

CS 430 Lecture 21 Activities

Opening Questions

1. Review the operations on a min (or max) binary heap and their runtimes. [Insert and Extract-Max](#) which are both $O(\lg n)$.
2. What if we needed a function to union two min binary heaps? How would you do it and what is the run time? [You would have to call EXTRACT-MAX on each element from the smaller heap and INSERT it into the other heap.](#)
3. Why did we use an array for a binary heap? Does a heap need to be binary? [Heaps match arrays well because both cannot have holes. It doesn't need to be binary, it could be trinary and have a difference in calling the child nodes \(\$3n\$, \$3n + 1\$, \$3n + 2\$ \) and the heap would be flatter, but at the tradeoff of HEAPIFY requiring more time.](#)

Mergeable (Min) Heaps

Consider the following operations on heaps. Our goal is to support all of these operations in no worse than $\Theta(\log n)$ and not be constrained to use an array or a binary tree.

- MAKE-HEAP: Creates a new empty heap.
- INSERT: Inserts a new element into a heap.
- MINIMUM: Returns the minimum element in a heap.
- EXTRACT-MIN: Returns the minimum element in a heap and removes it from the heap.
- UNION: Creates a new heap consisting of the elements of two existing heaps.


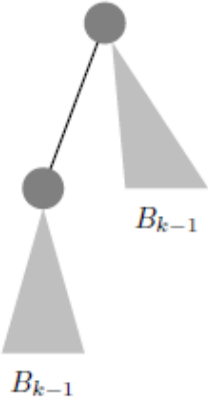
Two types of mergeable heaps are Binomial heaps and Fibonacci heaps¹. They also support these operations.

- DECREASE-KEY: changes the value of some element in a heap to a smaller value.
- DELETE: removes an element from a heap.

¹Amortized $O(1)$

Binomial Heaps (utilizing binomial trees)

Table 21.0:

<p>A binomial tree is an ordered tree defined recursively:</p> <p>$B_0 =$ </p> <p>$B_k =$ </p>	<ol style="list-style-type: none"> 1. How many nodes does B_k have, what is its height, and how many children does its root have? There are 2^k nodes. Its height is k. The root has k children. 2. Why are they called binomial trees? The binomial numbers represent how many nodes there are on each level.
---	--

A binomial heap is a collection (linked list) of binomial trees in which each binomial tree is **heap-ordered**: each node is greater than or equal to its parent. Also, in a binomial heap at most one instance of B_i may occur for any i .

3. How many binomial trees are needed at most in the linked list of roots to make a binomial heap of n nodes? You would need $\lg n$ binomial trees at most to make a binomial heap of n nodes.
4. See <https://www.cs.usfca.edu/~galles/visualization/BinomialQueue.html> to help describe how each operation is done, and its run time:

Make-Heap : $O(1)$, just give me a node that acts as B_0 and put the value in.

Minimum : Linear walk of root list: $O(\lg n)$.

Union :

Insert : Create a new B_0 heap and insert it at the front of the root list. Look at the next Binomial tree in the root list, and if they have the same size, merge the 2 “heap order” $O(\lg n)$.

Extract-Min : It would take $O(\lg n)$ to find the minimum value, and then you merge the children of the max back into the root list.

Decrease-Key : $O(\lg n)$. Assume we are at the node in the binomial heap. You remove the node and, if the value of the child is smaller than the value of the root, you would switch the values and keep walking that switch up.

Delete : $O(\lg n)$. Assume ___ at that node, make its value $-\infty$, call DECREASE-KEY and then EXTRACT-MIN to remove the $-\infty$.