Lecture 30

Heaps 3

- Sum of depths
- Sum of heights
- $O(n)$ algorithm for building a heap

Binary trees should not be drawn like this:

They should be drawn like this:

because level $L$ has $2^L$ nodes.

Consider a complete binary tree of height $h$, with all levels full.

Level $L$ has $2^L$ nodes: $2^0, \ldots, 2^{L-1}$

Number of nodes: $N = 2^{h+1} - 1$

Height of tree: $h = \log(N+1) - 1$

What is sum of depths of nodes?

**Note:** average depth = $\frac{1}{N}$ (sum of depths)

What is sum of heights of nodes?

**Note:** average height = $\frac{1}{N}$ (sum of heights)

Inserting $n$ nodes into a binary tree:

Best case (for number of nodes traversed): sum of depths.
\[
\sum_{l=0}^{h} l \cdot 2^l = 2 \sum_{l=0}^{h} l \cdot 2^{l-1}, \quad x=2
\]

= \[2 \sum_{l=0}^{h} \frac{d}{dx} x^l, \quad x=2\]

= \[2 \frac{d}{dx} \sum_{l=0}^{h} x^l\]

= \[2 \frac{d}{dx} \left( \frac{x^{h+1} - 1}{x-1} \right)\]

and substitute \( x=2 \)

---

**Sum of depths**

= \((h-1)2^{h+1} + 2\)

= \((\log(n+1) - 2) \cdot (n+1) + 2\)

Which is \( O(n \log n) \) and \( \Omega(n \log n) \)

---

**Generalizing to complete \( k \)-ary trees**

(Recall Assignment 3 tries)

number of nodes = \[\sum_{l=0}^{h} k^l\]

sum of depths = \[\sum_{l=0}^{h} k^l \cdot l\]

Use same trick \( x=k \).

\[\sum \text{of depths is } O(n \log n) \quad \Omega(n \log n)\]

---

**Parent-child index relations**

\[\begin{aligned}
&1 \\
&2 \\
&3 \\
&4 \\
&5 \\
&6 \\
&7 \\
&8 \\
&9 \\
&10 \\
&11 \\
&12 \\
&13 \\
&14 \\
&15 \\
&16 \\
&17 \\
&18 \\
&19 \\
&20 \\
\end{aligned}\]

\[\begin{aligned}
\text{parent} &= \text{child}/2 \\
\text{left child} &= 2 \times \text{parent} \\
\text{right child} &= 2 \times \text{parent} + 1
\end{aligned}\]

---

**Claim:** Left child of \( i \) is \( 2i \).

**Proof:** Let \( i \) be the \( m \)th node in level \( l \).

Then \( i = 2^l + m - 1 \).

But the last node of level \( l \) is \( 2^{l+1} - 1 \).

Thus, the children of nodes \( 2^l, \ldots, i \) are \( 2^{l+1}, \ldots, 2^{l+1} + 2m - 1 \).

Thus, the left child of \( i \) is \( 2^{l+1} + 2m - 2 \).

Thus \( 2 \times i \).
Last lecture: buildHeap()

for i = 1 to n
  upHeap(i)

worst case: total number of swaps t(n)

\[ t(n) < n \log n \]

A second way to build a heap

for i = \( \lfloor n/2 \rfloor \) down to 1
  downHeap(a, n, i)

downHeap(a, n, k)?
i = k
  while (2 * i ≤ n)
    child \( = 2 * i \)  // left child
    if (child < n)
      if a[child+1] < a[child]
        child += 1
      end if
    end if
    if a[child] < a[i] {
      swap keys (a, i, child)
    }
  end while

Example

1 2

Summary

Most nodes are near level \( h \)

for i = 1 to n
  upHeap(a, n, i)
end for

for i = n to 1
  downHeap(a, n, i)
end for

\( O(n \log n) \) \hspace{1cm} \( O(n) \)

faster!