CSE 326: Data Structures

Binomial Queues

Brian Curless

Spring 2008

Announcements (4/14/08)

HW #2 due on Friday at beginning of class.

Can still partner up this afternoon.

Comparing Heaps

Worst case [avg/amort. case]	**Binary Heap**	**Leftist Heap**	**Skew Heap**
insert	O(log n) [O(1)]	O(log n)	O(n) [O(log n)]
findMin	O(1)	O(1)	O(1)
deleteMin	O(log n)	O(log n)	O(n) [O(log n)]
merge	O(n)	O(log n)	O(n) [O(log n)]
build	O(n)	O(n)	O(n)

Our Last Priority Queue

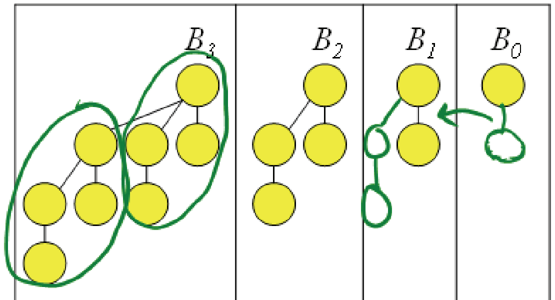
In exchange for getting fast merges with leftist and skew heaps, we have lost the fast (average case) inserts of binary heaps.

We'll look at one more priority queue data structure that doesn't make this trade-off: **binomial queues**.

But first, a brief diversion into **binomial *trees***…

Binomial Trees

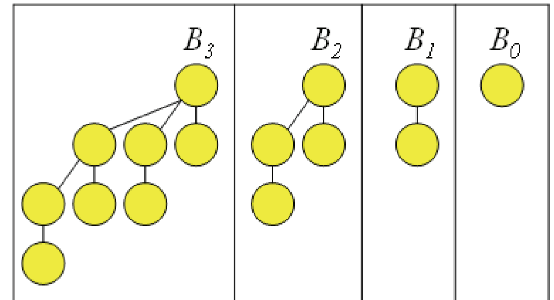
- A binomial tree B_h has height h and exactly 2^h nodes.
- B_h is formed by making B_{h-1} a child of another B_{h-1}.



Height (h)	3	2	1	0
number of elements	$2^3 = 8$	$2^2 = 4$	$2^1 = 2$	$2^0 = 1$

Binomial Trees

- The root of B_h has exactly h children.
- Every subtree of a binomial tree is also a binomial tree.



Height (h)	3	2	1	0
number of elements	$2^3 = 8$	$2^2 = 4$	$2^1 = 2$	$2^0 = 1$

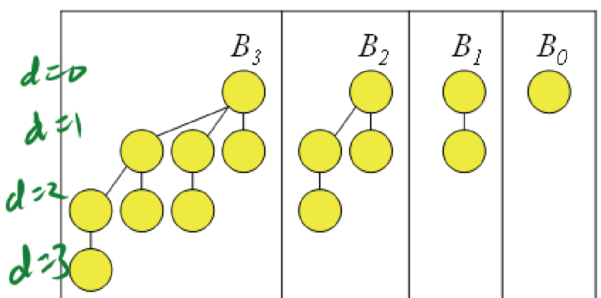
Binomial Trees

Number of nodes at depth d is a binomial coeff.:

$$\binom{h}{d} = \frac{h!}{(h-d)!\,d!}$$

Pascal's triangle:

1
1 1
1 2 1
1 3 3 1
d=0 d=1 d=2 d3



Height (h)	3	2	1	0
number of elements	$2^3 = 8$	$2^2 = 4$	$2^1 = 2$	$2^0 = 1$

Binomial Queues

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

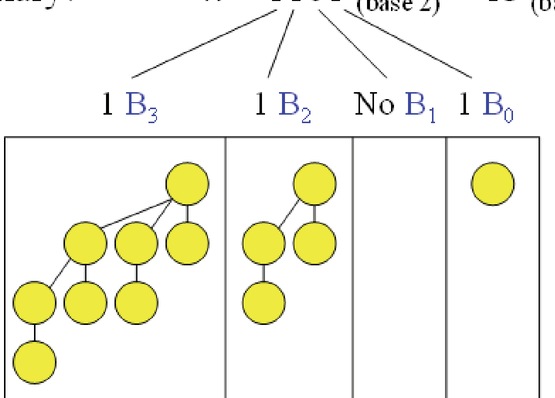
What's a forest?

