

Mergeable Heaps

- Binomial Heaps
- Fibonacci Heaps

Data structures maintaining a dynamic set of items with keys, supporting

Insert

Delete

Min, Extract-Min

Update

Union

Binary heap for priority queue

Ops: insert, delete, extract-min (max)

A balanced binary tree satisfying the heap property –

Key of a node $>$ key of its parent

$O(n)$ nodes, $O(\log n)$ height.

Each op $O(h)=O(\log n)$

First Relaxation – leading to Binomial heaps

A collection of trees (not necessarily binary).

Heap property.

Insert to smallest tree.

No two trees of the same [→]size. (To contain cost of extract-min).

Example: Insert 5,4,3,2,1,7,8,9 → a binomial tree

Binomial tree

B_0 : a single node.

B_k : merging two B_{k-1} , root of one B_{k-1} becomes a child of the root of the other B_{k-1} .

Properties:

(1) $B(k, i) = \# \text{ nodes at level } i \text{ of } B_n.$

level i of B_k : level $i - 1$ of left B_{k-1} +
level i of left B_{k-1}

$$B(k, i) = B(k - 1, i - 1) + B(k - 1, i) = \binom{k}{i}$$

(2) B_k has 2^k nodes.

$$|B_k| = 2 |B_{k-1}|.$$

(3) B_k has height k .

$$h(B_0) = 0, h(B_k) = h(B_{k-1}) + 1.$$

(4) Root of B_k has k subtrees : B_0, \dots, B_{k-1} .

Degree of every node $\leq k$.

Binomial Heaps: a set of binomial trees s.t.

- Each tree satisfies the heap property.
- No two roots have the same degree.

$n_i = \# B_i$ in the heap $H = 0$ or 1 .

$$n := |H| = \sum_i n_i 2^i$$

$\Rightarrow \# \text{ trees in } H \leq \log n$.

Operations: Min, Union, Insert, Extract-min,
decrease, delete

(1) $\text{Min}(H) = \text{min of roots of the binomial trees.}$

$O(\log n)$

(2) $\text{Union}(H_1, H_2)$

Analogous to adding $|H_1|$ and $|H_2|$ in binary.

(3) $\text{Insert}(H, x)$: make x a Binomial heap H_1 , then
 $\text{Union}(H, H_1)$.

(4) Extract - Min(H)

Let x be the root with minimum key.

Cut the subtrees of x from x and create a new heap H_1 with them as member binomial trees.

Remove x from H .

Union (H, H_1).

$O(\log n) : \deg(x) \leq \log n$.

(5) Decrease(x) : Bubble - up the tree until heap property is restored. $O(\log n)$

(6) Delete(x)

Decrease $k(x)$ to " $-\infty$ ".

Extract - min

Fibonacci Heaps

General idea of relaxation – delay work as much as possible.

First relaxation – Delay enforcement until an extract-min. op; consolidate the work after an extract-min.

Defn Fibonacci heap

- A collection of unordered trees.
- Each satisfying the heap property.
- Roots are linked in a circular list.
- A pointer to the root with the minimum key.

A nice property to maintain --

Heap size be exponential in max. degree of nodes.

Implications:

- Max. degree of any node = $O(\log n)$
- $O(\log n)$ trees in the heap, if no two roots have the same degree.

Primitives—

$O(1)$ per op, including cost for maintaining $\min(H)$.

Create(x) -- A new node x is created.

Remove(x) – Node x, a singleton, is removed from the F-heap.

Cut(x)

- x cannot be a root.
- The subtree rooted at x is cut from $\text{parent}(x)$ and become a new tree.
- When – delete or update, or a second child of x is cut-off from x (cascade-cut).

Link(x,y)

- x and y are roots.
- $\text{degree}(x) = \text{degree}(y)$.
- $\text{key}(x) \geq \text{key}(y)$.
- When – only applied during consolidation.

Mark(x)

- x cannot be a root.
- When – after first cut happens to any child of x.

Policy

1. Heap property.
2. (Link policy) Two roots are linked only if their degrees are the same.
3. (Cut policy) After becoming a non-root, a node can have at most one child cut off. (2nd cut-off causes the node itself to be cut off from its parent and become a root.)
4. Consolidate only after an extract-min is performed.

Lemma 1

#links \leq #cuts + n_0 + #insertions

where $n_0 =$ #trees initially

Proof : $\Phi(H) :=$ #trees in H .

cut $\Rightarrow \Delta\Phi(i) = 1$

link $\Rightarrow \Delta\Phi(i) = -1$

insert $\Rightarrow \Delta\Phi(i) = 1$

$$\Delta\Phi = \sum_i \Delta\Phi(i) = \#cuts - \#links + \#insertions$$

$$\Delta\Phi = \Phi(H_N) - \Phi(H_0)$$

$$\#cuts - \#links + \Phi(H_0) = \Phi(H_N) \geq 0$$

Remark: proof does not depend on Policy.

Lemma 2 #cascade - cuts \leq #actual cuts.

Proof :

$\Phi(H) := \sum_v lost(v)$, where $lost(v) = \#$ children v has lost.

($lost(v)$ is initially 0; 0 for roots; $1 \Leftrightarrow$ marked; $2 \rightarrow$ cascade - cut)

Each time step - - an application of a primitive op.

i th step :

actual cut $\Rightarrow \Delta\Phi(i) \leq 1$.

cascade cut $\Rightarrow \Delta\Phi(i) \leq -1$.

$\Rightarrow \Delta\Phi = \sum_i \Delta\Phi(i) \leq \#$ actual cuts - # cascade - cuts

$\Delta\Phi = \Phi(T) - \Phi(0) = \Phi(T) \geq 0$. ($\Phi(0) = 0$)

$\Rightarrow \#$ cascade - cuts $\leq \#$ actual cuts.

Lemma 3

$$\# \text{ cuts} + \# \text{ links} \leq 4 \# \text{ actual cuts} + \# \text{ insertions} + n_0,$$

where $n_0 = \# \text{ trees initially}$

Proof :

Lemma 1 : $\# \text{ links} \leq \# \text{ cuts} + n_0 + \# \text{ insert}$.

Lemma 2 : $\# \text{ cascade - cuts} \leq \# \text{ actual cuts}$.

$$\Rightarrow \# \text{ cuts} \leq 2 \# \text{ actual cuts}$$

$$\# \text{ cuts} + \# \text{ links} \leq 2 \# \text{ cuts} + n_0 \leq 4 \# \text{ actual cuts} + \# \text{ insert} + n_0$$

Remark : $n_0 = 0 \Rightarrow \text{amortized cost} = O(1 + \# \text{ actual cuts})$

Lemma 4

Any node has degree $\leq 2 \log n + 1$, where $n = \# H$.

Remark : a consequence of Cut and Link policies.

Proof : $G(k) := \min \# \text{nodes in a subtree rooted at a node of degree } k$.

(min over all possible configurations of an F-heap which is initially empty.)

$w :=$ a node of degree k in an F-heap of size n .

$\text{deg}_t(w) :=$ degree of w at time t .

One step : application of one primitive op.

Suppose $\deg_t(w) = k$.

v_1, \dots, v_k : children of w at time t .

$t_i \leq t$: time at which v_i was linked to w

and remains a child of w to time t .

$t_1 < t_2 < \dots < t_k \leq t$.

$\deg_{t_i}(w) \geq i - 1$, since v_1, \dots, v_{i-1} were children of w

at that time.

So, $\deg_{t_i}(v_i) = \deg_{t_i}(w) \geq i - 1$ (two nodes are linked only if they have the same degree).

From time t_i to t , v_i has at most one child cut off, else v_i would have been cut off from w .

(Inv.3 - Cut policy)

So, $\deg_t(v_i) \geq \deg_{t_i}(v_i) - 1 \geq i - 2$.

| Subtree rooted at v_i at time t | $\geq G(i - 2)$

| Subtree rooted at w at time t | \geq

$G(k - 2) + G(k - 3) + \dots + G(1) + 1 + 1 + 1$.

So, $G(k) \geq G(k - 2) + G(k - 3) + \dots + G(1) + 1 + 1 + 1$.

Fibonacci sequence :

$$F(1) = F(2) = 1,$$

$$F(k) = F(k-1) + F(k-2).$$

$$F(k) \geq \alpha^{k-2}, \alpha = \frac{\sqrt{5}+1}{2}.$$

$$F(k+2) = F(k) + \dots + F(1) + 1.$$

$$G(0) = 1, G(1) = 2.$$

Inductively assume $G(i) \geq F(i+2), i = 0, \dots, k-1$.

$$\begin{aligned} G(k) &\geq G(k-2) + G(k-3) + \dots + G(1) + 1 + 1 + 1 \\ &\geq F(k) + F(k-1) + \dots + F(3) + 1 + 1 + 1 \\ &= F(k+2). \end{aligned}$$

$$n \geq G(k) \geq F(k+2) \geq \alpha^k \Rightarrow k \leq \log_{\alpha} n.$$

Lemma 5

A Fibonacci heap of size n has no more than $2 \log n + 1$ trees if no two roots are of the same degree.

Proof :

Lemma 4 \Rightarrow degree of a node $\leq 2 \log n$.

Amortized cost of a data structure operation

σ : a data structure operation such as delete, extract - min.

$\hat{c}(\sigma) := 1 + \# \text{ actual cuts due to } \sigma$

$\sigma_1, \dots, \sigma_n$: a sequence of ops

Then

$$\sum_{i=1}^n c(\sigma_i) \leq \text{total \# cuts} + \text{total \# links} + n$$

$$= O(\text{total \# actual cuts} + n).$$

$$= O\left(\sum_{i=1}^n \hat{c}(\sigma_i)\right)$$

Extract-min

- The root with the minimum key is extracted.
- The subtrees of this root become new trees.
- Consolidation work – merge trees using link op until no two roots have the same degree.

Amortized cost = $O(\log |H|)$.

Delete(x)

- cut(x), then cut (y) for each child y of x.
- Remove(x).

Amortized cost = $O(\deg(x))=O(\log|H|)$.

Decrease(x)

Update on x and if heap property is violated, then cut(x).

Amortized cost = $O(1)$.