**Answers 01 Algorithms**

For all the following exercises, if you are given a list of numbers, they are unordered

NOTE: With e.g. {x} in my output I mean the value of ‘x’.

1. Put a list of two integers in ascending sequence

START

DECLARE VARIABLES

x, y: INTEGER

IF x < y THEN

OUTPUT ‘Order is: {x}, {y}’

ELSE

OUTPUT ‘Order is: {y}, {x}’

END IF

END

1. Put a list of ‘n’ integers in ascending sequence

(‘n’ is an unknown number, therefore two or more integers in the list).

Note: ‘a’ is an array of ‘n’ values. We want to sort ‘a’ in ascending order. That is, a[0] should be the smallest and a[n-1] should be the largest.

START

DECLARE VARIABLES

n, i, j, indexOfMinValue, temp: INTEGER

a: ARRAY[1, 2, ..., n] of INTEGER

FOR i = 0 to n-1 DO:

//Find array element with minimum value among a[i], a[i+1], ..., a[n-1]

indexOfMinValue = i

FOR j = i + 1 to n-1 DO:

IF a[j] < a[indexOfMinValue] THEN

indexOfMinValue = j

END IF

END FOR

//swap a[i] and a[indexOfMinValue]

temp = a[i]

a[i] = a[indexOfMinValue]

a[indexOfMinValue]= temp

END FOR

END

1. Do 1 and 2, but in descending sequence

* Put a list of two integers in descending sequence

START

DECLARE VARIABLES

x,y: INTEGER

IF x > y THEN

OUTPUT ‘Order is: {x}, {y}’

ELSE

OUTPUT ‘Order is: {y}, {x}’

END IF

END

* Put a list of ‘n’ integers in descending sequence

(‘n’ is an unknown number, therefore two or more integers in the list).

START

DECLARE VARIABLES

n, i, j, indexOfMaxValue, temp: INTEGER

a: ARRAY[1, 2, ..., n] of INTEGER

FOR i = 0 to n-1 DO:

//Find array element with maximum value among a[i], a[i+1], ..., a[n-1]

indexOfMaxValue = i

FOR j = i + 1 to n-1 DO:

IF a[j] > a[indexOfMaxValue] THEN

indexOfMaxValue = j

END IF

END FOR

//Swap a[i] and a[indexOfMaxValue]

temp = a[i]

a[i] = a[indexOfMaxValue]

a[indexOfMaxValue]= temp

END FOR

END

1. Find a specified integer in a list of other integers (bearing in mind that it might not be in the list)

START

DECLARE VARIABLES

x, n: INTEGER

a: ARRAY[1, 2, ..., n] of INTEGER

INPUT x

FOR i = 0 to n-1 DO:

IF x = a[i] THEN

OUTPUT ‘Integer {x} is found’

END IF

END FOR

END

1. Find the average of a list of integers: mean (the standard average), median and mode

* **Mean** (standard average) -> the sum of the values divided by the number of values

START

DECLARE VARIABLES

x, n, sum, count: INTEGER

a: ARRAY[1, 2, ..., n] of INTEGER

avg: DECIMAL

count = 0

sum = 0

FOR i = 0 to n-1 DO:

count = count + 1

sum = sum + a[i]

END FOR

avg = sum/count

OUTPUT ‘The mean is {avg}’

END

* **Median** - order the numbers (lowest to highest); the median is the one is in the middle of that list.
* If there are two middle values the median is their mean (standard average) – their sum divided by 2 (their number)

START

DECLARE VARIABLES

n, i, j, indexOfMinValue, temp: INTEGER

a: ARRAY[1, 2, ..., n] of INTEGER

avg: DECIMAL

//Order the numbers – lowest to highest

FOR i = 0 to n-1 DO:

//Find array element with minimum value among a[i], a[i+1], ..., a[n-1]

indexOfMinValue = i

FOR j = i + 1 to n-1 DO:

IF a[j] < a[indexOfMinValue] THEN

indexOfMinValue = j

END IF

END FOR

//Swap a[i] and a[indexOfMinValue]

temp = a[i]

a[i] = a[indexOfMinValue]

a[indexOfMinValue]= temp

END FOR

//Find the middle value in the ordered array; % refers to modulo

IF (n – 1) % 2 = 0 THEN

avg = a[(n – 1)/2]

ELSE

avg = (a[(n – 2)/2] + a[n/2])/2

END IF

OUTPUT ‘The median is {avg}’

END

* **Mode** – order the numbers lowest to highest; the mode is the number which appears the most often.

