

2.1-1

Using Figure 2.2 as a model, illustrate the operation of INSERTION-SORT on an array initially containing the sequence $\{31, 41, 59, 26, 41, 58\}$.

31	41	59	26	41	58
----	----	----	----	----	----



31	41	59	26	41	58
----	----	----	----	----	----



31	41	59	26	41	58
----	----	----	----	----	----



26	31	41	59	41	58
----	----	----	----	----	----



26	31	41	41	59	58
----	----	----	----	----	----



26	31	41	41	58	59
----	----	----	----	----	----



2.1-2

Consider the procedure SUM-ARRAY on the facing page. It computes the sum of the n numbers in array $A[1:n]$. State a loop invariant for this procedure, and use its initialization, maintenance, and termination properties to show that the SUM-ARRAY procedure returns the sum of the numbers in $A[1:n]$.

SUM-ARRAY(A, n)

```
1  sum = 0
2  for i = 1 to n
3      sum = sum + A[i]
4  return sum
```

● Loop invariant ::

before each iteration, sum is equal to the sum of $i-1$ numbers in array $A[1:i-1]$

● Initialization ::

before the first iteration $i=1$, the subarray $A[1:i-1]$ consists of empty array $A[1:0]$ and sum equals 0 which trivially the sum of empty array.

● Maintenance ::

for each iteration i , the body of the loop adds element $A[i]$ to sum which represents the sum of $i-1$ numbers of subarray $A[1:i-1]$ in line 3. After line 3 is executed, sum represents the sum of i numbers of subarray $A[1:i]$ which maintains the loop invariant for next iteration.

● Termination :-

The loop terminates when $i=n+1$ so sum represents the sum of n numbers of subarray $A[1:n]$. we then return sum.

2.1-3

Rewrite the INSERTION-SORT procedure to sort into monotonically decreasing instead of monotonically increasing order.

REVERSED INSERTION-SORT (A, n)

for $i=2$ to n

 key = $A[i]$

$j = i-1$

 while $j > 0$ and $A[j] < \text{key}$

$A[j+1] = A[j]$

$j = j-1$

$A[j+1] = \text{key}$

2.1-4

Consider the **searching problem**:

Input: A sequence of n numbers $\langle a_1, a_2, \dots, a_n \rangle$ stored in array $A[1:n]$ and a value x .

Output: An index i such that x equals $A[i]$ or the special value NIL if x does not appear in A .

Write pseudocode for **linear search**, which scans through the array from beginning to end, looking for x . Using a loop invariant, prove that your algorithm is correct. Make sure that your loop invariant fulfills the three necessary properties.

LINEAR SEARCH (A, n, x)

1. $i = \text{NIL}$
2. for $j=1$ to n
3. if $A[j] = x$
4. $i = j$
5. break

Loop Invariance:-

If loop reaches iteration j ,
then $x \notin A[1:j-1]$ and $i=\text{NIL}$

Initialization :

before first iteration $j=1$, $A[1:j-1]$ is the empty array $A[1:0]$. so $x \notin A[1:0]$ and thus $i = \text{NIL}$.

Maintenance ::

If we reach iteration j , then $x \notin A[1:j-1]$. if $A[j]=x$, then $x \in A[1:j]$ so we break from loop and don't proceed to next iteration after setting $i=j$ (smallest i such that $x=A[i]$) otherwise $x \notin A[1:j]$ and proceed to next iteration $j+1$.

Termination ::

we have two scenarios for termination ::

1. we exit loop when $j=n+1$ so $x \notin A[1:j]$ and $i = \text{NIL}$
2. we break from loop on line 5 and $A[i]=x$. i is the first index that has key x

2.1-5

Consider the problem of adding two n -bit binary integers a and b , stored in two n -element arrays $A[0:n-1]$ and $B[0:n-1]$, where each element is either 0 or 1, $a = \sum_{i=0}^{n-1} A[i] \cdot 2^i$, and $b = \sum_{i=0}^{n-1} B[i] \cdot 2^i$. The sum $c = a + b$ of the two integers should be stored in binary form in an $(n+1)$ -element array $C[0:n]$, where $c = \sum_{i=0}^n C[i] \cdot 2^i$. Write a procedure ADD-BINARY-INTEGERS that takes as input arrays A and B , along with the length n , and returns array C holding the sum.

ADD-BINARY-INTEGERS(A, B, n)

let C be an empty array of size $n+1$

$\text{carry} = 0$

for $i=1$ to n

$$C[i] = (A[i] + B[i] + \text{carry}) \bmod 2$$

$$\text{carry} = \lfloor (A[i] + B[i] + \text{carry}) / 2 \rfloor$$

$$C[n+1] = \text{carry}$$