**CS430 Lecture 22 Activities**

Fibonacci Heaps

Fibonacci heaps which support heap operations that do not delete elements in constant amortized time. From a theoretical standpoint, Fibonacci heaps are especially desirable when the number of EXTRACT-MIN and DELETE operations is small relative to the number of other operations performed. This situation arises in many graph algorithms.

In essence, a Fibonacci heap is a “lazy” binomial heap in which the necessary housekeeping is delayed until the last possible moment: deletion.

|  |  |
| --- | --- |
| * Set of Heap ordered trees (each parent smaller than children) * Maintain pointer to minimum element (find-min takes O(1) time) * Set of marked nodes (if one of its children has been removed) * n – number of nodes in the heap * rank(x) – number of children of node x * rank(H) – max rank of any node in heap H * trees(H) – number of trees in heap H * marks(H) – number of marked nodes in H |  |

1. See <https://www.cs.usfca.edu/~galles/JavascriptVisual/FibonacciHeap.html> and <https://www.cs.princeton.edu/~wayne/cs423/fibonacci/FibonacciHeapAnimation.html>

to help describe how each operation is done, and a rough estimate on its run time:

Make-Heap

Insert:

Minimum:

Union:

Extract-Min:

Decrease-Key:

Delete:

|  |  |
| --- | --- |
|  | Note that the times indicated for the Fibonacci heap are amortized times while the times for binary and binomial heaps are worst-case per-operation times. |