CS3000: Algorithms & Data — Summer '25 — Laney Strange

Heapsort

The Heapsort algorithm repeatedly finds the largest element and puts it in its correct sorted position.

The input to this procedure is an array, A, which represents a max-heap. The array $A[1:n]$ also maintains an attribute $A.heapsize$, which distinguishes the part of A which belongs to the heap (unsorted) vs. the part of A that has already been sorted.

A Max-Heap maintains two properties: (1) it is a complete binary tree (the levels get filled in top-to-bottom and left-to-right), and (2) every node is larger than all of its descendents. We make the simplifying assumption that the input array A is already a Max-Heap. (Can we convert a plain unsorted array to a Max-Heap? Absolutely! It's jut not part of this particular algorithm.)

Heapsort relies on a procedure MAX-HEAPIFY, that takes in the array A and the position of the current root i. Max-Heapify's job is to assume whatever's in the root might not be correct, but the whole rest of the heap is. It "bubbles down" the root node to its correct position within the heap.

HEAPSORT(A, n)

```
1  A.heapsize = n
2  for i = n downto 2
3        swap A[1], A[i]
4        A.heapsize = A.heapsize − 1
5        MAX-HEAPIFY(A, 1)
```

MAX-HEAPIFY(A, i)

```
 1  l = 2i
 2  r = 2i + 1
 3  if l ≤ A.heapsize and A[l] > A[i]
 4        largest = l
 5  else
 6        largest = i
 7  if r ≤ A.heapsize and A[r] > A[largest]
 8        largest = r
 9  if largest ≠ i
10        swap A[i], A[largest]
11        MAX-HEAPIFY(A, largest)
```

We typeset the procedures above with the following LaTeX:

```
\begin{codebox}
\Procname{$\proc{Heapsort}(A, n)$}
\li $\id{A.heapsize} = n$
\li \For $i \gets n \Downto 2$
\Do
```

```
\li swap $A[1], A[i]$
\li $\id{A.heapsize} = \id{A.heapsize} - 1$
\li $\proc{Max-Heapify}(A, 1)$
\End
\end{codebox}

\begin{codebox}
\Procname{$\proc{Max-Heapify}(A, i)$}
\li $l \gets 2i$
\li $r \gets 2i + 1$
\li \If $l \le A.\id{heapsize}$ and $A[l] > A[i]$
\Then
\li $\id{largest} \gets l$
\End
\li \Else
\Then
\li $\id{largest} \gets i$
\End
\li \If $r \le A.\id{heapsize}$ and $A[r] > A[\id{largest}]$
\Then
\li $\id{largest} \gets r$
\End
\li \If $\id{largest} \ne i$
\Then
\li swap $A[i], A[\id{largest}]$
\li $\proc{Max-Heapify}(A, \id{largest})$
\End
\end{codebox}
```