Ejercicios Cormen

Luis Felipe Dávila Goyenche

September 2018

1 Exercises

1.1 Problem 2.1-1

Using Figure 2.2 as a model, illustrate the operation of INSERTION-SORT on the array $[31, 41, 59, 26, 41, 58]$.

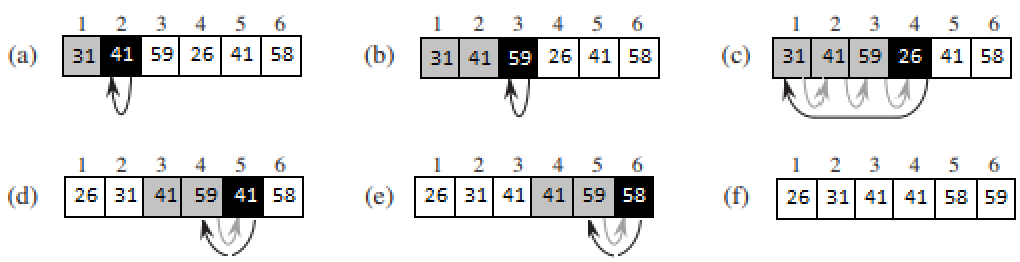

Figure 1: The operation of INSERTION-SORT on the array $[31, 41, 59, 26, 41, 58]$.

1.2 Problem 2.1-2

Rewrite the INSERTION-SORT procedure to sort into nonincreasing instead of nondecreasing order.

Algorithm 1 INSERTION-SORT

procedure INSERTIONSORT($array$)
    for $j \leftarrow 2$ `to lenthg(array)` **do**
        $key \leftarrow array[j]$
        while $j > 0$ `and` $key > array[j-1]$ **do**
            $array[j] \leftarrow array[j-1]$
            $j \leftarrow j-1$
        $array[j] \leftarrow key$

1.3 Problem 2.1-3

Consider the searching problem:
Input: A sequence of n numbers $[a_1, a_2, ..., a_n]$ and a value .
Output: An index i such that $A[i]$ or the special value NIL if does not appear in A.
Write pseudocode for *linearsearch*, which scans through the sequence, looking for . 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

procedure LINEARSEARCH(A, n)
  $index \leftarrow NULL$
  $\triangleright$ Invariant$(I_j) := \{\forall A_j \subseteq A$ de tamaño $j \leq length(A)$ con $A_j = [A[0], A[1], .., A[j-1]]$ se cumple que $n \notin A_j\}$
  for $j \leftarrow 0$ `to length(`A`)` **do**
    $\triangleright \{(j < length(A)) \cap (A[j] \notin A_j) \cap I_{j-1} \longrightarrow I_j\}$
    if $A[j] == n$ **then**
      $\triangleright$ $\{n \notin A_j \ \rightarrow I_j\} A[j] == n \{A[j] \notin A_j\}$
      $index \leftarrow j$
      $\triangleright \{(A[index] = A[j]) == n\} index = j \ \{A[index]\}$
      return $index$ $\triangleright$ $index \neq NULL$ is solution y $I_j ==$ `true para el final del loop`
  $\triangleright \{(j = length(A), A_j = A) \rightarrow (n \notin A, index = NULL)\}$
  return $index$

1.3.1 Prueba por inducción

`Initialization:` $j = 0$

$$(A_0 = \emptyset) \subseteq A \rightarrow A[0] \notin A_0 \tag{1}$$

Si $A[0] == n$ o $A[0]! = n$ entonces $n \notin A_j$ y el invariante de loop es verdadero justo antes del for.
`Maintenance:` `Suponer para` $j = k < length(A)$ y $A[j] \neq n$

$$(A_k = [A[0], A[1], ..., A[k-1]]) \subseteq A \rightarrow A[k] \notin A_k \tag{2}$$

`Se demuestra para` $j = k + 1$

$$A_{k+1} = [A[0], A[1], .., A[k-1], A[k]] = A_k \cup [A[k]] \tag{3}$$

$$A_k \cup [A[k]] \subseteq A \tag{4}$$

Dado que $n \notin A_k$ y $A[k] \neq n$ entonces $n \notin A_{k+1}$ por tanto la ivariante de loop se mantiene durante el cuerpo del loop siempre que $A[j]$ sea distinto de n
`Termination:` El loop finaliza cuando $A[j] == n$ o cuando $j = length(A)$.

En el el primer caso es evidente que $A[j] \notin A_j$ y por tanto si $A[j] == n$ entonces $n \notin A_j$ y la invariante se mantiene hasta el instante en que se realiza el realiza el retorno de la función, es decir, cuando finaliza el loop. En el caso que $j = length(A)$ entonces significa que $A_j = A$ y por tanto $A[n]$ no es un índice valido para acceder al arreglo, esto significa que no se encontró un objeto en el arreglo que fuese igual al valor del entrada o $n \notin (A_j = A)$ y nuevamente el invariante se cumple incluso en la finalización del loop.

1.4 Problem 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 Suma binaria

```
procedure BinaryAdding(A,B, n)
    C ← Array[n + 1]
    acarreo ← 0
    for i ← n to 0 do
        C[i] ← (A[i − 1] + B[i − 1] + acarreo)%2
        acarreo ← (A[i − 1] + B[i − 1] + acarreo)/2
    C[i] ← acarreo
    return C
```

References

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to Algorithms. McGraw-Hill, 2001.

James Apnes. Correctness proof. 2003