CSCE 750
10/31/2023

Merging Binomial Heaps
Fibonacci Heaps

Recall (last weak): to merge two binomial heaps:

Merge trees onto a single list in order by degree

Let L be this list.

$p := L$ initially

Let d_1, d_2, d_3 be the degrees of

p, $p.\text{right\_sibling}$, $p.\text{right\_sibling}.\text{right\_sibling}$

Convention: If ~~the 3rd tree~~ a tree does not exist, we take its degree to be ∞.

while $p.\text{right\_sibling} \neq \text{Nil}$ // p not pointing to last tree in list.

Let d_1, d_2, d_3 be as above

$O(1)$ If $d_1 < d_2$ or $(d_1 == d_2$ & $d_2 == d_3)$ then advance p $(p := p.\text{right\_sibling})$

else // $d_1 == d_2$ & $d_2 < d_3$

$O(1)$ → combine d_1 & d_2 (p & $p.\text{right\_sibling}$)

// p points to a tree of degree $d_1 + 1$

// don't advance p!

One can show that the final list L has trees in strictly increasing order by degree, so set $H.trees := L$.

Time to Merge H_1 (size m) with H_2 (size n)

$= \Theta(\text{length of } L)$

$\lg m + \lg n \in O(\lg(n+m))$

Claim: Don't need a degree field with nodes (provided you include $H.size$ attribute (total heap size)), without changing the asymptotic run time of any operation.

Hint: - degrees of children determined by deg of parent
- degree of roots determined by the binary rep of $H.size$.

Fibonacci Heaps supports (min-heaps)

- Insert(H, k)
- FindMin(H)
- ExtractMin(H) // same as DeleteMin
- Union(H_1, H_2) // same as Merge
- DecreaseKey(H, x, k)
- Delete(H, x)

operation	Binary heap	Binomial Heap	Fibonacci Heap
Insert(H, k)	$\Theta(\lg n)$	$\Theta(\lg n)$	$\Theta(1)$
FindMin(H)	$\Theta(1)$	$\Theta(\lg n)$	$\Theta(1)$
ExtractMin(H)	$\Theta(\lg n)$	$\Theta(\lg n)$	$\Theta(\lg n)$
Union(H_1, H_2)	$\Theta(n)$	$\Theta(\lg n)$	$\Theta(1)$
DecreaseKey(H, x, k)	$\Theta(\lg n)$	$\Theta(\lg n)$	$\Theta(1)$ ☆
Delete(H, x)	$\Theta(\lg n)$	$\Theta(\lg n)$	$\Theta(\lg n)$
	worst-case	worst-case	amortized

Structure: each node x has attrs

x.key

x.parent (a pointer)

x.child (a pointer to <u>any</u> of its children)

x.left, x.right } siblings of a common parent

x.degree

x.mark (boolean: true iff x has lost a child since the most recent time it was made a child of another node.)

The heap H has attrs

H.min — pointer to the ~~min key~~ min key (must be a root)

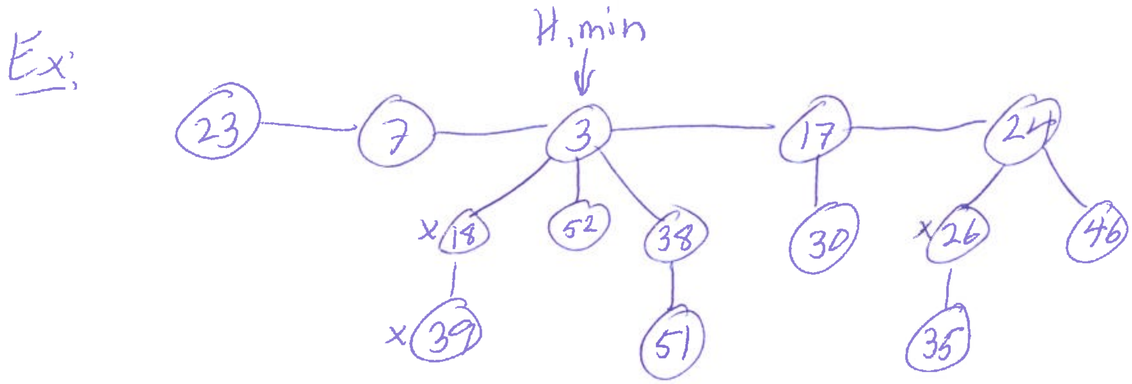
H.n — # of keys in H.

