day 9: 2/9/98 "This is a big mess."

1 Binomial Tree's, continued

1.1 some definitions

- $|B_i| = 2^i$
- $rank(node) = \#children$
- $rank(tree) = rank(root)$

1.2 insertion

Make a singleton tree then meld it with the tree you wish to insert into. This is amortized $O(1)$.

1.3 lazy meld

Relax the restriction that there be only one tree of a particular B_i. Concatenate 2 lists of trees in the meld, updating the min pointer to the min of each heap.

1.3.1 Find min

Just follow the min pointer through the heap

1.3.2 delete min

Remove some root and consider each subtree a tree. It will be necessary to inspect all trees to find the new min pointer. Might as well do real melds at the same time. $O(log(n))$.

2 Fibonacci heaps

Fibonacci heaps are binomial heaps with these exceptions:

- each node may be missing at most 1 child
- Root may be missing any number of children

2.1 Invariant

- $rank(t) \leq O(log(n))$

2.2 decrement

To decrement, promote nodes to the top level and decrement most of the time.
$decrement(h, x, \Delta)$

1. change priority of x
2. if $x \neq root$ then let $y = parent(x)$

- promote x to a new tree and mark(x)
- cascade-cut(y)

$cascade - cut(y)$

1. $z = parent(y)$
2. if $y \neq root$ then if $mark(y) = \emptyset$ then $mark(y) = true$ else $cut(y)$ and $cascade - cut(z)$

$cut(y)$

1. promote y to a root

2.3 analysis

let f = a potential function = # of trees + 2 * # of marks.

Hmm.. These notes are indecipherable. I'll grab some stuff from a book of mine. "Algorithms from P to NP" by Moret and Shapiro. This is pretty much verbatim.

Lemma: Let x be any node in a Fibonacci heap and let the current children of x be orderd from ealiest to latest according to the time at whichc they were linked to x. Then the rank of the ith child is at least $i - 2$. $rank(y) \geq i - 2$ where y = i'th child of x

proof: When y was linked to x, x must alread have had at least i-1 children, because children could since have disappeared (due to cuts), but any children added subsequently would be farther down in the order. Since x and y were linked, they had the same rank at that time, so that y itself had at least $i - 1$ children. Since y has not been cut, it can have lost at most one child since that time and hence still has at least $i - 2$ children.

corollary: A node of ranke k in a fibonacci heap has at least F_{k+2} descendents where F_k is the kth fibonacci number.

proof: Let S_k be the size of the smallest possible subtree in any Fibonacci heap such that its root has rank k. We claim that we have $S_k = F_{k+2}$. We have $S_0 = 1$ and $S_1 = 2$. The preceeding lemma gives the recurrence $S_k \geq \sum_{i=2}^{k} S_{i-2} + 2$. The inequality is exact because S_k was was specified to be minimal and such a tree does exist, giving: $S_k = S_{k-1} + S_{k-2}$. This implies $S_k = F_{k+2}$.