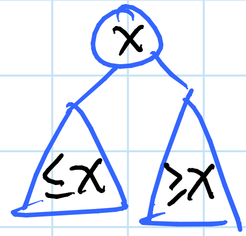
6.851	Lecture 5	Feb. 29, 2012

TODAY: Dynamic Optimality I (of 2)

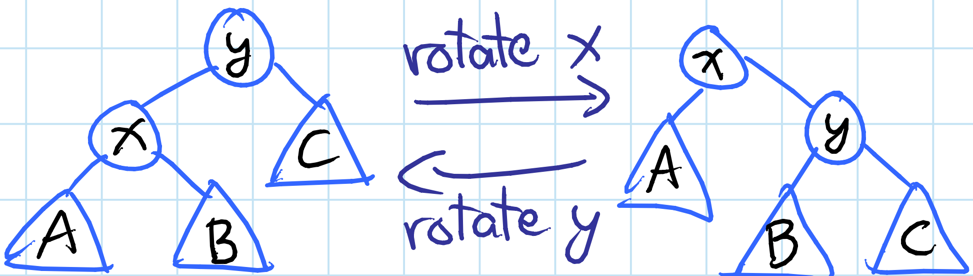
- binary search trees
- analytic bounds
- splay trees
- geometric view
- greedy algorithm

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)
- BB[α] trees
- 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 $\{1, 2, \dots, n\}$ & search for $x_1, x_2, \dots, x_m$

Sequential access property:
$1, 2, \dots, n \Rightarrow O(1)$ 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
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 $\leq \lg \frac{1}{p_k} + 1$]

[Iacono - SWAT 2000]

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 $x_i, \dots, x_j$

then x_j costs $O(\lg \min_i [\underbrace{|x_i - x_j|}_{\text{space}} + \underbrace{t_{ij}}_{\text{time}} + 2])$

"fast if close to something recent" ⊛

- e.g. $1, \frac{n}{2}, 2, \frac{n}{2}+1, 3, \frac{n}{2}+3, \dots \Rightarrow O(1)/\text{op.}$
- implies both working set & dynamic finger
- possible on pointer machine [Iacono; Bădoiu, Cole, Demaine, Iacono – Algorithmica 2007]
- possible on BST up to additive $O(\lg \lg n)$ [Bose, Douïeb, Dujmović, Howat – Algorithmica 2012]
- $\boxed{\text{OPEN}}$: possible on a BST?

Dynamic optimality / O(1)-competitive:

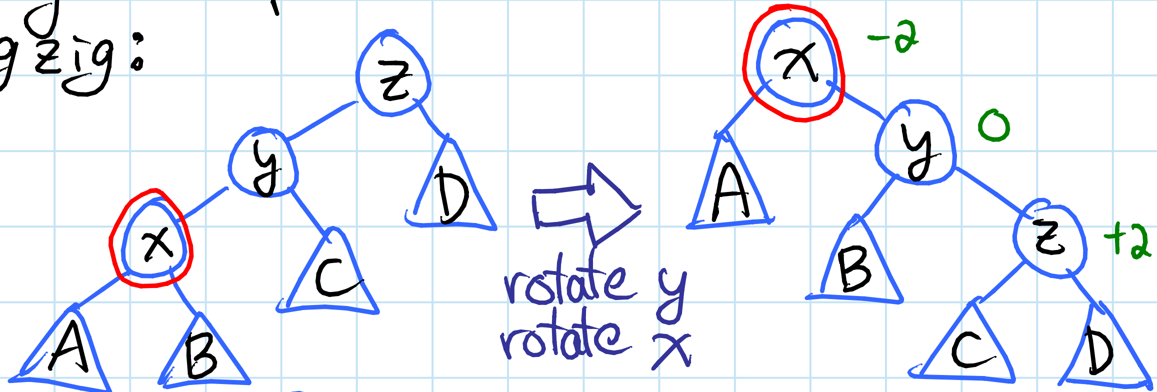
total cost $= O(\underbrace{\text{OPT}}_{\text{min. cost of any BST on this access sequence}})$

- $\boxed{\text{OPEN}}$: possible for any (online) BST?
  for any pointer-machine DS?
- $\boxed{\text{OPEN}}$: is any pointer-machine DS $= O(\text{OPT offline pointer-machine DS})$?

