CSE 326: Data Structures

d-Heaps - Addendum

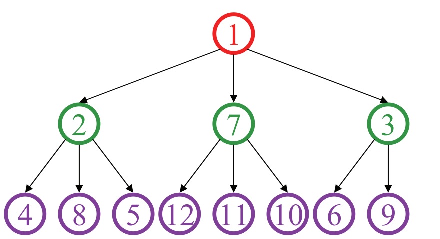
Brian Curless

Spring 2008

Announcements (4/16/08)

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

d-Heap complexity



1	2	7	3	4	8	5	12	11	10	6	9

Insert (worst case):

$O(\log_d n)$

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:

$$T_{worst}^{d-ins}(n) = k^{d-ins} \log_d n + \text{low-order-terms}^{d-insert}$$

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

$$\frac{T_{worst}^{d-ins}(n)}{T_{worst}^{2-ins}(n)} = \frac{k^{d-ins}\log_d n}{k^{2-ins}\log_2 n} = \frac{k^{d-ins}\log_d 2 \ \log_2 n}{k^{2-ins}\log_2 n} = \frac{k^{d-ins}}{k^{2-ins}}\log_d 2$$

We'll say the *d*-heap "speed-up" factor for insert is the reciprocal:

$$\frac{k^{2-ins}}{k^{d-ins}}\frac{1}{\log_d 2} = \frac{k^{2-ins}}{k^{d-ins}}\log_2 d$$

Relative cost of deleteMin

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

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

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

$$\frac{T_{worst}^{d-del}(n)}{T_{worst}^{2-del}(n)} = \frac{k^{d-del} d \log_d n}{k^{2-del} 2 \log_2 n} = \frac{k^{d-del} \log_d 2 \, \log_2 n}{k^{2-del} 2 \log_2 n} = \frac{k^{d-del}}{2k^{2-del}} d \log_d 2$$

We'll say the d-heap "slow-down" factor for deleteMin is:

$$\frac{k^{d-del}}{2k^{2-del}} d \log_d 2$$

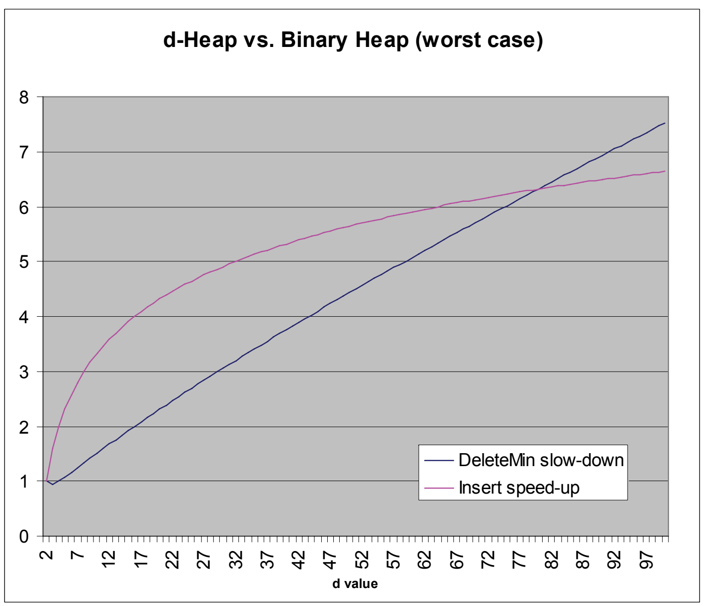
A strong assumption…

Suppose now we assume:

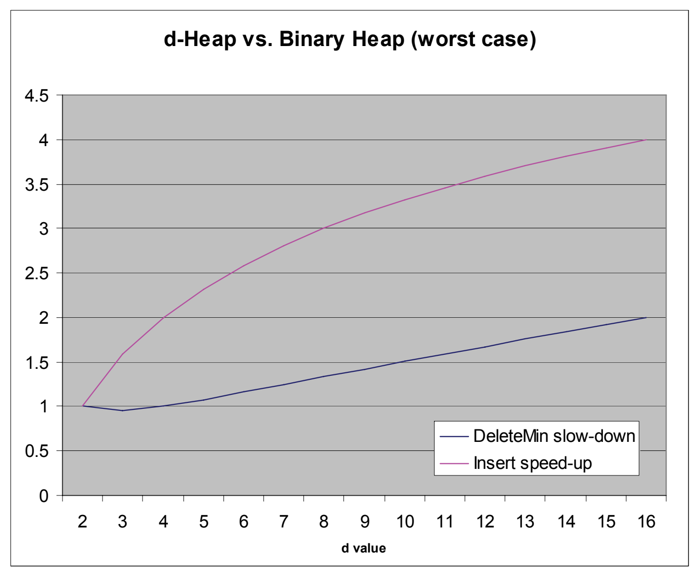
$$k^{d-ins} = k^{2-ins}$$

$$k^{d-del} = k^{2-del}$$

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).