Binomial Queues

Section 6.8
Heap Operations: Merge

Given two binary heaps $H_1$ and $H_2$, produce a new heap $H'$ combining $H_1$ and $H_2$

- Binary heaps take $\Theta(n_1 + n_2)$ time to merge
- i.e. they can never merge in better than linear time

We can do better, however

- Merge in $O(\log N)$ time
- this comes at the expensive of a slight performance hit on our other operations
Binomial Trees

- **Binomial trees** are recursive defined
- Start with one node
  - This is a binomial tree of **height 0**
  - To form a tree of height $k$, attach two trees of height $k - 1$ together
    - Attach one as a child of the root of the other
$B_4$
Binomial Tree Size

- A binomial tree of height $k$ has $2^k$ nodes

- Conversely, a binomial tree with $n$ nodes has $\log_2(n)$ height

- The number of nodes at level $d$ of a tree with height $k$ is the binomial coefficient:

$$\binom{k}{d} = \frac{k!}{d!(k-d)!}$$
Binomial Queues

- **Binomial Heaps / Binomial Queues**
  - use a *forest* of binomial trees
    - use each binomial tree \(0,1\) times
  - impose heap ordering on each binomial tree
  - no relationship between the roots of each tree
Binomial Queues

$H_1$:
Binomial Queue Size

- A binomial queue $H$ with $N$ nodes has $O(\log N)$ binomial trees.
  - Let $k$ be the largest integer such that $2^k \leq N$.
  - Observe that $k \leq \log_2(N)$.
  - $N$ can be written as the sum of unique powers of 2, the largest of which is $2^k$.
    - This sum uses each power of 2 \{0,1\} times.
  - The sum has at most $k + 1$ terms in it.
  - Each term corresponds to a binomial tree of $2^n$ nodes in the forest of $H$. 
Merge

- “Add” corresponding trees from the two forests
- For $k$ from 0 to maxheight
  - If neither queue has a $B_k$, skip
  - If only 1, leave it
  - If two, attach the larger priority root as a child of the other, producing a tree of height $k + 1$
  - If three, pick two to merge, leave 1

$H_1$:  

$H_2$:  

$O(\log N)!$
After Merging $H_1$ and $H_2$
To insert a node $X$ into a binomial queue $H$:

- Observe that a single node is a binomial tree of height 0
- So treat $X$ as a binomial queue
- Merge $X$ and $H$

Merge operation takes $\log(N)$ time

Therefore so does insert
insert(1)
insert(2)
insert(3)
insert(4)
insert(5)
insert(6)
insert(7)
**deleteMin**

- To **deleteMin** from a binomial queue $H$
- Find the binomial tree with the smallest root, let this be $B_k$
- Remove $B_k$ from $H$, leaving the rest of the trees to form queue $H'$
  - Delete (and return to user) the root of $B_k$
    - this leaves us with the children of $B_k$’s root, which are binomial trees of size $B_0, B_1, ..., B_{k-1}$
    - then let the trees $B_0, B_1, ..., B_{k-1}$ form a new binomial queue $H''$
- Merge $H'$ and $H''$ to repair the tree

Also $O(\log N)$!
deleteMin
deleteMin

$H''$: 

12
  /   
21   24
  
14

65

26

16

18

21

24

14

65

26

16

18
deleteMin

$H'$:

$H''$:
deleteMin
Non-Standard Operations

- percolateUp
  - identical to binary heap
- decreaseKey
  - percolateUp as far as root of binomial tree

- delete (an arbitrary node)
  - decreaseKey to $-\infty$, then deleteMin