Fix. Continue from last time....

Stack:

(after 2nd)

(roll to 3rd)

Topological sort
Recall DFS has prototype:

```c
void DFS(Graph G, list S);
```
call it like this

\[ S: 1 \ 2 \ \ldots \ n \]

\[ \text{DFS}(G, S) \]

\[ S: \text{dec. finish times} \]

\[ \text{DFS}(G^T, S) \]

To find strong components in topologically sorted order, step through \( S \) from bottom to top.
Appendix E.5

A rooted tree is a tree with a distinguished vertex called \textit{root}.

\textbf{Ex.}

\texttt{height(x)} = 2

\texttt{depth} = 0

\texttt{height(T)} = 3

Subtree rooted at \texttt{x}

\texttt{depth of a node} = dist. to root.

\texttt{children}: adjacent \& 1 further away

\texttt{parent} \hspace{1em}: 1 closer to root

\texttt{leaves} \hspace{1em}: no children

\texttt{internal nodes} \hspace{1em}: non-leaves.
height of a tree: depth of deepest leaf.

height of a node: height of subtree rooted at that node.

A Binary Tree is a rooted tree where each node has at most 2 children, called left & right.

Ex,

- Binary Tree
- Binary Tree
Recursive definition of height of a binary tree. Let $T$ be a binary tree, with $n$ nodes.

$$h(T) = \begin{cases} \infty & n = 0 \\ 0 & n = 1 \\ 1 + \max(h(L), h(R)) & n \geq 2 \end{cases}$$

Where $L$ is left subtree, $R$ is right subtree.

Ex.

\[ \begin{array}{c}
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\end{array} \]
Exercise:

Prove \( h(T) = \lceil \log n \rceil \).

Define

A complete binary tree (CBT) is a B.T. in which all leaves have the same depth & all internal nodes have 2 children.

Example:

```
  o
 / \ /
 o  o o
/ \ / \ / \
o  o o  o  o
```

Depth:

0 1 2 3 4
Note: Let $n = \# \text{nodes}, h = \text{height}(T)$:

$$n = \sum_{d=0}^{h} 2^d = \frac{2^{h+1} - 1}{2 - 1} = 2^{h+1} - 1$$

So $h+1 = \log_2(n+1)$

$$\therefore h = \log_2(n+1) - 1$$

**Define** An Almost Complete Binary Tree (ACBT) is a B.T. that is filled at every level, except possibly the deepest, where it is filled from left to right.
Ex

Let $n$ be a node, $h$ be height(T).

Note $h$ must satisfy:

$2^h - 1 < n \leq 2^{h+1} - 1$

$2^h \leq n < 2^{h+1}$

$h \leq \lfloor \log n \rfloor < h + 1$

$\therefore h = \lceil \log n \rceil$
6.1 Heaps

A **binary heap** is an array object used to represent an ACBT.

**Example:**

![Binary heap diagram]

**Array:**

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>17</td>
<td>20</td>
<td>11</td>
<td>16</td>
<td>15</td>
<td>12</td>
<td>5</td>
<td>8</td>
<td>10</td>
<td>2</td>
<td>6</td>
</tr>
</tbody>
</table>

Attributes: `length[A]` `heapSize[A]`
helper functions:

\[
\begin{cases}
\text{parent}(i) = \left\lfloor \frac{i}{2} \right\rfloor & (\text{for } i \geq 2) \\
\text{left}(i) = 2i \\
\text{right}(i) = 2i + 1
\end{cases}
\]

Two kinds of heaps: \underline{max-heap}, \underline{min-heap}.

- max-heap property
  \[ A[\text{parent}(i)] \geq A[i] \]

- min-heap property
  \[ A[\text{parent}(i)] \leq A[i] \]

Assume this from now on.