Today’s Outline

- Announcements
  - Written HW #2 due NOW
  - Project 2A due next Friday, 1/29
  - Written HW #3 due next Monday, 2/1

- Today’s Topics:
  - Priority Queues
    - Skew Heaps
    - Binomial Queues

Yet Another Data Structure: Binomial Queues

- Structural property
  - Forest of binomial trees with at most one tree of any height
    - What’s a forest?
    - What’s a binomial tree?

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

The Binomial Tree, \( B_h \)

- Height \( h \)
- 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

The Binomial Tree, \( B_h \)

- Number of nodes at depth \( d \) is binomial coeff. \( \binom{h}{d} \)
  - Hence the name; we will not use this last property
- Every subtree of a binomial tree is a binomial tree

Binomial Queues

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

- Order property
  - Each binomial tree has the heap-order property
Binomial Queue with \( n \) elements

Binomial Q with \( n \) elements has a unique structural representation in terms of binomial trees!

Every binomial Q with \( n \) elements has this structure

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

\( \text{Binomial Q with } n \text{ nodes has height } \Theta(\log n) \)

Operations on Binomial Queue

- Will again define \textit{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 1 to maxheight {
   a. \( m \leftarrow \text{total number of } B_k\text{'s in the two BQs} \)
   b. if \( m=0 \): continue; \( 0+0 = 0 \)
   c. if \( m=1 \): continue; \( 1+0 = 1 \)
   d. if \( m=2 \): combine the two \( B_k\text{'s to form a } B_{k+1} \)
      \( 1+1 = 0+c \)
   e. if \( m=3 \): retain one \( B_j \) and combine the other two to form a \( B_{k+1} \)
      \( 1+1+c = 1+c \)
   }

**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: 

Example: Binomial Queue Merge

H1: 
H2: 

Example: Binomial Queue Merge

H1: 
H2: 

Example: Binomial Queue Merge

H1: 
H2: 

Example: Binomial Queue Merge

H1: 
H2: 

Complexity of Merge

Constant time for each tree
Max height is:
Number of trees is:
⇒ worst case running time = $\Theta( )$

Insert in a Binomial Queue

Insert(x): Similar to leftist or skew heap

runtime
Worst case complexity: same as merge
O( )

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: