CSE 373
Data Structures & Algorithms

Lecture 09
Binary Heaps (Part II)
The Midterm

• Friday, October 23, 12:30, in class

• Closed books, closed notes

• Topics covered:
  – Asymptotic complexity, Big-O, stacks, queues, trees, AVL trees, B trees
Six Questions on the Midterm

1. Compute asymptotic complexities of programs
2. Write programs with better asymptotic complexities
3. Big O notation basics
4. Stacks and Queues
5. Trees: general concepts
6. Search trees (AVL, B): insertion, deletion

Speed and familiarity with the material are critical
Building a Heap

• At every point, the new item may need to percolate all the way through the heap

• Adding the items one at a time is $O(n \log n)$ in the worst case (what is the worst case?)

• Today we get clever and do it in $O(n)$
BuildHeap: Floyd’s Method

Add elements arbitrarily to form a complete tree. Pretend it’s a heap and fix the heap-order property!
Buildheap pseudocode

```java
private void buildHeap() {
    for (int i = currentSize/2; i > 0; i--) {
        percolateDown(i);
    }
}
```
BuildHeap: Floyd’s Method
Finally...
Note they’re not the same

O(N log N)

But that doesn’t matter, they’re both heaps

O(N)

Floyd’s method runs in time O(n); read the proof in Ch. 6.3.4
Facts about Heaps

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

• Realities:
  – Division/multiplication by powers of two are equally fast
  – With huge data sets, disk accesses dominate
  – Looking at only two new pieces of data: bad for cache!
Extension: $d$-Heaps

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

How does height compare to bin heap? (less)
This example has height 2, vs. 3 for bin
Operations on $d$-Heap

• Insert: runtime =

• deleteMin: runtime =

Does this help insert or deleteMin more?

depth of tree decreases, $O(\log_d n)$ worst

percolateDown requires comparison to find min, $O(d \log_d n)$, worst/ave