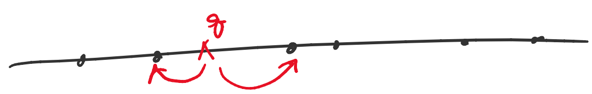
March 31, 2021 9:07 AM

Randomized Data Structures

store a set S of n numbers to support:

- search:
  - given q, is $q \in S$? (membership)
  - given q, find predecessor/successor in S
    - (may not be in S)



- insert
- delete

Known:

Static: $O(n)$ space, $O(n\log n)$ preproc time, $O(\log n)$ query time

dynamic: $O(\log n)$ query & update time, $O(n)$ space

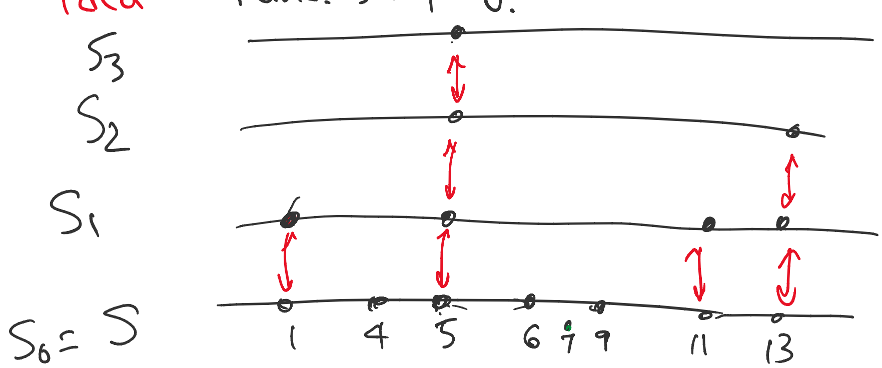
by
- AVL trees
- red-black trees
- 2-3 trees, 2-3-4 trees,

messy cases

- BB(α) trees
- splay tree
- AA trees
- ⋮

Rand. Method 1: Skip Lists (Pugh '90)

idea - rand. sampling!



expected space $O(n + \frac{n}{2} + \frac{n}{4} + \dots) = O(n)$.

$$S = S_0 \supseteq S_1 \supseteq S_2 \supseteq \dots \quad \text{till } S_\ell = \emptyset.$$

Let $S_0 = S$

$\forall x \in S_i$, we put $x \in S_{i+1}$ w. prob. $\frac{1}{2}$ by indep coin flip.

Store each S_i in a sorted linked list

add ptrs between S_i & S_{i+1}.

Obs # levels $= O(\log n)$ w.h.p.

Pf: Fix an elem x.

level of x is geom distrib. w. prob. $\frac{1}{2}$

$\Rightarrow$ mean 2

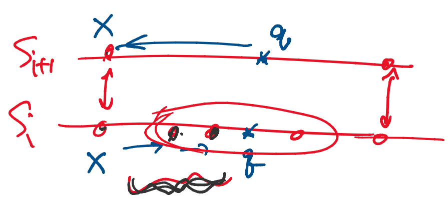
$\Rightarrow \Pr(\text{level of } x = i) = \left(\frac{1}{2}\right)^{i-1} \cdot \frac{1}{2} = \frac{1}{2^i}$

$\Rightarrow \Pr(\text{level of } x \geq c\log n)$

$\leq O\left(\left(\frac{1}{2}\right)^{c\log n}\right)$

$= O\left(\frac{1}{n^c}\right).$

Union bd. $\square$



pred-search(S_i, q):

x = pred-search(S_{i+1}, q)

do linear search in S_i from x

$\Rightarrow$ for fixed q, Query time at level i is geom distrib. w. prob $\frac{1}{2}$

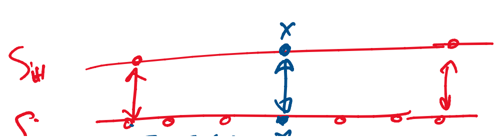
$\Rightarrow E[\text{query time per level}] = O(1)$

