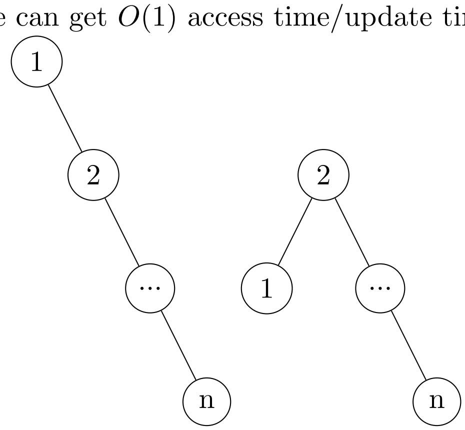
$$Pred(x) := max\{y \in S | y \leq x\}$$

For example, $Pred(57)$ when $S = \{9, 17, 23, 50, 76, 79\}$ returns 50. We let AS be the sequences of searches such that $AS = \{x_1 ... x_n\}$. NB: We generally assume AS to be longer than n.

2.2 Self-Balancing BSTs

- Examples: RBTs, AVLs, Rand BSTs
- Local rotations: $t_n = t_q = \Theta(\log n)$ op. (am (amortized)/wc (worst-case))

Can we do better? Depends on AS. For example, take the sequential / monotone $AS := \{1, 2, 3...n\}$. We can get $O(1)$ access time/update time if we rotate after each access:



Need instance specific benchmarks!

3 BST Model

3.1 Overview

BST Model $\leq$ Pointer Machine (PM)

- Keys are stored as nodes of a BST
- Allowed ops starting at route:

  (i) Walk up/left/right

  (ii) Local rotation:

  x → y

  y C   A x

  A B   B C

- No RAM in BST-Model! Reduces to PM-Model.

Remarks:

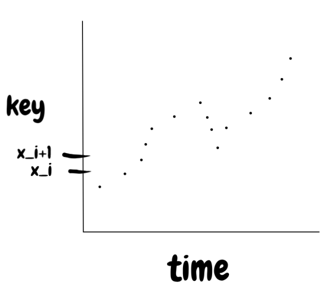
- BST Model is weak.
- $\exists$ a single access sequence AS simultaneously hard $\Omega(n \log n)$ for all BSTs!
- We focus only on searches. For now, rotations are used to speed up search time.

3.2 Properties

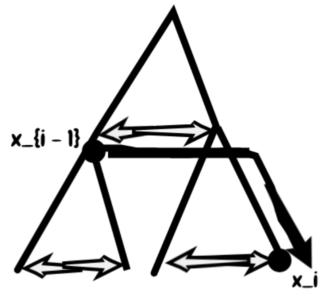
(I) Sequence Access Property: Monotone $AS := \{1...n\}$ $O(1)$ / op (am).

(II) Dynamic Finger Property: (Spatial Locality)

$$|x_i - x_{i-1}| \leq k \implies O(\log k) \text{ / op}$$



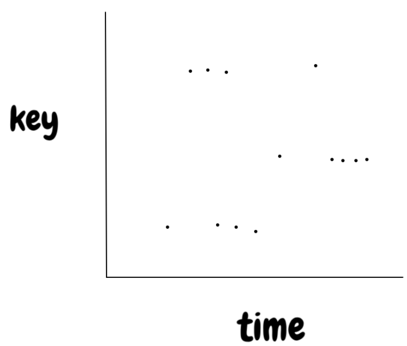
Ex. Finger Trees (level-linked trees):



Does BST have this property? $\approx$ (50 p. SICOMP [Cole 2000])

(III) Working-Set/Move-To-Front (MTF Property): (Temporal Locality)

$$\text{If } t_i \text{ distinct keys accessed since last } S(x_i) \implies O(\log t_i) \text{ / op}$$



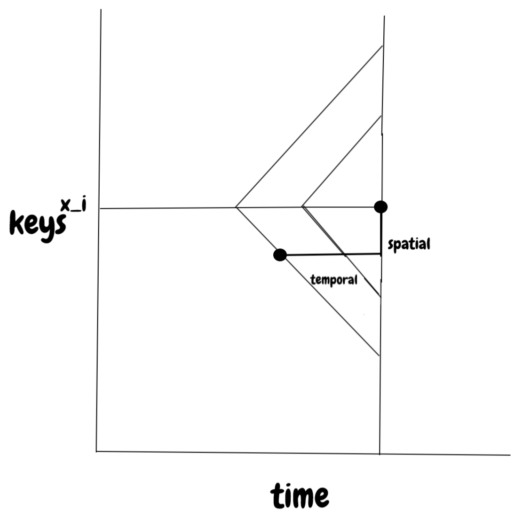
(IV) Entropy Property:

$$\text{Frequency of } x_i := p_i \implies O(\sum_i p_i \log \frac{1}{p_i}) \text{ / op (am)}$$