- 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!

Write n in binary: $n = 1101_{\text{(base 2)}} = 13_{\text{(base 10)}}$

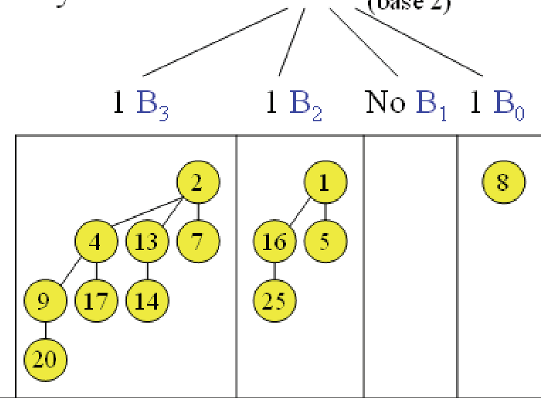


height (h)	3	2	1	0
number of elements	$2^3 = 8$	$2^2 = 4$	$2^1 = 2$	$2^0 = 1$

Binomial Queue with n elements

Each binomial tree obeys heap order.

Write n in binary: $n = 1101_{\text{(base 2)}} = 13_{\text{(base 10)}}$

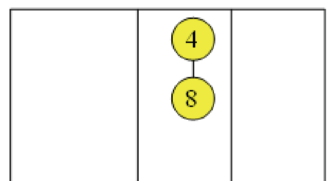


height (h)	3	2	1	0
number of elements	$2^3 = 8$	$2^2 = 4$	$2^1 = 2$	$2^0 = 1$

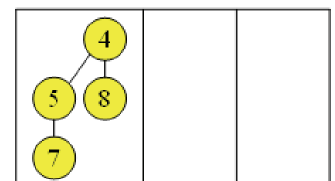
Properties of Binomial Queue

- At most one binomial tree of any height.
- n nodes
  - $\Rightarrow$ # of bits in binary representation of n: $\lfloor \log n \rfloor + 1$ $O(\log n)$
  - $\Rightarrow$ number of trees: $\leq \lfloor \log n \rfloor + 1 \in O(\log n)$
  - $\Rightarrow$ deepest tree has height: $O(\log n)$
- Is each subtree a binomial queue? Yes.

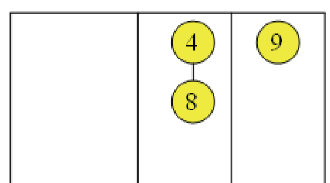
A Few More Examples



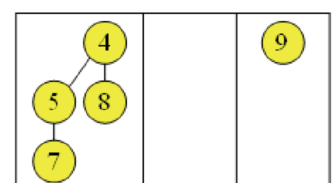
$N = 2_{10} = 10_2$	$2^2 = 4$	$2^1 = 2$	$2^0 = 1$



$N = 4_{10} = 100_2$	$2^2 = 4$	$2^1 = 2$	$2^0 = 1$



$N = 3_{10} = 11_2$	$2^2 = 4$	$2^1 = 2$	$2^0 = 1$



$N = 5_{10} = 101_2$	$2^2 = 4$	$2^1 = 2$	$2^0 = 1$

How to merge queues?

