

## Analysis of Algorithms CS 3343 Lecture Twelve

Prof. William Winsborough  
November 4, 2008

## Business

- Turn in Homework 5 (version 2.0)
  - Problem 7.1 (p.159)
- Reading: 6.1-6.4
- Homework 7 due Tuesday 11/11
  - 6.1-1, 6.1-2, 6.1-3 (hint: use induction on the height of the subtree), 6.1-7
  - 6.2-3, 6.2-4, 6.2-5
  - 6.4-2, 6.4-3
- Midterm 2 will be Thursday November 20
  - Tuesday November 18 will be devoted to review

4 November 2008

Winsborough CS 3343 Lecture 12

2

## Heaps and Heapsort

- A heap is a nearly complete binary tree
  - Lowest level may not be complete
- Represented as an array with two attributes:
  - $\text{heap-size}[A] \leq \text{length}[A]$
- Root is  $A[1]$
- Navigation from a given node:
  - $\text{Parent}(i)$  return  $\lfloor i/2 \rfloor$
  - $\text{Left}(i)$  return  $2i$
  - $\text{Right}(i)$  return  $2i+1$
  - Programming: bit shift implementations are done in line

4 November 2008

Winsborough CS 3343 Lecture 12

3

## Terminology

- Max-heaps
  - Max-heap property: for all  $i$ ,  $A[\text{parent}(i)] \geq A[i]$
- Min-heaps
  - Min-heap property: for all  $i$ ,  $A[\text{parent}(i)] \leq A[i]$
- Height of a node is the number of edges in the longest downward path from the node to a leaf
  - Height of the heap is the height of its root  $O(\lg n)$

4 November 2008

Winsborough CS 3343 Lecture 12

4

## Max-Heapify

- Subroutine used for manipulating max-heaps
- Inputs:  $A, i$ 
  - Assumes that binary trees rooted at  $\text{Left}(i)$  and  $\text{Right}(i)$  are max-heaps
  - But  $A[i]$  may be smaller than its children
  - The routine moves such values down in the tree, making the tree rooted at  $i$  a max-heap
- Refer to p.130 for routine

4 November 2008

Winsborough CS 3343 Lecture 12

5

## Runtime Complexity of Max-Heapify

- Constant time to organize top three nodes
- One recursive call with worst case input size  $2n/3$
- $T(n) \leq T(2n/3) + \Theta(1)$
- $T(n) = O(\lg n)$

4 November 2008

Winsborough CS 3343 Lecture 12

6

## Building a Heap

- Build the heap bottom up by using Max-Heapify (see p.133)
- Tight upper bound:  $O(n)$ 
  - $O(n \lg n)$  is clear
  - An  $n$ -element heap has height  $\lceil \lg n \rceil$  and at most  $\lceil n/2^{h+1} \rceil$  nodes of height  $h$
- Refer to p.135
- Work 6.3-3: there are at most  $\lceil n/2^{h+1} \rceil$  nodes of height  $h$

4 November 2008

Winsborough CS 3343 Lecture 12

7

## Heapsort

Heapsort(A)

1. Build-Max-Heap(A)
  2. for  $i \leftarrow \text{length}[A]$  downto 2
  3. do exchange  $A[1] \leftrightarrow A[i]$
  4.  $\text{heap-size}[A] \leftarrow \text{heap-size}[A] - 1$
  5. Max-Heapify(A,1)
- $O(n \lg n)$ : Build-Max-Heap is  $O(n)$  and each of the  $n - 1$  calls to Max-Heapify is  $O(\lg n)$

4 November 2008

Winsborough CS 3343 Lecture 12

8

## Example

- Refer to p.137

4 November 2008

Winsborough CS 3343 Lecture 12

9