6.851	Lecture 1	Feb. 2, 2010

Prof. Erik Demaine & Dr. André Schulz
TA: Aleksandar Zlateski

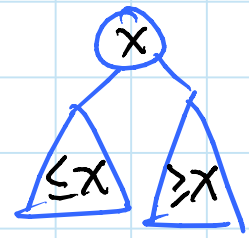
Topics: comparison, graph, integer, geometric, string, succinct, cache-efficient DSs

Administration:
- signup sheet
- requirements: homework, scribing, project
- listeners welcome
- problem session (starting ~ week 3)
- scribe for today

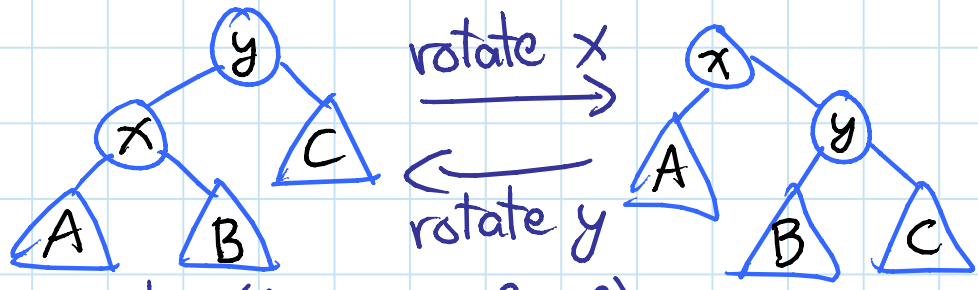
TODAY: Dynamic Optimality I
- binary search trees
- analytic bounds
- splay trees
- geometric view
- greedy algorithm

Q: is there one best binary search tree (BST)?

BST: comparison data structure supporting search
(& predecessor/successor, insert/delete)



Also a model of computation (for DSs)
- data must be stored in a BST
- unit-cost operations:
  - walk left, right, or up (parent)
  - rotate this node & its parent



(- create/destroy leaf)

⇒ search cost = length of root-to-node path

DSs in this model:
- vanilla BST (no rotations)
- AVL trees
- red-black trees (B-trees)
- weight-balanced trees, treaps
- splay trees
- Tango trees
- Greedy

$O(\lg n)$ / op.

focus here

Is $O(\lg n)$/search optimal?

- depends on sequence of searches
- say we're storing keys $\{1, 2, \ldots, n\}$ & search for $x_1, x_2, \ldots, x_m$

Sequential access property:

$1, 2, \ldots, n \Rightarrow O(1)$ amortized/op.

[in-order traversal in any BST]

Dynamic finger property:

$|x_i - x_{i-1}| = k \Rightarrow O(\lg k)$/op. possible

[think level-linked B-trees ~ but BST] ⊛

Entropy bound / static optimality: best possible without rotation

[Iacono-SWAT2000]

k appears p_k fraction of the time $\Rightarrow O\left(\sum_{k=1}^{n} p_k \lg \frac{1}{p_k}\right)$/op.

[store x_i at height $\leq \lg \frac{1}{p_k} + 1$]

Working-set property:

if t_i distinct keys accessed since last access to x_i, then $O(\lg t_i)$ possible

[intuition: store most recent higher up] ⊛

$\Rightarrow$ if all $x_i \in S$ then $O(\lg |S|)$/op. possible

[form BST on S, put rest below]

⊛ = hard to do with BST, but possible!

Unified property: [Iacono – SODA 2001]

if t_{ij} distinct keys accessed in $x_i, \ldots, x_j$
then x_j costs $O\left(\lg \min_i \left[\underbrace{|x_i - x_j|}_{\text{space}} + \underbrace{t_{ij}}_{\text{time}} + 2\right]\right)$

"fast if close to something recent" ⊛

- e.g. $1, \frac{n}{2}, 2, \frac{n}{2}+1, 3, \frac{n}{2}+3, \ldots \Rightarrow O(1)/\text{op.}$
- implies both working set & unified
- possible on pointer machine [Iacono; Bădiou, Cole, Demaine, Iacono – Algorithmica 2007]
- possible on BST up to additive $O(\lg \lg n)$ [Bose, Douïeb, Dujmović, Howat – arXiv 2009]
- $\boxed{\text{OPEN}}$: possible on a BST?

