**HEAP SORT**

Heaps and **MAX-HEAPIFY** review

Building a Heap

Heap Sort

Priority Queues (recitation)

Readings: 6.1 - 6.4

Parent \( i \) = \( \left\lfloor \frac{i}{2} \right\rfloor \)

Left \( i \) = \( 2i \)

Right \( i \) = \( 2i + 1 \)

Max-heap-property:

\[ A[\text{Parent}(i)] \geq A[i] \]

**MAX-HEAPIFY** \((A, 2)\)

heap-size \((A) = 10\)


\( \text{MAX-HEAPIFY}(A, 4) \)


\( O(\log n) \) time
Building a Heap

A [1...n] converted to a max-heap

Observation: Elements A [\lceil n/2 \rceil ... 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 = \lceil Length[A]/2 \rceil  down to 1
do MAX-HEAPIFY (A, i)
O(lgn) time
O(nlgn) overall

Example

A 4 132 16 9 10 4 8 7

MAX-HEAPIFY (A, 5)
no change

MAX-HEAPIFY (A, 4)
MAX-HEAPIFY (A, 3)

MAX-HEAPIFY (A, 2)

MAX-HEAPIFY (A, 1)

final
**Sorting Strategy**

Build max-heap from unordered array
Find maximum element \( A[i] \)
Discard node \( n \) from heap (decrement heap size)
   - new root could violate max-heap property
   - but children remain max heaps!

**Heap Sort**

\[ O(n \log n) \] \textbf{BUILD-MAX-HEAP} (\( A \))

\( n \) times for \( i = \text{length}(A) \) down to 2
  do exchange \( A[i] \leftrightarrow A[i] \)
  heap-size[\( A \)] = heap-size[\( A \)] - 1

\[ O(\log n) \] \textbf{MAX-HEAPIFY} (\( A, i \))

\[ O(n \log n) \] overall
heap-size = 9

\text{MAX-HEAPIFY}(A, 1)

Note: Cannot run MAX-HEAPIFY with heap size of 10.

\text{MAX-HEAPIFY}(A, 1)

\text{MAX-HEAPIFY}(A, 1)

not part of heap

\text{MAX-HEAPIFY}(A, 1)

\text{MAX-HEAPIFY}(A, 1)

not part of heap

\text{MAX-HEAPIFY}(A, 1)

and so on
Abstract Data Type: can be implemented in different ways

\textbf{INSERT} (S, x) : inserts x into set S
\textbf{MAXIMUM} (S) : returns element of S with largest key
\textbf{EXTRACT-MAX} (S) : removes and returns element with largest key
\textbf{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)