LAMPIRAN A

LISTING PROGRAM

```
Private Sub Command3_Click()
    End
End Sub

Private Sub compress_Click()
    m1.Visible = True
    m2.Visible = True
End Sub

Private Sub d1_Click()
    CompDecomp = 2
    Dim wktmulai As Single
    Dim decompress As Boolean
    Dim Dummy As Boolean
    Dim StTime As Double
    Dim Text As String
    Dim LastUsed As Integer
    If UBound(OriginalArray) = 0 Then
        MsgBox "There is nothing to compress/Decompress"
        Exit Sub
    End If
    If AutoDecodeIsOn = False Then
        decompress = True
        If CompDecomp = 0 Then Exit Sub
        If CompDecomp = 1 Then decompress = False
    Else
        decompress = True
    End If
    If decompress = True Then
        LastUsed = UsedCodecs(UBound(UsedCodecs))
        If JustLoaded = True Then LastUsed = 0
        If WichCompressor <> LastUsed Then
            Text = "This is not compressed with ." & Chr(13)
            MsgBox Text
            Exit Sub
        End If
    Else
        LastCoder = WichCompressor
    End If
    Call Copy_Orig2Work
    LastDeCoded = decompress
    If JustLoaded = True Then
        JustLoaded = False
        ReDim UsedCodecs(0)
    End If
    compression.MousePointer = MousePointerConstants.vbHourglass
    wktmulai = Timer
    Call DeCompress_arithmetic_Dynamic(hasil)
    Label12.Caption = Timer - wktmulai & " detik"
    compression.MousePointer = MousePointerConstants.vbDefault
    Call Show_Statistics(False, hasil)
```

```
    Call afterContent(hasil)
    compression.save.Visible = True
    d1.Visible = False
    d2.Visible = False
End Sub


Private Sub d2_Click()
    CompDecomp = 2
    Dim wktmulai As Single
    Dim decompress As Boolean
    Dim Dummy As Boolean
    Dim StTime As Double
    Dim Text As String
    Dim LastUsed As Integer
    If UBound(OriginalArray) = 0 Then
        MsgBox "There is nothing to
compress/Decompress"
        Exit Sub
    End If
    If AutoDecodeIsOn = False Then
        decompress = True
        If CompDecomp = 0 Then Exit Sub
        If CompDecomp = 1 Then decompress = False
    Else
        decompress = True
    End If
    If decompress = True Then
        LastUsed =
UsedCodecs(UBound(UsedCodecs))
        If JustLoaded = True Then LastUsed = 0
        If WichCompressor <> LastUsed Then
            Text = "This is not compressed with ." &
Chr(13)
            MsgBox Text
            Exit Sub
        End If
    Else
        LastCoder = WichCompressor
    End If
    Call Copy_Orig2Work
    LastDeCoded = decompress
    If JustLoaded = True Then
        JustLoaded = False
        ReDim UsedCodecs(0)
    End If
    compression.MousePointer =
MousePointerConstants.vbHourglass
    wktmulai = Timer
    Call DeCompress_ReducerDynamicGol(hasil)
    Label12.Caption = Timer - wktmulai & " detik"
    compression.MousePointer =
MousePointerConstants.vbDefault
    Call Show_Statistics(False, hasil)
    Call afterContent(hasil)
    compression.save.Visible = True
    d1.Visible = False
    d2.Visible = False
```

```
End Sub


Private Sub Form_Load()
    Dim X As Integer
    Dim Y As Integer
    Dim MaxWidth As Double
    ReDim OriginalArray(0)
    ReDim hasil(0)
    ReDim UsedCodecs(0)
    CompDecomp = 0
End Sub


Private Sub m1_Click()
        CompDecomp = 1
    Dim wktmulai As Single
    Dim decompress As Boolean
    Dim Dummy As Boolean
    Dim StTime As Double
    Dim Text As String
    Dim LastUsed As Integer
    If UBound(OriginalArray) = 0 Then
        MsgBox "There is nothing to
compress/Decompress"
        Exit Sub
    End If
    If AutoDecodeIsOn = False Then
        decompress = True
        If CompDecomp = 0 Then Exit Sub
        If CompDecomp = 1 Then decompress = False
    Else
        decompress = True
    End If
    If decompress = True Then
        LastUsed =
UsedCodecs(UBound(UsedCodecs))
        If JustLoaded = True Then LastUsed = 0
        If WichCompressor <> LastUsed Then
            Text = "This is not compressed with ." &
Chr(13)
            MsgBox Text
            Exit Sub
        End If
    Else
        LastCoder = WichCompressor
    End If
    Call Copy_Orig2Work
    LastDeCoded = decompress
    If JustLoaded = True Then
        JustLoaded = False
        ReDim UsedCodecs(0)
    End If
    compression.MousePointer =
MousePointerConstants.vbHourglass
    wktmulai = Timer
    Call Compress_arithmetic_Dynamic(hasil)
    Label12.Caption = Timer - wktmulai & " detik"
```

```
    compression.MousePointer =
MousePointerConstants.vbDefault
    Call Show_Statistics(False, hasil)
    Call afterContent(hasil)
    compression.save.Visible = True
    m1.Visible = False
    m2.Visible = False
End Sub


Private Sub m2_Click()
        CompDecomp = 1
    Dim wktmulai As Single
    Dim decompress As Boolean
    Dim Dummy As Boolean
    Dim StTime As Double
    Dim Text As String
    Dim LastUsed As Integer
    If UBound(OriginalArray) = 0 Then
        MsgBox "There is nothing to
compress/Decompress"
        Exit Sub
    End If
    If AutoDecodeIsOn = False Then
        decompress = True
        If CompDecomp = 0 Then Exit Sub
        If CompDecomp = 1 Then decompress = False
    Else
        decompress = True
    End If
    If decompress = True Then
        LastUsed =
UsedCodecs(UBound(UsedCodecs))
        If JustLoaded = True Then LastUsed = 0
        If WichCompressor <> LastUsed Then
            Text = "This is not compressed with ." &
Chr(13)
            MsgBox Text
            Exit Sub
        End If
    Else
        LastCoder = WichCompressor
    End If
    Call Copy_Orig2Work
    LastDeCoded = decompress
    If JustLoaded = True Then
        JustLoaded = False
        ReDim UsedCodecs(0)
    End If
    compression.MousePointer =
MousePointerConstants.vbHourglass
    wktmulai = Timer
    Call Compress_ReducerDynamicGol(hasil)
    Label12.Caption = Timer - wktmulai & " detik"
    compression.MousePointer =
MousePointerConstants.vbDefault
    Call Show_Statistics(False, hasil)
```

```
    Call afterContent(hasil)
    compression.save.Visible = True
    m1.Visible = False
    m2.Visible = False
End Sub
Private Sub open_Click()
    Dim OldFileName As String
    OldFileName = LoadFileName
    Cdlg.DialogTitle = "Select the file you want to
explore"
    Cdlg.FileName = ""
    Cdlg.ShowOpen
    LoadFileName = Cdlg.FileName
    Call load_File(LoadFileName)
    If LoadFileName = "" Then LoadFileName =
OldFileName
    Call Show_Contents(OriginalArray)
End Sub
Private Sub Decompress_Click()
    d1.Visible = True
    d2.Visible = True
End Sub
Private Sub Save_Click()
    Call Save_File_As(hasil, False)
End Sub
```

