CSE 326: Data Structures
Lecture #24
Odds ‘n Ends

Henry Kautz
Winter Quarter 2002

Treap Delete
- Find the key
- Increase its value to $\infty$
- Rotate it to the fringe
- Snip it off

Moral
- Yes, Virginia, you can maintain the Binary Search Tree property while restoring the heap property.

Treap Delete, cont.

Traveling Salesman
Recall the Traveling Salesperson (TSP) Problem:
Given a fully connected, weighted graph $G = (V,E)$, is there a cycle that visits all vertices exactly once and has total cost $\leq K$?
- NP-complete: reduction from Hamiltonian circuit
- Occurs in many real-world transportation and design problems
- Randomized simulated annealing algorithm demo

Final Statistics
- multiple choice – 33 points
- true / false – 27 points
- solving recurrence relations – 10 points
- calculating various quantities – 16 points
- creating a novel algorithm – 8 points
- data structure simulation – 6 points
Final Review

("We’ve covered way too much in this course… What do I really need to know?")

Final Review: What you need to know

- **Basic Math**
  - Logs, exponents, summation of series
  - Proof by induction
- **Asymptotic Analysis**
  - Big-Oh, Theta and Omega
  - Know the definitions and how to show f(N) is big-Oh/Theta/Omega of g(N)
  - How to estimate Running Time of code fragments
  - *e.g.* nested “for” loops
- **Recurrence Relations**
  - Deriving recurrence relation for run time of a recursive function
  - Solving recurrence relations by expansion to get run time

Final Review: What you need to know

- Lists, Stacks, Queues
  - Brush up on ADT operations: Insert/Delete, Push/Pop etc.
  - Array versus pointer implementations of each data structure
  - Amortized complexity of stretchy arrays
- **Trees**
  - Definitions/Terminology: root, parent, child, height, depth etc.
  - Relationship between depth and size of tree
  - Depth can be between $O(\log N)$ and $O(N)$ for $N$ nodes

Final Review: What you need to know

- **Binary Search Trees**
  - How to do Find, Insert, Delete
    - Bad worst case performance – could take up $O(N)$ time
  - AVL trees
    - Balance factor is +1, 0, -1
    - Know single and double rotations to keep tree balanced
    - All operations are $O(\log N)$ worst case time
  - Splay trees – good amortized performance
    - A single operation may take $O(N)$ time but in a sequence of operations, average time per operation is $O(\log N)$
    - Every Find, Insert, Delete causes accessed node to be moved to the root
    - Know how to zig-zig, zig-zag, etc. to “bubble” node to top
  - B-trees: Know basic idea behind Insert/Delete

Final Review: What you need to know

- **Priority Queues**
  - Binary Heaps: Insert/DeleteMin, Percolate up/down
    - Array implementation
    - BuildHeap takes only $O(N)$ time (used in heapsort)
  - Binomial Queues: Forest of binomial trees with heap order
    - Merge is fast – $O(\log N)$ time
    - Insert and DeleteMin based on Merge
- **Hashing**
  - Hash functions based on the mod function
  - Collision resolution strategies
    - Chaining, Linear and Quadratic probing, Double Hashing
  - Load factor of a hash table

Be Sure to Bring

- 1 page of notes
- A hand calculator!
- Several #2 pencils
Final Review: What you need to know

- **Sorting Algorithms**: Know run times and how they work
  - Elementary sorting algorithms and their run time
    - Selection sort
    - Heapsort – based on binary heaps (max-heaps)
    - BuildHeap and repeated DeleteMax’s
    - Mergesort – recursive divide-and-conquer, uses extra array
    - Quicksort – recursive divide-and-conquer, Partition in-place
      - fastest in practice, but $O(N^2)$ worst case time
      - Pivot selection – median-of-three works best
    - Know which of these are stable and in-place
    - Lower bound on sorting, bucket sort, and radix sort

- **Disjoint Sets and Union-Find**
  - Up-trees and their array-based implementation
  - Know how Union-by-size and Path compression work
  - No need to know run time analysis – just know the result:
    - Sequence of $M$ operations with Union-by-size and P.C. is $\Theta(M \alpha(M,N))$ – just a little more than $\Theta(1)$ amortized time per op

- **Graph Algorithms**
  - Adjacency matrix versus adjacency list representation of graphs
  - Know how to Topological sort in $O(|V| + |E|)$ time using a queue
  - Breadth First Search (BFS) for unweighted shortest path

- **Multidimensional Search Trees**
  - k-d Trees – find and range queries
  - Depth logarithmic in number of nodes
  - Quad trees – find and range queries
    - Depth logarithmic in inverse of minimal distance between nodes
    - But higher branching factor means shorter depth if points are well spread out (log base 4 instead of log base 2)

- **Randomized Algorithms**
  - expected time vs. average time vs. amortized time
  - Treaps, randomized Quicksort, primality testing