

## 1. Dawkins' Algorithm

In order to demonstrate the effects of a proper implementation of natural selection on the results of evolutionist algorithms, it would be pertinent to first duplicate the results of one of the better known and most widely cited algorithms to have been put forward by evolutionists so far. This algorithm is Richard Dawkins' famous "Weasel" algorithm, described in Chapter 3 of his book, "The Blind Watchmaker":

We again use our computer monkey, but with a crucial difference in its program. It again begins by choosing a random sequence of 28 letters, just as before ... it duplicates it repeatedly, but with a certain chance of random error – 'mutation' – in the copying. The computer examines the mutant nonsense phrases, the 'progeny' of the original phrase, and chooses the one which, *however slightly*, most resembles the target phrase, METHINKS IT IS LIKE A WEASEL.<sup>1</sup>

Dawkins uses a single point mutation per generation for each offspring derived from the original. There are 100 offspring per generation, and the final result is obtained in under 50 generations (generally). It is clear from the above quotation that the mutant with the highest fitness value will *always* be selected for in this algorithm.

A fairly simple and relatively quick version of this algorithm is listed below, along with its output. A more realistic version of this algorithm that uses more realistic calculations in its implementation of natural selection will be presented in the next section.

### Source Code Listing (VB6):

```
Option Explicit

Public Const TARGET_STRING As String = "METHINKS@IT@IS@LIKE@A@WEASEL"
Public Const OFFSPRING_PER_GENERATION = 100

Private mlngTarget() As Long
Private mlngParent() As Long
Private mlngLength As Long
Private mlngGeneration As Long

Public Sub Main()

    Dim lngDisplay As Long

    Randomize Timer

    Call StringToLong(TARGET_STRING, mlngTarget)
```

---

1 Dawkins, R. (1986) The Blind Watchmaker, Oxford University Press.

```

Call GenerateInitial

lngDisplay = 1
Do While CalculateScore(mlngParent) > 0
    If mlngGeneration = lngDisplay Then
        Call ShowResult
        lngDisplay = lngDisplay * 10
    End If

    Call GenerateOffspring
Loop

If mlngGeneration * 10 <> lngDisplay Then
    ShowResult
End If

End Sub

Private Sub GenerateInitial()

    Dim lngIndex As Long

    mlngLength = Len(TARGET_STRING)
    ReDim mlngParent(1 To mlngLength)
    mlngGeneration = 1
    For lngIndex = 1 To mlngLength
        mlngParent(lngIndex) = Int(Rnd * 27)
    Next lngIndex

End Sub

Public Sub StringToLong(ByRef StringIn As String, ByRef LongOut() As
Long)

    Dim lngIndex As Long

    ReDim LongOut(1 To Len(StringIn))
    For lngIndex = 1 To Len(StringIn)
        LongOut(lngIndex) = AscW(Mid$(StringIn, lngIndex, 1)) - 64
    Next lngIndex

End Sub

Public Function LongToString(ByRef LongIn() As Long) As String

    Dim lngIndex As Long

    LongToString = vbNullString
    For lngIndex = LBound(LongIn) To UBound(LongIn)
        LongToString = LongToString & Chr$(LongIn(lngIndex) + 64)
    Next lngIndex

```

```

End Function

Private Sub GenerateOffspring()

    Dim lngIndex As Long
    Dim lngBest() As Long
    Dim lngNext() As Long

    lngBest = mlngParent
    Call Mutate(lngBest(Int(Rnd * mlngLength) + 1))

    For lngIndex = 2 To OFFSPRING_PER_GENERATION
        lngNext = mlngParent
        Call Mutate(lngNext(Int(Rnd * mlngLength) + 1))

        If CalculateScore(lngNext) < CalculateScore(lngBest) Then
            lngBest = lngNext
        End If
    Next lngIndex

    mlngParent = lngBest
    mlngGeneration = mlngGeneration + 1

End Sub

Private Sub Mutate(ByRef FromVal As Long)

    FromVal = (FromVal + Int(Rnd * 26) + 1) Mod 27

End Sub

Private Function CalculateScore(ByRef CompareTo() As Long) As Long

    Dim lngIndex As Long

    For lngIndex = 1 To mlngLength
        If mlngTarget(lngIndex) <> CompareTo(lngIndex) Then
            CalculateScore = CalculateScore + 1
        End If
    Next lngIndex

End Function

Private Sub ShowResult()

    Debug.Print "Generation " & mlngGeneration & ": " &
        Replace$(LongToString(mlngParent), "@", " ") &
        " [" & CalculateScore(mlngParent) & "]"

End Sub

```

## Program Output:

```
Generation 1: XFLOOWJIW X KEFZBWO XXGVBUP [28]  
Generation 10: XELHWOWJII S FIBWO XWGVBUP [20]  
Generation 45: METHINKS IT IS LIKE A WEASEL [0]
```