GOLOMB

```
Option Explicit

'This is a 1 run method but we have to keep the
whole contents
'in memory until some variables are saved wich are
needed bij the decompressor
Private Type BytePos
    Data() As Byte
    Position As Long
    Buffer As Integer
    BitPos As Integer
End Type
Private Stream(1) As BytePos    '0=control
1=BitStreams
Private CharCount(256) As Long
Private Dictionary As String
Private BitsForHeader As Integer   '1=max 6 chars
2=max 30 chars  3=more then 30 chars
Private Golomb(8) As Integer
Private RetGolomb(15) As Integer
Private BitsToFollow(8) As Integer
Public Sub
Compress_ReducerDynamicGol(ByteArray() As
Byte)
    Dim X As Long
    Dim Y As Long
    Dim NoMore As Boolean
    Dim Most As Long
    Dim NewFileLen As Long
```

```
    Dim Nuchar As Byte
    Dim CharCount(255) As Long
    Call Init_ReducerDynamicGol
'whe only read the stream and convert them to
bitstreams
    For X = 0 To UBound(ByteArray)
        Call AddValueToStream(CInt(ByteArray(X)))
    Next
'send the EOF-marker
    Call AddValueToStream(256)
'lets fill the leftovers
    For X = 0 To 1
        Do While Stream(X).BitPos > 0
            Call AddBitsToStream(Stream(X), 0, 1)
        Loop
    Next
'Lets restore the bounderies
    For X = 0 To 1
        ReDim Preserve
Stream(X).Data(Stream(X).Position - 1)
    Next
'whe calculate the new length of the new data
    NewFileLen = 0
    For X = 0 To 1
        NewFileLen = NewFileLen +
UBound(Stream(X).Data) + 1
    Next
    ReDim ByteArray(NewFileLen + 3)
'here we store the compressed data
    NewFileLen = 0
    For X = 0 To 0
        ByteArray(NewFileLen) =
(UBound(Stream(X).Data) And &HFF0000) /
&H10000
        NewFileLen = NewFileLen + 1
        ByteArray(NewFileLen) =
(UBound(Stream(X).Data) And &HFF00) / &H100
        NewFileLen = NewFileLen + 1
        ByteArray(NewFileLen) =
UBound(Stream(X).Data) And &HFF
        NewFileLen = NewFileLen + 1
    Next
    For X = 0 To 1
        For Y = 0 To UBound(Stream(X).Data)
            ByteArray(NewFileLen) = Stream(X).Data(Y)
            NewFileLen = NewFileLen + 1
        Next
    Next
End Sub
Public Sub
DeCompress_ReducerDynamicGol(ByteArray() As
Byte)
    Dim OutStream() As Byte
    Dim OutPos As Long
    Dim InposCont As Long
    Dim InContBit As Integer
    Dim InposData As Long
    Dim InDataBit As Integer
```

```
    Dim Char As Integer
    Dim NumBits As Integer
    Dim X As Long
    Dim Temp As Byte
    ReDim OutStream(500)
    Call Init_ReducerDynamicGol
    InposCont = 0
    InposData = 0
    For X = 0 To 2
        InposData = CLng(InposData) * 256 + ByteArray(InposCont)
        InposCont = InposCont + 1
    Next
    InposData = InposData + InposCont + 1
    InContBit = 0
    InDataBit = 0
    OutPos = 0
    Do
        NumBits = 0
        Temp = 0
        Temp = Temp * 2 + ReadBitsFromArray(ByteArray, InposCont, InContBit, 2)
        Do While RetGolomb(Temp) = 0
            Temp = Temp * 2 + ReadBitsFromArray(ByteArray, InposCont, InContBit, 1)
        Loop
        NumBits = RetGolomb(Temp)
        Char = ReadBitsFromArray(ByteArray, InposData, InDataBit, NumBits)
        Char = ExpanderBits(NumBits, Char)
        If Char = 256 Then Exit Do
        Call AddCharToArray(OutStream, OutPos, CByte(Char))
    Loop
    ReDim ByteArray(OutPos - 1)
    For X = 0 To OutPos - 1
        ByteArray(X) = OutStream(X)
    Next
End Sub

Private Sub Init_ReducerDynamicGol()
    Dim X As Integer
    Dictionary = ""
    For X = 0 To 255
        Dictionary = Dictionary & Chr(X)
        CharCount(X) = 0
    Next
    CharCount(256) = 0
    BitsForHeader = 3
    For X = 0 To 1
        ReDim Stream(X).Data(500)
        Stream(X).BitPos = 0
        Stream(X).Buffer = 0
        Stream(X).Position = 0
    Next
    Golomb(1) = 0: BitsToFollow(1) = 2    '00
```

```
Golomb(2) = 1: BitsToFollow(2) = 2    '01
Golomb(3) = 4: BitsToFollow(3) = 3    '100
Golomb(4) = 5: BitsToFollow(4) = 3    '101
Golomb(5) = 12: BitsToFollow(5) = 4   '1100
Golomb(6) = 13: BitsToFollow(6) = 4   '1101
Golomb(7) = 14: BitsToFollow(7) = 4   '1110
Golomb(8) = 15: BitsToFollow(8) = 4   '1111
For X = 0 To 15
   RetGolomb(X) = 0
Next
RetGolomb(0) = 1
RetGolomb(1) = 2
RetGolomb(4) = 3
RetGolomb(5) = 4
RetGolomb(12) = 5
RetGolomb(13) = 6
RetGolomb(14) = 7
RetGolomb(15) = 8
End Sub

Private Function ReducerBits(Char As Integer) As
Integer
   Dim DiPos As Integer
   Dim TotPos As Integer
   Dim Y As Integer
   If Char = 256 Then ReducerBits = 8: Char = 255:
Exit Function
   DiPos = InStr(Dictionary, Chr(Char)) - 1
   Call update_Model(Char)
   For Y = 1 To 8
      If DiPos >= TotPos And DiPos < TotPos + 2 ^
Y Then
         ReducerBits = Y
         Char = DiPos - TotPos
         Exit Function
      End If
      TotPos = TotPos + 2 ^ Y
   Next
End Function

Private Function ExpanderBits(BitsNum As Integer,
BytePos As Integer) As Integer
   If BitsNum = 8 And BytePos = 255 Then
ExpanderBits = 256: Exit Function
   Dim TotPos As Integer
   Dim Y As Integer
   For Y = 1 To BitsNum - 1
      TotPos = TotPos + 2 ^ Y
   Next
   TotPos = TotPos + BytePos + 1
   ExpanderBits = Asc(Mid(Dictionary, TotPos, 1))
   Call update_Model(ExpanderBits)
End Function

Private Sub update_Model(Char As Integer)
   Dim Dictpos As Integer
   Dim OldPos As Integer
```

```
    Dim Temp As Long

    Dictpos = InStr(Dictionary, Chr(Char))

    OldPos = Dictpos

    CharCount(Dictpos) = CharCount(Dictpos) + 1

    Do While Dictpos > 1 And CharCount(Dictpos) >=
CharCount(Dictpos - 1)

        Temp = CharCount(Dictpos - 1)

        CharCount(Dictpos - 1) = CharCount(Dictpos)

        CharCount(Dictpos) = Temp

        Dictpos = Dictpos - 1

    Loop

    If OldPos = Dictpos Then Exit Sub

    Dictionary = Left(Dictionary, Dictpos - 1) &
Chr(Char) & Mid(Dictionary, Dictpos, OldPos -
Dictpos) & Mid(Dictionary, OldPos + 1)

End Sub



Private Sub AddValueToStream(Number As
Integer)

    Dim BitsDeep As Integer

    BitsDeep = ReducerBits(Number)

    Call AddBitsToStream(Stream(0),
Golomb(BitsDeep), BitsToFollow(BitsDeep))

    Call AddBitsToStream(Stream(1), Number,
BitsDeep)

End Sub



'this sub will add an amount of bits to a sertain
stream

Private Sub AddBitsToStream(Toarray As BytePos,
Number As Integer, NumBits As Integer)

    Dim X As Long

    If NumBits = 8 And Toarray.BitPos = 0 Then

        If Toarray.Position > UBound(Toarray.Data)
Then ReDim Preserve
Toarray.Data(Toarray.Position + 500)

        Toarray.Data(Toarray.Position) = Number And
&HFF

        Toarray.Position = Toarray.Position + 1

        Exit Sub

    End If

    For X = NumBits - 1 To 0 Step -1

        Toarray.Buffer = Toarray.Buffer * 2 + (-1 *
((Number And 2 ^ X) > 0))

        Toarray.BitPos = Toarray.BitPos + 1

        If Toarray.BitPos = 8 Then

            If Toarray.Position > UBound(Toarray.Data)
Then ReDim Preserve
Toarray.Data(Toarray.Position + 500)

            Toarray.Data(Toarray.Position) =
Toarray.Buffer

            Toarray.BitPos = 0

            Toarray.Buffer = 0

            Toarray.Position = Toarray.Position + 1

        End If

    Next

End Sub



'this function will return a value out of the amaunt of
bits you asked for

Private Function ReadBitsFromArray(FromArray()
As Byte, FromPos As Long, FromBit As Integer,
NumBits As Integer) As Long

    Dim X As Integer
```

```
    Dim Temp As Long
    For X = 1 To NumBits
        Temp = Temp * 2 + (-1 *
((FromArray(FromPos) And 2 ^ (7 - FromBit)) > 0))
        FromBit = FromBit + 1
        If FromBit = 8 Then
            If FromPos + 1 > UBound(FromArray) Then
                Do While X < NumBits
                    Temp = Temp * 2
                    X = X + 1
                Loop
                FromPos = FromPos + 1
                Exit For
            End If
            FromPos = FromPos + 1
            FromBit = 0
        End If
    Next
    ReadBitsFromArray = Temp
End Function


'this sub will add a char into the outputstream
Private Sub AddCharToArray(Toarray() As Byte,
ToPos As Long, Char As Byte)
    If ToPos > UBound(Toarray) Then ReDim
Preserve Toarray(ToPos + 500)
    Toarray(ToPos) = Char
    ToPos = ToPos + 1
End Sub
```