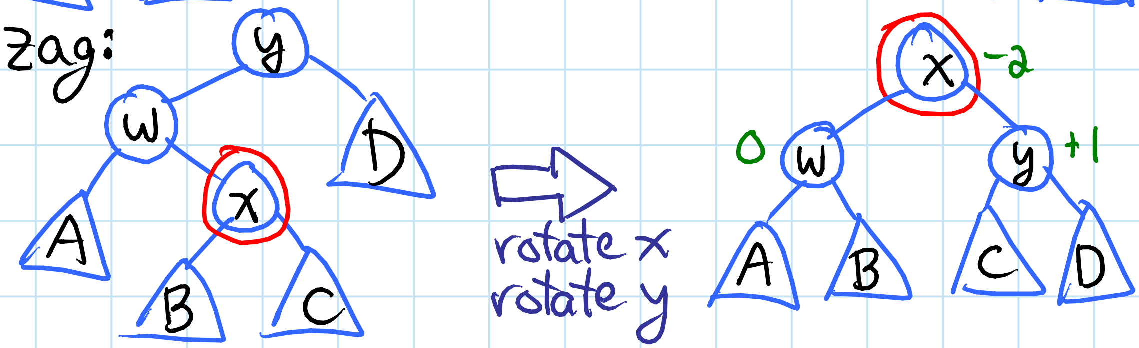
- balanced BST is $O(\lg n)$-competitive
- Tango trees are $O(\lg \lg n)$-competitive [L6]

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

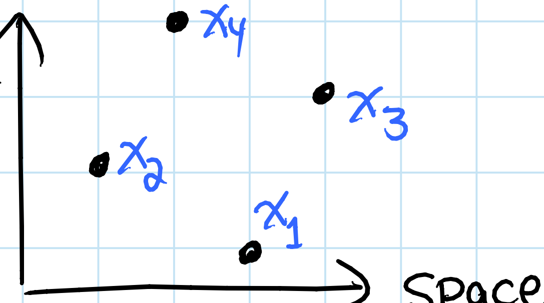
Performance: (amortized)

- has working-set property [Sleator & Tarjan]
- has dynamic-finger property [Cole – SICOMP 2000]
- CONJECTURE: has unified property [Iacono]
- CONJECTURE: dynamically optimal [Sleator & Tarjan]

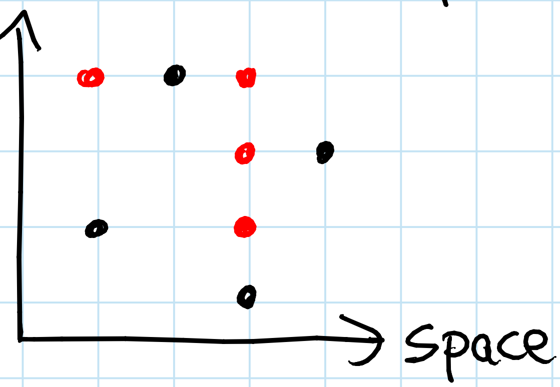
Geometric view:

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

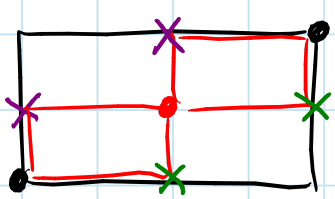
access sequence
→ point set
$\{(x_i, i)\}$



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:

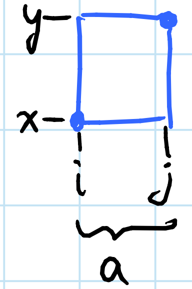


Corollary: OPT = smallest ASS containing input

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

Proof of Theorem:

($\Rightarrow$) consider rectangle spanned by $(i, x) \to (j, y)$

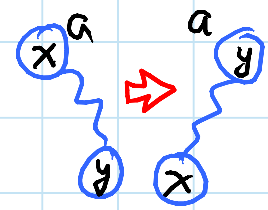


- let a_t = lca of x & y just before time t
- for all t: $x \le a_t \le y$ & a_t is an ancestor of x & y

$\Rightarrow (a_i, i)$ & $(a_j, j) \in$ execution

(need to touch all ancestors of touched nodes)

- want a third point in the rectangle
- if $a_i \neq x$ then use (a_i, i)
- if $a_j \neq y$ then use (a_j, j)
- else: a changes from x to y between times i & j

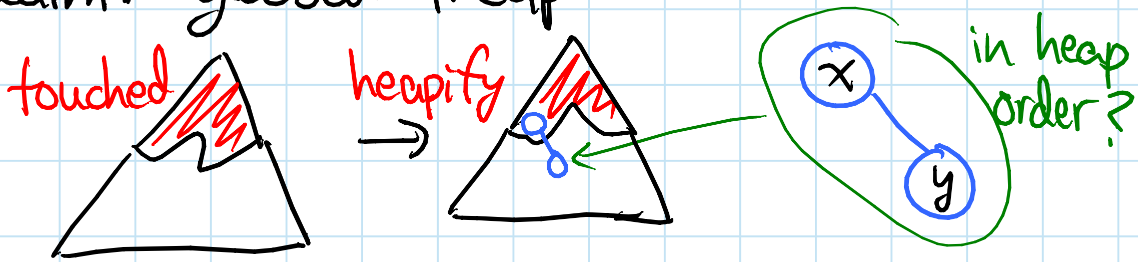


$\Rightarrow$ y rotated before time j

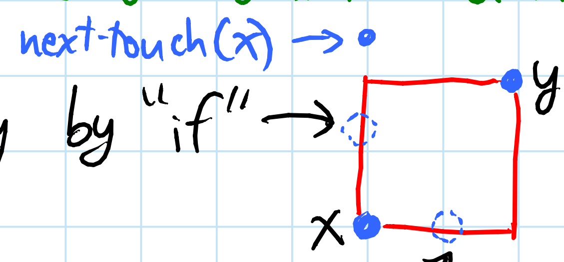
$\Rightarrow (y, t) \in$ execution for some $i \le t < j$

