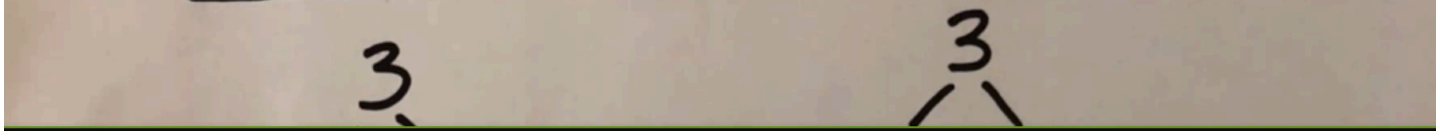
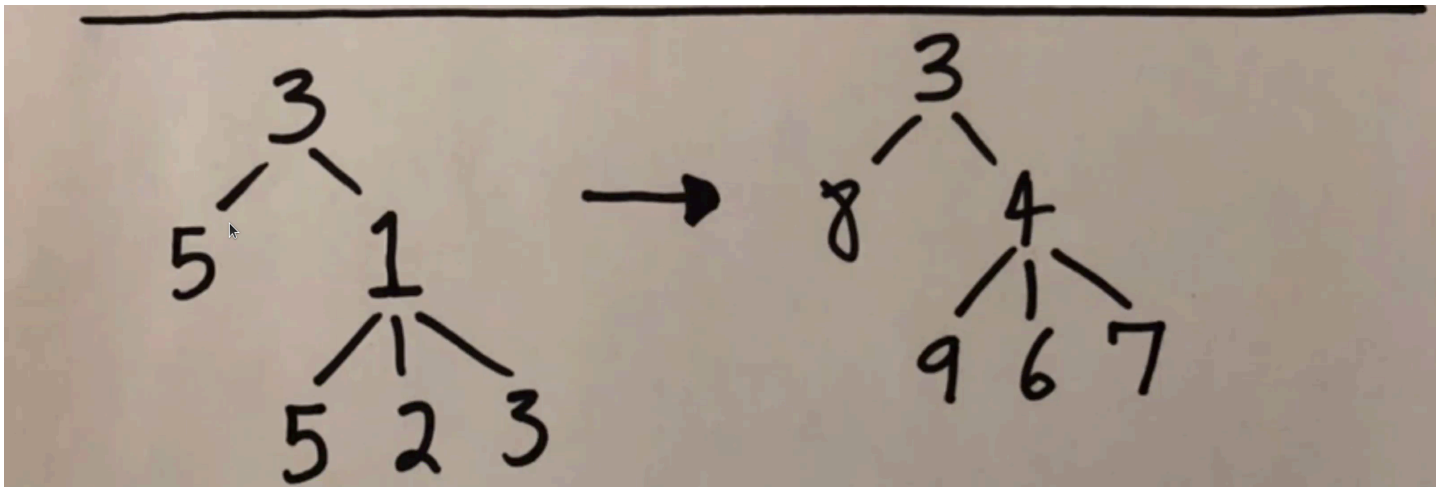
4.a

Midterm Practice (10/21/22)

Problem 4: T a tree. v.val holds non-negative integers.

a.) Compute v.PthTo for each v, the sum of v.val from the root to the node v.





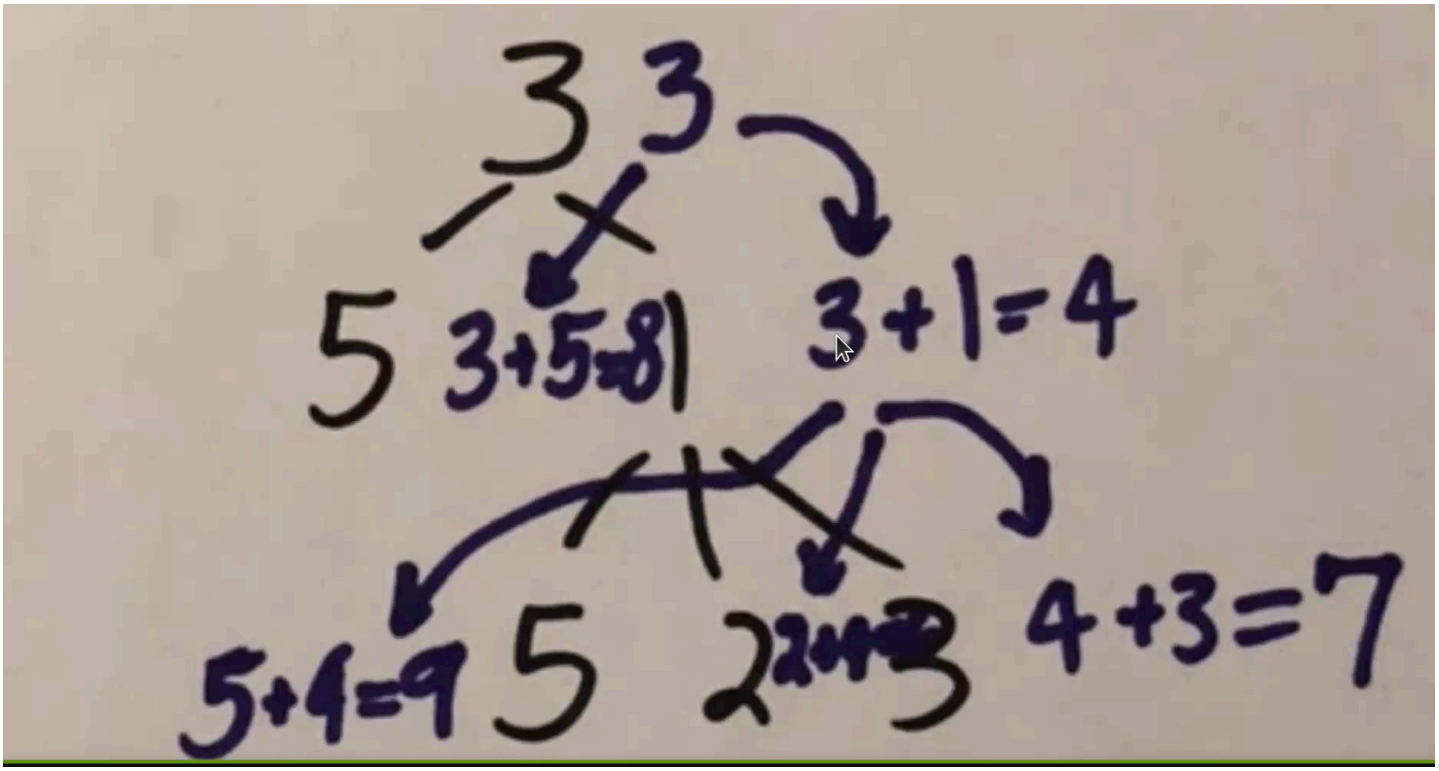
What Strategies do we have?

* Iteration
* Recursion
* Divide-and-Conquer
* Randomized Algorithms
(* Data Structures)

Trees have recursive structure.
$\Rightarrow$ Try recursion.

What's the main idea?

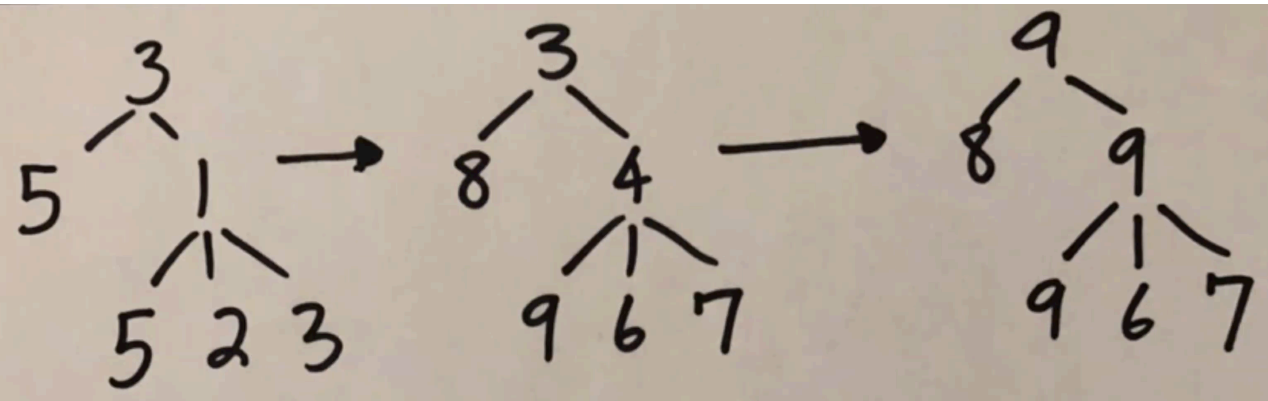
- Start at root.
- Using Path to Parent, Compute Child
- Recurse.




```
PathCalc(v, PthSoFar);
    v.PthTo ← v.val + PthSoFar;
    for each child w of v do
        PathCalc(w, v.PthTo)
    end
end
Initial call: PathCalc(T, 0)
```

4.b

b.) Compute v.PthToLf, the max of x.PthTo for all leaves x.



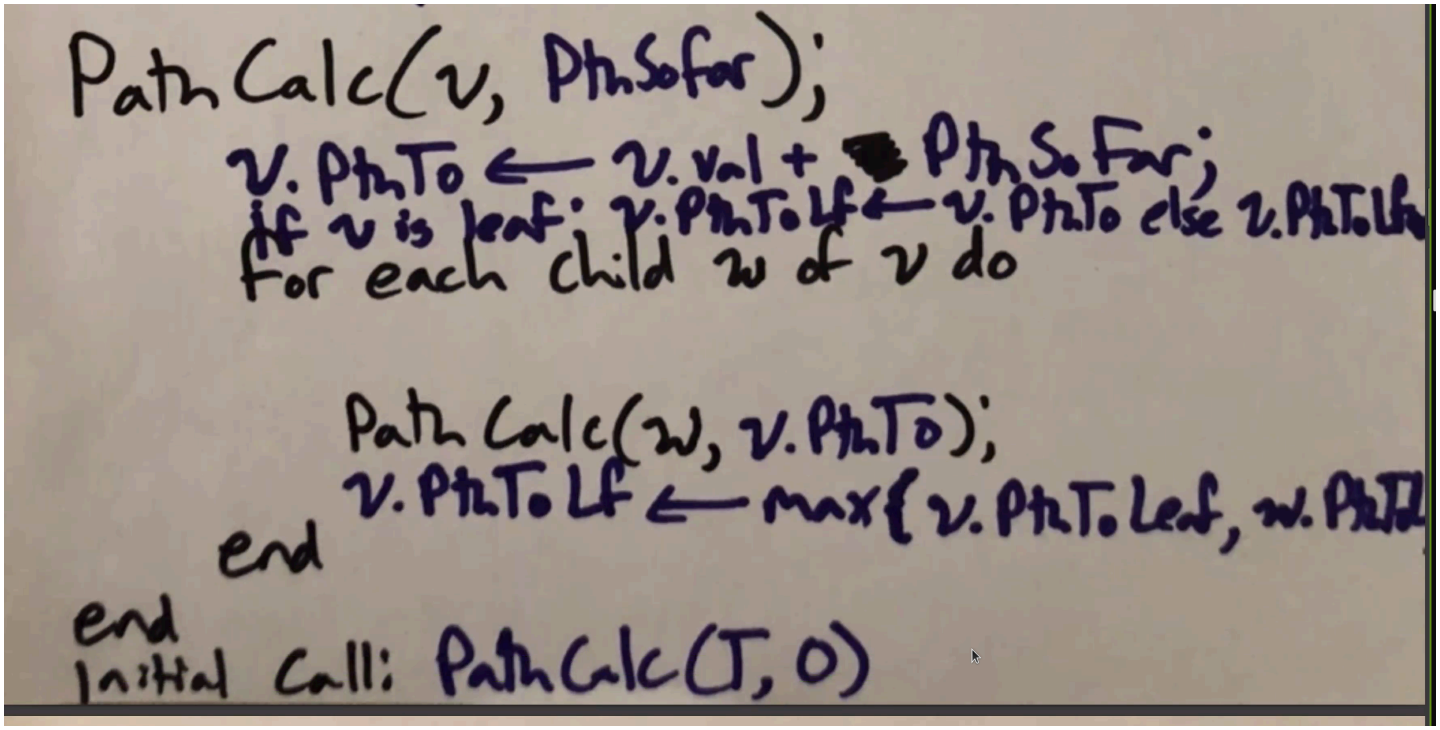
When should we compute PthToLf?

When should we compute PthToLf?

↳ What information do we need?

