A Fibonacci Heaps

A.1 Mergeable Heaps

A *mergeable heap* is a data structure that stores a collection of *keys*[1] and supports the following operations.

- INSERT: Insert a new key into a heap. This operation can also be used to create a new heap containing just one key.
- FINDMIN: Return the smallest key in a heap.
- DELETEMIN: Remove the smallest key from a heap.
- MERGE: Merge two heaps into one. The new heap contains all the keys that used to be in the old heaps, and the old heaps are (possibly) destroyed.

If we never had to use DELETEMIN, mergeable heaps would be completely trivial. Each "heap" just stores to maintain the single record (if any) with the smallest key. INSERTs and MERGEs require only one comparison to decide which record to keep, so they take constant time. FINDMIN obviously takes constant time as well.

If we need DELETEMIN, but we don't care how long it takes, we can still implement mergeable heaps so that INSERTs, MERGEs, and FINDMINs take constant time. We store the records in a circular doubly-linked list, and keep a pointer to the minimum key. Now deleting the minimum key takes $\Theta(n)$ time, since we have to scan the linked list to find the new smallest key.

In this lecture, I'll describe a data structure called a *Fibonacci heap* that supports INSERTs, MERGEs, and FINDMINs in constant time, even in the worst case, and also handles DELETEMIN in $O(\log n)$ *amortized* time. That means that any sequence of n INSERTs, m MERGEs, f FINDMINs, and d DELETEMINs takes $O(n + m + f + d \log n)$ time.

A.2 Binomial Trees and Fibonacci Heaps

A *Fibonacci heap* is a circular doubly linked list, with a pointer to the minimum key, but the elements of the list are not single keys. Instead, we collect keys together into structures called *binomial heaps*. Binomial heaps are trees[2] that satisfy the *heap property* — every node has a smaller key than its children — and have the following special structure.