6.006
Spring 2008
L8

Sorting

Review: Insertion Sort
Merge Sort

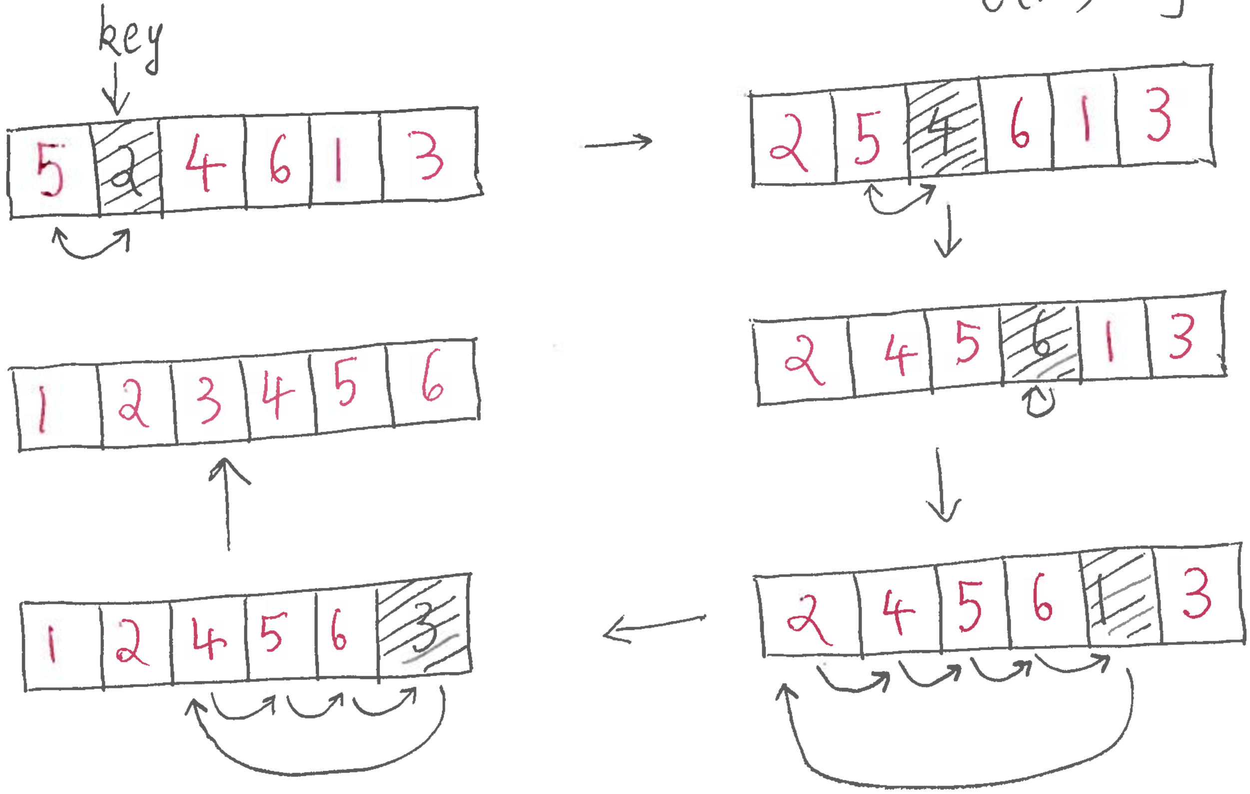
Readings: 2.1, 2.2, 2.3
$6.1, 6.2, \underbrace{6.3, 6.4}_{\text{Thursday}}$

Selection Sort

Heaps

Insertion Sort

$\Theta(n^2)$ algorithm

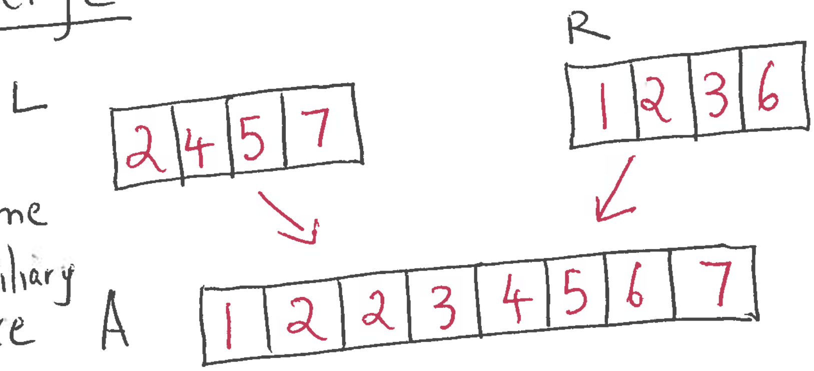


Merge Sort

Divide n-element array into two subarrays of $n/2$ elements each

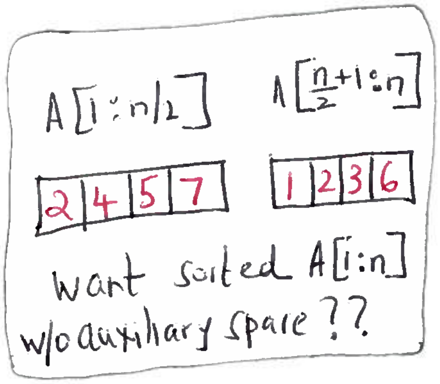
Recursively sort sub-arrays using mergesort