Dynamic optimality / $O(1)$-competitive:

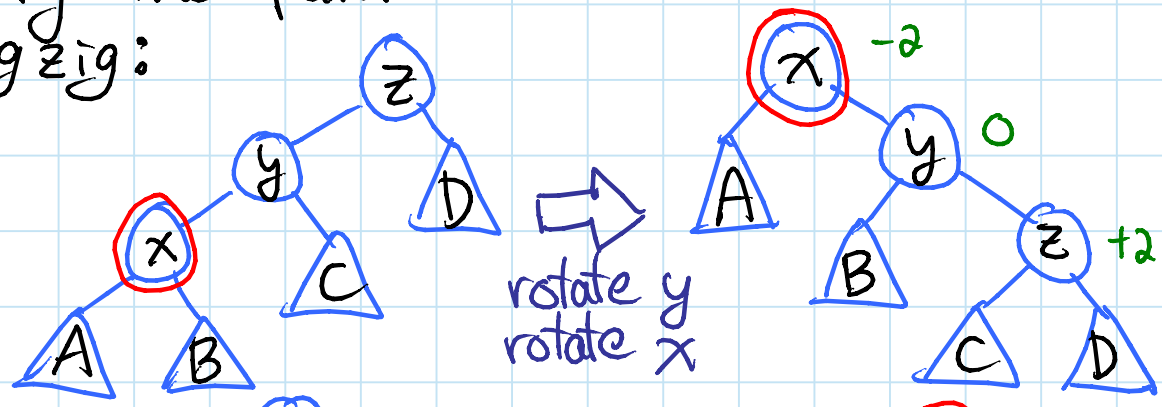
total cost $= O(\underbrace{OPT}_{\text{min. cost of any BST on this access sequence}})$

- $\boxed{\text{OPEN}}$: is this possible for any BST? (online)

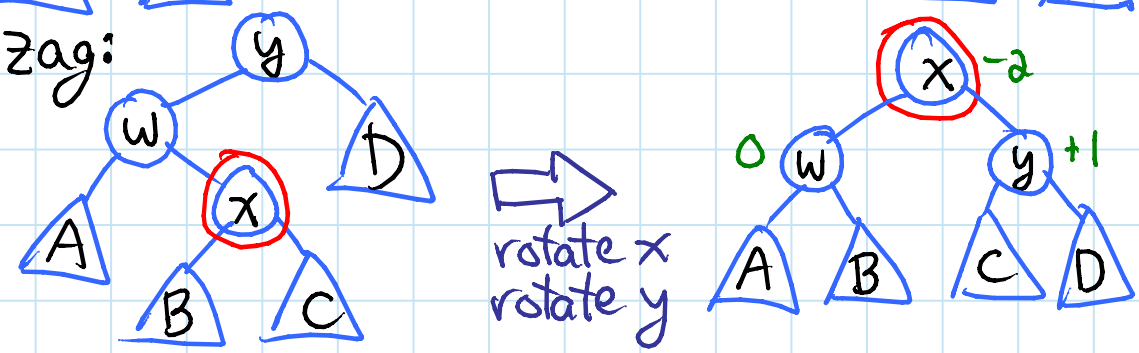
  for any pointer-machine DS?

Splay trees: [Sleator & Tarjan – JACM 1985]

- binary search for x
- modify the path:
  - zigzig:



  - zigzag:



- at the end, possible single rotation to put x at root
- key feature: at most half the nodes on the path go down in the tree

Performance: (amortized)

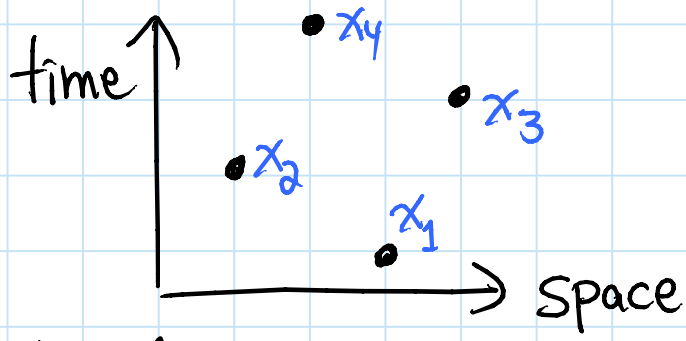
- has working-set property [Sleator & Tarjan]
- has dynamic-finger property [Cole – SICOMP 2000]