- There is a direct correlation between
  - the number of nodes in the tree
  - the representation of that number in base 2
  - and the actual structure of the tree
- When we merge two queues, the number of nodes in the new queue is the *sum of* N_1+N_2
- We can use these facts to help see how fast merges can be accomplished
- E.g., add 3+3 in base 2 arithmetic:

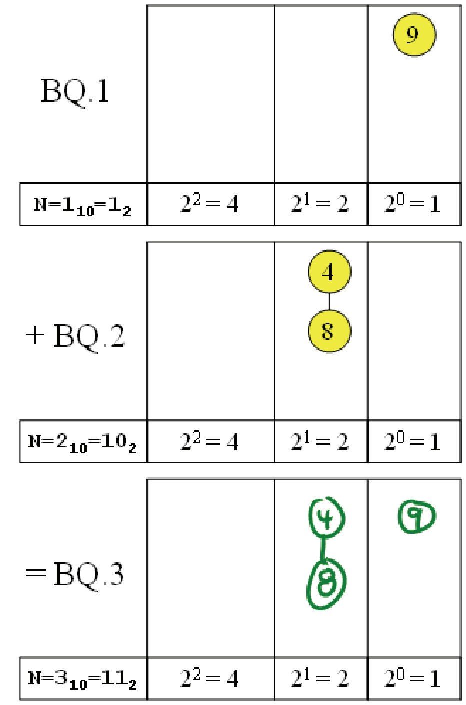
$$\begin{array}{r} 11 \\ +\ 11 \\ \hline 110 \end{array}$$

Example 1.

Merge BQ.1 and BQ.2

Easy Case.

There are no comparisons and there is no restructuring.

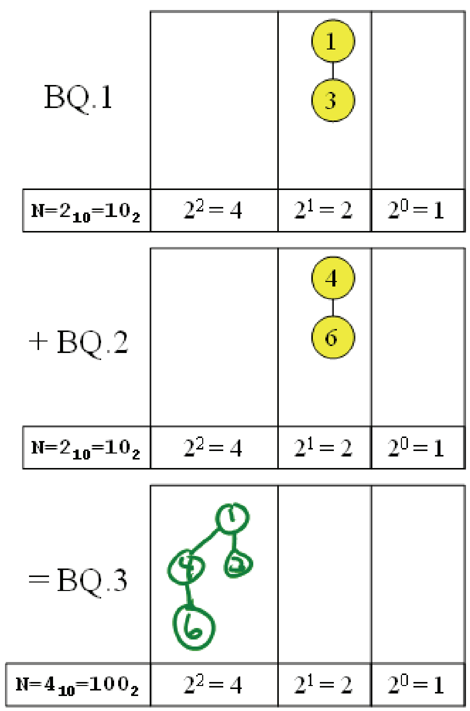


Example 2.

Merge BQ.1 and BQ.2

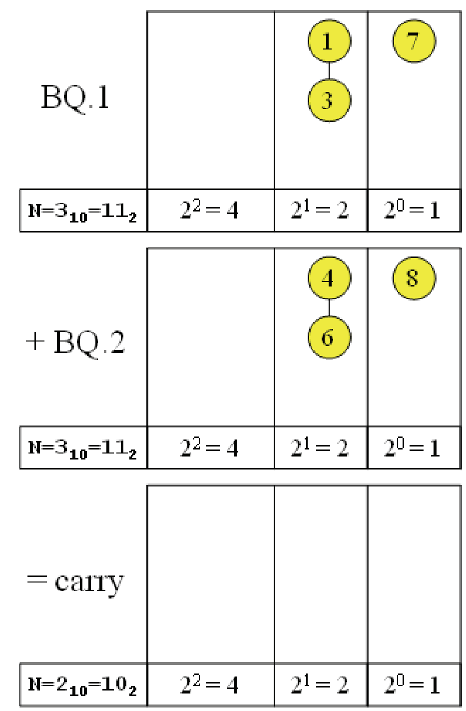
This is an add with a carry out.

It is accomplished with one comparison and one pointer change: O(1)



Example 3.

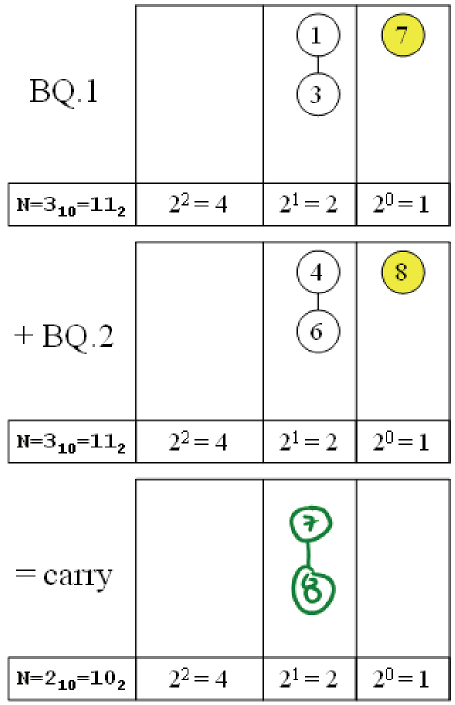
Merge BQ.1 and BQ.2



Example 3.

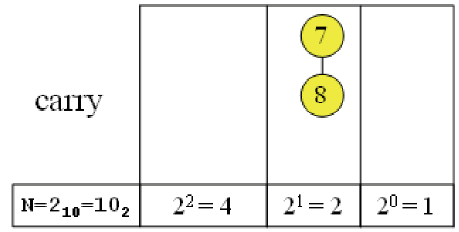
Merge BQ.1 and BQ.2

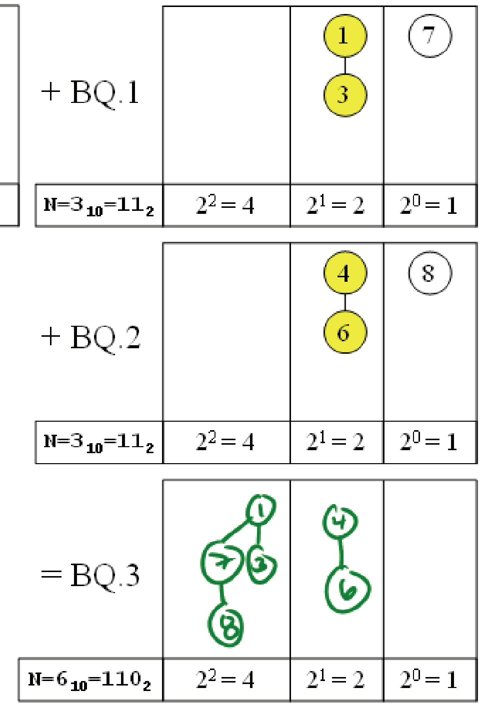
Part 1 – Add B_0 trees, form the carry.



Example 3.

Part 2 - Add the existing B_1 trees and the carry.





Merge Algorithm

- Just like binary addition algorithm
- Assume trees $X_0,\ldots,X_t$ and $Y_0,\ldots,Y_t$ are binomial queues
  - X_i and Y_i are of type B_i or null

```
C0 := null; //initial carry is null//
for i = 0 to t do
  combine Xi,Yi, and Ci to form Zi and new Ci+1
Zt+1 := Ct+1
```

