Yet Another Data Structure: Binomial Queues

- Structural property
  - Forest of binomial trees with at most one tree of any height

- Order property
  - Each binomial tree has the heap-order property

The Binomial Tree, $B_h$
- $B_h$ has height $h$ and exactly $2^h$ nodes
- $B_h$ is formed by making $B_{h-1}$ a child of another $B_{h-1}$
- Root has exactly $h$ children
- Number of nodes at depth $d$ is binomial coeff. $\binom{h}{d}$
  - Hence the name; we will not use this last property

Binomial Queue with $n$ elements
Binomial Q with $n$ elements has a unique structural representation in terms of binomial trees!
Write $n$ in binary: $n = 1101_{(base\ 2)} = 13_{(base\ 10)}$
Properties of Binomial Queue

- At most one binomial tree of any height

- \( n \) nodes \( \Rightarrow \) binary representation is of size ?
  \( \Rightarrow \) deepest tree has height ?
  \( \Rightarrow \) number of trees is ?

*Define:* \( \text{height(forest } F) = \max_{\text{tree } T \in F} \{ \text{height}(T) \} \)

*Binomial Q with \( n \) nodes has height \( \Theta(\log n) \)

Operations on Binomial Queue

- Will again define *merge* as the base operation
  - insert, deleteMin, buildBinomialQ will use merge

- Can we do increaseKey efficiently?
  decreaseKey?

- What about findMin?

Merging Two Binomial Queues

Essentially like adding two binary numbers!

1. Combine the two forests
2. For \( k \) from 0 to max height {
   a. \( m \leftarrow \) total number of \( B_k \)'s in the two BQs
   b. if \( m=0 \): continue;
   c. if \( m=1 \): continue;
   d. if \( m=2 \): combine the two \( B_k \)'s to form a \( B_{k+1} \)
   e. if \( m=3 \): retain one \( B_k \) and combine the other two to form a \( B_{k+1} \)
}

*Claim:* When this process ends, the forest has at most one tree of any height

Example: Binomial Queue Merge

- H1:
- H2:
**Example: Binomial Queue Merge**

H1:  
H2:

```
1
/|
2 1 3
/ \|
5 1 5 7 3
/ \  /\  \
6 4 2 4 6 7
```

**Complexity of Merge**

Constant time for each height  
Max height is: \( \log n \)

\[ \Rightarrow \text{worst case running time} = \Theta(\quad ) \]

**Insert in a Binomial Queue**

Insert\((x)\): Similar to leftist or skew heap

- **runtime**  
  Worst case complexity: same as merge  
  \( O(\quad ) \)

- Average case complexity: \( O(1) \)

Why??  
*Hint: Think of adding 1 to 1101*

**deleteMin in Binomial Queue**

Similar to leftist and skew heaps....
deleteMin: Example

find and delete smallest root

merge BQ (without the shaded part) and BQ'

Result:

runtime: