binary tree
up to \(2^k\) nodes on level \(k\)

\[1 = 2^0\]
\[2 = 2^1\]
\[4 = 2^2\]
\[8 = 2^3\]

implies up to \(2 \times 2^k = 2^{k+1}\) nodes
on level \(k+1\)

Full binary of height \(h\)
has \(2^h\) leaves

Number of internal nodes is
\[2^h - 1 = 2^0 + 2^1 + 2^2 + 2^3 + \cdots + 2^{h-1}\]

Total number of nodes is
\[2^h + 2^h - 1 = 2^{h+1} - 1\]
Full binary tree of height $h$

has $2^{h+1} - 1 = n$ nodes

<table>
<thead>
<tr>
<th># of nodes</th>
<th>work</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\frac{n}{2^{h+1}}$</td>
<td>$2^{h} \times 1$</td>
</tr>
<tr>
<td>$\frac{n}{2^h}$</td>
<td>$2 \times n$</td>
</tr>
<tr>
<td>$\frac{n}{4}$</td>
<td>$4 \times n$</td>
</tr>
<tr>
<td>$\frac{n}{8}$</td>
<td>$6 \times n$</td>
</tr>
</tbody>
</table>

$2 \times \frac{n}{4} + 4 \times \frac{n}{8} + 6 \times \frac{n}{16} + 8 \times \frac{n}{32} + 10 \times \frac{n}{64} + 12 \times \frac{n}{128}$

like a decreasing geometric series

$1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{6} + \ldots = 2$

$1 + \frac{2}{3} + \frac{4}{9} + \frac{8}{27} + \ldots \approx 3$ maybe

so the sum of work is $n \times$ some constant

careful analysis gets $2n$
Assume \( n = A\. \text{length} \)

**BUILD-MAX-HEAP(A)**

1. \( A\.\text{heap-size} = A\.\text{length} \) \( \text{// need to set heap-size} \)
2. \( \text{for } i = \lfloor A\.\text{length}/2 \rfloor \text{ downto } 1 \)
3. \( \text{MAX-HEAPIFY}(A, i) \)

\( A[1..n] \) are max-heaps

at end \( A[1..n] \) is max-heap

Assume \( n = A\.\text{length} \)

**HEAPSORT(A)**

1. \( \text{BUILD-MAX-HEAP}(A) \) \( A[1..n] \) is a max-heap
2. \( \text{for } i = A\.\text{length} \text{ downto } 2 \)
3. \( \text{exchange } A[1] \text{ with } A[i] \) \( A[1..i] \) is a max-heap
4. \( A\.\text{heap-size} = A\.\text{heap-size} - 1 \) \( A[1..i] \) are the largest \( n-i \) numbers in sorted order
5. \( \text{MAX-HEAPIFY}(A, 1) \)

\( A[1..n] \) are the largest numbers sorted

\( A[1..i-1] \) is a max-heap
Build-Map-Heap is $\Theta(n)$
from $2 \times \frac{n}{2}$ to $2 \times n$ comparisons

Map-Heapify $(A, 1)$ is $\Theta(\lg n)$? no, might not be $\Omega(1)$
size of max-heap

Call Map-Heapify $(A, 1)$ $n-1$ times
size of max-heap on these calls changes:

\[ n-1, n-2, \ldots, 4, 3, 2, 1 \]

About half of these are $\geq \frac{n}{2}$

and $\lg \left(\frac{n}{2}\right) = (\lg n) - (\lg 2) = (\lg n) - 1$

Worst-case total $O(n \lg n)$

Best-case total $\Omega(2(n))$ not really true

Actually $\Theta(n \lg n)$ on average