CSCI 136:
Data Structures
and
Advanced Programming

Lecture 32

Heap implementation

Instructor: Dan Barowy

Williams

Topics

Heap implementation

Announcements

1. **Final exam**: Saturday, Dec 17, 1:30pm. Room TBD.
2. **Final exam review session**, in class, last day of class, **Friday 12/9**.

Your to-dos

1. **Last quiz**, due **Sat**.
2. Lab 10 (partner lab), **due Tuesday 12/6 by 10pm**.
3. **Review readings** from *Bailey*.
4. **Study** for the final exam.
   a. Pro tip: **review quizzes**.
   b. **Do problems** in study guide/practice exam.
   c. **Don't stress out!** Just be methodical and do your best.
5. **Work on resubmissions** you plan to submit.

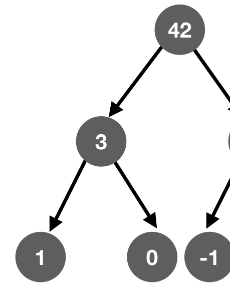
Announcements

Sean Barker '09, Bowdoin College

Friday, Dec 2 @ 2:35pm
Computer Science Colloquium – Wege TCL 123
Smart Meters for Smart Cities: Data Analytics in Energy-Aware Buildings

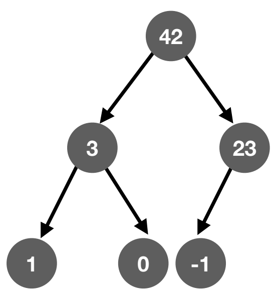
The proliferation of smart energy meters has resulted in many opportunities for next-generation buildings. Energy-aware "smart buildings" may optimize their energy consumption and provide convenience and economic benefits through analysis of their meter data. However, storing and analyzing this data presents computational challenges, especially when conducted at scale. In this talk, I discuss our work on several problems in this space, focusing particularly on efficient compression of smart meter data and the disaggregation of building-wide consumption into individual device consumption. Our work in these areas aims to support the development of sustainable, energy-efficient smart cities and smart grids.

Refresher: binary max heap



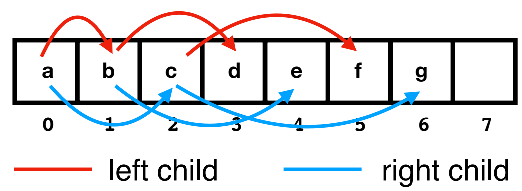
Max heap property: for any given node **n**, if **p** is a parent node of **n**, then the **key** of **p** is ≥ the **key** of **n**.

Insertion



A **binary heap** is usually implemented as an **always-complete binary tree**.

Implementation



A binary heap is often implemented using an implicit binary tree data structure. In other words, heap nodes are actually stored in an array or vector.

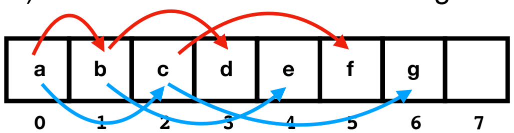
$$\texttt{leftChild(i)} = 2 \times i + 1$$
$$\texttt{rightChild(i)} = 2 \times i + 2$$
$$\texttt{parent(i)} = \lfloor (i - 1) / 2 \rfloor$$

Max heap in action

Build a max heap from the following elements:

56 5 57 0 -7 99

But store the elements in an array (i.e., an implicit binary tree). Process nodes from left to right.