ARITHMATIC

```
Option Explicit
'This is a 1 run method
Private OutStream() As Byte
Private OutPos As Long
Private OutBitCount As Integer
Private OutByteBuf As Byte
Private CharCount(257) As Long
Private Const MaxBits As Integer = 24
Private Bits_To_Follow As Integer
Private Const EOF_Symbol = 256
Public Sub
Compress_arithmetic_Dynamic(ByteArray() As
Byte)
    Dim InpPos As Long
    Dim Low As Long
    Dim High As Long
    Dim Range As Long
    Dim Half As Long
    Dim First_Qtr As Long
    Dim Third_Qtr As Long
    Dim Mid As Long
    Dim TotChars As Long
    Dim Char As Integer
    Dim Index As Integer
    Dim X As Integer
    Call Init_Arithmetic_Dynamic
```

```
    Low = 0
    High = (2 ^ MaxBits) - 1
    Half = High / 2
    First_Qtr = Half / 2
    Third_Qtr = Half + First_Qtr
    Char = 0
    Do
       If InpPos > UBound(ByteArray) Then
          Char = EOF_Symbol
       Else
          Char = ByteArray(InpPos)
       End If
       InpPos = InpPos + 1
       Range = High - Low
       High = Low + CLng(Range *
(CharCount(Char) / CharCount(0)))
       Low = Low + CLng(Range *
(CharCount(Char + 1) / CharCount(0)))
       Do
          If High < Half Then
             Call Bit_Plus_Follow(0)            '*
Output 0 if in low half. *'
          ElseIf Low >= Half Then            '*
Output 1 if in high half.*'
             Call Bit_Plus_Follow(1)
             Low = Low - Half
             High = High - Half                 '*
Subtract offset to top.  *'
          ElseIf Low >= First_Qtr And High <
Third_Qtr Then          '* Output an opposite
bit   *'
             Bits_To_Follow = Bits_To_Follow + 1
'* later if in middle half. *'
             Low = Low - First_Qtr              '*
Subtract offset to middle*'
             High = High - First_Qtr
          Else        '* Otherwise exit loop.    *'
             Exit Do
          End If
          Low = 2 * Low
          High = 2 * High + 1       '* Scale up code
range.    *'
       Loop
       If Char = EOF_Symbol Then Exit Do
       Call update_Model(Char)
    Loop
    For X = MaxBits - 1 To 0 Step -1
       If (Low And 2 ^ X) = 0 Then
          Call AddBitsToOutStream(0, 1)
       Else
          Call AddBitsToOutStream(1, 1)
       End If
    Next
    Do While OutBitCount > 0
       Call AddBitsToOutStream(1, 1)
    Loop
    ReDim ByteArray(OutPos - 1)
    Call CopyMem(ByteArray(0), OutStream(0),
OutPos)
```

```
End Sub


Public Sub
DeCompress_arithmetic_Dynamic(ByteArray()
As Byte)
   Dim InpPos As Long
   Dim InBitPos As Integer
   Dim Low As Long
   Dim High As Long
   Dim Range As Long
   Dim Half As Long
   Dim First_Qtr As Long
   Dim Third_Qtr As Long
   Dim Mid As Long
   Dim Value As Long
   Dim TotChars As Long
   Dim Char As Integer
   Dim Index As Integer
   Dim Counter As Long
   Dim Temp As Integer
   Dim X As Integer
   Call Init_Arithmetic_Dynamic
   Value = 0
   InpPos = 0
   InBitPos = 0
   Value = ReadBitsFromArray(ByteArray,
InpPos, InBitPos, MaxBits)
   Low = 0
   High = (2 ^ MaxBits) - 1
   Half = High / 2
   First_Qtr = Half / 2
   Third_Qtr = Half + First_Qtr
   Char = 0
   Do
      If InpPos > UBound(ByteArray) Then
         Exit Do
      End If
      If OutPos = 15 Then
         OutPos = 15
      End If
      Range = High - Low
      Counter = Int((Value - Low + 1) *
(CharCount(0) / Range))
      For Char = 0 To 256
         If CharCount(Char) <= Counter Then
            Exit For
         End If
      Next
      Char = Char - 1
      If Char = EOF_Symbol Then Exit Do
      High = Low + CLng(Range *
(CharCount(Char) / CharCount(0)))
      Low = Low + CLng(Range *
(CharCount(Char + 1) / CharCount(0)))
      Call update_Model(Char)
      Call AddValueToOutStream(Char)
```

```
    Do                          '* Loop to get rid of
bits. *'

        If InpPos <= UBound(ByteArray) Then

          If High < Half Then

            '* nothing *'               '* Expand
low half.       *'

            Value = 2 * Value +
ReadBitsFromArray(ByteArray, InpPos,
InBitPos, 1)       '* Move in next input bit.  *'

          ElseIf Low >= Half Then           '*
Expand high half.       *'

            Value = Value - Half

            Low = Low - Half                '*
Subtract offset to top.  *'

            High = High - Half

            Value = 2 * Value +
ReadBitsFromArray(ByteArray, InpPos,
InBitPos, 1)       '* Move in next input bit.  *'

          ElseIf Low >= First_Qtr And High <
Third_Qtr Then '* Expand middle half.     *'

            Value = Value - First_Qtr

            Low = Low - First_Qtr           '*
Subtract offset to middle*'

            High = High - First_Qtr

            Value = 2 * Value +
ReadBitsFromArray(ByteArray, InpPos,
InBitPos, 1)       '* Move in next input bit.  *'

          Else        '* Otherwise exit loop.    *'

            Exit Do

          End If

          Low = 2 * Low

          High = 2 * High + 1               '* Scale
up code range.    *'

        Else

          Exit Do

        End If

      Loop

    Loop

    ReDim ByteArray(OutPos - 1)

    Call CopyMem(ByteArray(0), OutStream(0),
OutPos)

End Sub


Private Sub Init_Arithmetic_Dynamic()

    Dim X As Integer

    ReDim OutStream(500)

    OutPos = 0

    OutBitCount = 0

    OutByteBuf = 0

    Bits_To_Follow = 0

    For X = 0 To 257

      CharCount(X) = 258 - X

    Next

End Sub


Private Sub update_Model(Index As Integer)

    Dim I As Integer

    I = Index

    Do While I >= 0
```

```
      CharCount(I) = CharCount(I) + 1
      I = I - 1
   Loop
End Sub


Private Sub Bit_Plus_Follow(Bit As Integer)
   Call AddBitsToOutStream(CLng(Bit), 1)
'* Output the bit.          *'
   Do While Bits_To_Follow > 0
      Call AddBitsToOutStream(1 - Bit, 1)
'* Output bits_to_follow    *'
      Bits_To_Follow = Bits_To_Follow - 1
'* opposite bits. Set       *'
   Loop                                '*
bits_to_follow to zero.  *'
End Sub


Private Sub AddValueToOutStream(Number
As Integer)
   If OutPos > UBound(OutStream) Then
ReDim Preserve OutStream(OutPos + 100)
   OutStream(OutPos) = Number
   OutPos = OutPos + 1
End Sub


Private Sub AddBitsToOutStream(Number As
Long, NumBits As Integer)
   Dim X As Long
   For X = NumBits - 1 To 0 Step -1
      OutByteBuf = OutByteBuf * 2 + (-1 *
((Number And CDbl(2 ^ X)) > 0))
      OutBitCount = OutBitCount + 1
      If OutBitCount = 8 Then
         OutStream(OutPos) = OutByteBuf
         OutBitCount = 0
         OutByteBuf = 0
         OutPos = OutPos + 1
         If OutPos > UBound(OutStream) Then
            ReDim Preserve OutStream(OutPos +
500)
         End If
      End If
   Next
End Sub


'this function will return a value out of the
amaunt of bits you asked for
Private Function
ReadBitsFromArray(FromArray() As Byte,
FromPos As Long, FromBit As Integer, NumBits
As Integer) As Long
   Dim X As Integer
   Dim Temp As Long
   For X = 1 To NumBits
      Temp = Temp * 2 + (-1 *
((FromArray(FromPos) And 2 ^ (7 - FromBit)) >
0))
      FromBit = FromBit + 1
      If FromBit = 8 Then
         If FromPos + 1 > UBound(FromArray)
Then
```

```
            Do While X < NumBits
               Temp = Temp * 2
               X = X + 1
            Loop
            FromPos = FromPos + 1
            Exit For
         End If
         FromPos = FromPos + 1
         FromBit = 0
      End If
   Next
   ReadBitsFromArray = Temp
End Function
```