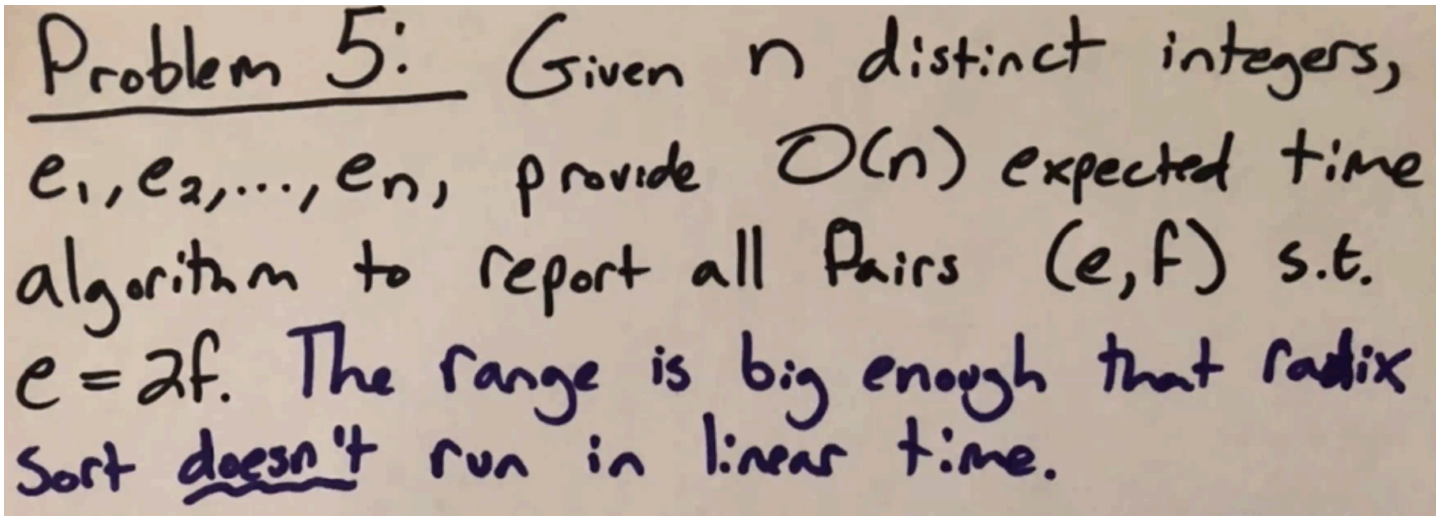
w.PthTo for all leaves of v.

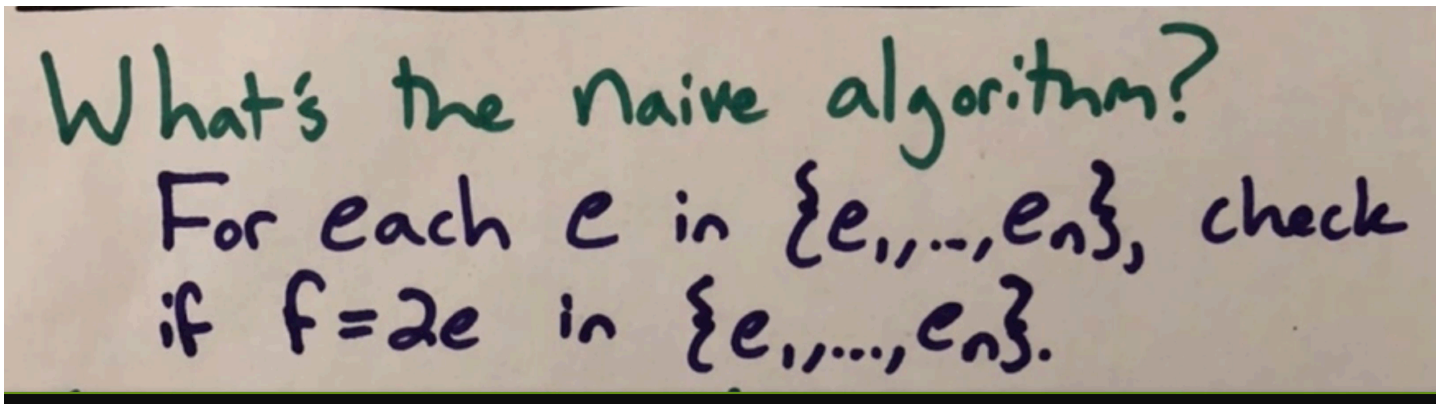
$\Rightarrow$ Compute PthToLf after PthTo.



Else v.pthToLF =0

5





Sanity Check: How many times is a pair counted?