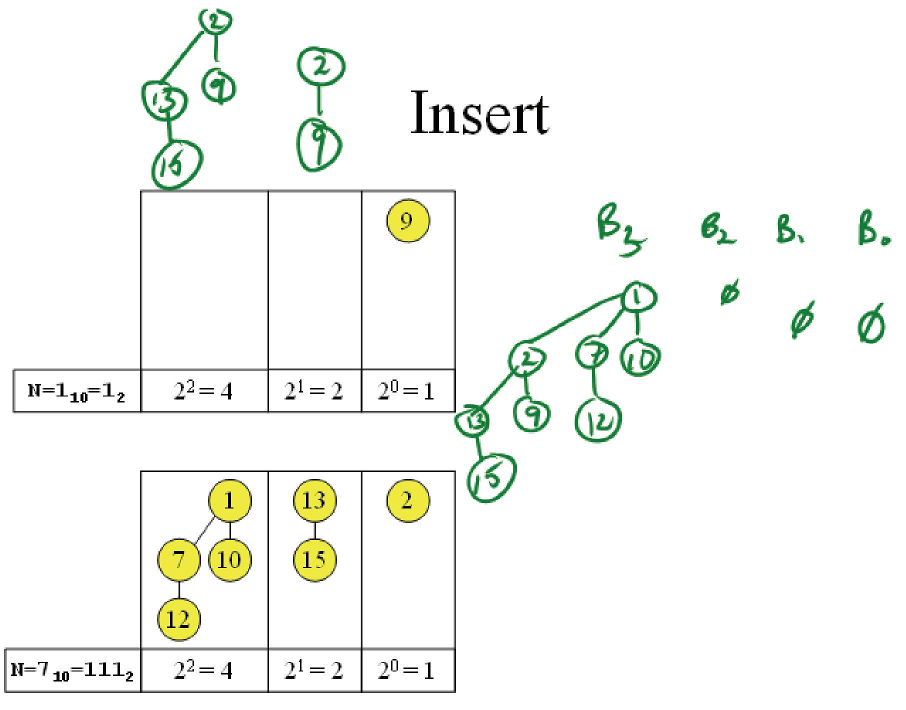
Complexity of Merge

Constant time for each tree.

Max number of trees is: $O(\log n)$

$\Rightarrow$ worst case running time = $O(\log n)$

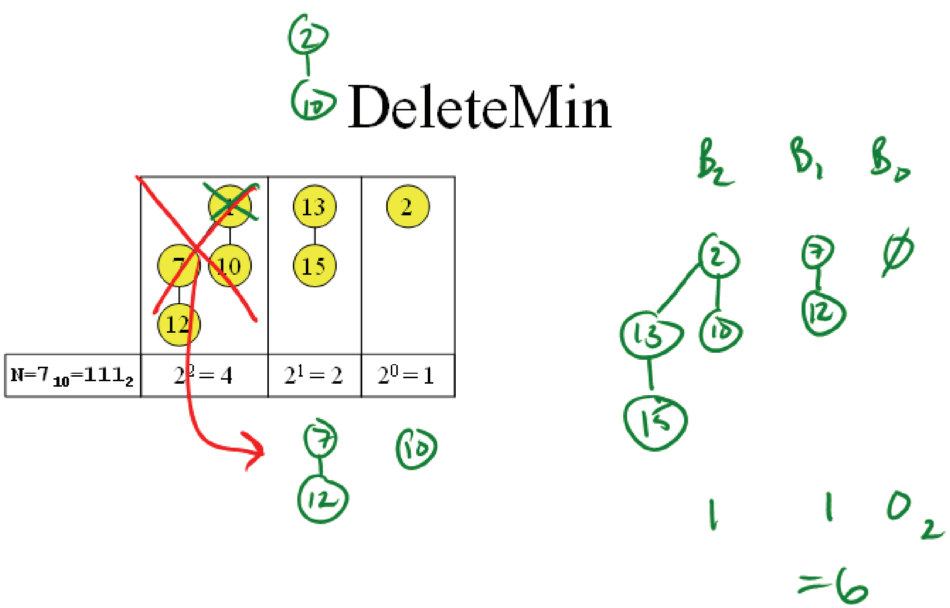
Insert



Insert

- Create a single node queue B_0 with the new item and merge with existing queue.
- Total time (worst case) = O(log n)
- Total time (average case) = O(1)
  - Hint: Think of adding 1 to 1101

DeleteMin



DeleteMin

1. Assume we have a binomial queue $X_0, \ldots, X_m$
2. Find tree X_k with the smallest root
3. Remove X_k from the queue
4. Remove root of X_k (return this value)
   - This yields a binomial queue $Y_0, Y_1, \ldots, Y_{k-1}$.
5. Merge this new queue with remainder of the original (from step 3)

- Total time (worst case) = O(log n)

More Operations on Binomial Queue

- buildBinomialQ can be done with repeated inserts in O(n) time.
- Can we do decreaseKey efficiently? increaseKey?
- What about findMin?

0
1
10
11
100

Comparing Heaps

Worst case [avg/amort. case]	Binary Heap	Leftist Heap	Skew Heap	Binomial Queue
insert	O(log n) [O(1)]	O(log n)	O(n) [O(log n)]	O(log n) [O(1)]
findMin	O(1)	O(1)	O(1)	O(log n)
deleteMin	O(log n)	O(log n)	O(n) [O(log n)]	O(log n)
merge	O(n)	O(log n)	O(n) [O(log n)]	O(log n)
build	O(n)	O(n)	O(n)	O(n)