CS 2100

Treaps
ReCAP

- HW: next Thursday
  - Friday is review,
    following Monday is test
- Midterm content:
  through AVL trees
Last time
End of Binary Search trees

Rule:

Runtime of find (and insert/delete as a result):

\[ O(\text{height}(T)) = O(n) \]

AVL trees:

At every node \( x \), heights of \( x \)'s children must be within \( \pm 1 \).

Result: \[ \text{height} \leq 2\left\lfloor \log_2 n \right\rfloor \]
find (and others): \[ O(\log n) \]
Another recap:

**Heaps:** (not BSTs) priorities

At every node, the key value will be $\geq$ key at either child.

Example:

- Insert (60)

```
55
/   \
55   60
/  \
33  55
/ \  /  \
22 15 6 9
```

Runtimes: $O(\log n)$
Treaps: a new binary tree structure

Goal: Each node will contain a value (like a BST) and a priority (like a heap).

- BST over values
- Heap over priorities

Ex: Suppose values are names and priorities are integers. Both can be “sorted”:

- Values/names have alphabetical order
- Integers (obviously)
Names: BST ✓ heap over integers

```
   "Michael" 100
     /     
  "Helen" 5.0  "Tom" 88
    |       /  
  "Erin" 28  "Jesse" 44
     |       |
  "Frank" 15
```

Sam
In BST order, but violates heap!

Now: insert ("Sam", 101)
In heaps, we "bubbled" up. Can we do that here?

```

```

"Michael"

100

"Helen"

50

"Erin"

28

"Frank"

15

"Jesse"

44

"Tom"

88

"Sam"

not BST
Well - can't violate BST!
What did we do to move things around in AVL trees?

Pivot

```
    "Michael"
     /
   /   
"Helen" "Tom"
/     /
50   88
```

```
"Erin" "Jesse"
/     /
28     44
```

```
"Frank" "Sam"
/     /
15     [0]
```
Result of pivoting: insert \((x, 2)\)

Clearly, we're still a BST!
Can \(T1, T2,\) or \(T3\) be non-heaps after this?
(Note: just inserted \(x\), so what are \(T1 + T2\)?)

\(T3\): \(T3\) had priorities all \(\leq y's\) priority
\(\rightarrow\) \(y\) is still above, so still in heap order
Same for other case: $\text{insert}(x, 2)$

pivot ($y$)

Still happy!
Result:

Insert (val, key):
Run BST insert (on data)
Save its location, it
while (it.priority > it's parent's priority)
pivot(it)
and
it != root()
Example: Insert:
(P, S), (5, 9), (E, 7), (4, 1),
(W, 13), (D, 8), (J, 2), (K, 4), (P, 11)

Diagram of a data structure with nodes labeled with coordinates. The diagram illustrates the process of inserting and pivoting through a set of nodes.
Observation:
Take a step back. What must the root be?

Example:
Insert:
(P, 11), (S, 9), (E, 7), (H, 1),
(W, 13), (D, 8), (J, 2), (K, 4), (P, 11)
Observation

Troops are unique
(BSTs + heaps are not)
(at AVLs)

→ This is like giving 2 traversals:

Next:
- remove
- run times
 (randomized)