DSA notes - 2

Heaps

Let A be an array of length n that represents a complete binary tree, i.e. each array element represents a node of the complete binary tree. The value in node i is given by $A[i]$. The parent of node i is given by $parent(i) = \lfloor \frac{i-1}{2} \rfloor$. The left child of node i is given by $left(i) = 2i + 1$ and the right child of node i is given by $right(i) = 2i + 2$. If $left(i)$ (respectively $right(i)$) is greater than or equal to n, then node i does not have a left (respectively right) child. The pseudocode given below assumes that the array A and its length n are passed by reference to each of these functions; i.e. any modifications to A or n changes the values of the corresponding variables in the scope from which the function was called.

Algorithm 1 HEAPIFY(A, n, i)   //A is an array of length n

```
small = i
if left(i) < n and A[small] > A[left(i)] then
    small = left(i)
end if
if right(i) < n and A[small] > A[right(i)] then
    small = right(i)
end if
if small = i then
    return
else
    Exchange A[small] and A[i]
    HEAPIFY(A, n, small)
end if
```

The algorithm HEAPIFY assumes that the complete binary trees having roots $left(i)$ and $right(i)$ (if they exist) obey the min-heap property. The algorithm modifies the array A so that the complete binary tree rooted at i also now obeys the min-heap property.

Algorithm 2 BUILD-HEAP(A, n)   //A is an array of length n

```
i = ⌊n/2⌋ − 1        //nodes ⌊n/2⌋ to n − 1 have no children
while i ≥ 0 do
    HEAPIFY(A, n, i)
    i ← i − 1
end while
```

The algorithm BUILD-HEAP takes an arbitrary array A as input and changes the order of elements in it so that the complete binary tree represented by the array obeys the min-heap property.

Algorithm 3 EXTRACT-MIN(A, n)   //A is an array of length n representing a min-heap

---

$x = A[0]$
$A[0] = A[n-1]$
$n \leftarrow n - 1$ //Decrease array size
HEAPIFY(A, n, 0)
return x

The algorithm EXTRACT-MIN returns the first element of the array A, or the value contained in the root of the min-heap represented by A.

Algorithm 4 FLOAT-UP(A, n, i)   //A is an array of length n

---

if $i = 0$ or $A[parent(i)] \leq A[i]$ **then**
    return
end if
Exchange $A[parent(i)]$ and $A[i]$
FLOAT-UP(A, n, $parent(i)$)

The algorithm FLOAT-UP makes the value in node i "float up" as many levels as it needs to so that the min-heap property becomes valid between it and its parent (or until it floats up to the root). If i was the only node in the heap for which the min-heap property was violated between it and its parent, then after the "floating up" operation, every node in the heap obeys the min-heap property.

Algorithm 5 INSERT(A, n, val)   //A is an array of length n representing a min-heap

---

$A[n] \leftarrow val$
$n \leftarrow n + 1$ //Increase size of array
FLOAT-UP(A, n, $n - 1$)

The algorithm INSERT inserts a new value val into the min-heap represented by the array A.