- CONJECTURE: has unified property
- CONJECTURE: dynamically optimal

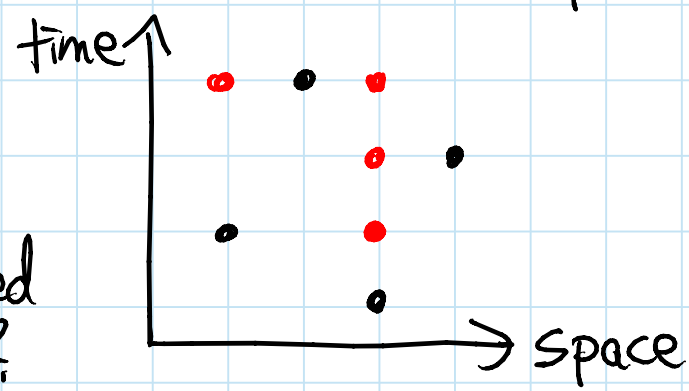
Geometric view:

[Demaine, Harmon, Iacono, Kane, Pătrascu – SODA 2009]

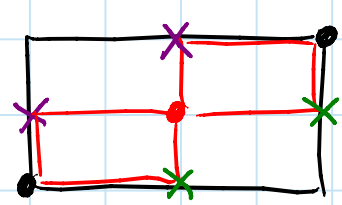
access sequence
→ point set
$\{(x_i, i)\}$



BST execution
→ point set:
which nodes touched during search(x_i)?



Theorem: point set is a valid BST execution
⟺ Arborally Satisfied Set (ASS)
↳ rectangle spanned by two points in set, not on horizontal/vertical line, contains another point
– in fact must have another point on a rectangle side incident to either corner:



Corollary: OPT = smallest ASS containing input

$\boxed{\text{OPEN}}$: complexity? $O(1)$-approximation?

Proof of Theorem:

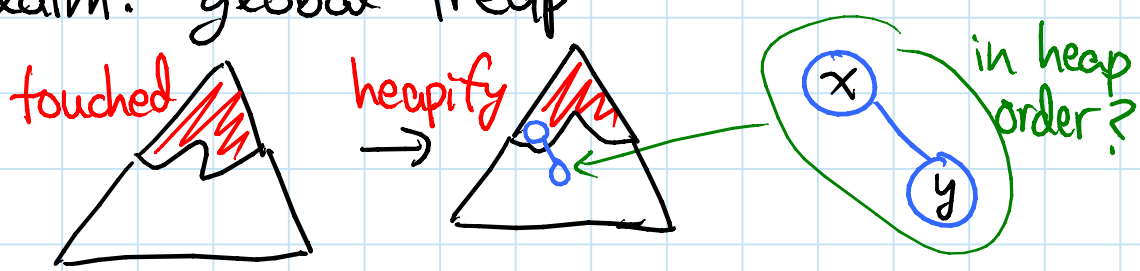
($\Rightarrow$) let a = lca of x_i & x_j (changes over time)

- so $x_i \le a \le x_j$ & a is ancestor of x_i & x_j
- if $a \ne x_i$ just before search(x_i) then $(a, i) \in$ execution
- if $a \ne x_j$ just before search(x_j) then $(a, j) \in$ execution
- else: a changes from x_i to x_j between times i & j
  $\Rightarrow$ x_j rotated before search(x_j)

$\Rightarrow$ get third point in $(x_i, i) \to (x_j, j)$ rectangle in all cases

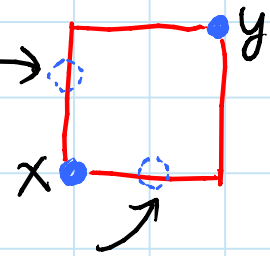
($\Leftarrow$) define tree at all times to be treap: BST & heap ordered by next-touch-time

- note: next-touch-time has some ties, so this is not uniquely defined
- when we reach time i, nodes to touch form a connected subtree at the top (by heap-order property)
- these nodes get new next-touch-time
- re-arrange into local treap (this still may be ambiguous — break ties arbitrarily — but still restricts global choice)
- claim: global treap



