Trees/BSTs

- Insert(78) $\Theta(h)$
- Delete(50) — swap with 61 and delete $\Theta(h)$
- Successor $\Theta(h)$
- In-order walk $\Theta(n)$
- And: Constructing a BST from a sorted list... (also $\Theta(n)$)

Heaps

- Nearly-Complete Binary Tree

B. Extract-Min* $\Theta(lg n)$ *for min-heaps
A. Heapify $\Theta(lg n)$
C. Build-Heap $\Theta(n)$
D. Priority Queue
   - Decrease-Key*

Array Representation:

```
90 74 82 61 2...
0 1 2 3 4...
```

- Parent[i] = $\left\lfloor \frac{i-1}{2} \right\rfloor$
- Child[i]: Left[i] = 2i + 1 Right[i] = 2i + 2

AVL Trees: Balanced BSTs

1 $\geq |h(left) - h(right)|$

Rebalancing only requires rotations.

Easy Case:

Hard Case:

The above won't work... we need two rotations.

Then it becomes the easy case...
HASHING/HASH TABLES

A. Collision Resolution

B. Insert

C. Delete

D. Lookup

E. Rolling hashes

Amortized Analysis

Hash table (ignore deletions) — want to keep $\alpha < \frac{4}{5}$

On insert, if $\alpha \geq \frac{4}{5}$, allocate a new table of $2m$ size and rehash everything

$\rightarrow O(m+n) = O(n)$ time to resize

$O(1)$ otherwise

In a series of $n$ insertions:
- Suppose resizes occur at $k, 2k, 4k, \ldots$
- Then cost of insertions is $O(n + (k + 2k + \ldots + n)) = O(n + \left(\frac{n}{2} + \frac{n}{4} + \ldots + n\right))$
  $= O(n + (2n))$
  $= O(3n) = O(n)$

Thus each insert is $\frac{O(n)}{n} = O(1)$ time