CSE 326: Data Structures

Topic #4: Putting Our Heaps Together, The Leftist Way!

Ashish Sabharwal

Autumn, 2003

Today's Outline

- Finish Binary Heaps
- d-heaps from last weeks notes
- Leftist Heaps

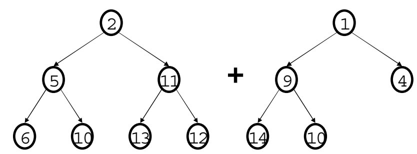
New Operation: Merge

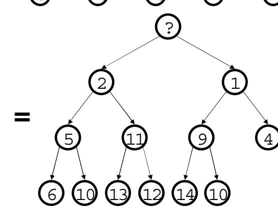
Given two heaps, merge them into one heap

- first attempt: insert each element of the smaller heap into the larger.
  runtime:
- second attempt: concatenate binary heaps' arrays and run buildHeap.
  runtime:

How about O(log n) time?

Idea: Hang Heaps From a New Root



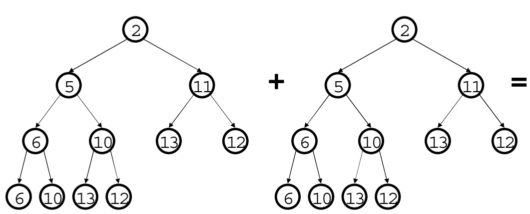


IncreaseKey and percolate down!

runtime:

Problem?

Idea: Hang Heaps From a New Root



Problem?

Leftist Heaps

Idea:

Focus all heap maintenance work in one small part of the heap

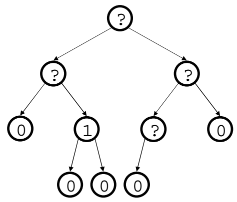
Leftist heaps:

1. Most nodes are on the left
2. All the merging work is done on the right

Definition: Null Path Length

The null path length (npl) of a node x is the number of nodes between x and a null in its subtree

- npl(null) = -1
- npl(leaf) = 0
- npl(single-child node) = 0



Equivalent definitions:

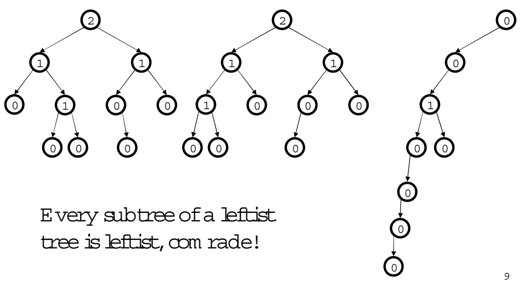
1. npl(x) is the height of largest complete subtree rooted at x
2. npl(x) = 1 + max{npl(left(x)), npl(right(x))}

Leftist Heap Properties

- Heap-order property
  - Parent's priority is at most the childrens' priority
  - result: minimum element is at the root
- Leftist property
  - For every node x, npl(left(x)) ≥ npl(right(x))
  - result: tree is at least as "heavy" on the left as the right

Are leftist trees…
complete?
balanced?

Are These Leftist?



Every subtree of a leftist tree is leftist, comrade!

Right Path in a Leftist Tree is Short (#1)

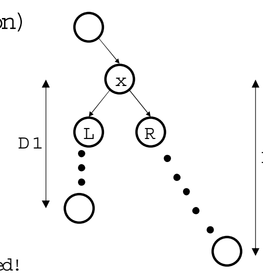
Claim: The right path is as short as any in the tree.

Proof: (By contradiction)

Pick a shorter path:
D1 < D2

npl(L):

npl(R):



Leftist property at x violated!

Right Path in a Leftist Tree is Short (#2)

Claim: If the right path has **r** nodes, then the tree has at least **2^r-1** nodes.

Proof: (By induction)

Base case : **r=1**. Tree has at least **2^1-1 = 1** node

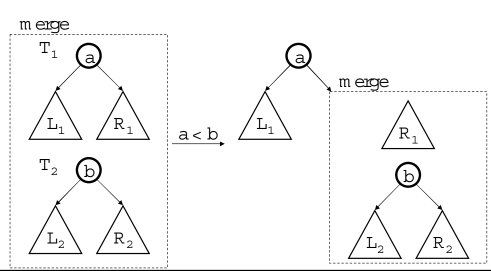
Inductive step : assume true for **r' < r**. Prove for tree with right path at least **r**.

1. Right subtree: right path of **r-1** nodes
   ⇒ **2^{r-1}-1** right subtree nodes (by induction)
2. Left subtree: also right path of length at least **r-1** (by previous slide)
   ⇒ **2^{r-1}-1** left subtree nodes (by induction)

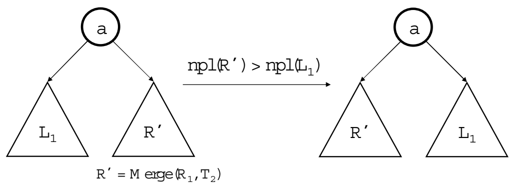
Total tree size: **(2^{r-1}-1) + (2^{r-1}-1) + 1 = 2^r-1**

Merging Two Leftist Heaps

- merge(T_1,T_2) returns one leftist heap containing all elements of the two (distinct) leftist heaps T_1 and T_2



Merge Continued



runtime:

13

Let's do an example, but first... Other Heap Operations

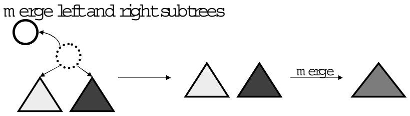
- insert ?
- deleteMin ?
- buildHeap ?

14

Operations on Leftist Heaps

- merge with two trees of total size n: Θ(log n)
- insert with heap size n: Θ(log n)
  - pretend node is a size 1 leftist heap
  - insert by merging original heap with one node heap

- deleteMin with heap size n: Θ(log n)
  - remove and return root
  - merge left and right subtrees



15

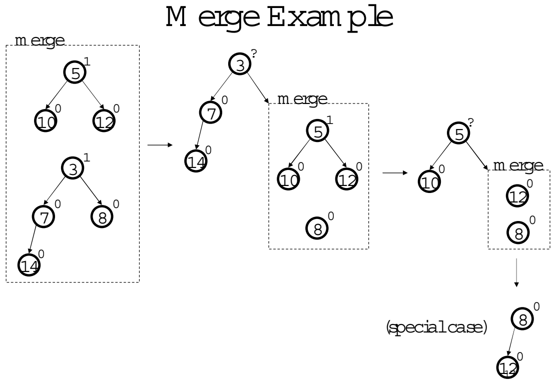
Operations on Leftist Heaps

- buildHeap: options are

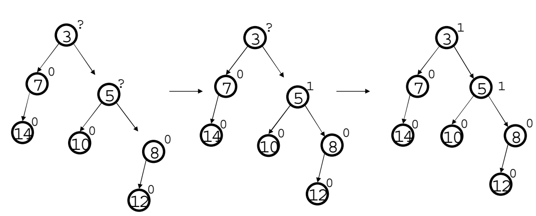
1. Use Floyd's method
   - but need pointer based implementation
   - unclear how to traverse right-to-left, bottom-up
2. Do n inserts
   - Takes Θ(n log n) time
3. Use merge in a smart way!
   - (Exercise in your next homework)

16

Merge Example



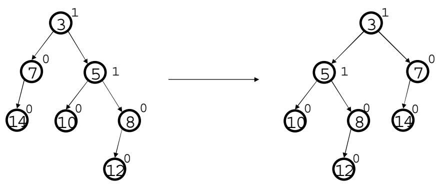
Sewing Up the Example



Done?

18

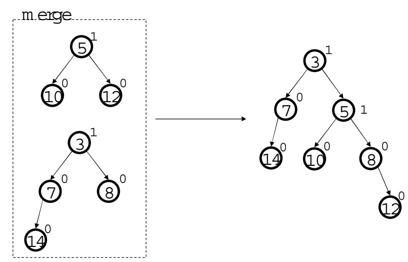
Finally…



19

Iterative Leftist Merging

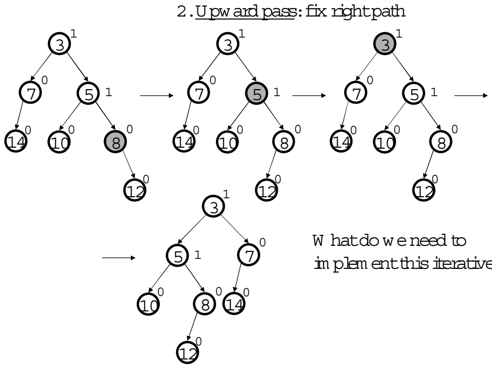
1. Downward pass: merge right paths



20

Iterative Leftist Merging

2. Upward pass: fix right path



What do we need to implement this iteratively?

21

Leftist Heaps: Summary

Good

-
-
-

Bad

-
-
-

22

To Do

- Continue project #1
- First written homework will be out Wednesday
- Finish reading chapter 6

23