CSE 373: Midterm Review

Pete Morcos
University of Washington
4/28/00

http://www.cs.washington.edu/education/courses/cse373/00sp

Asymptotic Analysis

- O(f(N)) means "no worse than, maybe better"
- Θ(f(N)) means "basically the same, tight bound"
- Ω(f(N)) means "no better than, maybe worse"
- All three are only true for "sufficiently large" N
- Recall formal definition:
  - $T(n) = O(f(n))$ iff. there are positive constants c and n_0 such that $T(n) \le c \cdot f(n)$ for all $n \ge n_0$
- Two typical ways to figure out cost of code
  - Summations: used for iterative code
  - Recurrences: used for recursive code

Important Summations

$$\sum_{i=0}^{N} i = O(N^2)$$

Example leading to above summation:

```
for(i=0; i<=N; i++)
    for(j=0; j<i; j++)
        // do constant time work
```

outer loop executes N times (i = 0 to N)
inner loop executes i times (j = 0 to i)

$$\sum_{i=0}^{N} 2^i = O(2^N)$$

$$\sum_{i=0}^{\log N} 2^i = O(N)$$

Writing down recurrences

- Usually given an input of size N. Define T(N) as cost of a function call.
- Look at each line of function, and write down its cost.
  - Only worry about cost of the line during *current* function call.
- For recursive calls, don't try to figure out cost; instead just write down in T form.
  - Example, if recursion uses a problem of half the original size, its cost is T(N/2)

Lists, Stacks, Queues

- Variants
  - Header nodes make pointer manipulations easier at ends of list
  - Doubly linked
  - Circularly linked
- Two ways to implement
  - Array-based: uses minimum space, but may need to shift O(N) items after some operations
  - Link-based: uses extra space for pointers, but rearrangements easier
- Array queues require slightly tricky modulo arithmetic
- Computer uses a "call stack" for all function calls—implicit use of space

Trees

- Some reminders
  - Single node tree has height, depth of 0
  - Height of a node is maximum path length to any leaf
- Recursive definition: a tree is either
  - null, or
  - a node with some number of trees as children
- Preorder, postorder, inorder traversals
- Depth can range from N to approx. log N depending on how balanced the tree is

Binary Trees

- Binary search trees: understand arrangement of values, how to do Find and Insert
- AVL trees have extra height info at each node
  - Limit to how unbalanced they are
  - Find just like BST
  - Insert has extra rotation step to fix things up
    - Triggered while we head back up the tree updating heights
    - Two kinds of rotation

B-Trees

- Nodes have up to M children, with up to M-1 “index values” to help with navigation
  - Child to the right of index value k contains values all greater than or equal to k
- Leaf nodes all at same height
- Inserting may cause nodes to overfill and split, adding new child to parent, which may split, ...
- Removal may cause node to become less than half full, causing merger with sibling, which may cause parent to become less than half-full...

Hash Tables

- Usually want table size prime, hash function to be highly variable
- Collisions can be handled in different ways
  - Chaining: add extra space for pointers to linked lists of colliding values
  - Linear Probing: scan linearly for next free item; may create clusters
  - Quadratic Probing: scan quadratically for next free item; may fail if table over half-full
- Probing techniques require rehashing if table is full; become slow when table close to full

Heaps

- Heaps only provide access to smallest item
- Understand the array implementation technique
- Insert/DeleteMin must maintain completeness of tree:
  - Insert adds item to end of array, then percolate up
  - DeleteMin moves item at end to top, then percolate down
- d-heap just has d children per node rather than 2

Binomial Queues

- Alternative to heaps
- Contains various B_i trees, each holding 2^i nodes
- Merge:
  - Join matching pairs of B_i, creating B_{i+1}
  - Move from 0 upwards
- Insert: merge original with new single-node BQ
- DeleteMin:
  - Chop off smallest root
  - Treat orphans as a new BQ, merge with original BQ

Selection

- Want to get kth smallest item of a set
- k = N/2 (the median) is hardest case
  - Naive techniques require $O(k\ N) = O(N^2)$
- Quickselect is simple method that is O(N) average
- Recurrence for quickselect:
  - $T(N) = T(N/2) + N$
  - $= N + N/2 + N/4 + ... + 1$
  - $= O(N)$

Sorting

- Bubblesort, Selection Sort, Insertion Sort
  - Won't ask questions about bubblesort
- Heapsort
  - Understand trick of using right end of array to avoid using extra space—requires a maxheap
- Mergesort

More Sorting

- Shellsort
  - Divide set into alternating colors
  - Sort within a color
  - Reduce the increment, causing items that used to have different colors to be the same
  - Repeat until increment = 1
- Quicksort
  - Divide-and-conquer recursion
  - Partition step trickiest part
  - Understand how to write the recurrence, given the code

Sorting III

- Stable sorts don't change order of duplicates
- Bucket sort is O(N) for "small enough" N
- Radix sort is technique for sorting on hierarchical values (e.g. first by major, then by name) or for sorting by dividing values into slices