$\Rightarrow E[\text{query time}] = \boxed{O(\log n)}.$

insert(S_i, x, p): // given ptr p to pred of x

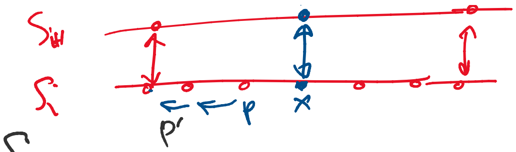
flip coin

if heads {



flip coin
if heads {

do linear search in S_i
from p to find pred p' in S_{i+1}
insert(S_{i+1}, x, p')
}



$\Rightarrow \quad E[\text{insert time at level } i \mid \text{level}(x) \geq i]$

$= O(1)$ geom distrib. w. prob $\frac{1}{2}$

$\Rightarrow \quad Pr[\text{level}(x) \geq i] = O\left(\frac{1}{2^i}\right).$

$\Rightarrow \quad E[\text{insert time}] = O\left(\sum_i \frac{1}{2^i}\right) = \boxed{O(1)}$ if given pred

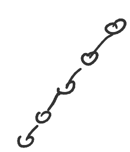
(if not, $O(\log n)$.

Same for delete.

Rand. Method 2: Treaps (Seidel-Aragon '96)

idea - back to binary search tree
pick root "randomly" ←

how? assign each elem
a rand. priority value in [0,1]
(indep).



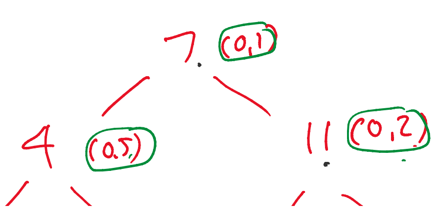
for each subtree,
choose elem w. lowest priority value as its root.

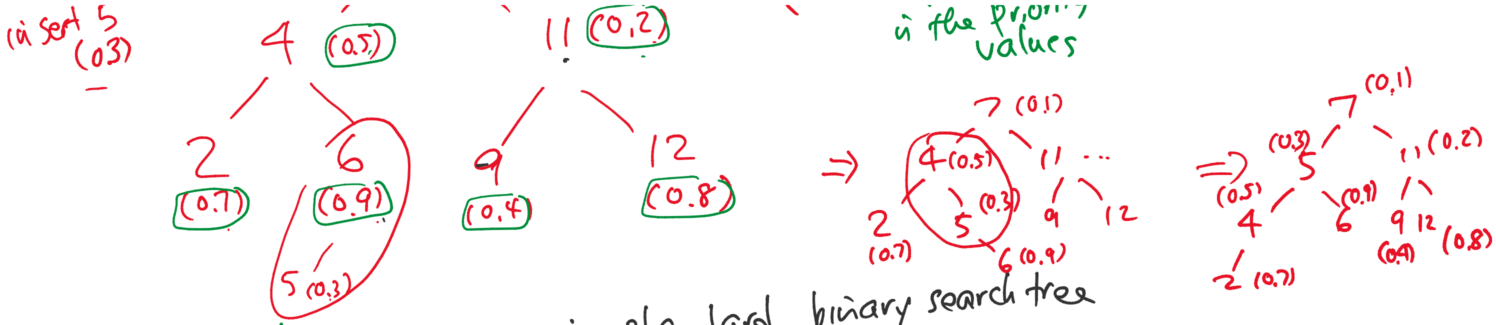
keys {2, 4, 6, 7, 9, 11, 12}
priorities (0.7) (0.5) (0.9) (0.1) (0.4) (0.2) (0.8)

note: simultaneously
binary search tree
(in the key values)

& heap
in the priority values

insert 5
(0.3)





Pred-search: same as in standard binary search tree

query time $\boxed{O(\log n)}$ w.h.p.

(equiv. to recurs. depth of rand. quicksort)

insert: pick rand. priority value

fix problems by rotations

$\boxed{O(1)}$ expected time if pred/succ given

(pf omitted)

Similar for delete

no messy cases

no extra ptrs.

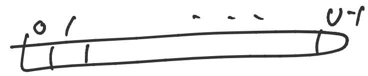
Question: Can we do better than $O(\log n)$ query time?

no for comp-based algms

but yes for membership queries for integers!

Assume all elems are in $\{0, 1, .., U-1\}$.

Easy method 0:



use bit vector of size U

query time $O(1)$

insert $O(1)$

delete $O(1)$

but space is $O(U)$. big!

Next: hashing . . .