Today’s Outline

• Announcements
  – Homework #3 due Wed, Feb 8th, 11pm.

• Today’s Topics:
  – Priority Queues
    • Binary Min Heap - buildheap
    • D-Heaps

Facts about Binary Min 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
• looking at only two new pieces of data: bad for cache!
• with huge data sets, disk accesses dominate

Representing Complete Binary Trees in an Array

From node i:

left child: right child: parent:

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 = $O(d \log d)$ worst
  - Depth of tree decreases
  - $O(d \log n)$ worst

- deleteMin: runtime = $O(d \log d)$ worst
  - percolateDown requires $d$ comparisons to find min child
  - $O(d \log n)$ worst