Two properties of a max-heap?
1) Any node is greater or equal to children
2) Almost-complete: added left-to-right

min-heap?
(smaller or equal)

```
in+ size
left(i) = 2*i
right(i) = 2*i + 1
parent(i) = i/2
```
Inserting on a heap

Three Steps
1) Add to next spot
2) Increment size
3) Up-Heapify

Insert 5
Insert 55

n = # of items in the heap
Insert = O(\log n)

# leaves \( \sqrt{n} \)

Complete Heap/Tree of height h
has \( 2^h - 1 \) nodes
and \( 2^h - 1 \) leaf nodes
Removing from a heap

Three Steps
1) Swap root w/ last
2) Decrement size
3) Down-Heapify

Remove max
Remove max

Remove = O(lgn)
Naive Heapsort

Steps: 1) Make empty heap
       2) Insert everything
       3) Remove Everything

\[ \footnotesize \begin{array}{c}
0(n) & 18 & 2 & 27 & 19 & 18 \\
\Theta(n) & @ & @ & @ & @ & @ \\
\end{array} \]

\[ \footnotesize O(n \lg n) - \text{In} \]
\[ \footnotesize O(n \lg n) - \text{Out} \]

Space? \( \Theta(n) \)

\[ \footnotesize 1 \ 2 \ 8 \ 18 \ 19 \ 27 \]
Smarter Heapsort

Let's fix the insert part: Down-heapify n times from the bottom up.

18 2 27 19 1 8 10 32 11

\[ T(n) = \sum_{l=0}^{\log_2 n - 1} l \cdot \frac{n}{2^l} \]

for \(|x| < 1\)

\[ \sum_{k=0}^{\infty} k \cdot x^k = \frac{x}{(1-x)^2} \]

Improves asymptotic bound? No, not of Heapsort

Improves space requirement? Yes,
Benefits of Heapsort

1) Worst-case $\Theta(n \log n)$

2) Smaller Space Requirement

3) Heaps have PQ application
Lower bounds

- Comparison Sorts: compares values of its elements with each other
- Decision trees: want to sort the list \([a, b, c]\)

\[ \begin{array}{c}
\text{a} \ ? \ \text{b} \\
\text{b} \ ? \ \text{c} \\
\text{a} \ ? \ \text{c} \\
\end{array} \]

\[ \begin{array}{c}
[a \ b \ c] \leq [a \ c \ b] \leq [a \ b \ c] \\
\text{[a \ c \ b]} \leq [c \ a \ b] \leq [b \ a \ c] \leq [b \ c \ a] \\
\end{array} \]

\[ \begin{array}{c}
\text{n elements} \\
\log(n!) \\
\text{n! permutations} \\
\lg(n!) = \Theta(n \lg n) \\
\end{array} \]