Priority Queues II

CSE 373
Data Structures & Algorithms
Ruth Anderson

Today’s Outline

• Announcements
  – Midterm 1, this Fri Oct 21
  – Homework #3 due Thurs, Oct 27, 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:

implicit (array) implementation:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>L</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
</tr>
</tbody>
</table>

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 = Depth of tree decreases. $O(d \log_d n)$ worst

• deleteMin: runtime = percolateDown requires $d$ comparisons to find min child. $O(d \log_d n)$ worst