

Binomial Queues

CSE 326
Data Structures
Lecture 15

Reading

- Reading
 - › Section 6.8, 11.2

3/2/05

Binomial Queues - Lecture 15

2

Merging heaps

- Binary Heap is a special purpose hot rod
 - › FindMin, DeleteMin and Insert only
 - › does not support fast merges of two heaps
- For some applications, the items arrive in prioritized clumps, rather than individually
- Is there somewhere in the heap design that we can give up a little performance so that we can gain faster merge capability?

3/2/05

Binomial Queues - Lecture 15

3

Binomial Queues

- Binomial Queues are designed to be merged quickly with one another
- Using pointer-based design we can merge large numbers of nodes at once by simply pruning and grafting tree structures
- More overhead than Binary Heap, but the flexibility is needed for improved merging speed

3/2/05

Binomial Queues - Lecture 15

4

Worst Case Run Times

	<u>Binary Heap</u>	<u>Binomial Queue</u>
Insert	$\Theta(\log N)$	$\Theta(\log N)$
FindMin	$\Theta(1)$	$O(\log N)$
DeleteMin	$\Theta(\log N)$	$\Theta(\log N)$
Merge	$\Theta(N)$	$O(\log N)$

3/2/05

Binomial Queues - Lecture 15

5

Binomial Queues

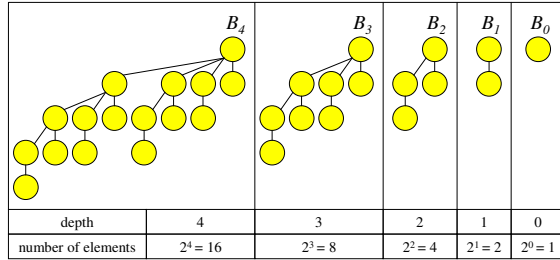
- Binomial queues give up simplicity in order to provide $O(\log N)$ merge performance
- A **binomial queue** is a collection (or *forest*) of heap-ordered trees
 - › Not just one tree, but a collection of trees
 - › each tree has a defined structure and capacity
 - › each tree has the familiar heap-order property

3/2/05

Binomial Queues - Lecture 15

6

Binomial Queue with 5 Trees



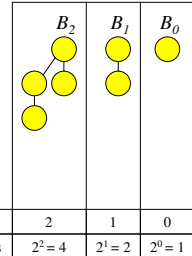
3/2/05

Binomial Queues - Lecture 15

7

Structure Property

- Each tree contains two copies of the previous tree
 - the second copy is attached at the root of the first copy
- The number of nodes in a tree of depth d is exactly 2^d



3/2/05

Binomial Queues - Lecture 15

8

Powers of 2

- Any number N can be represented in base 2
 - A base 2 value identifies the powers of 2 that are to be included

$2^3 = 8_{10}$	$2^2 = 4_{10}$	$2^1 = 2_{10}$	$2^0 = 1_{10}$		
1	1	1		Hex ₁₆	Decimal ₁₀
1	0	0		3	3
1	0	0		4	4
1	0	1		5	5

3/2/05

Binomial Queues - Lecture 15

9

Numbers of nodes

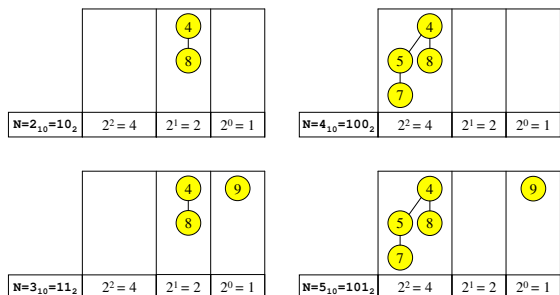
- Any number of entries in the binomial queue can be stored in a forest of binomial trees
- Each tree holds the number of nodes appropriate to its depth, ie 2^d nodes
- So the structure of a forest of binomial trees can be characterized with a single binary number
 - $100_2 \rightarrow 1 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 = 4$ nodes

3/2/05

Binomial Queues - Lecture 15

10

Structure Examples



What is a merge?

- 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 that fact to help see how fast merges can be accomplished

3/2/05

Binomial Queues - Lecture 15

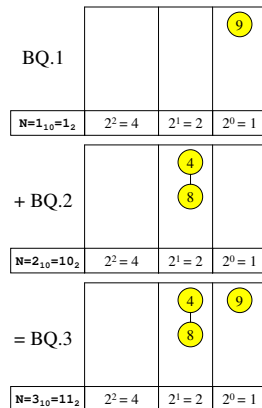
12

Example 1.

Merge BQ.1 and BQ.2

Easy Case.

There are no comparisons and there is no restructuring.

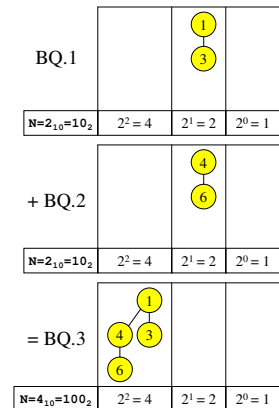


Example 2.

Merge BQ.1 and BQ.2

This is an add with a carry out.

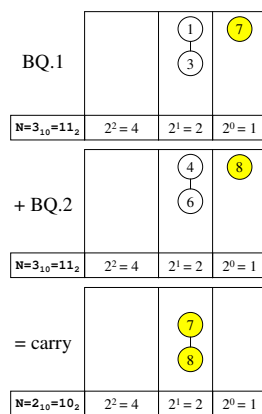
It is accomplished with one comparison and one pointer change: $O(1)$



Example 3.

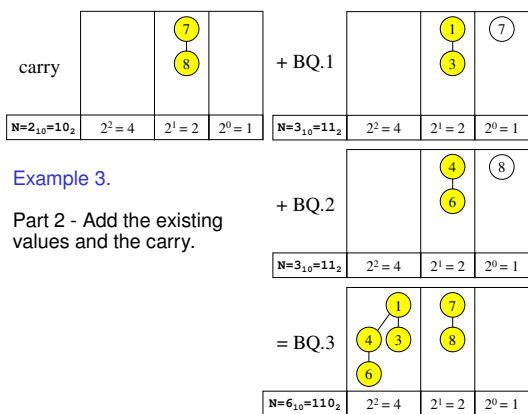
Merge BQ.1 and BQ.2

Part 1 - Form the carry.



Example 3.

Part 2 - Add the existing values and the carry.



Merge Algorithm

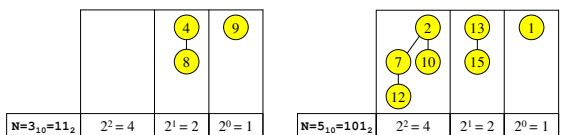
- Just like binary addition algorithm
- Assume trees X_0, \dots, X_n and Y_0, \dots, Y_n are binomial queues
 - X_i and Y_i are of type B_i or null

```

C0 := null; //initial carry is null//
for i = 0 to n do
    combine Xi, Yi, and Ci to form Zi and new Ci+1
Zn+1 := Cn+1

```

Exercise



$O(\log N)$ time to Merge

- For N keys there are at most $\lceil \log_2 N \rceil$ trees in a binomial forest.
- Each merge operation only looks at the root of each tree.
- Total time to merge is $O(\log N)$.

3/2/05

Binomial Queues - Lecture 15

19

Insert

- Create a single node queue B_0 with the new item and merge with existing queue
- $O(\log N)$ time

3/2/05

Binomial Queues - Lecture 15

20

DeleteMin

- Assume we have a binomial forest X_0, \dots, X_m
 - Find tree X_k with the smallest root
 - Remove X_k from the queue
 - Remove root of X_k (return this value)
 - This yields a binomial forest Y_0, Y_1, \dots, Y_{k-1} .
 - Merge this new queue with remainder of the original (from step 3)
- Total time = $O(\log N)$