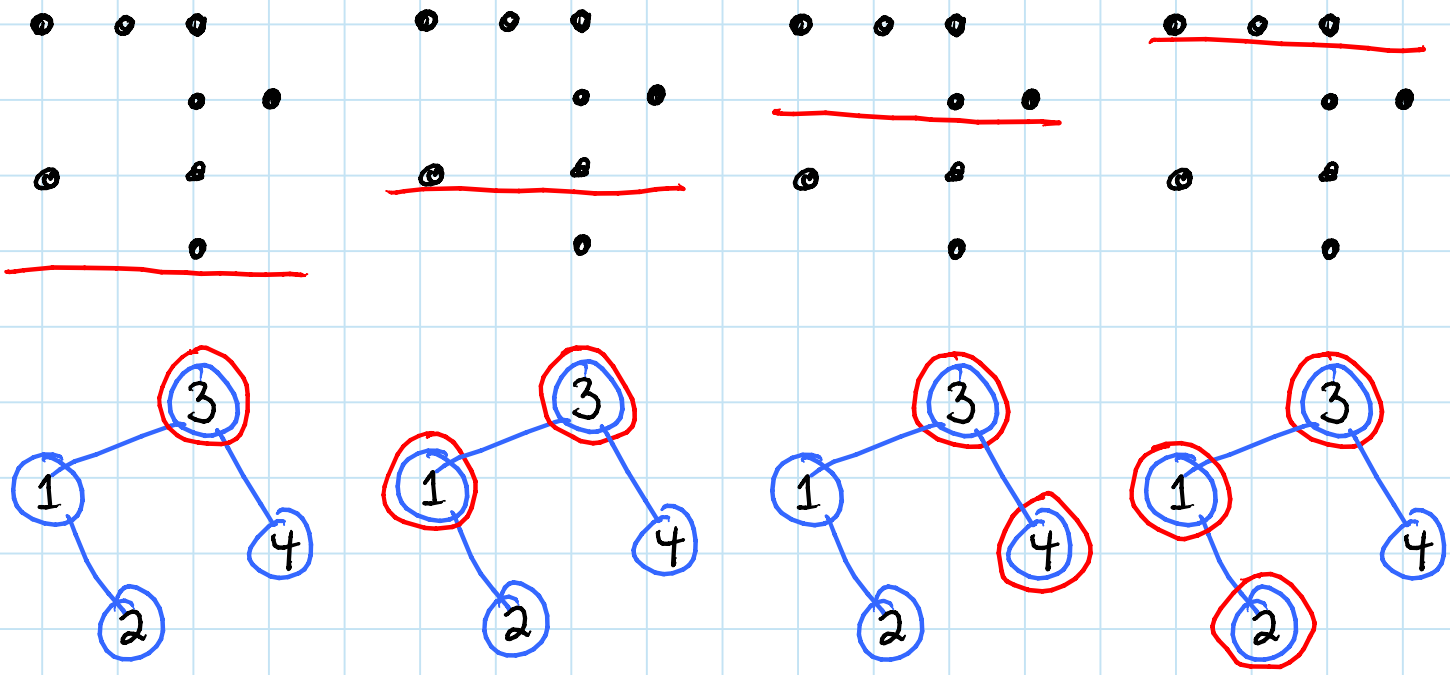
if y to be touched sooner than x then (x, next-touch) $\to$ (y, next-touch) is an unsatisfied rectangle: (according to 2nd definition of ASS)

empty by "if" $\to$



leftmost such point would be right child of x after search(x_i), not y $\square$

Simple example:



Greedy algorithm: [Lucas 1988; Munro 2000]

- consider point set one row at a time
- add the necessary points on that row
- in tree view: re-arrange root-to-x path optimally for future searches

$\boxed{\text{CONJECTURE}}$: Greedy $= O(OPT)$

or even: $= OPT + O(m)$

- seems obvious... "just" need to show you needn't stray from the access path

So what?

Theorem: online ASS algorithm $\rightarrow$ online BST (with $O(1)$ slowdown)

Corollary: Greedy is actually an online BST!

- Conjecture $\Rightarrow$ dynamically optimal