Global module

```
Public Declare Sub CopyMem Lib "kernel32" Alias "RtlMoveMemory" (Destination As Any, source As Any, ByVal Length As Long)

Public OriginalArray() As Byte          'array to store the original

Public OriginalSize As Long             'size of the original file

Public hasil() As Byte                  'array to store the results

Public LoadFileName As String
'which file is loeded

Public JustLoaded As Boolean            'to see if the original file is just loaded

Public DictionarySize As Integer        'for use with LZW and LZSS compressors

Public LastCoder As Integer             'wich coder is used last

Public UsedCodecs() As Integer          'to store the codecs used

Public LastDeCoded As Boolean           'to see what has happen lately

Public AritmaticRescale As Boolean      'to set rescale on

Public CompDecomp As Integer
'result from compress or decompress screen


'constants for the coder types

Public Const Coder_Differantiator = 1

Public Const Coder_FrequentieShifter = 2

Public Const Coder_BWT = 3

Public Const Coder_Fix128 = 4

Public Const Coder_Flatter64 = 5

Public Const Coder_MTF_No_Header = 6

Public Const Coder_MTF_With_Header = 7

Public Const Coder_SortSwap = 8

Public Const Coder_Used_To_Front = 9

Public Const Coder_ValueDownShift = 10

Public Const Coder_ValueUpShift = 11

Public Const Coder_ValueTwister = 12
```

```
Public Const Coder_Fix128B = 13

Public Const Coder_Seperator = 14

Public Const Coder_Flatter16 = 15

Public Const Coder_Base64 = 16

Public Const Coder_Numerator = 17

Public Const Coder_Numerator2 = 18

Public Const Coder_AddDifferantiator = 19

Public Const Coder_Scrambler = 20

Private CodName(20) As String

'constants for the compressor types

Public Const Compressor_EliasGamma =
16

Public CompName(61) As String

Private AutoDecodeIsOn As Boolean   'to
see if autodecode is used


'this routine is to initialize the text which
can be returned to

'the user to say which compressor/coder
is used

Public Sub Init_CoderNameDataBase()

   CompName(Compressor_EliasGamma)
= "Elias Gamma"

End Sub


'copy original array to the hasil so that
whe can work on it without

'changing the original contents

Public Sub Copy_Orig2Work()

   ReDim hasil(UBound(OriginalArray))

   Call CopyMem(hasil(0), OriginalArray(0),
UBound(OriginalArray) + 1)

End Sub


'copy the hasil to the original array so that
whe can apply a

'second compress/coder on the file

Public Sub Copy_Work2Orig()

   ReDim OriginalArray(UBound(hasil))

   Call CopyMem(OriginalArray(0), hasil(0),
UBound(hasil) + 1)

End Sub


'This sub is used to start a coder

'whichcoder is the constant of the
codertype

'Decode is used to say if we want to code
or decode

Public Sub Start_Coder(WichCoder)

   Dim Decode As Boolean

   Dim StTime As Double

   Dim Text As String

   Dim LastUsed As Integer

   If UBound(OriginalArray) = 0 Then

      MsgBox "There is nothing to
code/Decode"
```

```
    Exit Sub
  End If
  If AutoDecodeIsOn = False Then
    Decode = True
    frmCodeDecode.Show vbModal
    DoEvents
    If CompDecomp = 0 Then Exit Sub
    If CompDecomp = 1 Then Decode =
False
  Else
    Decode = True
  End If
  If Decode = True Then
    LastUsed =
UsedCodecs(UBound(UsedCodecs))
    If JustLoaded = True Then LastUsed =
0
    If (WichCoder Or 128) <> LastUsed
Then
      Text = "This is not coded with the "
& CodName(WichCoder) & Chr(13)
      If LastUsed = 0 Then
        Text = Text & "Its not coded at
all"
      Else
        If LastUsed > 128 Then
          Text = Text & "Its coded with
the " & CodName(LastUsed And 127)
        Else
          Text = Text & "Its compressed
with " & CompName(LastUsed)
        End If
      End If
      MsgBox Text
      Exit Sub
    End If
  Else
    LastCoder = WichCoder Or 128
  End If
  LastDeCoded = Decode
  Call Copy_Orig2Work
  If JustLoaded = True Then
    JustLoaded = False
    ReDim UsedCodecs(0)
  End If
  StTime = Timer
  compression.MousePointer =
MousePointerConstants.vbHourglass
  Select Case WichCoder
    Case 1
      If Decode = False Then Call
Difference_Coder(hasil)
      If Decode = True Then Call
Difference_DeCoder(hasil)
    Case 2
      If Decode = False Then Call
FrequentShifter_Coder(hasil)
```

```
        If Decode = True Then Call
FrequentShifter_DeCoder(hasil)

    Case 3

        If Decode = False Then Call
BWT_CodecArray4(hasil)

        If Decode = True Then Call
BWT_DeCodecArray4(hasil)

    Case 4

        If Decode = False Then Call
Fix128_Coder(hasil)

        If Decode = True Then Call
Fix128_DeCoder(hasil)

    Case 5

        If Decode = False Then Call
FlattenTo64(hasil)

        If Decode = True Then Call
DeFlattenTo64(hasil)

    Case 6

        If Decode = False Then Call
MTF_CoderArray(hasil)

        If Decode = True Then Call
MTF_DeCoderArray(hasil)

    Case 7

        If Decode = False Then Call
MTF_CoderArray2(hasil)

        If Decode = True Then Call
MTF_DeCoderArray2(hasil)

    Case 8

        If Decode = False Then Call
Sort_Swap_Coder(hasil)

        If Decode = True Then Call
Sort_Swap_DeCoder(hasil)

    Case 9

        If Decode = False Then Call
Used2Front_Coder(hasil)

        If Decode = True Then Call
Used2Front_DeCoder(hasil)

    Case 10

        If Decode = False Then Call
ValueDownShifter_Coder(hasil)

        If Decode = True Then Call
ValueDownShifter_DeCoder(hasil)

    Case 11

        If Decode = False Then Call
ValueUpShifter_Coder(hasil)

        If Decode = True Then Call
ValueUpShifter_Coder(hasil)

    Case 12

        If Decode = False Then Call
ValueTwister_Coder(hasil)

        If Decode = True Then Call
ValueTwister_DeCoder(hasil)

    Case 13

        If Decode = False Then Call
Fix128B_Coder(hasil)

        If Decode = True Then Call
Fix128B_DeCoder(hasil)

    Case 14

        If Decode = False Then Call
Seperator_Coder(hasil)
```

```
        If Decode = True Then Call
Seperator_DeCoder(hasil)

      Case 15

        If Decode = False Then Call
Flatter16_coder(hasil)

        If Decode = True Then Call
Flatter16_Decoder(hasil)

      Case 16

        If Decode = False Then Call
Base64Array_Encode(hasil)

        If Decode = True Then Call
Base64Array_Decode(hasil)

      Case 17

        If Decode = False Then Call
Numerator_EnCoder(hasil)

        If Decode = True Then Call
Numerator_DeCoder(hasil)

      Case 18

        If Decode = False Then Call
Numerator2_EnCoder(hasil)

        If Decode = True Then Call
Numerator2_DeCoder(hasil)

      Case 19

        If Decode = False Then Call
AddDiffer_Coder(hasil)

        If Decode = True Then Call
AddDiffer_DeCoder(hasil)

      Case 20

        If Decode = False Then Call
Scrambler_Coder(hasil)

        If Decode = True Then Call
Scrambler_DeCoder(hasil)

   End Select

   compression.MousePointer =
MousePointerConstants.vbDefault

   If AutoDecodeIsOn = False Then

      Call Show_Statistics(False, hasil,
Timer - StTime)

   End If

End Sub

'This sub is used to start a compressor

'whichcoder is the constant of the
compressortype

'Decode is used to say if we want to
compress or decompress

Public Sub
Start_Compressor(WichCompressor)

   Dim decompress As Boolean

   Dim Dummy As Boolean

   Dim StTime As Double

   Dim Text As String

   Dim LastUsed As Integer

   If UBound(OriginalArray) = 0 Then

      MsgBox "There is nothing to
compress/Decompress"

      Exit Sub

   End If

   If AutoDecodeIsOn = False Then
```

```
      decompress = True
      If CompDecomp = 0 Then Exit Sub
      If CompDecomp = 1 Then decompress
= False
   Else
      decompress = True
   End If
   If decompress = True Then
      LastUsed =
UsedCodecs(UBound(UsedCodecs))
      If JustLoaded = True Then LastUsed =
0
      If WichCompressor <> LastUsed Then
         Text = "This is not compressed with
" & CompName(WichCompressor) & "." &
Chr(13)
         If LastUsed = 0 Then
            Text = Text & "Its not
compressed at all"
         Else
            If LastUsed > 128 Then
               Text = Text & "Its coded with
the " & CodName(LastUsed And 127)
            Else
               Text = Text & "Its compressed
with " & CompName(LastUsed)
            End If
         End If
         MsgBox Text
         Exit Sub
      End If
   Else
      LastCoder = WichCompressor
   End If
   Call Copy_Orig2Work
   LastDeCoded = decompress
   If JustLoaded = True Then
      JustLoaded = False
      ReDim UsedCodecs(0)
   End If
   compression.MousePointer =
MousePointerConstants.vbHourglass
   StTime = Timer
   If decompress = False Then Call
Compress_arithmetic_Dynamic(hasil)
   End
   compression.MousePointer =
MousePointerConstants.vbDefault
   If AutoDecodeIsOn = False Then
      Call Show_Statistics(False, hasil,
Timer - StTime)
   End If
End Sub


'this sub is used to load a chosen file
Public Sub load_File(Name As String)
```

```
    Dim FreeNum As Integer

    If Name = "" Then Exit Sub

    FreeNum = FreeFile

    Open Name For Binary As #FreeNum

    ReDim OriginalArray(0 To
LOF(FreeNum) - 1)

    Get #FreeNum, , OriginalArray()

    Close #FreeNum

    JustLoaded = True

    Call
Split_Header_From_File(OriginalArray)

    compression.Caption = "loading" &
LoadFileName & ""

    OriginalSize = UBound(OriginalArray) +
1

    Call Show_Statistics(True, OriginalArray)

End Sub
```

```
'this sub is used to see if the file just
loaded is a file which is

'stored by this programm and is already
coded/compressed

Private Sub
Split_Header_From_File(ByteArray() As
Byte)

    Dim HeadText As String

    Dim X As Integer

