Data Structure

Binary Heap

Structure

\[
\begin{align*}
0, & \\
n=1, & \\
\end{align*}
\]

Rep.

Property

HEAPS ORDERED

the priority of each node is as great as that of its descendants
Array Representation

If they exist

- Parent(i) = \left\lfloor \frac{i-1}{2} \right\rfloor
- Left(i) = 2i + 1
- Right(i) = 2i + 2

Size

Space between n references

Heap

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>n-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td></td>
</tr>
</tbody>
</table>
Inserting an element

1. add new element to next open slot in the array
2. Swap new element with parent until the parent is at least as large or reach root

$O(\log n)$ time
Finding and Extracting (Removing) Max $O(\log n)$

1. remember root (store in a var)
2. replace root by last element
   $\text{heap}[0] = \text{heap}[\text{n-1}]$
   $\text{n} = \text{n} - 1$
3. Look at two children
   if HEAPORDERED is violated, swap
   repeat

---

TAF

max in root (element 0)
Increase Priority

\[ O(\log n) \]

tracker \( t = \text{p.q.}\text{.add\_tracked}(r) \)

\[ t.\text{Increase Priority}(z) \]

change element

swap with parent until its \text{HEAPOrdered} is restored
Fix Upward (i)

Fig 25.3

Requires:
Only possible violation of Heap Ordered is between i and its parent

Swap

continue
Decrease Priority

tracker $t$

t. decreasePriority($b$)

Like extractMax on the subtree root at the node being changed
$O(\log n)$

**Fix Downward(i)**

often called heapify

**Fig 25.4**

possible violations

Requires:
Only Violation of Heap Ordered between node $i$ and its children

continue here

selected larger child & swap $i$ with that child
Deletion

If replacement is larger, fix upward.
Otherwise, fix downward.

Example where heap[\(n-1\)] is larger than what it replaces.
Overview of binary heap

Advantage - very space efficient
very simple with low constants
hidden in asymptotic notation

Drawbacks
merging two binary heaps takes linear time
increasing priority (through tracker) takes logarithmic time
Merging two priority queues.

Given an arbitrary array $a[0]...a[n-1]$. Convert to a binary heap in linear time.

For (int $i=n-1$, $i \geq 0$; $i--$)

$\text{FixDownward}(i)$

$O(n)$

Successive inserts $O(n \log n)$