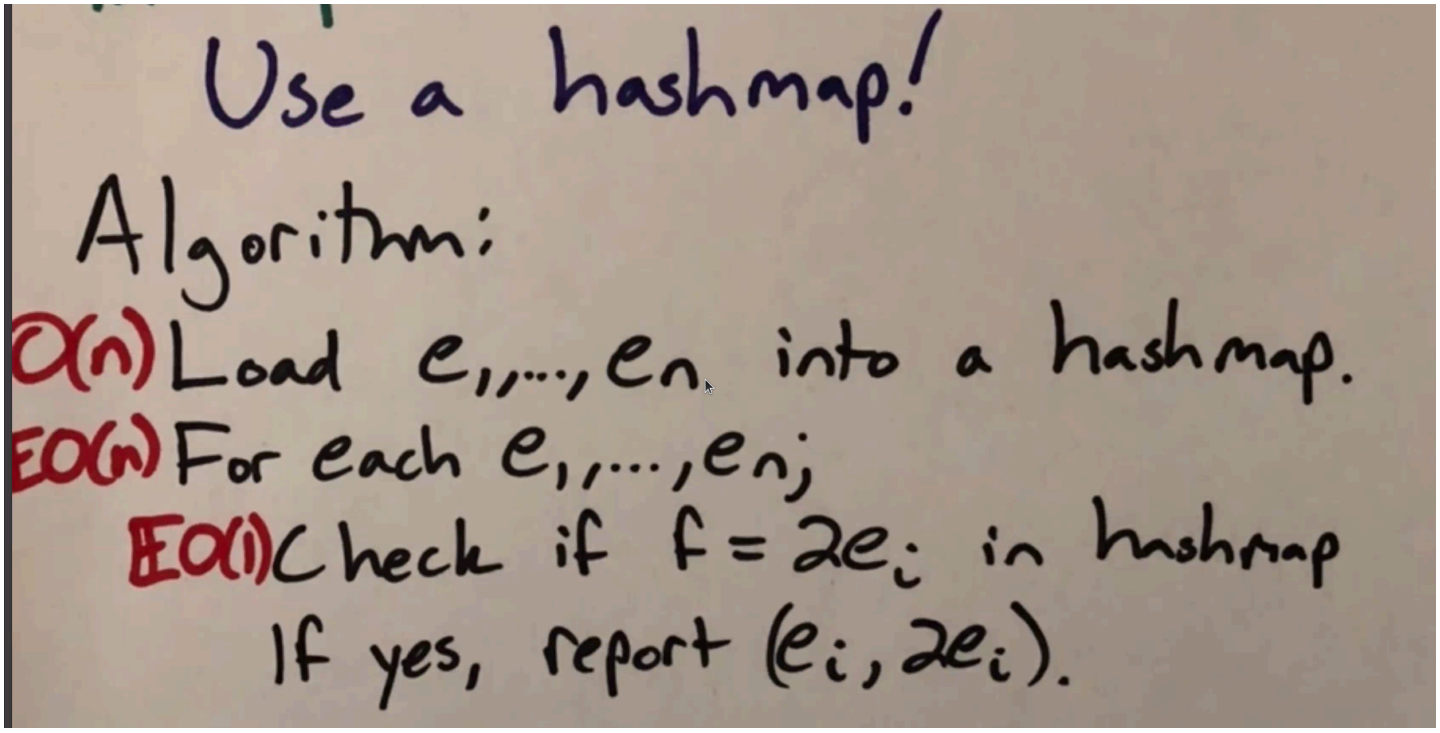
Once! ✓ (Why?)

$n\chi$ where χ is complexity of checking if f in $\{e_1, \ldots, e_n\}$.