    Dim CodecsUsed As Integer

    Dim Version As String

    Dim InPos As Long

    InPos = UBound(ByteArray)

    For X = 0 To 2

        HeadText = HeadText &
Chr(ByteArray(InPos))

        InPos = InPos - 1

    Next
```

```
    If HeadText <> "UCF" Then Exit Sub
'this is an un-UCF'ed file

    Version = Chr(ByteArray(InPos))

    InPos = InPos - 1

    Select Case Version

        Case "0"

            CodecsUsed = ByteArray(InPos)

            InPos = InPos - 1

            ReDim UsedCodecs(CodecsUsed)

            For X = 1 To CodecsUsed

                UsedCodecs(X) =
ByteArray(InPos)

                InPos = InPos - 1

            Next

            ReDim Preserve ByteArray(InPos)

    End Select

    ReDim hasil(0)

    JustLoaded = False
```

```
End Sub


'this sub is used to save a file

'if the file was coded/compressed the types of coders/compressors used

'will be saved with the file so that we can recall it later

Public Sub Save_File_As(ByteArray() As Byte, source As Boolean)

   Dim FileNr As Integer

   Dim HeadArray() As Byte

   Dim OutHead As Integer

   Dim HeadText As String

   Dim Answer As Integer

   Dim CodecsUsed As Integer

   Dim SaveName As String

   Dim ExtPos As Integer

   Dim Temp As Integer

   Dim X As Integer

   If UBound(ByteArray) = 0 Then

      MsgBox "There is nothing to be saved"

      Exit Sub

   End If

   If source = False And LastCoder <> 0 Then Call AddCoder2List(LastCoder)

   If UBound(UsedCodecs) = 0 And UBound(ByteArray) = UBound(OriginalArray) Then

      Answer = MsgBox("The file to save is the same as the original file" & Chr(13) & "Still want to save this file", vbYesNo + vbExclamation)

      If Answer = vbNo Then

         Exit Sub

      End If

   End If

Ask_SaveName:

   SaveName = ""

   compression.Cdlg.DialogTitle = "Type in the name you want to save with"

   compression.Cdlg.FileName = ""

   compression.Cdlg.ShowSave

   SaveName = compression.Cdlg.FileName

   If SaveName = "" Then

      If source = False And LastCoder <> 0 Then

         ReDim Preserve UsedCodecs(UBound(UsedCodecs) - 1)

         LastCoder = UsedCodecs(UBound(UsedCodecs))

      End If

      Exit Sub

   End If

   Temp = 0

   Do
```

```
      ExtPos = Temp
      Temp = InStr(ExtPos + 1, SaveName, ".")
   Loop While Temp <> 0
   If ExtPos = 0 Or ExtPos < Len(SaveName) - 5 Then
      SaveName = SaveName & ".hmf"
   End If
'store the header in reversed order at the end of the file
   HeadText = "UCF0"
   If LastCoder = 0 And source = False Then
      CodecsUsed = 0
   Else
      CodecsUsed = UBound(UsedCodecs)
   End If
   ReDim HeadArray(4 + CodecsUsed)
   OutHead = 0
   For X = CodecsUsed To 1 Step -1
      HeadArray(OutHead) = UsedCodecs(X)
      OutHead = OutHead + 1
   Next
   HeadArray(OutHead) = CodecsUsed
   OutHead = OutHead + 1
   For X = Len(HeadText) To 1 Step -1
      HeadArray(OutHead) = Asc(Mid(HeadText, X, 1))
      OutHead = OutHead + 1
   Next
   FileNr = FreeFile
   If Dir(SaveName, vbNormal) <> "" Then
      Answer = MsgBox("File already exists" & Chr(13) & Chr(13) & "Overwrite", vbCritical + vbYesNo)
      If Answer = vbNo Then
         GoTo Ask_SaveName
      End If
      Kill SaveName   'first remove it otherwise size is not adjusted
   End If
   Open SaveName For Binary As #FileNr
   Put #FileNr, , ByteArray()
   If CodecsUsed > 0 Then
      Put #FileNr, , HeadArray()
   End If
   Close #FileNr
End Sub


'this sub is used to show the statistics of a file
'it can display both the original as the hasil
Public Sub Show_Statistics(OrgData As Boolean, Data() As Byte)
   Dim StatWindow As Integer
```

```
    Dim Frequentie(255) As Long
    Dim SortFreq(1, 255) As Long
    Dim Counts() As Long
    Dim X As Long
    Dim Minval As Long
    Dim Maxval As Long
    Dim next_offset As Long
    Dim this_count As Long
    Dim HeightValue As Double
    Dim Entry As String
    Dim NewSize As String
    Dim NuSize As Long
    Dim BPB As String
    If OrgData = False Then StatWindow = 1
    NuSize = UBound(Data) + 1
    BPB = Format(((NuSize * 8) / OriginalSize), "###0.000") & " bpb"
    NewSize = NuSize & " Bytes ¬_¬ " & Format(100 - (OriginalSize - NuSize) / OriginalSize * 100, "##0.00") & "% "
    For X = 0 To UBound(Data)
        Frequentie(Data(X)) = Frequentie(Data(X)) + 1
    Next
    Minval = UBound(Data)
    For X = 0 To 255
        If Minval > Frequentie(X) Then Minval = Frequentie(X)
        If Maxval < Frequentie(X) Then Maxval = Frequentie(X)
    Next
' Lets use the counting sort to sort them into another array
' Create the Counts array.
    ReDim Counts(Minval To Maxval)
' Count the items.
    For X = 0 To 255
        Counts(Frequentie(X)) = Counts(Frequentie(X)) + 1
    Next X
' Convert the counts into offsets.
    next_offset = 0
    For X = Maxval To Minval Step -1
        this_count = Counts(X)
        Counts(X) = next_offset
        next_offset = next_offset + this_count
    Next X
' Place the items in the sorted array.
    For X = 0 To 255
        SortFreq(0, Counts(Frequentie(X))) = Frequentie(X)
        SortFreq(1, Counts(Frequentie(X))) = X
```

```
    Counts(Frequentie(X)) =
Counts(Frequentie(X)) + 1

  Next X

  compression.Size(StatWindow).Caption
= NewSize

End Sub

Public Sub Show_Contents(ByteArray() As
Byte)

  Dim X As Long

  Dim Y As Integer

  Dim AddText As String

  Dim Data As Byte

  Dim Text As String

  X = UBound(ByteArray)

  If X = 0 Then

    MsgBox "Ther is nothing to see
because there is no data"

    Exit Sub

  End If

  On Error GoTo 0

  compression.beforeC.Clear

  For X = 0 To UBound(ByteArray) Step 50

    AddText = String(50, " ")

    For Y = 0 To 49

      If X + Y <= UBound(ByteArray) Then

        Data = ByteArray(X + Y)

        If Data < 28 Then Text = Chr(1)
Else Text = Chr(Data)

        Mid(AddText, Y + 1, 1) = Text

      End If

    Next

      compression.beforeC.AddItem
AddText

  Next

End Sub

Public Sub afterContent(dataComp() As
Byte)

  Dim X As Long

  Dim Y As Integer

  Dim AddText As String

  Dim Data As Byte

  Dim Text As String

  X = UBound(dataComp)

  If X = 0 Then

    MsgBox "der is nothing to see
because der is no data"

    Exit Sub

  End If

  On Error GoTo 0

  compression.afterC.Clear

  For X = 0 To UBound(dataComp) Step 50

    AddText = String(50, " ")

    For Y = 0 To 49

      If X + Y <= UBound(dataComp) Then
```

```
            Data = dataComp(X + Y)

            If Data < 28 Then Text = Chr(1)
Else Text = Chr(Data)

            Mid(AddText, Y + 1, 1) = Text

         End If

      Next

         compression.afterC.AddItem
AddText

   Next

End Sub


'this sub is used to store a
coder/compressor type into an array

'so that we can keep up which
coders/compressors are used to get to

'the last file whe have standing in the
original array

Public Sub AddCoder2List(CodeNumber As
Integer)

   JustLoaded = False

   If LastDeCoded = True Then

      If UBound(UsedCodecs) > 0 Then

         ReDim Preserve
UsedCodecs(UBound(UsedCodecs) - 1)

         LastCoder =
UsedCodecs(UBound(UsedCodecs))

      End If

      Exit Sub

   End If

   ReDim Preserve
UsedCodecs(UBound(UsedCodecs) + 1)

   UsedCodecs(UBound(UsedCodecs)) =
CodeNumber

End Sub


'this sub is used to decode/uncompress
automaticly without the user

'having search which type of
coder/compressor was used

Public Sub Auto_Decode_Depack()

   Dim X As Integer

   Dim CodeNumber As Integer

   If UBound(OriginalArray) = 0 Then

      MsgBox "There is nothing to
Decode/Decompress"

      Exit Sub

   End If

   If UBound(UsedCodecs) = 0 Or
JustLoaded = True Then

      MsgBox "This file was'nt
Coded/Compressed"

      Exit Sub

   End If

   CodeNumber =
UsedCodecs(UBound(UsedCodecs))

   AutoDecodeIsOn = True
```

```
    If CodeNumber > 128 Then
        Call Start_Coder(CodeNumber And 127)
    Else
        Call Start_Compressor(CodeNumber)
    End If
    Call Copy_Work2Orig
    ReDim hasil(0)
    compression.AscTab(1).Clear
    compression.FreqTab(1).Clear
    compression.FileSize(1).Caption = " "
    AutoDecodeIsOn = False
    If UBound(UsedCodecs) > 0 Then
        ReDim Preserve UsedCodecs(UBound(UsedCodecs) - 1)
        LastCoder = UsedCodecs(UBound(UsedCodecs))
    End If
    Call Show_Statistics(True, OriginalArray)
End Sub


'this sub is used to compare the original array with the hasil
Public Sub Compare_Source_With_Target()
    Dim FileSize As Long
    Dim SameSize As Boolean
    Dim Text As String
    Dim Equal As Boolean
    Dim X As Long
    SameSize = True
    Equal = True
    If UBound(OriginalArray) = 0 Then
        MsgBox "There is nothing to compare"
        Exit Sub
    End If
    If UBound(hasil) = 0 Then
        MsgBox "There is nothing to compare with"
        Exit Sub
    End If
    FileSize = UBound(OriginalArray)
    If UBound(hasil) <> FileSize Then
        SameSize = False
        If UBound(hasil) < FileSize Then
            FileSize = UBound(hasil)
        End If
    End If
    For X = 0 To FileSize
        If OriginalArray(X) <> hasil(X) Then
            Equal = False
            Exit For
```

```
    End If
  Next
  If Equal = False Then
    Text = "The files are different at
position " & X
    If SameSize = False Then
      Text = Text & Chr(13) & "And They
dont have the same size"
    End If
    MsgBox Text
    Exit Sub
  End If
  If SameSize = False Then
    Text = "The files are almost the same
except that they don't have the same size"
  Else
    Text = "the two files are the same"
  End If
    MsgBox Text
End Sub
```

LAMPIRAN B

ISI FILE UNTUK PERCOBAAN

1. File 1

For Wikipedia guidelines, see Wikipedia:What is an article.

An article is a stand-alone section of a larger written work. These nonfictional prose compositions appear in magazines, newspapers, academic journals, the Internet or any other type of publication.

Articles can be divided into two main categories: news and features. Straight news stories deal with the timeliness and immediacy of breaking news, while feature articles are news stories that deal with human-interest topics[1] or which offer the opportunity for providing more breadth or depth, context of history or other explanatory background material.

Contents

[hide]

[edit] News Articles

See also: News style

A news article is an article published in a print or Internet news medium such as a newspaper, newsletter, news magazine or news-oriented website that discusses current or recent news of either general interest (i.e. daily newspapers) or on a specific topic (i.e. political or trade news magazines, club newsletters, or technology news websites).

A news article can include accounts of eyewitnesses to the happening event. It can contain photographs, accounts, statistics, graphs, recollections, interviews, polls, debates on the topic, etc. Headlines can be used to focus the reader's attention on a particular (or main) part of the article. The writer can also give facts and detailed information following answers to general questions like who, what, when, where, why and how.

Quoted references can also be helpful. References to people can also be made through written accounts of interviews and debates confirming the factuality of the writer's information and the reliability of her source. The writer can use redirection to ensure that the reader keeps reading the article and to draw his attention to other articles. For example, phrases like "Continued on page 3" redirect the reader to a page where the article is continued.

