Fibheap analysis (cont.)

1. Compute amortized cost of DeleteMin assuming all tree roots have degree $O(\lg n)$ [n = size of the heap]

2. Prove this assumption

1. DeleteMin(H) — amortized
   - Remove H.min — $O(1)$
   - Promote H.min's children to the root list — $O(\#\text{children}) = O(\lg n)$ by our assumption
   - Traverse the root list & consolidate
     - moving to the next tree $O(1)$
     - & consolidate $O(1)$

   when done: left with trees of distinct degrees
   degree bound is $O(\lg n)$;
   So: If root list starts with k many trees afterwards — have $O(\lg n)$ many trees, so
   $\Delta\Phi = -(k - O(\lg n)) = O(\lg n) - k$
   ($-k$ pays for the traversal & combining trees)
   does not pay for reading off the array & returning the list $O(\lg n)$

Add it up: amortized cost is $O(\lg n)$

Starting with an empty heap, do
- a many inserts — $O(1)$
- b findMin — $O(1)$
- c DecreaseKey's — $O(1)$
- d Merge — $O(1)$
- e DeleteMin — $O(\lg n)$

→ total actual time of any such sequence of ops is
$O(a + b + c + d + e\lg n)$
where n is the max # items in the heap(s) at any given time
$\le$ # insertions

2. Prove that degree of any tree is $O(\lg n)$.

Fibonacci sequence
$F_0 := 0$
$F_1 := 1$
$F_2 := 1$
$F_3 := 2$
$\vdots$
$(n \ge 2)\quad F_n := F_{n-1} + F_{n-2}$

Claim: For $k \ge -1$, $1 + \sum_{j=0}^{k} F_j = F_{k+2}$

Proof: $k = -1$:
$F_{k+2} = F_1 = 1 + \underbrace{\sum_{j=0}^{-1} F_j}_{0} = 1$ ✓

$k = 0$
$F_{k+2} = F_2 = 1 = 1 + \sum_{j=0}^{0} F_j = 1 + F_0 = 1$ ✓

$k \ge 0$, assume true for $k-1$
$F_{k+2} = F_{(k-1)+2} + F_k$
$= \left(1 + \sum_{j=1}^{k-1} F_j\right) + F_k$
$= 1 + \sum_{j=1}^{k} F_j$ ✓ //claim

Let $\varphi := \frac{1+\sqrt{5}}{2} \approx 1.619$ (golden ratio)

satisfies $\varphi^2 = 1 + \varphi$:
$\left(\frac{1+\sqrt{5}}{2}\right)^2 = \frac{1+2\sqrt{5}+5}{4} = \frac{6+2\sqrt{5}}{4} = \frac{3+\sqrt{5}}{2}$
$= 1 + \frac{1+\sqrt{5}}{2} = 1 + \varphi$

Claim: $\forall k \ge 0$, $F_{k+2} \ge \varphi^k$

Proof: $k = 0$: $F_{k+2} = F_2 = 1 = \varphi^0$ ✓

induction $k = 1$: $F_{k+2} = F_3 = 2 > \varphi$

$k \ge 2$. $F_{k+2} = F_{k+1} + F_k$
$\ge \varphi^{k-1} + \varphi^{k-2}$ (ind hyp)
$= \varphi^{k-2}(\varphi + 1)$
$= \varphi^{k-2}\varphi^2 = \varphi^k$ ✓ //claim

Prop: Let x be any node in a fibheap. Let size(x) be the # of nodes in x's subtree (incl x itself). Let deg(x) be the degree (# of children) of x. Show that
$\text{size}(x) \ge F_{\deg(x)+2}$

Prop: $\text{size}(x) \geq F_{\deg(x)+1}$

Proof: Suppose $d = \deg(x)$.

Let $y_1, y_2, \ldots, y_d$ be the children of x in order that they became children of x

y_1 first, then y_2, ...

Consider some y_i $(1 \leq i \leq d)$

when y_i is made a child of x, x already had children $y_1, \ldots, y_{i-1}$, so when y_i was made a child of x, degree of x was at least $i-1$.

Thus at that time $\deg(y_i) \geq i-1$.

At the present time, $\deg(y_i) \geq i-2$

because y_i could only lose ≤ 1 child in the interim (b/c node marking)

$\forall i \geq 2$, y_i has degree $\geq i-2$

Prop is proved by induction on the height of a node

height $= 0$: one node, so size $= 1$ & degree $= 0$

$1 = F_1$

$y_1, \ldots, y_d$ have height $<$ height of x

apply the ind hyp to the y_i's:

$$\text{size}(y_i) \geq F_{\deg(y_i)+1} \geq F_{i-1}$$

So

$$\text{size}(x) = 1 + \sum_{i=1}^{d} \text{size}(y_i)$$

(the 1 counts x itself)

$$\geq 1 + \sum_{i=1}^{d} F_{i-1}$$

$$= 1 + \sum_{i=0}^{d-1} F_i = F_{d+1}$$

$$= F_{\deg(x)+1}$$

prop. //

For any tree node x

$n \geq \text{size}(x) \geq F_{\deg(x)+1} \geq \varphi^{\deg(x)-1}$

[n = heap size]

Take $\log_\varphi$ of both sides & add 1

$$\deg(x) \leq \log_\varphi n + 1 = O(\lg n)$$

∴ assumption

Analysis complete for fib heaps

Next topic: Disjoint Set Systems ("Union-Find" data struct)

n items, maintained in a family of disjoint sets

• = item

○ = set:

Each set has an item that is its designated representative (could be any item, doesn't matter)

Basic ops:

MakeSet(x) — takes a new item x, adds it to the set system as a 1-element set (singleton) (x is the representative)

Find(x) — x is an item in the system, returns (representative of) the set containing x

Find(x) = Find(y) = Find(z) = ...

x & y are in the same set iff Find(x) = Find(y).

Union(x,y) — x, y items in the system.

does nothing if x & y are in the same set.

Else, merges the set containing x with the set containing y into a single set, destroying the original sets.

New rep of combined set is one of the reps of the original sets.

Toy app: maze building