6.006 Introduction to Algorithms
Spring 2008

Lecture 9: Sorting II: Heaps

Lecture Overview

- Review: Heaps and MAX_HEAPIFY
- Building a Heap
- Heap Sort
- Priority Queues (Recitation)

Readings

CLRS 6.1-6.4

Review

Heaps:

$$\begin{aligned} \text{Parent}(i) &= \lfloor i/2 \rfloor \\ \text{Left}(i) &= 2i \\ \text{Right}(i) &= 2i+1 \end{aligned}$$

Max_heap property:

A[Parent(i)] $\geq$ A[i]

- MAX_HEAPIFY($A, 2$)
  heap_size(A) = 10
  $A[2] \longleftrightarrow A[4]$
- MAX_HEAPIFY(A,4)
  $A[4] \longleftrightarrow A[9]$

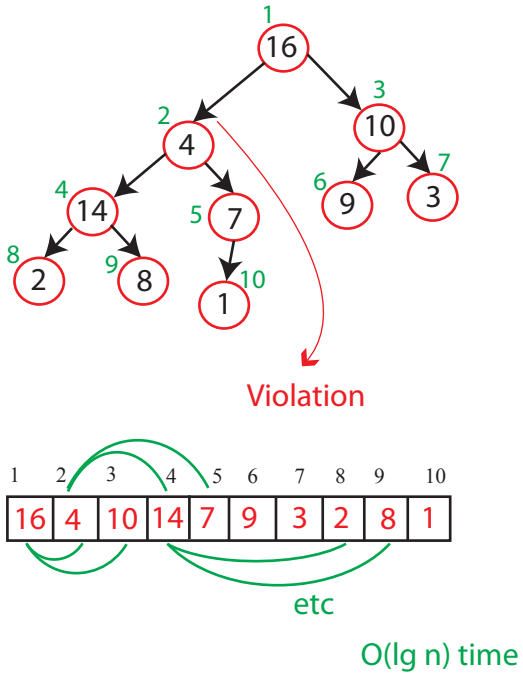

Figure 1: Review from last lecture

Building a Heap

$A[1 \cdots n]$ converted to a max_heap *Observation*: Elements $A[\lfloor n/2 + 1 \rfloor \cdots n]$ are all leaves of the tree and can't have children.

BUILD_MAX_HEAP(A):
heap_size(A) = length(A)
$O(n)$ times for $i \leftarrow \lfloor$ length$[A]/2 \rfloor$ downto 1
$O(\lg n)$ time do MAX_HEAPIFY(A, i)
$O(n \lg n)$ overall

See Figure 2 for an example.

