Introduction

Sorting algorithms have important applications in Value at Risk calculations and other risk and pricing functions in finance. For example, when calculating VaR via Monte Carlo simulation, we run a simulation of the asset prices many times, and save the change in portfolio value for each simulation. We then need to sort the many thousands of values in order to pick the Value at Risk at the required confidence interval.

Bubble Sort Algorithm and VBA Code

The bubble sort is a simple sorting algorithm that compares adjacent pairs of values within an array, adjusting the order of the values to place the smaller value before the larger one. It does this repeatedly for a cycle through the entire array. The algorithm counts whether or not any changes have been made during a cycle. The algorithm stops when no changes have been made.

The bubble sort is a relatively slow and non-scalable algorithm that is not recommended for use in larger data sets.

Function Bubblesort(Array_Values As Object)
    Application.Volatile (False)
    Application.ScreenUpdating = False

    Dim size As Long, changed As Long
    Dim tmp As Double, nums() As Double
    Dim limit As Long, i As Long, cycle As Long
    limit = Array_Values.Rows.Count
    ReDim Preserve nums(1 To limit)

    timer_val = Timer
    Do
        cycle = cycle + 1
        changed = 0
        For i = 1 To limit - 1
            If nums(i) > nums(i + 1) Then
                tmp = nums(i)
                nums(i) = nums(i + 1)
                nums(i + 1) = tmp
                changed = 1
            End If
        Next i
    Loop Until changed = 0

    Bubblesort = WorksheetFunction.Transpose(nums)
    Application.Volatile (True)
    Application.Calculation = xlCalculationAutomatic
    Application.ScreenUpdating = True

    End Function
3 Sorting Algorithms in Excel VBA: Bubble Sort, Insertion Sort, Quick Sort

**Insertion Sort Algorithm and VBA Code**

This compares each value x in an array to every other value in the array, and assigns a rank based on the number of values that are greater than x. x is then inserted in a new array at the same index as its rank in the original array. This process is repeated for all the values in the array.

The number of operations required in this approach is considerably lower than the bubble sort, so it performs better, but it is still a slower approach than recursive techniques such as the quick sort or merge sort.

```vba
Function InsertSort(Array_Values)
    Application.Volatile (False)
    Application.ScreenUpdating = False
    Dim nums() As Double
    Dim limit As Long
    Dim i As Long, j As Long
    Dim num_greater
    Dim new_array() As Double
    Dim base_variable As Double
    Dim Rank As Long
    limit = UBound(Array_Values)
    ReDim Preserve nums(1 To limit)
    ReDim Preserve new_array(1 To limit)
    For i = 1 To limit
        nums(i) = Array_Values(i)
    Next i
    For i = 1 To limit
        num_greater = 0
        base_variable = nums(i)
        For j = 1 To limit
            If base_variable < nums(j) Then
                num_greater = num_greater + 1
            End If
        Next j
        Rank = limit - num_greater
        new_array(Rank) = nums(i)
    Next i
    InsertSort = WorksheetFunction.Transpose(new_array)
    Application.Volatile (True)
    Application.Calculation = xlCalculationAutomatic
    Application.ScreenUpdating = True
End Function
```

**Quick Sort Algorithm and VBA Code**

```vba
Function QuickSort(Param_Array)
    Application.Volatile (False)
    Application.ScreenUpdating = False
    Dim nums() As Double
    Dim limit As Long
    Dim i As Long, j As Long
    Dim num_greater
    Dim new_array() As Double
    Dim base_variable As Double
    Dim Rank As Long
    limit = UBound(Param_Array)
    ReDim Preserve nums(1 To limit)
    ReDim Preserve new_array(1 To limit)
    For i = 1 To limit
        nums(i) = Param_Array(i)
    Next i
    For i = 1 To limit
        num_greater = 0
        base_variable = nums(i)
        For j = 1 To limit
            If base_variable < nums(j) Then
                num_greater = num_greater + 1
            End If
        Next j
        Rank = limit - num_greater
        new_array(Rank) = nums(i)
    Next i
    QuickSort = WorksheetFunction.Transpose(new_array)
    Application.Volatile (True)
    Application.Calculation = xlCalculationAutomatic
    Application.ScreenUpdating = True
End Function
```
Dim i As Long, j As Long
Dim less_or_equal() As Double
Dim greater() As Double
Dim Array_Values() As Double
Dim New_Array_Values()
Dim pivot As Double
Dim array1, array2

limit = UBound(Param_Array)
ReDim Preserve Array_Values(1 To limit)

