# Heaps

# Motivating Problems

1. Finding the min
2. Finding the min with insertions
3. Finding the min with insertions and deletions

# Motivating Problems

1. Finding the min
2. Finding the min with insertions
3. Finding the min with insertions and deletions

# Motivating Problems

1. Finding the min
2. Finding the min with insertions
3. Finding the min with insertions and deletions

# Heaps

Solve using a binary tree data structure

Heap invariant property: parent is smaller than (or equal to) both children

(This is specifically a min-heap. The max-heap invariant flips the inequality.)

# Insertion

1. insert at bottom of tree
2. re-establish invariant by pulling value up

# Insertion

1. insert at bottom of tree
2. re-establish invariant by pulling value up

# Extract Min

1. remove root
2. move last node into root
3. re-establish invariant by pushing value down

heapsort

# Extract Min

1. remove root
2. move last node into root
3. re-establish invariant by pushing value down

heapsort

# Array Implementation of Tree

Rules for array index:

parent of $i = \lfloor \frac{i-1}{2} \rfloor$

left child of $i = 2i + 1$

right child of $i = 2i + 2$

automatically balanced!

# Pull Up – Insertion Helper

assume heap except at element $i$, $A[i]$ might be too small

pullup($i$)

**if** $A[i] < A[parent_i]$ **then**
    swap $A[i]$ with $A[parent_i]$
    pullup($parent_i$)

invariant: initialization, maintenance, termination

# Push Down – Extraction Helper

assume heap except at element $i$, $A[i]$ might be too large

pushdown($i$)

$min_i \leftarrow$ index of smallest among $i$ and valid children of $i$
**if** $min_i \neq i$ **then**
  swap $A[i]$ with $A[min_i]$
  pushdown($min_i$)

invariant: initialization, maintenance, terminination

# Analysis



Correctness

What is the space complexity?

What is the time complexity?

# Construction

Given an array, how do we construct a heap?

Can we do better than $O(n \lg n)$ time?

bottom up creation:

$$\textbf{for } i \text{ from } \frac{length}{2} - 1 \text{ to } 0 \textbf{ do}$$
$$\quad \text{pushdown}(i)$$

what is this time complexity?

# Construction

Given an array, how do we construct a heap?

Can we do better than $O(n \lg n)$ time?

bottom up creation:

**for** $i$ from $\frac{length}{2} - 1$ to 0 **do**
  pushdown($i$)

what is this time complexity?

# Construction

Given an array, how do we construct a heap?

Can we do better than $O(n \lg n)$ time?

bottom up creation:

**for** $i$ from $\frac{length}{2} - 1$ to 0 **do**
pushdown($i$)

what is this time complexity?

# Heap Creation Time Complexity

*height* of a node is maximum distance to a leaf
$count_x$ is the number of nodes in a tree with height $x$

$$\sum_{h=0}^{\lg n} (O(h) \times count_h)$$

There are $\frac{n}{2^{h+1}}$ nodes with height $h$

$$\sum_{h=0}^{\lg n} O(h) \frac{n}{2^{h+1}} = O\left(n \sum_{h=0}^{\lg n} \frac{h}{2^{h+1}}\right)$$

# More Time Complexity

$$\sum_{h=0}^{\infty} \frac{h}{2^h} = 2, \text{ so } \sum_{h=0}^{\lg n} \frac{h}{2^h} < 2$$

$$O\left(n \sum_{h=0}^{\lg n} \frac{h}{2^{h+1}}\right) = O\left(n \sum_{h=0}^{\infty} \frac{h}{2^h}\right)$$

$$= O(n)$$

# Sizing The Array

resize by doubling array size

'amortized' analysis: the 'accounting method'

1. start with array half full, 0 accrued steps
2. each insertion takes 3 steps
   a. insert self now
   b. move self when array full
   c. move an existing element when array full
3. when array is full, we have 1 move step for each item
4. after move, array is half full, 0 accrued steps

# Amortization

'amortized' analysis: the 'aggregate method'
Let $c_i = i$ if $i-1$ is a power of 2, 1 otherwise

$$\begin{aligned}\sum_{i=1}^{n} c_i \leq n + \sum_{j=0}^{\lg n} 2^j \\ < n + 2n \\ < 3n\end{aligned}$$

# Sorting

# Lower Bounds

How many possible outputs are there for arranging $n$ items?