Merge two sorted subarrays



$\Theta(n)$ time

$\Theta(n)$ auxiliary space



want sorted $A[1:n]$ w/o auxiliary space??

In-Place Sorting

Numbers re-arranged in the array A with at most a constant number of them stored outside the array at any time

Insertion Sort : stores key outside array $\Theta(n^2)$
in-place

Merge Sort : Need $\Theta(n)$ auxiliary space $\Theta(n \log n)$
during merging

Q: can we have $\Theta(n \log n)$ in-place sorting?

Selection Sort

n times

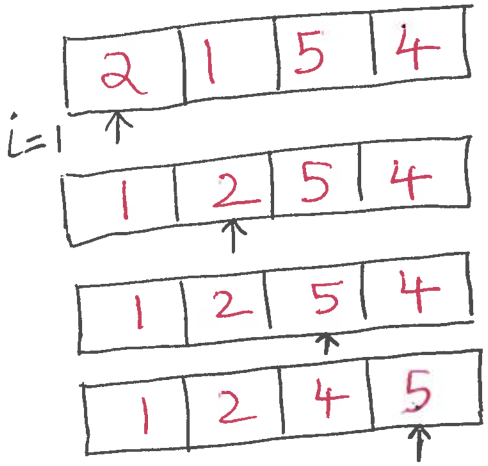
0. $i = 1$
1. Find minimum value in list beginning with i
2. Swap it with the value in the i^{th} position
3. $i = i+1$, stop if $i = n$

O(n) time

Can we improve to O(lg n)?

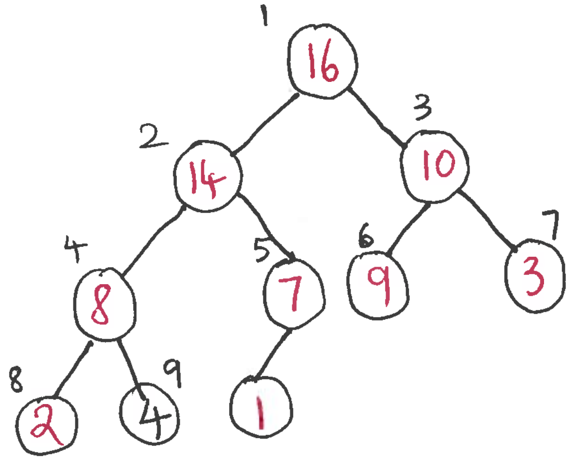
$\Theta(n^2)$ time

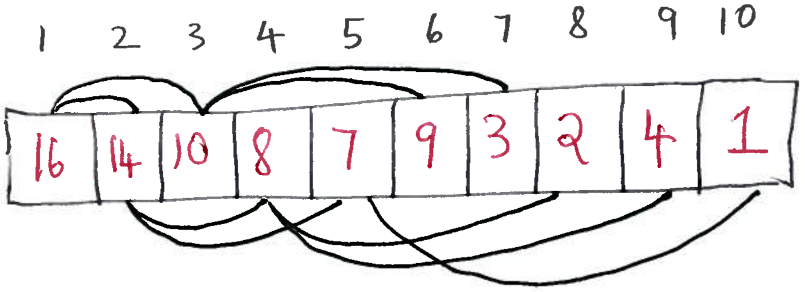
in-place



Heaps (not garbage collected storage!)

A heap is an array object that is viewed as a nearly complete binary tree





DATA STRUCTURE

root : $A[i]$

Node with index i

$\text{PARENT}(i) = \lfloor \frac{i}{2} \rfloor$ NO POINTERS!

$\text{LEFT}(i) = 2i$

$\text{RIGHT}(i) = 2i+1$

length[A] : number of elements in the array

heap-size[A] : number of elements in the heap stored within array A.

