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:

- **Parent**\((i) = \lfloor i/2 \rfloor\)
- **Left**\((i) = 2i\)
- **Right**\((i) = 2i + 1\)

Max_heap property:

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

- MAX.HEAPIFY\((A, 2)\)
  - heap_size\((A) = 10\)
- MAX.HEAPIFY\((A, 4)\)
Building a Heap

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

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

See Figure 2 for an example.
MAX-HEAPIFY (A,5)
no change
MAX-HEAPIFY (A,4)

MAX-HEAPIFY (A,3)

MAX-HEAPIFY (A,2)

MAX-HEAPIFY (A,1)

Figure 2: Example: Building Heaps
Sorting Strategy

- Build max_heap from unordered array
- Find maximum element \( A[1] \)
  New root could violate max_heap property but children remain max_heaps.
- Discard node \( n \) from heap (decrement heapsize)

Heap Sort Algorithm

\[
\begin{align*}
O(n \lg n) & \text{ BUILD_MAX_HEAP}(A): \\
n \text{ times} & \text{ for } i = \text{length}[A] \text{ downto } 2 \\
& \quad \text{ do exchange } A[1] \leftrightarrow A[i] \\
& \quad \text{ heap_size}[A] = \text{heap_size}[A] - 1 \\
O(\lg n) & \text{ MAX_HEAPIFY}(A, 1) \\
O(n \lg n) & \text{ overall}
\end{align*}
\]

See Figure 3 for an illustration.
heap_size = 9
MAX_HEAPIFY (A,1)

Note: cannot run MAX_HEAPIFY with heapsize of 10

MAX_HEAPIFY (A,1)

and so on . . .

Figure 3: Illustration: Heap Sort Algorithm
Priority Queues

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

\begin{itemize}
\item \textbf{INSERT}(S, X): inserts \( X \) into set \( S \)
\item \textbf{MAXIMUM}(S): returns element of \( S \) with largest key
\item \textbf{EXTRACT_MAX}(S): removes and returns element with largest key
\item \textbf{INCREASE_KEY}(S, x, k): increases the value of element \( x \)’s key to new value \( k \)
\end{itemize}

(assumed to be as large as current value)