Roots are kept on a doubly linked circular list (using left & right attrs for links)

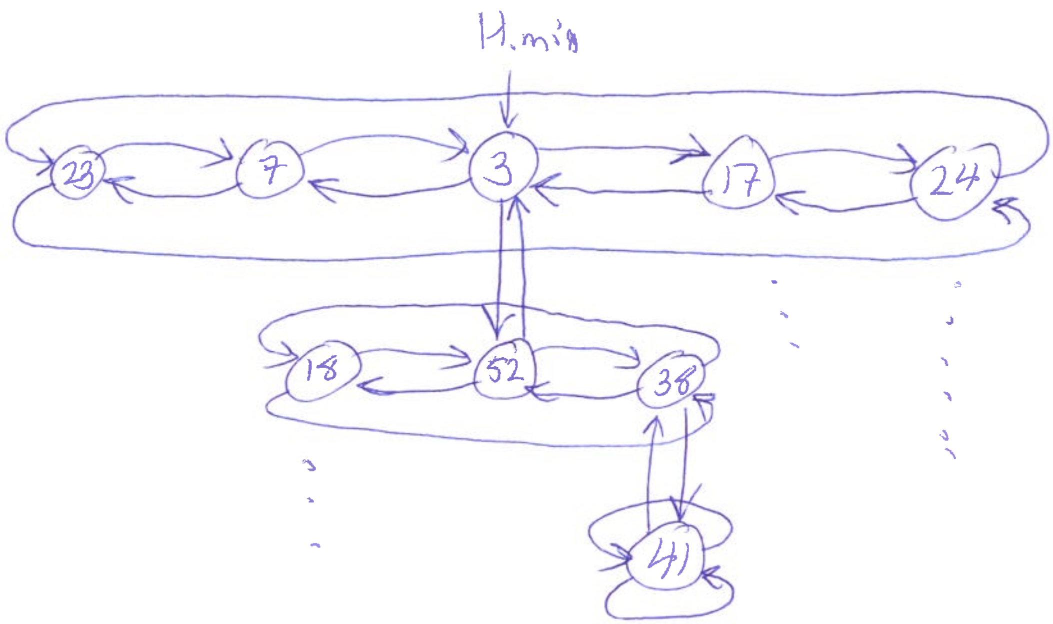
Each root is the root of a tree of keys in min heap order.

The children of any node are also in a doubly linked circular list (pointed to by ~~the~~ child attr).

Ex:



"x" means marked



Potential Function

$$\Phi(H) = t(H) + 2m(H) \quad \text{where}$$

- $t(H)$ = # of trees (roots) of H (not counting subtrees)
- $m(H)$ = # of marked nodes in H

Given a collection $H_1, \dots, H_n$ of Fib heaps.

$$\Phi(H_1, \dots, H_n) := \sum_{i=1}^{n} \Phi(H_i)$$

Simple Operations

Insert(H, k) — create new Fib heap H' containing just k, then Union(H, H')

- Actual runtime $\Theta(1 + \underbrace{\text{time to merge}}_{\Theta(1)}) = \Theta(1)$
- Amortized time $=$ actual time $+ \underbrace{\Delta\Phi}_{1} = \Theta(1)$

Union(H_1, H_2) — Join 2 lists of roots, select new H.min

Actual time: $\Theta(1)$

Amortized time: $\Theta(1)$ $\quad \Delta\Phi = 0$

ExtractMin

- Remove H.min from the root list
- Promote children of H.min by merging child list with root list
- Consolidate the heap, combining trees so that no two roots have the same degree.