(⇐) 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

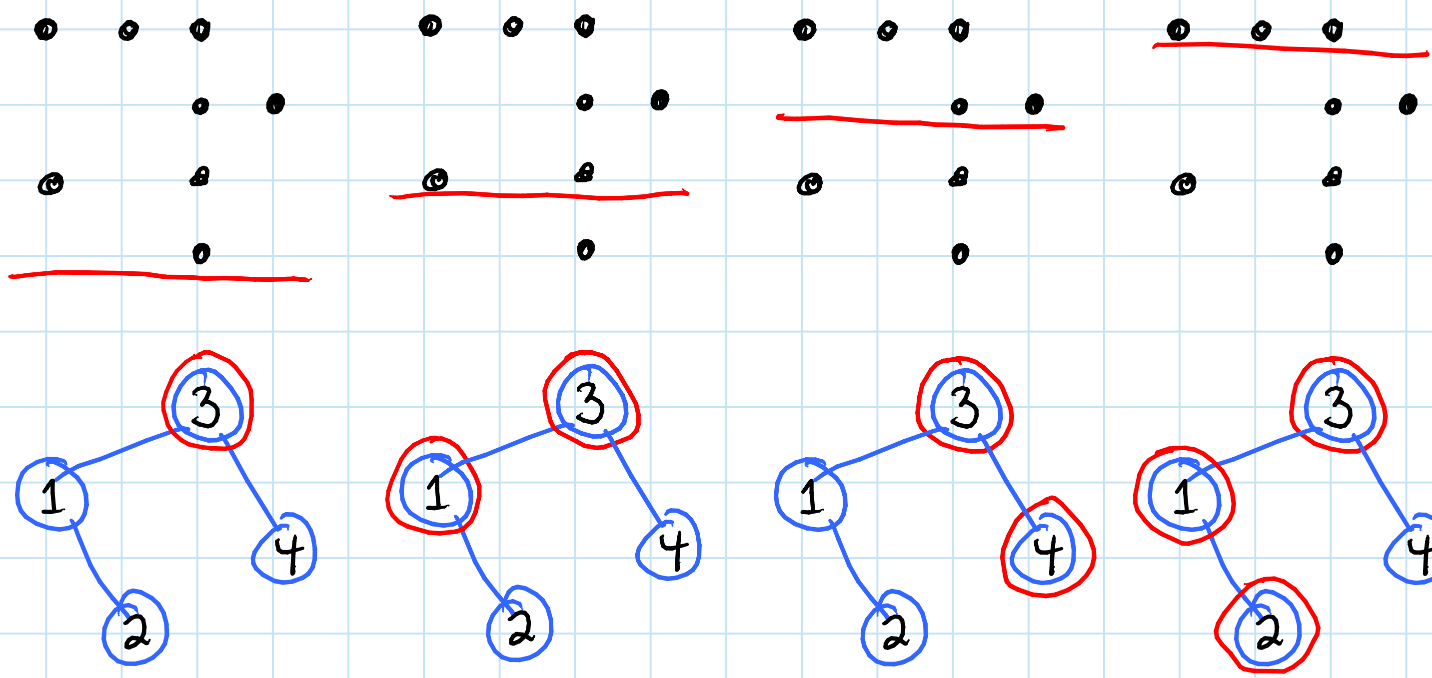


if y to be touched sooner than x then $(x, \text{now}) \to (y, \text{next-touch}(y))$ is an unsatisfied rectangle:
(according to 2nd definition of ASS)



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

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 $= O(\text{OPT})$

or even: $= \text{OPT} + O(m)$

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

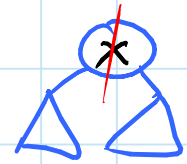
So what?

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

Corollary: Greedy is actually an online BST!
- Conjecture $\Rightarrow$ dynamically optimal

Proof sketch of theorem:

- store touched nodes from access in a <u>split tree</u>: split(x) moves x to root & deletes x, leaving 2 split trees in $O(1)$ amortized time ~ if fully split:



- really: all n splits in $O(n)$ time (& make split tree on n items in $O(n)$)
- 2-3-4 tree with min & max pointers can split into n' & n'' in $O(\lg \min\{n', n''\}) + O(n)$ total merges
- use potential $\Phi = \sum_{\text{split tree } T} (|T| - \lg |T|)$

$\Rightarrow O(1)$ amortized search cost for split
- simulate with BST: interleaved min/max search

$\Rightarrow$ BST is "treap of split trees", where heap order is by <u>previous</u> touch & ties mean in split tree ($\Rightarrow$ optimal order)
- use proof similar to ($\Leftarrow$) above
- by ASS, when touching node in split tree, also touch predecessor & successor in parent split tree $\Rightarrow$ cheap to reach