Algorithms: Solutions 6

Problem 1

Let $A[1..n]$ be a sorted array of n distinct integer numbers. Write an efficient algorithm INDEX-SEARCH(A, n) that finds an index i such that $A[i] = i$.

The algorithm is almost identical to BINARY-SEARCH, and its time complexity is $O(\lg n)$. It works only for integer arrays, since it is based on the assumption that, for every two indices p and r (where $p \leq r$), we have $A[r] - A[p] \geq r - p$.

```
INDEX-SEARCH(A, n)
p ← 1
r ← n
while p < r
    do q = ⌊(p + r)/2⌋
        if q ≤ A[q]
            then r ← q
            else p ← q + 1
if p = A[p]
   then return p
   else return 0
```

Problem 3

A *d-ary heap* is like a binary heap, but instead of 2 children, nodes have d children.

(a) What are the expressions for determining the parent of the given element, PARENT(i), and a j-th child of a given element, CHILD(i, j), where $1 \leq j \leq d$?

$$\begin{aligned} \text{PARENT}(i) &= \left\lfloor \frac{i + d - 2}{d} \right\rfloor \\ \text{CHILD}(i, j) &= (i - 1) \cdot d + j + 1 \end{aligned}$$

(b) Write an efficient implementation of HEAPIFY and HEAP-INSERT for a d-ary heap.

The HEAPIFY algorithm is somewhat different from the binary-heap version, whereas HEAP-INSERT is identical to the corresponding algorithm for binary heaps. The running time of HEAPIFY is $O(d \cdot \log_d n)$, and the running time of HEAP-INSERT is $O(\log_d n)$.

```
HEAPIFY(A, i, n, d)
largest ← i
for l ← CHILD(i, 1) to min(n, CHILD(i, d))      ▷ loop through all children of i
    do if A[l] > A[largest]
            then largest ← l
if largest ≠ i
    then exchange A[i] ↔ A[largest]
          HEAPIFY(A, largest)
```

```
HEAP-INSERT(A, key)
heap-size[A] ← heap-size[A] + 1
i ← heap-size[A]
while i > 1 and A[PARENT(i)] < key
      do A[i] ← A[PARENT(i)]
          i ←PARENT(i)
A[i] ← key
```