Binary decision tree with $n!$ leaves has height at least $lg(n!)$

Stirling approximation: $n! = \sqrt{2\pi n}(\frac{n}{e})^n(1+\Theta(\frac{1}{n}))$

so:

$$\lg(n!) = \lg(\sqrt{2\pi n}(\tfrac{n}{e})^n(1+\Theta(\tfrac{1}{n})))$$
$$= \lg\sqrt{2\pi} + \lg\sqrt{n} + \lg\left((\tfrac{n}{e})^n\right) + \lg\left(1+\Theta(\tfrac{1}{n})\right)$$
$$= \Theta\left(\lg\sqrt{n} + n\lg(\tfrac{n}{e}) + \lg\left(1+\Theta(\tfrac{1}{n})\right)\right)$$
$$= \Theta\left(n\lg n\right)$$

so comparison-based sorting takes $\Omega(n\lg n)$ time

# Lower Bounds

How many possible outputs are there for arranging $n$ items?
Binary decision tree with $n!$ leaves
Binary decision tree with $n!$ leaves has height at least $lg(n!)$
Stirling approximation: $n! = \sqrt{2\pi n}(\frac{n}{e})^n(1+\Theta(\frac{1}{n}))$
so:

$$\lg(n!) = \lg(\sqrt{2\pi n}(\tfrac{n}{e})^n(1+\Theta(\tfrac{1}{n})))$$
$$= \lg\sqrt{2\pi} + \lg\sqrt{n} + \lg\left((\tfrac{n}{e})^n\right) + \lg\left(1+\Theta(\tfrac{1}{n})\right)$$
$$= \Theta\left(\lg\sqrt{n} + n\lg(\tfrac{n}{e}) + \lg\left(1+\Theta(\tfrac{1}{n})\right)\right)$$
$$= \Theta\left(n\lg n\right)$$

so comparison-based sorting takes $\Omega(n \lg n)$ time

# Lower Bounds

How many possible outputs are there for arranging $n$ items?
Binary decision tree with $n!$ leaves has height at least $lg(n!)$
Stirling approximation: $n! = \sqrt{2\pi n}(\frac{n}{e})^n(1+\Theta(\frac{1}{n}))$
so:

$$\lg(n!) = \lg(\sqrt{2\pi n}(\frac{n}{e})^n(1+\Theta(\frac{1}{n})))$$
$$= \lg\sqrt{2\pi} + \lg\sqrt{n} + \lg\left((\frac{n}{e})^n\right) + \lg\left(1+\Theta(\frac{1}{n})\right)$$
$$= \Theta\left(\lg\sqrt{n} + n\lg(\frac{n}{e}) + \lg\left(1+\Theta(\frac{1}{n})\right)\right)$$
$$= \Theta\left(n\lg n\right)$$

so comparison-based sorting takes $\Omega(n\lg n)$ time

# Lower Bounds

How many possible outputs are there for arranging $n$ items?
Binary decision tree with $n!$ leaves has height at least $lg(n!)$
Stirling approximation: $n! = \sqrt{2\pi n}(\frac{n}{e})^n(1+\Theta(\frac{1}{n}))$

so:

$$\begin{aligned}
\lg(n!) &= \lg(\sqrt{2\pi n}(\tfrac{n}{e})^n(1+\Theta(\tfrac{1}{n}))) \\
&= \lg\sqrt{2\pi} + \lg\sqrt{n} + \lg\left((\tfrac{n}{e})^n\right) + \lg\left(1+\Theta(\tfrac{1}{n})\right) \\
&= \Theta\left(\lg\sqrt{n} + n\lg(\tfrac{n}{e}) + \lg\left(1+\Theta(\tfrac{1}{n})\right)\right) \\
&= \Theta\left(n\lg n\right)
\end{aligned}$$

so comparison-based sorting takes $\Omega(n\lg n)$ time

# Lower Bounds

How many possible outputs are there for arranging $n$ items?
Binary decision tree with $n!$ leaves has height at least $lg(n!)$
Stirling approximation: $n! = \sqrt{2\pi n}(\frac{n}{e})^n(1+\Theta(\frac{1}{n}))$
so:

$$\lg(n!) = \lg(\sqrt{2\pi n}(\tfrac{n}{e})^n(1+\Theta(\tfrac{1}{n})))$$
$$= \lg\sqrt{2\pi} + \lg\sqrt{n} + \lg\left((\tfrac{n}{e})^n\right) + \lg\left(1+\Theta(\tfrac{1}{n})\right)$$
$$= \Theta\left(\lg\sqrt{n} + n\lg(\tfrac{n}{e}) + \lg\left(1+\Theta(\tfrac{1}{n})\right)\right)$$
$$= \Theta\left(n\lg n\right)$$

so comparison-based sorting takes $\Omega(n\lg n)$ time

# Lower Bounds

How many possible outputs are there for arranging $n$ items?
Binary decision tree with $n!$ leaves has height at least $lg(n!)$
Stirling approximation: $n! = \sqrt{2\pi n}(\frac{n}{e})^n(1+\Theta(\frac{1}{n}))$
so:

$$\lg(n!) = \lg(\sqrt{2\pi n}(\tfrac{n}{e})^n(1+\Theta(\tfrac{1}{n})))$$
$$= \lg\sqrt{2\pi} + \lg\sqrt{n} + \lg\left((\tfrac{n}{e})^n\right) + \lg\left(1+\Theta(\tfrac{1}{n})\right)$$
$$= \Theta\left(\lg\sqrt{n} + n\lg(\tfrac{n}{e}) + \lg\left(1+\Theta(\tfrac{1}{n})\right)\right)$$
$$= \Theta\left(n\lg n\right)$$

so comparison-based sorting takes $\Omega(n\lg n)$ time

# Lower Bounds

How many possible outputs are there for arranging $n$ items?
Binary decision tree with $n!$ leaves has height at least $lg(n!)$
Stirling approximation: $n! = \sqrt{2\pi n}(\frac{n}{e})^n(1+\Theta(\frac{1}{n}))$
so:

$$\lg(n!) = \lg(\sqrt{2\pi n}(\tfrac{n}{e})^n(1+\Theta(\tfrac{1}{n})))$$
$$= \lg\sqrt{2\pi} + \lg\sqrt{n} + \lg\left((\tfrac{n}{e})^n\right) + \lg\left(1+\Theta(\tfrac{1}{n})\right)$$
$$= \Theta\left(\lg\sqrt{n} + n\lg(\tfrac{n}{e}) + \lg\left(1+\Theta(\tfrac{1}{n})\right)\right)$$
$$= \Theta\left(n\lg n\right)$$

so comparison-based sorting takes $\Omega(n\lg n)$ time

# Lower Bounds

How many possible outputs are there for arranging $n$ items?

Binary decision tree with $n!$ leaves has height at least $lg(n!)$

Stirling approximation: $n! = \sqrt{2\pi n}(\frac{n}{e})^n(1+\Theta(\frac{1}{n}))$

so:

$$\begin{aligned}
\lg(n!) &= \lg(\sqrt{2\pi n}(\tfrac{n}{e})^n(1+\Theta(\tfrac{1}{n}))) \\
&= \lg\sqrt{2\pi} + \lg\sqrt{n} + \lg\left((\tfrac{n}{e})^n\right) + \lg\left(1+\Theta(\tfrac{1}{n})\right) \\
&= \Theta\left(\lg\sqrt{n} + n\lg(\tfrac{n}{e}) + \lg\left(1+\Theta(\tfrac{1}{n})\right)\right) \\
&= \Theta\left(n\lg n\right)
\end{aligned}$$

so comparison-based sorting takes $\Omega(n\lg n)$ time

# Lower Bounds

How many possible outputs are there for arranging $n$ items?

Binary decision tree with $n!$ leaves has height at least $lg(n!)$

Stirling approximation: $n! = \sqrt{2\pi n}(\frac{n}{e})^n(1+\Theta(\frac{1}{n}))$

so:

$$\lg(n!) = \lg(\sqrt{2\pi n}(\tfrac{n}{e})^n(1+\Theta(\tfrac{1}{n})))$$
$$= \lg\sqrt{2\pi} + \lg\sqrt{n} + \lg\left((\tfrac{n}{e})^n\right) + \lg\left(1+\Theta(\tfrac{1}{n})\right)$$
$$= \Theta\left(\lg\sqrt{n} + n\lg(\tfrac{n}{e}) + \lg\left(1+\Theta(\tfrac{1}{n})\right)\right)$$
$$= \Theta\left(n\lg n\right)$$

so comparison-based sorting takes $\Omega(n\lg n)$ time