CSC236 - Week 1, Lecture 2

Cristyn Howard

January 10, 2018

- abstract data type (ADT) - an object and its operations
- data structure - specific implementation of some ADT

- One example of an ADT is a Priority Queue (PQ).
  - object: maintains a set S of elements with keys that can be compared
  - operations:
    * INSERT(x, S): insert item x into set S such that the order is maintained
    * MAX(S): return a value with the maximum priority
    * EXTRACTMAX(S): find an element with max priority and remove it from the set
- PQ application example: might be used by an operating system to organize processes that need to be run
- Some data structures for PQs:
  -

	Worst-case run time of:	
Data Structure	INSERT	EXTRACTMAX
unsorted linked list	$O(1)$	$O(n)$
sorted linked list	$O(n)$	$O(1)$
heaps	$O(logn)$	$O(logn)$

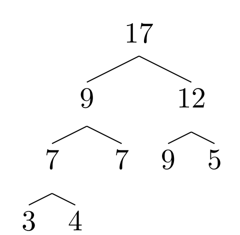
  - Note that heaps are superior! So what are they, and how do they work?

Recall: Complete Binary Tree (CBT): start at the leftmost node, every node has at most two children, fill every level before moving to the next

Recall: Height of binary tree: length of longest path from the root to any leaf of the tree.

  - For CBT of size n, height = $\lfloor log_2 n \rfloor$

- heaps: store set S of size n in an n-node complete binary tree
  - elements in a heap are stored such that the priority of any node is greater than the priority of its children
  - heaps for set S are not unique, i.e. there are many valid heap configurations of set S
  - Example: $S = \{3, 4, 5, 7, 7, 9, 9, 12, 17\}$, A heap containing s:



- In a computer, heaps are stored in a (1-indexed) array such that the elements represent the nodes of the heap ordered from top-to-bottom, left-to-right.
  - An array storing our heap of S:

Index:	1	2	3	4	5	6	7	8	9
Value:	17	9	12	7	7	9	5	3	4

- How do we relate parents to children (and vice versa) when the heap is stored in an array?

Parents to Children	Children to Parents
right child index = parent index × 2	parent index = $\lfloor$ child index / 2 $\rfloor$
left child index = (parent index × 2) + 1	

- How are the PQ operations implemented on a heap stored as an array?
  - INSERT(): (A) add new element to last space in array; (B) get parent index, get parent value; (C) if parent value is smaller than new element value, swap element locations, go to B.
    - <u>worst case run time of insert</u>: compare & swap take constant time, the upper bound on the number of swaps is the height of the tree, thus Insert $\in \Theta(logn)$.
      - $O(logn)$: every insert takes <u>at most</u> $c \times logn$ steps, for some constant c.
      - $\Omega(logn)$: $\exists$ some element for which insert takes <u>at least</u> $c \times logn$ steps, (e.g. $x > root$).
  - MAX(): $\Theta(1)$, get first element of the array
  - EXTRACTMAX(): (A) swap first & last elements of the array; (B) return last element & reduce heap size by one; (C) compare first element value to values of both children, swap element with larger of two children; (D) continue until element is larger than both children, or until the indices of both children are greater than heap size (i.e. element has no children)
    - $\Theta(logn)$ for reasons similar to insert
  - SORTING: extract max until all elements have been removed from the array, $\Theta(n \times logn)$