	Outline					
Computer Science 331 Heap Sort	1 HeapSort					
Mike Jacobson Department of Computer Science University of Calgary Lecture #26	 Description Correctness and Efficiency References 					
Mike Jacobson (University of Calgary) Computer Science 331 Lecture #26 1 / 16 HeapSort Description	Mike Jacobson (University of Calgary) Computer Science 331 Lecture #26 2 / 16 HeapSort Description Pseudocode					
 Idea: Use Build-Max-Heap to convert the input array into a representation of a heap Repeatedly use Delete-Max to extract the largest element in the unsorted part of the array and move it into position 	Heap Sort(A) n = heap-size(A) if $n > 1$ then Build-Max-Heap(A) i = n - 1 while $i > 0$ do largest = Delete-Max(A) A[i] = largest i = i - 1 end while end if					

Example: After Build-Max-Heap



Example

0	1	2	3	4	5	6	7
2	7	4	1	6	9	3	8



0	1	2	3	4	5	6	7
9	8	4	7	6	2	3	1

Mike Jacobson (University of Calgary)	Computer Science 331	Lecture #26	5 / 16	Mike Jacobson (University of Calgary)	Computer S	Science 331	Lecture #26	6 / 16
	HeapSort Description				HeapSort	Description		
					Theapeon	Decemption		
A[7] = Delete-Ma	$\mathbf{x}(A)$			A[6] = Delete-Ma	$\mathbf{x}(A)$			











A[4] =**Delete-Max**(A)

0	1	2	3	4	5	6	7
			1		1		1

Mike Jacobson (University of Calgary)	Computer	Science 331	Lecture #26	9 / 16
	HeapSort	Description		
A[3] = Delete-Max (A	۹)			

Mike Jacobson (University of Calgary)	Computer S	Science 331	Lecture #26	10 / 16
	HeapSort	Description		
A 2 = Delete-Ma	X (A)			

HeapSort Description









Mike Jacobson (University of Calgary)

0 1 2

34

5

6 7

A[1] =**Delete-Max**(A)

Partial Correctness

Loop invariant: if the loop is executed at least k times then, after the *k*th iteration:

- $0 \le i \le n 1$, and i = n 1 k
- heap-size(A) = i + 1 = n k
- A represets a max-heap with size i + 1
- the last n i 1 entries are all greater than or equal to each of the first i + 1 entries in A
- $A[j] \le A[j+1]$ for $i \le j \le n-2$
- the entries of A are reordered but otherwise unchanged

If true and loop terminates, then i = 0 and

			•			
			٠			
Mike Jacobson (University of Calgary)	Computer Science 331	Lecture #26 13 / 16	Mike Jacobson (University of Calgary)	Computer Science 331	Lecture #26	14 / 16
	HeapSort Correctness and Efficienc	cy .		References		
Termination and Eff	ficiency		References			
Loop variant: $f(n, i) = i$			Textbook, Section 10.8 A simplified presenta analyzed Build-Max-Heap slig	ation: "Max-Heapify" is no	ot separately dex 0)	
Worst-case cost of heaps	Sort:					
٩			Cormen, Leiserson, Rive	st and Stein, Introduction	n to Algorithms,	
٠			Second Edition, Chapter the presentation in these	6: A discussion of Heap notes.	Sort that is closer	r to
Advantage over mergeSo	ort:					