Announcements (4/16/08)

- Homework due Friday at start of class
- Graded homework and projects returned in section tomorrow

\[ \text{Insert (worst case): } O(\log_d n) \]
\[ \text{DeleteMin (worst case): } O(d \log_d n) \]

We'll compare \(d\)-heaps to binary heaps. First, a couple useful facts:
\[ \log_d n = \log_d 2 \log_2 n \]
\[ \log_d 2 = 1/\log_2 d \]

Relative cost of insert

If insert is \(O(\log_d n)\), then runtime must be:
\[ T_{\text{worst}}^{d-\text{ins}}(n) = k^{d-\text{ins}} \log_d n + \text{low-order-terms}^{d-\text{insert}} \]

Comparing \(d\)-heap to binary heap for large \(n\) (ignore low order terms):
\[ \frac{T_{\text{worst}}^{d-\text{ins}}(n)}{T_{\text{worst}}^{2-\text{ins}}(n)} = \frac{k^{d-\text{ins}} \log_d n}{k^{2-\text{ins}} \log_2 n} = \frac{k^{d-\text{ins}} \log_2 2}{k^{2-\text{ins}} \log_2 n} = \frac{k^{d-\text{ins}}}{k^{2-\text{ins}} \log_2 d} \]

We’ll say the \(d\)-heap “speed-up” factor for insert is the reciprocal:
\[ \frac{k^{2-\text{ins}}}{k^{d-\text{ins}} \log_2 d} = \frac{k^{2-\text{ins}}}{k^{d-\text{ins}} \log_2 d} \]
**Relative cost of deleteMin**

If deleteMin is \(O(d \log_d n)\), then runtime must be:

\[
T_{\text{worst}}^{d-\text{del}} (n) = k^{d-\text{del} \log_d n + \text{low-order-terms}^{d-\text{del}}}
\]

Comparing \(d\)-heap to binary heap for large \(n\) (ignore low order terms):

\[
\frac{T_{\text{worst}}^{d-\text{del}} (n)}{T_{\text{worst}}^{2-\text{del}} (n)} = \frac{k^{d-\text{del} \log_d n}}{k^{2-\text{del} \log_2 n}} = \frac{k^{d-\text{del} \log_d 2 \log_2 n}}{2k^{2-\text{del} \log_2 2}}
\]

We’ll say the \(d\)-heap “slow-down” factor for deleteMin is:

\[
\frac{k^{d-\text{del}}}{2k^{2-\text{del}}} \cdot d \log_d 2
\]

**A strong assumption…**

Suppose now we assume:

\[
k^{d-\text{ins}} = k^{2-\text{ins}}
\]

\[
k^{d-\text{del}} = k^{2-\text{del}}
\]

Then we can look at relative performance as a function of \(d\)…

If we zoom in on the left side of the graph:
Observations

- At $d=3$ and $d=4$, deleteMin appears to be about as fast as for binary heaps, and insert appears to be appreciably faster.
- But, $d=3$ requires division and multiplication by 3, which is probably not as fast as bit shifting.
- $d=4$ is still good, though. Weiss suggests it may be current best choice.
- In general, the constants can make a significant difference...especially for large enough $n$ to require disk accesses.
- And we’re only considering worst case here. Average case is quite important (e.g., insert is $O(1)$ average case for binary heap).