For i = 1 To limit
    Array_Values(i) = Param_Array(i)
Next i

Select Case limit
Case Is <= 1
    QuickSort = Param_Array
    Exit Function
Case Is = 2
    pivot = (Array_Values(1) + Array_Values(2)) / 2
Case Is > 2
    pivot = WorksheetFunction.RoundUp(limit / 2, 0)
    pivot = WorksheetFunction.Median(Array_Values(pivot),
        Array_Values(1), Array_Values(limit))
End Select

Dim limit As Long, lim_less_or_equal As Long, lim_greater As Long
lim_less_or_equal = 0
lim_greater = 0

For i = 1 To limit
    If Array_Values(i) <= pivot Then
        lim_less_or_equal = lim_less_or_equal + 1
        ReDim Preserve less_or_equal(1 To lim_less_or_equal)
        less_or_equal(lim_less_or_equal) = Array_Values(i)
    Else
        lim_greater = lim_greater + 1
        ReDim Preserve greater(1 To lim_greater)
        greater(lim_greater) = Array_Values(i)
    End If
Next i

New_Array_Values = Concatenate_VBA(QuickSort(less_or_equal), QuickSort(greater))
QuickSort = New_Array_Values

Function Concatenate_VBA(Param_Array1, Param_Array2)
    Dim new_array()

    Dim limit1 As Long, limit2 As Long, i As Long, j As Long
    Dim array1(), array2()

    limit1 = UBound(Param_Array1)
    limit2 = UBound(Param_Array2)

End Function
ReDim Preserve array1(1 To limit1)
ReDim Preserve array2(1 To limit2)

For i = 1 To limit1
    array1(i) = Param_Array1(i)
    Next i

For i = 1 To limit2
    array2(i) = Param_Array2(i)
    Next i

ReDim Preserve new_array(1 To limit1 + limit2)

For i = 1 To limit1
    new_array(i) = array1(i)
    Next i

j = 0
For i = (limit1 + 1) To (limit1 + limit2)
    j = j + 1
    new_array(i) = array2(j)
    Next i

Concatenate_VBA = new_array

End Function

How VBA Developer.net Can Save You Time and money

You can get complete Excel apps from VBA Developer.net containing the code in this document, customisation, VBA development of any Excel, Access and Outlook apps, as well as C# and C++ add-ins and technical documentation.

Visit VBA Developer.net

Examples of VBA documents from VBA Developer.net

How to build a Black Scholes VBA Option Pricer

How to build a Black Scholes C# Option Pricer

How to build a Black Scholes VBA Option Pricer for FX Options

How to build a Black Scholes VBA Option Pricer for Equity Options

How to build a Black Scholes VBA Option Pricer using Monte Carlo Simulation
3 Sorting Algorithms in Excel VBA: Bubble Sort, Insertion Sort, Quick Sort

How to build a Black Scholes VBA Option Pricer for Binary Options

How to build a Black Scholes VBA Option Pricer for Equity Barrier Options

How to build a Black Scholes VBA Option Pricer for Exotic Asian Options

How to build a Black Scholes VBA Option Pricer for Exotic Lookback Options

How to build an Equity Option Pricer using the Binomial Tree in Excel VBA

How to code a Choleskey Decomposition in VBA (Numerical Methods for Excel)

3 ways to sort in VBA

How to Code a Multivariate Value at Risk (VaR) VBA Monte Carlo Simulation

How To Code the Newton-Raphson Method in Excel VBA

How to Model Volatility Smiles, Volatility Term Structure and the Volatility Surface in Excel VBA

How To Write Use Cases for a Portfolio Reporting VBA Tool

How To Design Classes for Object Orientated VBA Programming

How To Write a User Interface Model For a Portfolio Reporting VBA Tool

How To Create a Semantic Object Model For a Portfolio Reporting VBA Tool

How To Normalise a Database For VBA Apps

How To Create a Database using SQL Scripts for a Portfolio Reporting VBA App

How to Write Stored Procedures in SQL/Access/VBA for a Portfolio Reporting VBA App

How to Use Cursors in SQL for a Portfolio Reporting VBA Tool

How to Move Data from Access to Excel with SQL for a Portfolio Reporting VBA App

Portfolio Reporting VBA Tool: Inserting Data into SQL/Access Databases from Excel
Portfolio Reporting VBA Tool: Connecting Excel with SQL & Access Databases

How To Design Classes For an Option Pricer in VBA: UML Concepts