Static optimality: $\Theta(\sum_i p_i \log \frac{1}{p_i})$ opt if we disallow rotations. Essentially Huffman compression trees.
Final Project (FP): Tree Compression vs Dynamic BST [Godin 2019]

(V) Unified Property:

$$t_{ij} \text{ distinct key accesses before } x_i \implies cost(x_i) \leq O(\log(min_{j \leq i}\{|x_i - x_j| + t_{ij}\}) + 2)$$



For example consider if we accessed 13 in the past. Now if we've never seen 15 we want it so that a recent access of 13 would shorten the search time for 15. Hence the L1 ball (bounded by the sum of the components).
Open Question: $\exists$ BST satisfying unified property?
Relationships

1. (II) $\implies$ (I) : this follows from the fact that (I) s the special case where $k = 1$.
2. (III) $\implies$ (IV) : intuitively consider the two sequences 111111222222 vs 111211221222 same frequencies but clearly (IV) would capture the correlation in the former.

Dynamic Optimality Conjecture [ST' 85]:

$$\exists \text{ a single dynamic BST } \mathcal{T} \text{ s.t. } \forall x \text{ (AS) } : C_{\mathcal{T}}(x) \leq O(C_{BST}(x))$$

$\implies$ Best online solution = Best offline solution
where $C_{BST}(x)$ is the optimal tree over all possible trees on that x.
Essentially posing that in the case of a resource limited BST model, being able to predict the future doesn't help you.

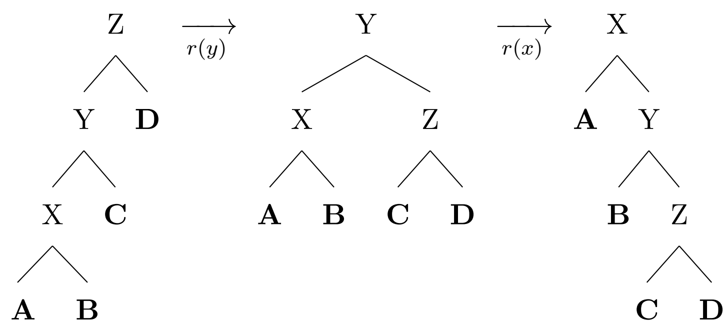
4 Splay Trees [ST '81]

4.1 Premise

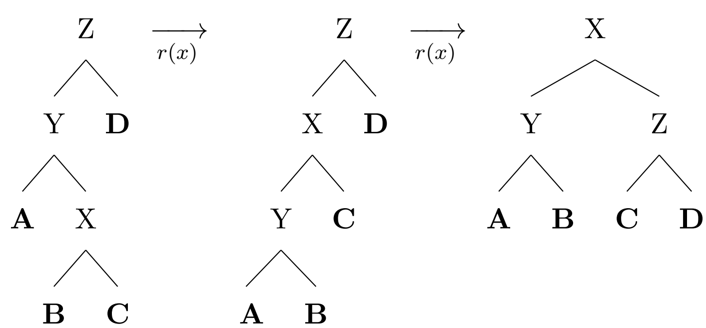
- We consider the restriction of BST and no updates on the task of search.
- Splay Trees are conjectured to be Dynamically Optimal (Open)
- Motivated by the Huffman Tree, which is optimal for the static case, we want a dynamic tree which will keep recent keys close to the root.

4.2 Algorithm

- Upon searching for a key, we propogate it to the root of the tree using one of 4 splaying rotations. Since rr and ll are similar and rl and lr are similar we just present one of each pair.
- Double Rotations ("splaying"):
  - rr-splay (Zig-Zig rotation)



  - lr-splay (Zig-Zag Rotation)



- We don't aim to maintain a balanced tree but rather a tree that adapts to recent events thereby increasing **temporal locality**.
- Concerning spacial locality, note that suppose w isn't in the search path of x but they the subpath share ℓ. Then dw doesn't change that much:

$$d'_w \leq d_w + \ell/2 + O(1)$$

. Effectively, it halves the 'shared' depth each time a **spatially local** query is made.

4.3 Theorem MTF property of Splay Trees

Theorem 1. *Splay trees have amortized search/insert/delete time of $O(\log n)$/op (insert/delete left as an exercise).*

Proof. Via potential argument. Let $\mathcal{T}_i$ be the i^{th} tree (ie the tree after the i^{th} search). For potential argument, we choose a w such that $w(x) > 0$ and in particular for this theorem take $w(x) = 1$.

