Problem Solutions to CLRS

Zeaiter Zeaiter

Contents

1 Chapter 2

2.1–2

```
procedure INSERTION SORT(A)
    for i = 1 to A.length − 1 do
        key = A[i]
        j = i − 1
        while j > 0 and A[j] < key do
            A[j + 1] = A[j]
            j = j − 1
        end while
        A[i + 1] = key
    end for
end procedure
```

2.1–3

```
1: procedure LINEAR SEARCH(A, v)
2:     for i = 0 to A.length − 1 do
3:         if A[i] == v then
4:             return i
5:         end if
6:         return NIL
7:     end for
8: end procedure
```

At the start of each iteration of the **for** loop (lines 2–7) $i-1$ is not an index of A such that $A[i-1] = v$.

Let us now prove the correctness of our algorithm. Suppose $i = 0$, then $i-1$ is clearly not an index of A and hence $A[i-1]$ is undefined. Now suppose the loop invariant is true for some i, that is, $i-1$ is not an index of A such that $A[i-1] = v$, or equivalently, $A[i-1] \neq v$. Then at line 3 the **if** loop will **return** i if $A[i] = v$, in which case the **for** loop terminates and there is no further iteration. Otherwise, if $A[i] \neq v$ then at the start of the next for loop iteration $(i+1)-1$ is not an index of A such that $A[(i+1)-1] = v$. Finally, for termination to occur we have either $i = n+1$ where $n = A.length$ in which case the algorithm returns NIL indicating v is not an element of A. Otherwise, termination occurs because of the nested **if** on line 3 which causes the algorithm to return i which indicates the index of A such that $A[i] = v$.

2.1–4

Input: Two sequences of n integers, $A = (a_1, \ldots, a_n)$ and $B = (b_1, \ldots, b_n)$, such that $0 \leq a_i, b_i \leq 1$ for $i = 1, \ldots, n$. Least significant digits are first.

Output: An array $C = (c_1, \ldots, c_n, c_{n+1})$ such that $0 \leq c_i \leq 1$ for $i = 1, \ldots, n+1$ and $C' = A' + B'$ where $\cdot'$ is the integer represented by $\cdot$.

```
procedure BINARY ADDITION(A, B)
    Define integer C[A.length + 1]
    overflow = 0
    for i = 0 to A.length − 1 do
        C[i] = (A[i] + B[i] + overflow) % 2
        overflow = (A[i] + B[i] + overflow)/2
    end for
    C[i] = overflow
    return C
end procedure
```