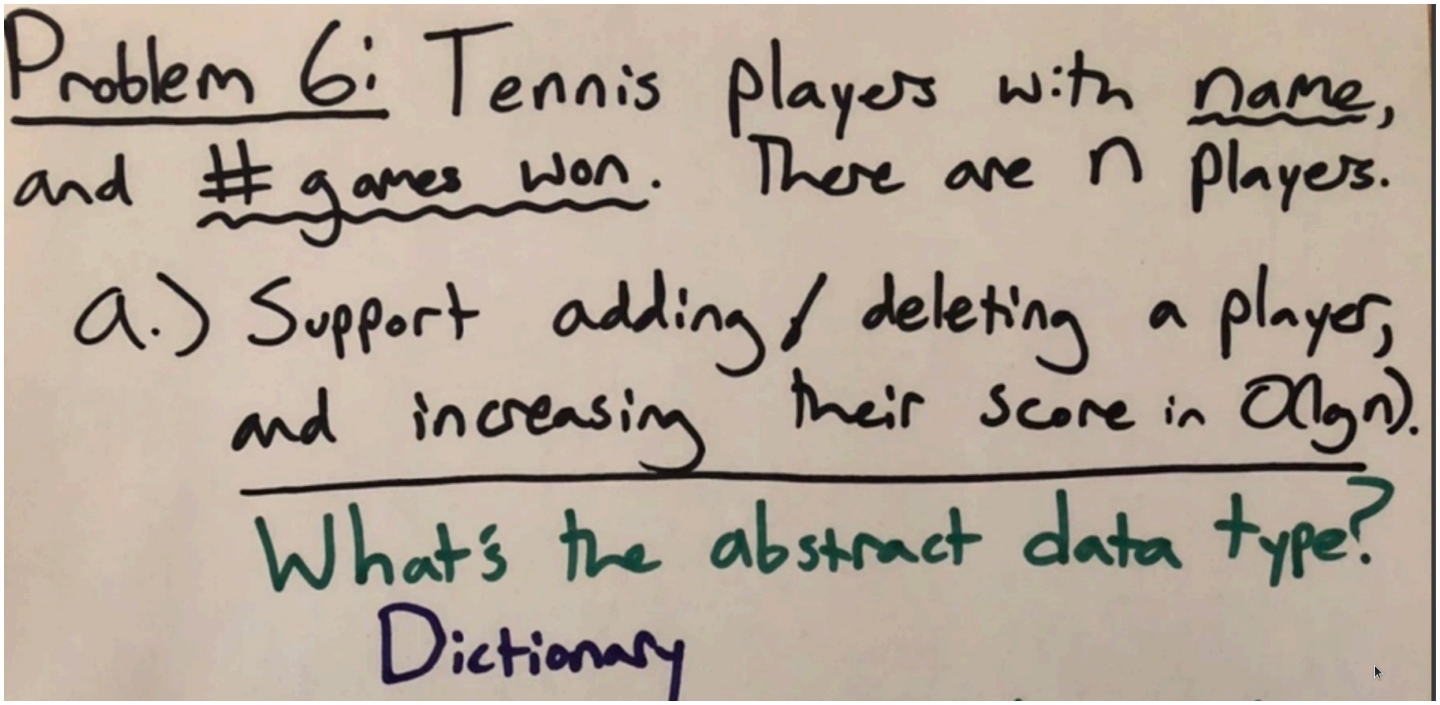
What χ can we afford?

Expected $O(1)$.

How can we check f in $\{e_1, \ldots, e_n\}$ in expected $O(1)$?


Use a hashmap!
Algorithm:
O(n) Load $e_1, \ldots, e_n$ into a hashmap.
O(n) For each $e_1, \ldots, e_n$;
O(1) Check if $f = 2e_i$ in hashmap
If yes, report $(e_i, 2e_i)$.

6


Problem 6: Tennis players with name, and # games won. There are n players.
a.) Support adding / deleting a player, and increasing their score in O(lgn).
What's the abstract data type?
Dictionary

What dictionary implementations do we have? With $O(\log n)$ time?

Hashmap? $O(1)$ expected ✗

Use a 2-3 tree with name as the key and # games won as the value.

Add ⇝ 2-3 tree add. $O(\log n)$

Delete ⇝ 2-3 tree delete. $O(\log n)$

Update Score ⇝ 2-3 tree find then overwrite Score

$O(\log n)$

b.) Support finding # players that won at least x games in $O(\log n)$.

What operations would be useful?

Count/range

Have we seen similar problems?

* 2-3 trees handout, Ex. 3

* Report fraction of green items with value less than k.
  - ↳ Recall: Augment tree with total count of items + count of green items.

* HW5, Problem 4

How many plants have heights in range $[h_1, h_2]$?

↳ Recall: Use two 2-3 trees, one for height and one for ID.

Which ideas seem applicable?

Use two 2-3 trees for the different kinds of information: name / # games won.

Use second 2-3 tree for # of

games won. ⇒ just store # players.

↳ "Compressing" the data often gives cleaner/faster algorithms.

Store sum of subtree at each node.

To maintain the data structure:

- Decrement count when deleting player.
- Increment count when adding player.
- Update (increment new/decrement old) counts when changing # games won.

To answer query: Sum subtrees as you search for x.