While a good conclusion is an important ingredient for newspaper articles, the immediacy of a deadline environment means that copy editing often takes the form of deleting everything past an arbitrary point in the story corresponding to the dictates of available space on a page. Therefore, newspaper reporters are trained to write in inverted pyramid style, with all the most important information in the first paragraph or two. If less vital details are pushed towards the end of the story, the potentially destructive impact of draconian copy editing will be minimized.

[edit] Feature Articles

See also: Feature writing

Feature articles are nonfiction articles that intend to inform, teach or amuse the reader on a topic. The topic centers around human interests. Feature stories may include conventions found in fiction such as dialogue, plot and character. A feature article is an umbrella term that includes

many literary structures: personality sketches, essays, how-to's, interviews and many others.[2] The following are examples of feature articles:

* Column — A short newspaper or magazine piece that deals specifically with a particular field of interest, or broadly with an issue or circumstance of far-reaching scope. They appear with bylines on a regular basis (daily, weekly, etc.). They may be written exclusively for one newspaper or magazine, they may be marketed by a syndicate, or they may be self-syndicated by the author.

* Essay — A short, literary, nonfiction composition (usually prose), in which a writer develops a theme or expresses an idea.

* Evergreen — A timeless article that editors can hold for months and publish when needed. They need little or no updating.[3]

* Exposé — These articles use in-depth reporting with heavy research and documentation. Used to expose corruption in business, politics or celebrities. Also called the investigative article.[2][3]

* Filler — Short non-fiction items, usually just under 300 words, used to fill in small spaces on a page of a magazine or newspaper page.[4]

the number of times that their articles are cited

.

Publishing: 1990. ISBN 0844259616, pp. 8, 31, 50, 96-97

6. ^ a b Jacobi, Peter, The Magazine Article: How to Think It, Plan It, Write It. Writer's Digest Books: 1991, ISBN 0898794501, pp. 50-77, 90

[edit] External links

* How-to Articles

* How to Write Articles

* How to Write Special Feature Articles

* Getting Published: Your Article Submission Checklist

* Article Submission

Retrieved from "http://en.wikipedia.org/wiki/Article_%28publishing%29"

Categories: Narrative structures | Writing

2. File 2

Eyetracking points the way to effective news article design

OJR's design experts review usability research and offer suggestions on how you can make your online articles better connect with readers.

By Laura Ruel and Nora Paul

Posted: 2007-03-13

When one of world's best-known usability experts, Jakob Nielsen, conducts eyetracking research to test what his usability work has shown, the results generate some beneficial tips for online editors. This is what happened in late 2005, when Nielsen and Kara Pernice Coyne, the Nielsen/Norman Group's director of research, conducted an eyetracking test with 255 people in New York City.

With a little more than half of the participants (63 percent) ages 30 to 49, the test generated results applicable to the target audience for most news sites. Additionally, 20 percent were 18-29 and 16 percent were 50-64. Fifty-eight percent were female, 42 percent were male. Every test subject was given 50 tasks to complete. Sessions with each test subject lasted about one to two hours.

Coyne (who we interviewed for this column) stresses that crucial to understanding the testing results is an awareness of the user's motivation or goal behind each task. Some of the testing scenarios included asking the user to "read the news" or "read/learn", making a number these results particularly helpful to journalists. She said eyetracking is valuable in these cases because it indicates not only where

our users look, but where key usability problems exist.

"[With eyetracking] we can see that a user may navigate the page of an interface that houses the info she wants," she said, "but if the text is poorly presented, or the navigation is cluttered, or there are too many superfluous images so she cannot easily find what she needs. This is a lost opportunity."

We've featured three of the more interesting journalistic study results below.

Featured finding #1:

Rewrite + reformat = remember

What if you could engage users in a story for about half the time, yet have them remember about 34 percent more of the content? That's exactly what one test showed. Spending less than two hours rewriting and reformatting a story about New York City restaurants really paid off according to this study.

The image below shows the two stories tested:

Image

The original version (left) was revised to increase white space, make the main idea concise, remove unnecessary images, shorten lines of text and add a graphic for each restaurant ranking. (Nielsen/Norman Group images, used with permission)

The eyetracking data is featured in the image below. Red areas indicate the areas where the fixation length (or the length of time the users spent look at that area of the screen) was longest. Dark areas indicate low or no fixation length on that part of the presentation.

Image

Users spent a longer amount of time (about one minute) viewing the original version of the content (left) but remembered 34 percent less than those who received the reformatted story (right). In both cases a greater amount of time was spent on the left-hand side of the page. (Nielsen/Norman Group images, used with permission)

3. File 3

Love that garlic? Fresh may be healthier than bottled

The next time you use garlic for its renowned antibacterial effects, consider fresh garlic instead of those bottles of chopped garlic. Researchers in Japan report that fresh garlic maintains higher levels of a key healthy ingredient than preserved versions and may be better for you. Their study is scheduled for the June 25 issue of ACS' Journal of Agricultural and Food Chemistry, a bi-weekly publication.

In the new study, Toyohiko Ariga and colleagues point out that allicin is one of the main active ingredients in garlic. Other studies have shown that allicin has beneficial effects in preventing blood clots, cancer, and bacterial infection. Although commercially bottled garlic is often stored in oil or water, researchers did not know how various storage and preservation methods affect levels of allicin, which is fragile and disappears quickly.

To find out, Ariga's group compared allicin levels in extracts of fresh garlic after 1-2 weeks of storage in water, alcohol, and vegetable oil. Garlic stored in water at room temperature lost about half its allicin in 6 days and garlic in vegetable oil lost half its allicin in less than an hour. The garlic lost its antibacterial action as allicin broke down. However, allicin broke down into materials that still are believed to have some anticancer and anti-blood clot effects. - MTS

ARTICLE: "Biological and Chemical Stability of Garlic-Derived Allicin"

CONTACT:

Toyohiko Ariga, Ph.D.

Nihon University

Fujisawa, Japan

Sniffing out a broad-spectrum of airborne threats in seconds

Scientists in California are reporting successful laboratory and field tests of a new device that can sniff out the faintest traces of a wide range of chemical, biological, nuclear, and explosive threats - and illicit drugs - from the air in minutes with great accuracy. The ultra-sensitive detector, known as the single-particle aerosol mass spectrometry (SPAMS) system, could tighten security at airports, sports stadiums and other large-scale facilities, according to their report, scheduled for the July 1 issue of ACS' Analytical Chemistry, a semi-monthly journal.

Matthias Frank and colleagues explain that chemical, biological, nuclear, and explosive materials, as well as illicit drugs, all release minute amounts of aerosol particles into the air. Detecting these particles requires a device with a high sensitivity, low probability of false alarms, and a fast response time. "SPAMS uniquely meets these requirements in realistic field environments," the report states. While other aerosol detectors exist, SPAMS is specifically designed for the rapid detection of low-concentration aerosols, it adds.

The study describes laboratory tests of SPAMS and extended field tests at San Francisco International Airport. It showed that within seconds, SPAMS detected a diverse set of materials including simulants for potentially hazardous biological, chemical and radiological materials, as well as actual explosives and drugs. The study terms SPAMS a "significant and important advance in rapid aerosol threat detection." - AD

ARTICLE: "Autonomous, Broad-Spectrum Detection of Hazardous Aerosols in Seconds"

CONTACT:

Matthias Frank, Ph.D.

Lawrence Livermore National Laboratory

Livermore, California 94550

Inhalable form of gene-therapy takes aim at lung cancer and inflammatory lung disease

A new inhalable form of gene therapy - based on technology recognized in the 2006 Nobel medicine prize, shows increasing promise for treating lung cancer, infectious diseases and inflammatory lung disease, scientists have concluded after an exhaustive review of worldwide research on the topic. Their report is scheduled for the June 2 issue of ACS' Molecular Pharmaceutics, a bi-monthly journal.

In the article, Sally-Ann Cryan, Niamh Durcan, and Charlotte Murphy focus on research efforts to develop an inhalable form of RNA interference (RNAi), a gene-therapy technique that interferes with or "silences" genes that make disease-causing proteins. The authors explain that RNAi has advantages over other gene therapies. It is potent, very specific, and appears to have a low risk of side effects.

