Cormen Exercises - Lab 2

Fabio Steven Tovar Ramos

September 2018

1 Exercises

2.1-1 Illustrate the operation of INSERTION-SORT on the array $\langle 31, 4159, 26, 41, 58\rangle$
.

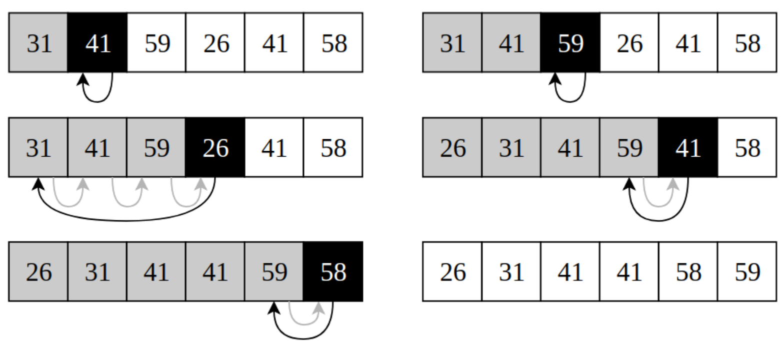

Figure 1: Insertion sort for an given array.

2.1-2 Rewrite the INSERTION-SORT procedure to sort into non-increasing instead of non-decreasing order.

Algorithm 1 Non-increasing Insertion Sort

procedure REVERSE-INSERTION-SORT(A)
  for $j \leftarrow 2$ to $A.length$ **do**
    $key \leftarrow A[1]$
    $i \leftarrow j - 1$
    while $i > 0$ and $key > A[i]$ **do**
      $A[i+1] \leftarrow A[i]$
      $i \leftarrow i - 1$
    $A[i+1] \leftarrow key$

2.1-3 Consider the ***searching problem:***
Input: A sequence of n numbers $A = \langle a_1, a_2, ..., a_n\rangle$ and a value v.

Output: https://dictionary.cambridge.org/es/gramatica/gramatica-britanica/can-could-or-mayAn index i such that $v = A[i]$ or the special value *NIL* if v does not appear in A.

Write pseudocode for ***linear search***, which scans through the sequence, looking for v. Using a loop invariant, prove that your algorithm is correct. Make sure that your loop invariant fulfills the three necessary properties.

Algorithm 2 Linear Search

```
1: procedure LINEAR-SEARCH(A, v)
2:     for i ← 1 to A.length do
3:         if v is equals to A[i] then
4:             return i
5:     return NIL
```

Loop Invariant and correctness of Linear Search

Loop invariant At the start of i iteration loop, $v \neq A[k]$ for all k integers in $[1, i)$, i.e., v is not in the sub-array $A[1...i-1]$ (sub-array of elements evaluated in previous iterations).

Initialization: It will start showing that the invariant holds true before the first iteration of *for loop*. In this state $i = 1$, so v is not in the sub-array due that $[1, 1)$ interval is empty.

Maintenance: The for-loop traverse the array incrementally, element by element supported by i counter. So that, in each iteration the intervals will grow by one making sure to check each element in the array.
Before the loop starts, v is not in the elements already evaluated because, in lines $3-4$ of Algorithm 2 the condition that v is different from the value of the array in this index ($A[i]$) will be checked. In case of these values are the same, the algorithm returns the i index and breaks the loop before that the counter is increased.

Termination: The Algorithm could be terminate in two cases:

(a) In some point of the iterations, v was found, so that the for loop is broken and the i counter is returned. Due of this, it can be affirmed that v is not in $A[1...i-1]$, otherwise the loop could not have reached i iteration.

(b) The for-loop finished and was not found v in the array. So v is not in A for all its values, especially v is not in $A[1...i-1]$ where $i = A.length$.

2.1-4 Consider the problem of adding two n-bit binary integers, stored in two n-element arrays A and B. The sum of the two integers should be stored in binary form in an $(n+1)$-element array C. State the problem formally and write pseudocode for adding the two integers.

Algorithm 3 Binary sum of two n-bit integers

procedure BINARY-SUM(A,B)
    $C :=$ array of size $A.length + 1$
    $i \leftarrow A.length$
    $carry \leftarrow 0$
    while $i > 0$ **do**
        $sum \leftarrow A[i] + B[i] + carry$
        $carry \leftarrow \lfloor sum/2 \rfloor$
        $C[i] \leftarrow sum \mod 2$
        $i \leftarrow i - 1$
    return C