Heap Algorithms

```
Parent($A, i$)
    // *Input*: $A$: an array representing a heap, $i$: an array index
    // *Output*: The index in $A$ of the parent of $i$
    // *Running Time*: $O(1)$
    if $i == 1$ return NULL
    return $\lfloor i/2 \rfloor$
```

```
Left($A, i$)
    // *Input*: $A$: an array representing a heap, $i$: an array index
    // *Output*: The index in $A$ of the left child of $i$
    // *Running Time*: $O(1)$
    if $2 * i \leq \textit{heap-size}[A]$
        return $2 * i$
    else return NULL
```

```
Right($A, i$)
    // *Input*: $A$: an array representing a heap, $i$: an array index
    // *Output*: The index in $A$ of the right child of $i$
    // *Running Time*: $O(1)$
    if $2 * i + 1 \leq \textit{heap-size}[A]$
        return $2 * i + 1$
    else return NULL
```

```
Max-Heapify($A, i$)
    // *Input*: $A$: an array where the left and right children of $i$ root heaps (but $i$ may not), $i$: an array index
    // *Output*: $A$ modified so that $i$ roots a heap
    // *Running Time*: $O(\log n)$ where $n = \textit{heap-size}[A] - i$
    $l \leftarrow$ Left($i$)
    $r \leftarrow$ Right($i$)
    if $l \leq \textit{heap-size}[A]$ and $A[l] > A[i]$
        $\textit{largest} \leftarrow l$
    else $\textit{largest} \leftarrow i$
    if $r \leq \textit{heap-size}[A]$ and $A[r] < A[\textit{largest}]$
        $\textit{largest} \leftarrow r$
    if $\textit{largest} \neq i$
        exchange $A[i]$ and $A[\textit{largest}]$
        Max-Heapify(A, largest)
```

```
Build-Max-Heap($A$)
    // *Input*: $A$: an (unsorted) array
    // *Output*: $A$ modified to represent a heap.
    // *Running Time*: $O(n)$ where $n = \textit{length}[A]$
    $\textit{heap-size}[A] \leftarrow \textit{length}[A]$
    for $i \leftarrow \lfloor \textit{length}[A]/2 \rfloor$ downto 1
        Max-Heapify($A, i$)
```