START

DECLARE VARIABLES

n, i, j, count, maxCount, avg: INTEGER

a: ARRAY[1, 2, ..., n] of INTEGER

//Count how many times each number occurs

maxCount = 0

FOR i = 0 to n-1 DO:

count = 0

FOR j = 0 to i DO:

IF a[i] = a[j]

count = count + 1

END IF

END FOR

//Find the number which occurs the most often

IF count > maxCount

maxCount = count

avg = a[i]

END IF

END FOR

IF maxCount = 1 THEN

OUTPUT ‘There is no mode’

ELSE

OUTPUT ‘The mode is {avg}’

END IF

END

1. Find the even numbers in a given list of integers

START

DECLARE VARIABLES

n, i: INTEGER

a: ARRAY[1, 2, ..., n] of INTEGER

FOR i = 0 to n-1 DO:

IF a[i] % 2 = 0 THEN

OUTPUT ‘{a[i]} is an even number’

END IF

END FOR

END

* **Biggest number in the list**

Given a list of numbers

Identify the largest number in the list

START

DECLARE VARIABLES

n, i, j, indexOfMinValue, temp: INTEGER

a: ARRAY[1, 2, ..., n] of INTEGER

//Order the numbers – lowest to highest

FOR i = 0 to n-1 DO:

//Find array element with minimum value among a[i], a[i+1], ..., a[n-1]

indexOfMinValue = i

FOR j = i + 1 to n-1 DO:

IF a[j] < a[indexOfMinValue] THEN

indexOfMinValue = j

END IF

END FOR

//Swap a[i] and a[indexOfMinValue]

temp = a[i]

a[i] = a[indexOfMinValue]

a[indexOfMinValue]= temp

END FOR

OUTPUT ‘The biggest number in the array is {a[n-1]}’

END

* Put a list of three integers in ascending sequence

START

DECLARE VARIABLES

n, i, j, indexOfMinValue, temp: INTEGER

a: ARRAY[1, 2, 3] of INTEGER

FOR i = 0 to 2 DO:

//Find array element with minimum value among a[0], a[1], a[2]

indexOfMinValue = i

FOR j = i + 1 to 2 DO:

IF a[j] < a[indexOfMinValue] THEN

indexOfMinValue = j

END IF

END FOR

//swap a[i] and a[indexOfMinValue]

temp = a[i]

a[i] = a[indexOfMinValue]

a[indexOfMinValue]= temp

END FOR

END

* Put a list of five integers in ascending sequence

START

DECLARE VARIABLES

n, i, j, indexOfMinValue, temp: INTEGER

a: ARRAY[1, 2, …, 5] of INTEGER

FOR i = 0 to 4 DO:

//Find array element with minimum value among a[0], a[1], …, a[4]

indexOfMinValue = i

FOR j = i + 1 to 4 DO:

IF a[j] < a[indexOfMinValue] THEN

indexOfMinValue = j

END IF

END FOR

//swap a[i] and a[indexOfMinValue]

temp = a[i]

a[i] = a[indexOfMinValue]

a[indexOfMinValue]= temp

END FOR

END

* Put a list of ten integers in ascending sequence

START

DECLARE VARIABLES

n, i, j, indexOfMinValue, temp: INTEGER

a: ARRAY[1, 2, …, 10] of INTEGER

FOR i = 0 to 9 DO:

//Find array element with minimum value among a[0], a[1], …, a[9]

indexOfMinValue = i

FOR j = i + 1 to 9 DO:

IF a[j] < a[indexOfMinValue] THEN

indexOfMinValue = j

END IF

END FOR

//swap a[i] and a[indexOfMinValue]

temp = a[i]

a[i] = a[indexOfMinValue]

a[indexOfMinValue]= temp

END FOR

END

* At this stage you have implemented a **sorting algorithm**. Take the opportunity to review what type of Sorting Algorithm you have implemented and briefly research the other types of sorting algorithms there are.

<http://faculty.cs.niu.edu/~hutchins/csci241/sorting.htm>

**Pseudocode for 3 Elementary Sort Algorithms**

If we want to sort an array, we have a wide variety of algorithms we can use to do the job. Three of the simplest algorithms are Selection Sort, Insertion Sort and Bubble Sort. None of these is especially efficient, but they are relatively easy to understand and to use.

In each of the three methods, we traverse all or part of an array repeatedly. There is a sorted part, which starts out empty and keeps growing, and there is an unsorted part, which initially is the whole array and keeps shrinking. We continue until we are done.

**Selection Sort**

Suppose A is an array of N values. We want to sort A in ascending order. That is, A[0] should be the smallest and A[N-1] should be the largest.

The idea of Selection Sort is that we repeatedly find the smallest element in the unsorted part of the array and swap it with the first element in the unsorted part of the array.

For I = 0 to N-1 do:

Smallsub = I

For J = I + 1 to N-1 do:

If A(J) < A(Smallsub)

Smallsub = J

End-If

End-For

Temp = A(I)

A(I) = A(Smallsub)

A(Smallsub) = Temp

End-For

A refinement of the above pseudocode would be to avoid swapping an element with itself.

An alternate way to sort in ascending order is to find the largest value and swap with the last element in the unsorted part of the array.

Selection Sort does roughly N\*\*2 / 2 comparisons and does N swaps.

**Insertion Sort**

Suppose A is an array of N values. We want to sort A in ascending order.

Insertion Sort is an algorithm to do this as follows: We traverse the array and insert each element into the sorted part of the list where it belongs. This usually involves pushing down the larger elements in the sorted part.

For I = 1 to N-1

J = I

Do while (J > 0) and (A(J) < A(J - 1)

Temp = A(J)

A(J) = A(J - 1)

A(J - 1) = Temp

J = J - 1

End-Do

End-For

Insertion Sort does roughly N\*\*2 / 2 comparisons and does up to N - 1 swaps.

**Bubble Sort**

Suppose A is an array of N values. We want to sort A in ascending order.

Bubble Sort is a simple-minded algorithm based on the idea that we look at the list, and wherever we find two consecutive elements out of order, we swap them. We do this as follows: We repeatedly traverse the unsorted part of the array, comparing consecutive elements, and we interchange them when they are out of order. The name of the algorithm refers to the fact that the largest element "sinks" to the bottom and the smaller elements "float" to the top.

For I = 0 to N - 2

For J = 0 to N - 2

If (A(J) > A(J + 1)

Temp = A(J)

A(J) = A(J + 1)

A(J + 1) = Temp

End-If

End-For

End-For

Bubble Sort does roughly N\*\*2 / 2 comparisons and does up to N\*\*2 / 2 swaps.

**Notes**

To sort in descending order instead, we would use > instead of < when we compare array elements.

It may be possible to improve the above algorithms slightly.

None of these is an especially efficient sorting algorithm. Each of them requires a number of steps proportional to N\*\*2. They vary in their efficiency, depending on whether the array is filled with random values or is already partly sorted or is already sorted in one direction or the other.

Bubble Sort is the best to use for an array which is already mostly in order, but it is the slowest to use for an array initially filled with random values. Insertion Sort is very fast if the array is already in ascending order. Selection Sort (unless we avoid self-swapping) is always about equally fast or slow.

* Put the numbers 1 to 5 in descending sequence

START

DECLARE VARIABLES

i, indexOfMaxValue, temp: INTEGER

a: ARRAY[1, 2, ..., 5] of INTEGER

a[] = [1, 2, 3, 4, 5]

indexOfMaxValue = 4

FOR i = 0 to 1 DO:

//Swap a[i] and a[indexOfMaxValue]

temp = a[i]

a[i] = a[indexOfMaxValue]

a[indexOfMaxValue]= temp

indexOfMaxValue = indexOfMaxValue - 1

END FOR

END

* Allow the user to specify if they want a list of ‘n’ integers in ascending or descending sequence

START

DECLARE VARIABLES

n, i, j, indexOfMinValue, temp: INTEGER

whatOrder: STRING

a: ARRAY[1, 2, ..., n] of INTEGER

INPUT whatOrder

IF whatOrder = ‘ascending’ THEN

FOR i = 0 to n-1 DO:

//Find array element with minimum value among a[i], a[i+1], ..., a[n-1]

indexOfMinValue = i

FOR j = i + 1 to n-1 DO:

IF a[j] < a[indexOfMinValue] THEN

indexOfMinValue = j

END IF

END FOR

//swap a[i] and a[indexOfMinValue]

temp = a[i]

a[i] = a[indexOfMinValue]

a[indexOfMinValue]= temp

END FOR

END IF

IF whatOrder = ‘desceding’ THEN

FOR i = 0 to n-1 DO:

//Find array element with maximum value among a[i], a[i+1], ..., a[n-1]

indexOfMaxValue = i

FOR j = i + 1 to n-1 DO:

IF a[j] > a[indexOfMaxValue] THEN

indexOfMaxValue = j

END IF

END FOR

//Swap a[i] and a[indexOfMaxValue]

temp = a[i]

a[i] = a[indexOfMaxValue]

a[indexOfMaxValue]= temp

END FOR

END IF

END