Recursive binary search

Bop: divide, base case: low > high

Binsearch (M, low, high) {

If low <= high then

Mid = (low + high)/2

If M[mid] = Target Then return mid

If M[mid] < Target Then result = binsearch(M, mid + 1, high)

If M[mid] > Target Then result = binsearch(M, low, mid – 1)

Else

Return -1

End If

Return result

}

Function Binsearch(M, low, high, target)

Dim mid As Integer

Dim result As Integer

If low <= high Then

mid = (low + high) / 2

If M(mid) = target Then Return mid

If M(mid) < target Then result = Binsearch(M, mid + 1, high, target)

If M(mid) > target Then result = Binsearch(M, low, mid - 1, target)

Else

Return -1

End If

Return result

End Function

M = [1,3,4,5,6] target = 3 binsearch (M, 0, len(m) -1)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Call Number | Low | High | Mid | Return |
| 1 | 0 | 4 | 2 |  |
| 2 | 0 | 1 | 0 |  |
| 3 | 1 | 1 | 1 | 1 |
| 2 | 0 | 1 | 0 | 1 |
| 1 | 0 | 4 | 2 | 1 |

M = [1,3,4,5,6] target = 7 binsearch (M, 0, len(m) -1)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Call Number | Low | High | Mid | Return |
| 1 | 0 | 4 | 2 |  |
| 2 | 3 | 4 | 3 |  |
| 3 | 4 | 4 | 4 |  |
| 4 | 5 | 4 | 4 | -1 |
| 3 | 4 | 4 | 4 | -1 |
| 2 | 3 | 4 | 3 | -1 |
| 1 | 0 | 4 | 2 | -1 |

M = [1,3,4,5,6] target = 2 binsearch (M, 0, len(m) -1)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Call Number | Low | High | Mid | Return |
| 1 | 0 | 4 | 2 |  |
| 2 | 0 | 1 | 0 |  |
| 3 | 1 | 1 | 1 |  |
| 4 | 1 | 0 | 0 | -1 |
| 3 | 1 | 1 | 1 | -1 |
| 2 | 0 | 1 | 0 | -1 |
| 1 | 0 | 4 | 2 | -1 |

Module Module1

Sub Main()

Dim ar() As Integer = {7, 34, 56, 76, 34, 2, 457, 7, 53, 24, 76, 735, 32465, 3}

Dim t(ar.Length - 1) As Integer

t = mergesort(ar)

For i = 0 To t.Length - 1

Console.WriteLine(t(i))

Next

Console.ReadLine()

End Sub

Function mergesort(ByVal ar() As Integer)

If ar.Length <= 1 Then Return ar

Dim left() As Integer = mergesort(ar.Take(ar.Length / 2).ToArray)

Dim right() As Integer = mergesort(ar.Skip(ar.Length / 2).Take(ar.Length - (ar.Length / 2)).ToArray)

Return merge(left, right)

End Function

Function merge(ByVal left() As Integer, ByVal right() As Integer)

Dim temp(left.Length + right.Length - 1) As Integer

Dim countL As Integer = 0

Dim countR As Integer = 0

Dim countT As Integer = 0

While countT < temp.Length

If countL < left.Length And countR < right.Length Then

If left(countL) < right(countR) Then

temp(countT) = left(countL)

countL += 1

countT += 1

ElseIf left(countL) > right(countR) Then

temp(countT) = right(countR)

countR += 1

countT += 1

Else

temp(countT) = right(countR)

countR += 1

countT += 1

temp(countT) = left(countL)

countL += 1

countT += 1

End If

ElseIf countL = left.Length Then

For i = countR To right.Length - 1

temp(countT) = right(i)

countT += 1

Next

ElseIf countR = right.Length Then

For i = countL To left.Length - 1

temp(countT) = left(i)

countT += 1

Next

End If

End While

Return temp

End Function

End Module