Define:

$$S_i(s) = \sum_{y \in \mathcal{T}_i(x)} w(y)$$

and Rank:

$$r_i(x) = \log(S_i(x))$$

Define the potential Φ to be:

$$\Phi(i) := \Phi(\mathcal{T}_i) = \sum_{x \in \mathcal{T}_i} r_i(x)$$

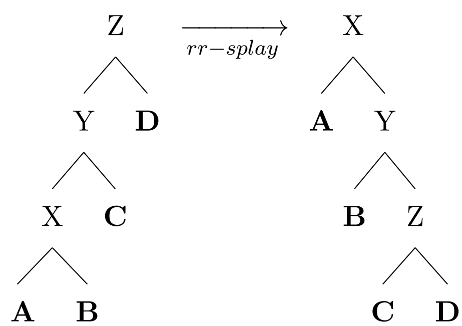
Effectively it represents the average depth over nodes in $\mathcal{T}_i$.

Lemma 2 (Access Lemma)**.** *The amortized* $Cost(\Phi)$ *of the* i^{th} *splay operation on* x *is* $\leq 3(r_i(x) - r_{i-1}(x))$

Proof of Access Lemma (rr-splay):

$$\hat{Cost}_x(i) = 2 + \Delta_i \Phi$$

as we are considering the amortized cost of operation on the node x on the i^{th} operation which consists of two rotations in addition to the change in Φ induced by i. Recall the rr-splay we do out the analysis for rr but not that rl (by extension also lr and ll) are similar:



To compute $\Delta_x(i)$, we note that the subtrees internally are unchanged by the splay operation, so we only need to worry about the x, y, z. We use the following inequalities:

$$\textbf{(I)}\ r_{i+1}(x) = r_i(z)$$

$$\textbf{(II)}\ r_i(x) \leq r_i(y)$$

$$\textbf{(III)}\ r_{i+1}(y) \leq r_{i+1}(x)$$

By inspection, we see that **(I)** holds since the entire tree remains in the subtree of the rotation root. Next note (**II**) holds because y is a parent of x, while (**III**) holds since x is a parent of y after the rotation.

Going back to our expression of cost:

$$\begin{aligned}
\hat{Cost}_x(i) &= 2 + \Delta_i \Phi \\
&= 2 + r_{i+1}(x) + r_{i+1}(y) + r_{i+1}(z) - r_i(x) - r_i(y) - r_i(z) \\
&\leq_{\textbf{(I)}} 2 + r_{i+1}(x) + r_{i+1}(y) + r_{i+1}(z) - r_i(x) - r_i(y) - r_{i+1}(x) \\
&= 2 + r_{i+1}(y) + r_{i+1}(z) - r_i(x) - r_i(y) \\
&\leq_{\textbf{(II)}} 2 + r_{i+1}(y) + r_{i+1}(z) - 2r_i(x)
\end{aligned}$$

Aside:

$$\begin{aligned}
\frac{r_i(x) + r_{i+1}(z)}{2} &= \frac{\log(S_i(x)) + \log(S_{i+1}(z))}{2} \\
&\leq \frac{\log(S_i(x) + S_{i+1}(z))}{2} && \text{via log concavity} \\
&\leq \frac{\log(S_{i+1}(x))}{2} && \text{A and B are disjoint from C and D} \\
&\leq r_{i+1}(x) - 1
\end{aligned}$$

rearranging:

$$r_{i+1}(z) \leq 2r_{i+1}(x) - r_i(x) - 2$$

Applying this to the cost function:

$$\begin{aligned}
\hat{Cost}_x(i) &\leq 2 + r_{i+1}(y) + r_{i+1}(z) - 2r_i(x) \\
&\leq 2 + r_{i+1}(y) + (2r_{i+1}(x) - r_i(x) - 2) - 2r_i(x) \\
&= r_{i+1}(y) + 2r_{i+1}(x) - 3r_i(x) \\
&\leq_{\textbf{(III)}} 3(r_{i+1}(x) - r_{i+1}(x)) \\
&= 3(r_{i+1}(x) - r_{i+1}(x))
\end{aligned}$$

As we have proven the lemma as we set out to. $\square$

Now to prove the Theorem:

$$\begin{aligned}
\hat{Cost}(x) &= \sum_{i=1}^{k} \hat{Cost}_i(x) \\
&\leq_{AL} 3 \sum_{i=1}^{k} r_i(x) - r_{i-1}(x) \\
&= 3(r_k - r_1) \\
&\leq 3(\log n - 0) \\
&= O(\log n)
\end{aligned}$$

$\square$