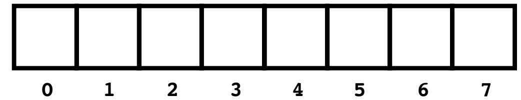
left child — right child

$$\text{leftChild}(i) = 2 \times i + 1$$

$$\text{rightChild}(i) = 2 \times i + 2$$

$$\text{parent}(i) = \lfloor (i - 1) / 2 \rfloor$$

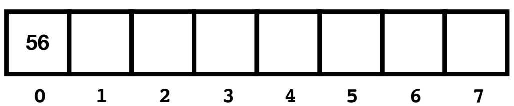
Max heap in action



left child — right child

56 5 57 0 -7 99

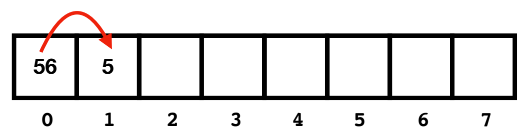
Max heap in action



left child — right child

5 57 0 -7 99

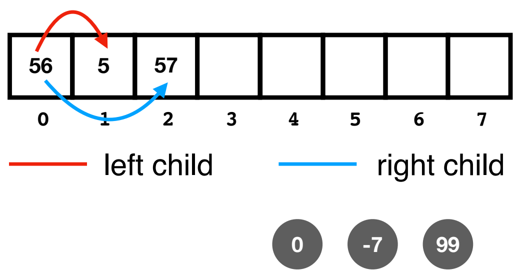
Max heap in action



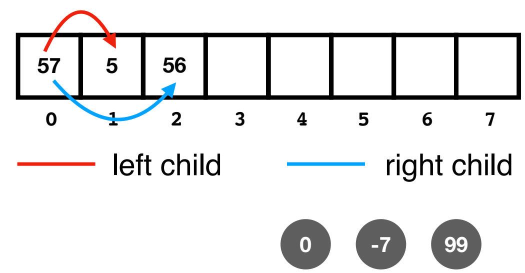
left child — right child

57 0 -7 99

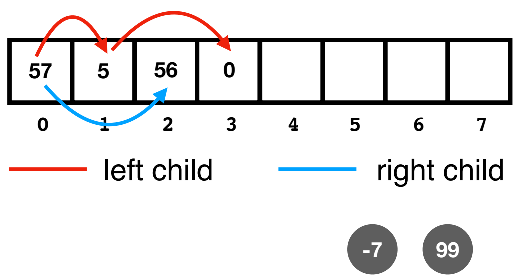
Max heap in action



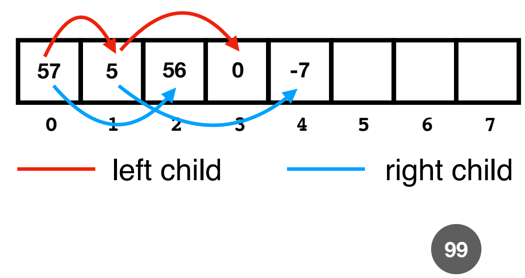
Max heap in action



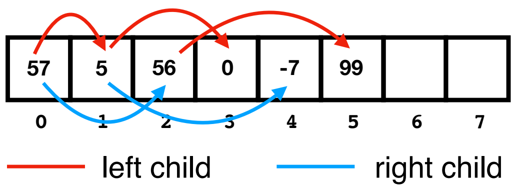
Max heap in action



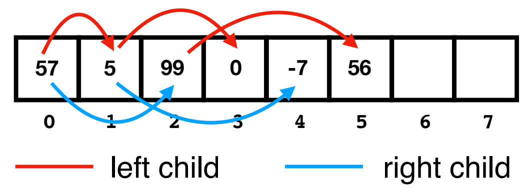
Max heap in action



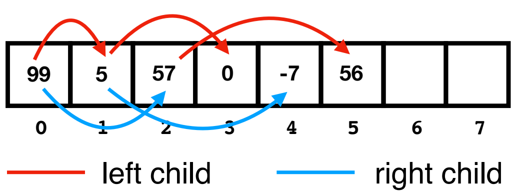
Max heap in action



Max heap in action

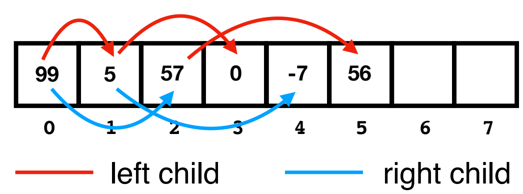


Max heap in action



Done!

Max heap in action



<u>Advantages</u>:

find-max: O(1)
insert: O(log n)
extract: O(log n)

How is a binary heap implemented?
(code)

Recap & Next Class

Today:

Heaps

Next class:

Dijkstra’s algorithm