$\text{heap-size}[A] \leq \text{length}[A]$

MAX-HEAPS (and MIN-HEAPS)

max-heap property : For every node i other than the root $A[\text{PARENT}(i)] \geq A[i]$

height of a binary heap $O(\lg n)$

MAX-HEAPIFY : $O(\lg n)$ maintains max-heap property

BUILD-MAX-HEAP : $O(n)$ produces max-heap from unordered input array

HEAP-SORT : $O(n \lg n)$

Heap operations insert, extract-max etc $O(\lg n)$

MAX-HEAPIFY (A, i)

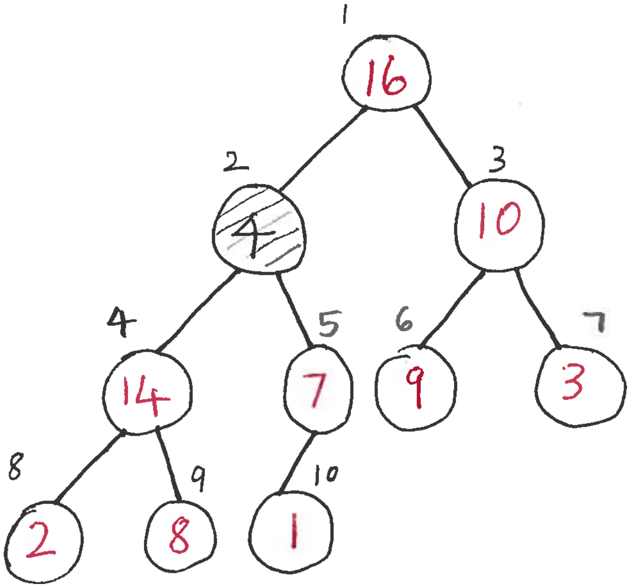
```
l ← left(i)
r ← right(i)
if l ≤ heap-size(A) and A[l] > A[i]
        then largest ← l
        else largest ← i
if r ≤ heap-size(A) and A[r] > largest
        then largest ← r
if largest ≠ i
    then exchange A[i] and A[largest]
         MAX-HEAPIFY (A, largest)
```

Assumes trees rooted at left(i) and Right (i) are max-heaps

A[i] may be smaller than children violating max-heap property

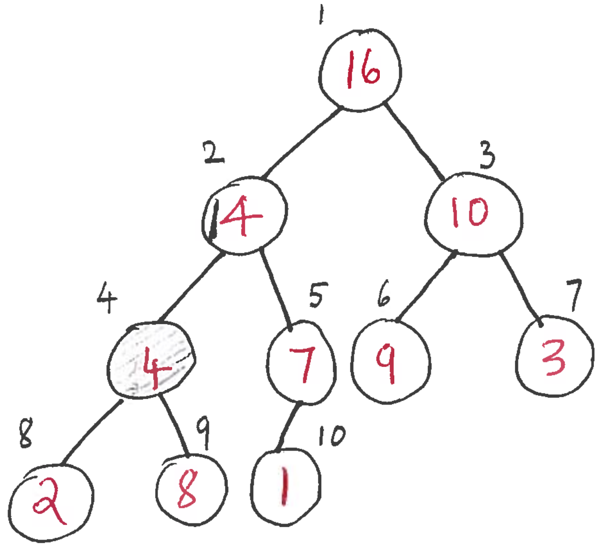
Let the A[i] value "float down" so subtree rooted at index i becomes a max-heap.

EXAMPLE



MAX-HEAPIFY (A, 2)

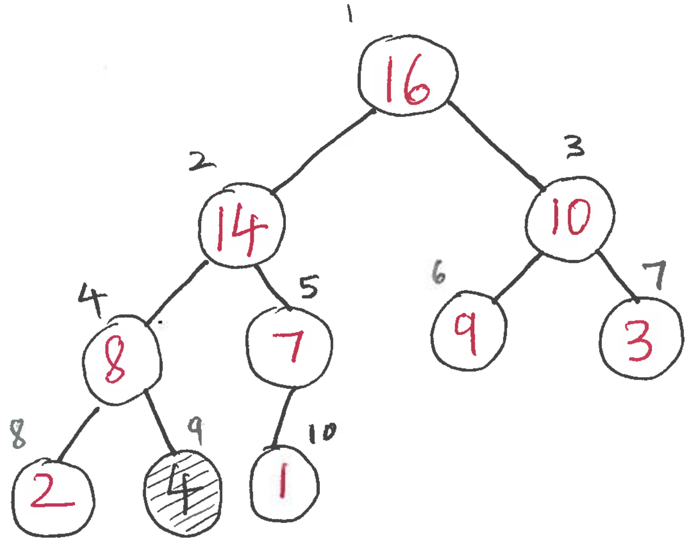
heap-size[A] = 10



exchange A[2] with A[4]

Call MAX-HEAPIFY (A, 4)

because max-heap property is violated



exchange A[4] with A[9]

no more calls