CSE 326: Data Structures

d-Heaps

Brian Curless
Spring 2008

Announcements (4/9/08)

- Project #1 due today
- Homework due Friday at start of class
- Project #2a assigned Friday
  - You may work with a partner if you choose.
  - If you do so, find a partner by Friday.

(Aside)

A Perfect binary tree – A binary tree with all leaf nodes at the same depth. All internal nodes have 2 children.

- Height \( h \)
  - \( 2^{h+1} - 1 \) nodes
  - \( 2^h - 1 \) non-leaves
  - \( 2^h \) leaves

Number of nodes:

\[
\sum_{i=0}^{h} 2^i = 2^{h+1} - 1
\]

Facts about Heaps

Observations:
- Finding a child/parent index is a multiply/divide by two
- Operations jump widely through the heap
- Each percolate step looks at only two new nodes
- Inserts are at least as common as deleteMins

Realities:
- Division/multiplication by powers of two are equally fast
- Hopping through the array, looking at only two new pieces of data: bad for cache!
- With huge data sets, disk accesses dominate
A Solution: $d$-Heaps

- Each node has $d$ children
- Still representable by array
- Good choices for $d$:
  - (choose a power of two for efficiency)
  - fit one set of children in a cache line
  - fit one set of children on a memory page/disk block

Operations on $d$-Heap

- Insert: runtime $= O(\log_d n)$

- deleteMin: runtime $= O(d \log_d n)$

Does this help insert or deleteMin more?

One More Operation

Merge two heaps:

- Add the items from one into another?
- Start over and build it from scratch?