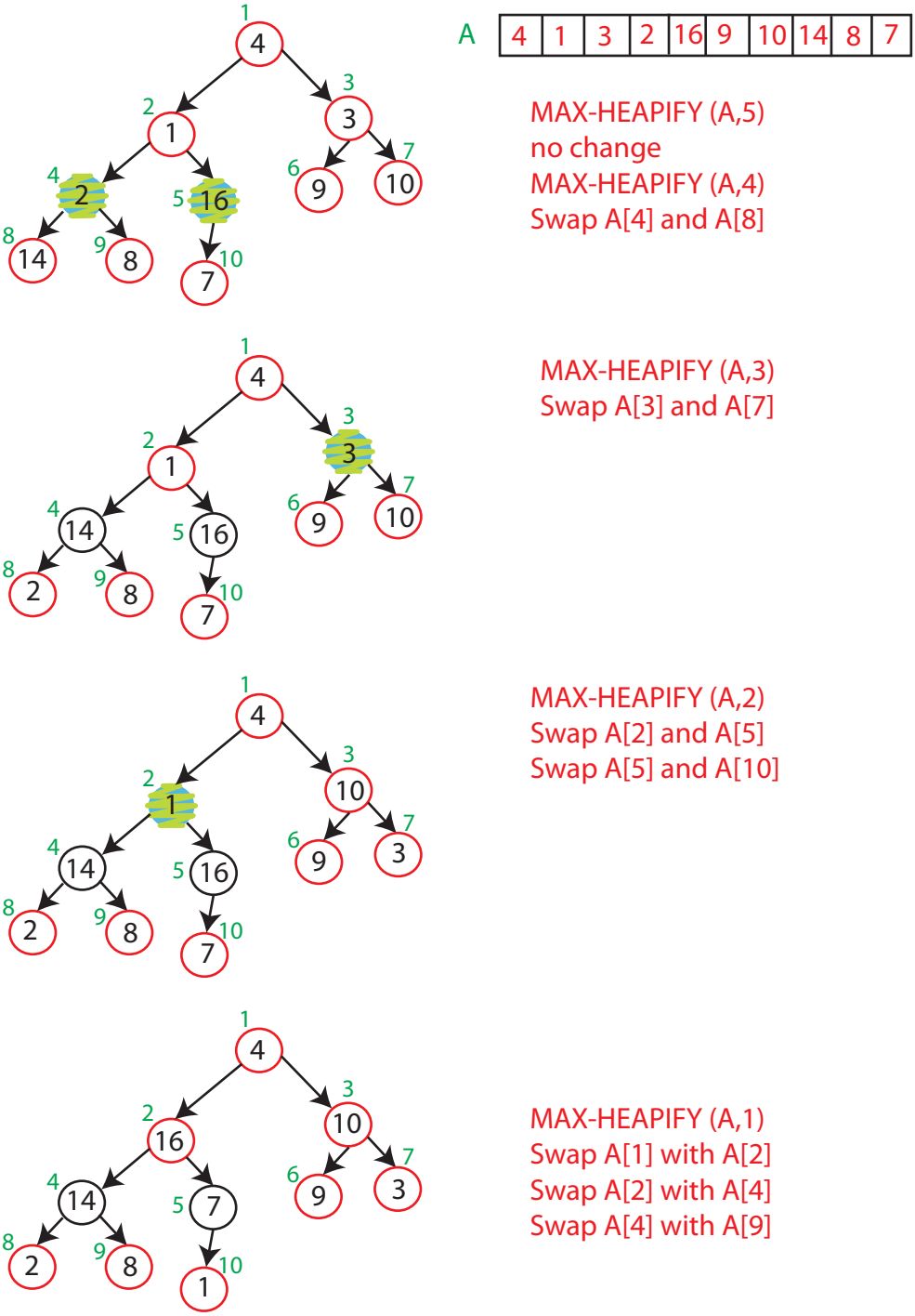

Figure 2: Example: Building Heaps

Sorting Strategy

- Build max_heap from unordered array
- Find maximum element ($A[1]$)
- Put it in correct position $A[n]$, $A[n]$ goes to $A[1]$

  New root could violate max_heap property but children remain max_heaps.
- Discard node n from heap (decrement heapsize)

Heap Sort Algorithm

$O(n \lg n)$ BUILD_MAX_HEAP(A):
n times for i =length[A] downto 2
do exchange $A[1] \longleftrightarrow A[i]$
heap_size[A] = heap_size[A] $- 1$
$O(\lg n)$ MAX_HEAPIFY($A, 1$)
$O(n \lg n)$ overall

See Figure 3 for an illustration.

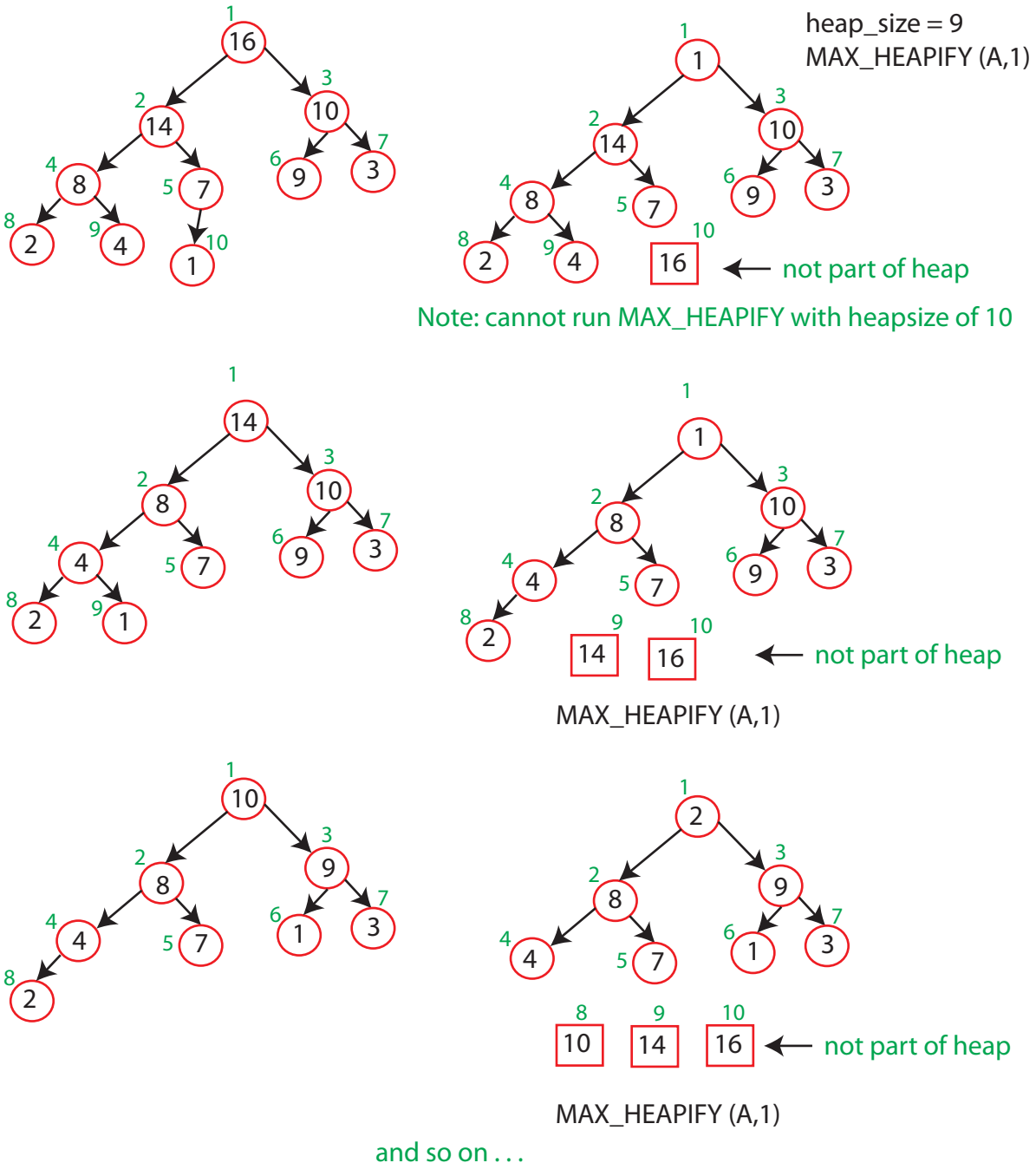

Figure 3: Illustration: Heap Sort Algorithm

Priority Queues

This is an abstract datatype as it can be implemented in different ways.

INSERT(S, X) :	inserts X into set S
MAXIMUM(S):	returns element of S with largest key
EXTRACT_MAX(S):	removes and returns element with largest key
INCREASE_KEY(S, x, k):	increases the value of element x's key to new value k (assumed to be as large as current value)