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

Henry Kautz

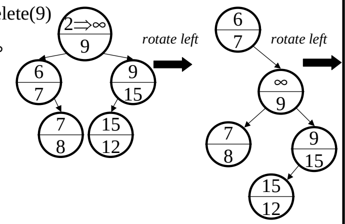
Winter Quarter 2002

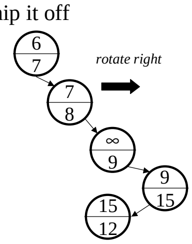
1

Treap Delete

- Find the key
- Increase its value to ∞
- Rotate it to the fringe
- Snip it off

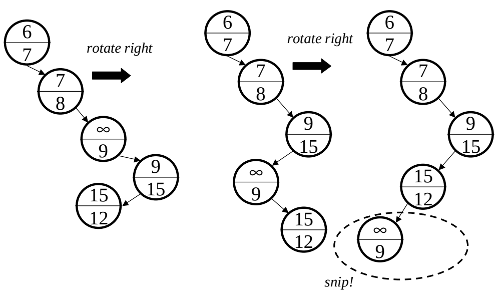
delete(9)





2

Treap Delete, cont.



3

Moral

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

4

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 ≤ K?

- NP-complete: reduction from Hamiltonian circuit

- Occurs in many real-world transportation and design problems
- <u>Randomized simulated annealing algorithm demo</u>

5

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

6

Final Review

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

7

Be Sure to Bring

- 1 page of notes
- A hand calculator!
- Several #2 pencils

8

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-O/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

$$\sum_{i=1}^{N} i = \frac{N(N+1)}{2}$$

$$\sum_{i=0}^{N} A^i = \frac{A^{N+1}-1}{A-1}$$

9

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

10

Final Review: What you need to know

- Binary Search Trees
  - How to do Find, Insert, Delete
    - Bad worst case performance – could take up to 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

11

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

12

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

13

Final Review: What you need to know

- 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

14

Final Review: What you need to know

- Graph Algorithms (cont.)
  - Dijkstra's shortest path algorithm
  - Depth First Search (DFS) and Iterated DFS
    - Use of memory compared to BFS
  - A* - relation of g(n) and h(n)
  - Minimum Spanning trees – Kruskal's algorithm
  - Connected components using DFS or union/find
- NP-completeness
  - Euler versus Hamiltonian circuits
  - Definition of P, NP, NP-complete
  - How one problem can be "reduced" to another (e.g. input to HC can be transformed into input for TSP)

15

Final Review: What you need to know

- 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 fractor 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

16