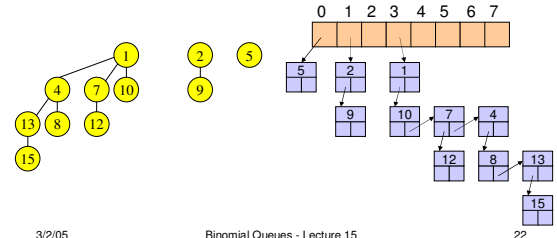
3/2/05

Binomial Queues - Lecture 15

21

Implementation

- Binomial forest as an array of multiway trees
 - FirstChild, Sibling pointers

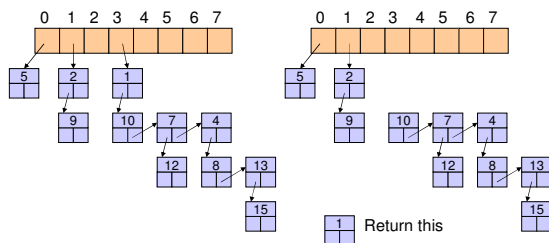


3/2/05

Binomial Queues - Lecture 15

22

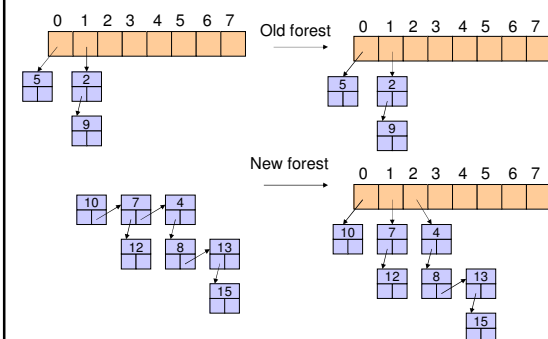
DeleteMin Example



3/2/05

Binomial Queues - Lecture 15

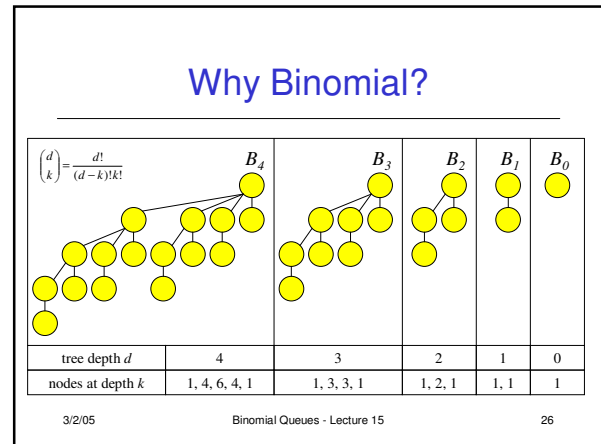
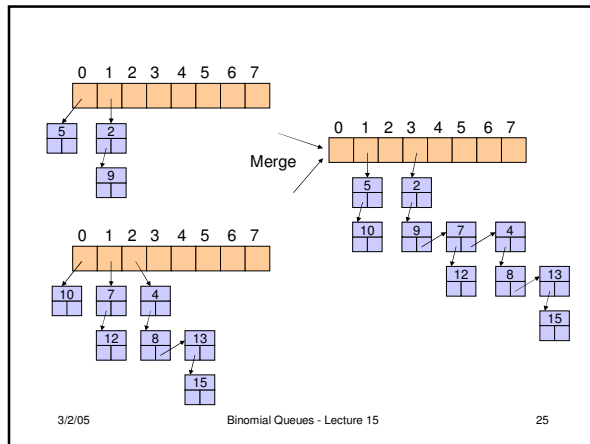
23



3/2/05

Binomial Queues - Lecture 15

24



Other Priority Queues

- Leftist Heaps
 - › $O(\log N)$ time for insert, delete, merge
- Skew Heaps
 - › $O(\log N)$ amortized time for insert, delete, merge
- Calendar Queues
 - › $O(1)$ average time for insert and delete
 - › Assuming insertions are "random"

3/2/05 Binomial Queues - Lecture 15 27

