CSC207.01 2014S, Class 45: Heaps

Overview

- Preliminaries.
  - Admin.
  - Questions.
  - Looking at phase 1.
- Priority queues, revisited.
- Recent implementation techniques.
- Heaps.
- Adding elements to heaps.
- Removing elements from heaps.
- Asymptotic analysis.
- Storing heaps in arrays.
- Heap sort.

Preliminaries

Upcoming Work

- Reading for Tuesday: No reading.
- Today's writeup: No writeup.
- Part 2 of the project is due 10:30 p.m. Friday.

Admin

- Welcome to our prospective
- I've reviewed about half of your projects before running out of time.
  We'll talk about some common (and not so common) issues for about ten minutes.
- Given the problems with MathLAN, I'm not sure what to do about the due date of the project. What do you think?
  - Those of you trying to fix the problem using Wireshark or whatever probably need to stop.
- I need two graders for next semester who can do detailed comments. Yes, I will provide training.

Extra Credit

- Math extra Thursday: Sphere Packing.
- CS Table Friday: Heartbleed. Readings TBD.
- Iowater project April 26 - Tag drains. Mail iowater@grinnell.edu for details.
- Any one pride week activity.

Questions on the Project.

Short discussion of Phase 1.

Priority queues, revisited

ADT

- A queue that's prioritized.
- A normal queue is first in first out.
- A priority queue has a priority that indicates which things are more important
- We could store numbers and the largest/smallest has highest priority.
- We could associate a number with each value we store, largest (or smallest) has highest priority.
- Use a comparator!

How might we implement one?

- Linked list in which values are in order of priority.
  - O(1) to get
  - O(n) to put

Can we do better?

- Use a skip list. O(logn) to put, O(logn) to get.

Recent implementation techniques

- Hash tables - Using keys and hashing: convert to a number
  - Expandable
- Binary search trees - Divide and conquer applied to data structures
- We could hope that put is O(logn) if it's balanced
- We don't know how to keep them balanced.

Heaps

- An approach for building priority queues.
- Binary trees, nearly complete
  - Complete at every level (except maybe the last)
- Also with the heap property: The root of the tree has the highest priority (and that holds for all the subtrees)
- Note: Peeking at the highest-priority element is O(1)
- How do we build and maintain a heap structure

Adding elements to heaps

- When I add something, I have to decide where it goes and then I have two properties to achieve.
  - Nearly complete
  - Heap property
- To maintain near completeness, we need to add something at the end of the last row (or, if that row is complete, at the beginning of the next row)
- And then "swap up" - if larger than thing above, swap the values.
  - No need to look at subtrees - We know that it’s larger than the other subtree.
- Question: How do we find the end of the last row?
  - Forthcoming.
- How fast is add? path to the root: O(logn)

Removing elements from heaps

- Remove the root
- Put the last element at the last level at the root. Nearly complete!
- Repeatedly swap with the larger of the two children (provided it’s smaller than the larger of the two children).
- Yay! O(logn)

Asymptotic analysis

Storing heaps in arrays

Heap sort

Copyright (c) 2013-14 Samuel A. Rebelsky.

This work is licensed under a Creative Commons Attribution 3.0 Unported License. To view a copy of this license, visit `http://creativecommons.org/licenses/by/3.0/` or send a letter to Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA.