Binomial Heaps

((Mergeable Heaps))

Binomial Trees

Def: A binomial tree of height \( k \) is a rooted ordered tree whose shape is specified inductively:

\( k = 0 \): \( B_0 \) (one node)

\( k > 0 \): \( B_k \) constructed from 2 \( B_{k-1} \)'s like this:

- new leftmost child of root

size of \( B_k = 2^k \)
height of \( B_k = k \)

\( B_0 \cup B_1 \cup B_2 \)

\( B_3 \)

Degree of a node is \# of children.
Degree of root of \( B_k = k \)
all other nodes have degree < k.

\[ \text{B}_k \]

(all easy by induction).

\[ \text{max degree of any node in binomial tree of size } n \text{ is } \log_2 n. \text{ (} n = 2^k \text{ and all nodes of degree } \leq k. ) \]

A Binomial Heap \( H \) is a list of binomial trees \( B_{k_1}, B_{k_2}, \ldots, B_{k_n} \) so that:

- \( k_1 < k_2 < \ldots < k_n \)
- Each \( B_{k_i} \) contains items with key values in its nodes (one item per node. (Assuming min heap) These items are in min-heap order.

The items of \( H \) are all items contained in the trees in the list.

H empty: empty list.

Spec \( H \) has size \( n > 0 \)
let $k_0 < k_1 < \ldots < k_{l-1}$
be heights of the trees
in $H$’s list.
Since $k_{l-1} \geq l-1$,
\[
    n = \sum_{i=0}^{l-1} k_i \geq 2^{l-1}.
\]
\[
    l = O(\log n)
\]
Exactly one way to represent
any $n \geq 0$ as a sum of
distinct powers of 2,
so shape of $H$ is uniquely
determined by its size.

Implementing $H$:
Use usual linked rep
for each tree, with
\[
    \begin{align*}
    \text{parent} & \quad \text{other-fields:} \\
    \text{leftmost-child} & \quad \text{key} \\
    \text{right-sibling} & \quad \text{satellite} \\
    \text{fields} & \quad \text{data} \\
    \end{align*}
\]
Link tree together in list
by the right-sibling
fields of their roots.

Basic Ops:
Making an empty $H$:
\[
    \text{head}[H] \text{ points to list of trees. Set to null.}
\]
\[
    \mathcal{O}(1) \text{ time}
\]
Finding the minimum in H: search roots of trees for minimum key value and return it. 

$O(\log n)$ where 

$n =$ size of $H$ 

( # trees is $O(\log n)$ )

Merging two binomial heaps $H_1$, $H_2$.

 Traverse list of roots of both heaps, merge the lists in order of increasing height.

 Problem: duplicate tree heights.

 May encounter a $B_k$ in $H_1$ and a $B_k$ in $H_2$

 if $x \leq y$, combine:

 if $x > y$, swap trees; always make lesser key new root.
Takes $O(1)$ time!
- Compare root values
- Insert one root at the front of the linked list of the other root's children.

Now may have up to $3B_{k+1}$'s. Combine 2 of them and include the third (if it exists) in the result. Continue.

[like binary add with carry]

Time: $n = \text{size of new heap linear in list lengths}$
\[= O(\lg n) \]

Inserting into a binomial heap $H$ a new item $x$:
- $H$
- Form $H_1$ with just $x$ (in $B_0$)
- Merge $H$ with $H_1$

Total time $= O(\lg n)$.

\underline{ExtractMin}:
- Find tree containing minimum $O(\lg n)$ time
-unlink this tree from H's list
0(1) time

reverse temp list.
temp now points to a binomial heap.
list has length \( k \leq \lg n \),
so reversal takes time \( O(\lg n) \).
Merge \( H \) with this temp heap. \( O(\lg n) \) time
\( O(\lg n) \) time total.

Decrease Key: same
For a binary heap:
cascade (bubble) up
the node with decreased key towards the root.
Time: \( O(\text{height of tree}) \)
\( = O(\lg n) \).
Delete (x):
  O(1n) Decrease x's key to -\infty (some small value)
  O(1n) ExtractMin

Disjoint set systems.
Collection \( C = \{S_1, \ldots, S_k\} \)
where each \( S_i \) is a set of items, and \( S_i \cap S_j = \emptyset \)
for \( i \neq j \).

3 basic ops:
MakeSet(x): create
a new set whose
only element is x.
(x not in any other
set of the system).

Find(x) finds the
unique set containing x

Union(x,y): merges the
set containing x with
the set containing y.
(the two original sets
are destroyed).

Represent each set by some
distinguished element of the
called the representative of the set.

\(\text{Find}(x)\) returns the representative of the set containing \(x\).

\(x\) and \(y\) are in same set iff \(\text{Find}(x) = \text{Find}(y)\).