They cite encouraging results with RNAi in laboratory studies in cells and animals with a range of lung diseases, including lung cancer, certain respiratory infections and inflammatory lung disease. Keys to successful therapy in humans include careful design of the gene-silencing agents, determining the most effective doses, and developing better ways of delivering RNAi agents to the lungs, the scientists say. - MTS

ARTICLE: "Inhalable siRNA: Potential as a Therapeutic Agent in the Lungs"

CONTACT:

Sally-Ann Cryan, Ph.D.

Royal College of Surgeons in Ireland

Dublin, Ireland

Researchers band together in global battle on bacterial biofilms

The discovery that bacteria are not loners, but social creatures that congregate and chemically communicate in communities - termed biofilms - has sparked a global scientific effort to control

spread of these slimy coatings that grow on hospital surfaces, inside tubing, and a multitude of other places. That's the topic of an article scheduled for the June 9 issue of Chemical & Engineering News, ACS' weekly newsmagazine.

In the C&EN cover story, Senior Editor Lisa M. Jarvis points that biofilms are the major culprit behind hospital-acquired infections that are now the fourth leading cause of death in the United States, claiming thousands of lives each year. Biofilms also cause other problems ranging from dental plaque to the biofouling of ship hulls. The films are large, complex communities of bacteria that are difficult to kill.

But researchers from academia and industry are now collaborating in a global effort to develop promising new strategies to combat this problem. New approaches include the development of non-stick surfaces and the identification of chemicals that silence bacterial communication or starve them of key nutrients. The first commercial compound to specifically target biofilms is still a few years away, according to the article.

ARTICLE: "Communal Living"

4. File 4

IBM launched on Tuesday an application that seeks to harness the power and time of Internet users around the globe to make the Web more accessible to the visually impaired.

ADVERTISEMENT

Many blind or partially sighted users run screen reading software that describes the content of a Web page but often encounter problems. The screen readers rely on text or descriptive tags to explain the items on a page but these are often added as an after thought or are incomplete

Using the new IBM software users can report these problems to a central database and ask for additional descriptive text to be added to a site. Other Internet users that want to contribute can then check the database, select one of the submitted problems and "start fixing it" by added text labels. The additional information isn't incorporated into the original site's HTML code but into a metadata file that is loaded each time a visually impaired user subsequently visits the site.

"This idea came from my own experience with inaccessible Web sites," said Chieko Asakawa, a researcher at IBM in Tokyo who led a six-person team on development of the software. Asakawa is blind herself so knows well the problems of navigating the Web and its increasing rich multimedia pages.

"As users we face a lot of problems everyday but currently we don't have any mechanism to report what we have found. Every day we find images without alternative text (the text description of an image that usually accompanies it in the HTML code) but there is no way for me to say 'I want to have a description for this image.' It's a simple motivation but if we can report this kind of problem without difficulty and have it easily understood by sighted people I think it's going to be great."

IBM began offering the software from Tuesday as a beta release through its AlphaWorks Web site.

The software for blind or partially sighted users runs with Internet Explorer and the "Jaws" screen reader while the software for supporters of the project is available as a plug-in for Firefox. It runs in English or Japanese.

Demonstrating the system, Asakawa typed in the address for the White House Web site and soon found problems. While the site appears to have been designed with accessibility in mind, the headings at the top of the three main columns had no data attached that would allow her screen-reading software to make sense of what they were.

5. File 5

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "Gdefns.h"

// uncompressor (Golomb) for sparse file.

// usage: GolombDecode < file.GZ > file.decoded

// (c) David J.C. MacKay

// License: GPL
http://www.gnu.org/copyleft/gpl.html

// Originates from:
http://www.inference.phy.cam.ac.uk/mackay/itpr
nn/code/c/compress/

void      printzeroes( int , int *  ) ;

// prints a string of r zeroes, PRECEDED by "1" if
the second flag is set.

void      printzeroes( int r , int *needtoprint1 ) {

  if ( *needtoprint1 ) {

    printf("1\n");

    *needtoprint1 = 0 ;

  }

  for ( ; r>0 ; r-- ) {

    printf("0\n");

  }

}

int main( int argc , char *argv[] ){

  FILE *fp;

  int r , u , needtoprint1 = 0 ;

  //  unsigned char c ;

  int c ;

  fp=stdin;

  r = 0 ;

  while( (c=getc(fp)) != EOF ) {

    if( c == '1' ) {

      printzeroes(MGolomb, &needtoprint1 );

      r = 0 ;

    }    else if (c=='0') {

      // read in the next m bits as an integer

      r=0;

      for ( u=1 ; ( (u<=mGolomb) && (c=getc(fp)) !=
EOF ) ; u++ ) {

            if( c == '1' ) {          r++   ;    }

            if ( u < mGolomb ) { r = r * 2 ; }

      }

      printzeroes(r, &needtoprint1); // Ideally we
should print a "1" immediately, but

      // instead we postpone this printing event
and make a note that we need to do it later;

      // this is an ugly hack to cope with the "add a
virtual trailing 1" behaviour of the encoder.

      // The final "1" does not get printed.

      needtoprint1 = 1;

    } else {

      // carriage returns

    }

  }

  return(0);

}
```

6. File 6

```
"Sleepy" -- A Sample Adventure Game in Prolog

David Matuszek, Villanova University

You are welcome to use any and all parts of this
program in your own adventure game.

/* "Sleepy" -- a sample adventure game, by
David Matuszek. */

/* In standard Prolog, all predicates are
"dynamic": they

  can be changed during execution. SWI-Prolog
requires such

  predicates to be specially marked. */

:- dynamic at/2, i_am_at/1, i_am_holding/1,
alive/1,

        lit/1, visible_object/1.

/* This routine is purely for debugging purposes.
*/

dump :- listing(at), listing(i_am_at),
listing(i_am_holding),

      listing(alive), listing(lit),
listing(visible_object).

/* This defines my current location. */


i_am_at(bedroom).


/* These facts describe how the rooms are
connected. */

path(bedroom, n, den) :- lit(bedroom).

path(bedroom, n, den) :-

            write('You trip over something in the
dark.'), nl,

            !, fail.

path(den, s, bedroom).

path(bedroom, d, bed).

path(bed, u, bedroom).

/* These facts tell where the various objects in
the game

  are located. */

at(flyswatter, den).

at(fly, bedroom).

at('light switch', den).

at('light switch', bedroom).

/* These facts specify some game-specific
information. */

alive(fly).

lit(bedroom).

lit(den).

visible_object('light switch').

/* These rules describe how to pick up an object.
*/

take(fly) :-

            write('It''s too fast for you!'), nl,

            !, fail.

take('light switch') :-

            take(switch).

take(switch) :-

            write('It''s firmly embedded in the
wall!'), nl,

            !, fail.

take(X) :-

      i_am_holding(X),
```

```
        write('You"re already holding it!'),
        nl.
take(X) :-
        i_am_at(Place),
        at(X, Place),
        retract(at(X, Place)),
        assert(i_am_holding(X)),
        write('OK.'),
        nl.
take(_) :-
        write('I don"t see it here.'),
        nl.
/* These rules describe how to put down an
object. */
drop(X) :-
        i_am_holding(X),
        i_am_at(Place),
        retract(i_am_holding(X)),
        assert(at(X, Place)),
        write('OK.'),
        nl.
drop(_) :-
        write('You aren"t holding it!'),
        nl.
/* These rules define the six direction letters as
calls to go/1. */
n :- go(n).
s :- go(s).
e :- go(e).
w :- go(w).
u :- go(u).
d :- go(d).
```

7. File 7

(erabaru.or.id) Pada jaman dahulu, seorang nenek tua menganut kepercayaan pada Dewi Kwan-im, ia selalu membaca kitab suci dan menyembahyangi Dewi Kwan-im. Suatu ketika ia pergi ke gunung Ku Tuo, daerah Zhejiang, memberi penghormatan kepada Dewi Kwan-im. Ketika si nenek tua berjalan ke Qian Busha untuk berziarah, dihadapannya datang seorang wanita tua yang sengaja menghadang. Karena terhalang, nenek tua itu menghindar, tapi selalu dihadang, sehingga membuat keduanya berdiri berhadapan, dan saat itu, si nenek yang akan berziarah ke gunung itu betanya, “Saya datang untuk memberi penghormatan kepada Dewi Kwan-im, kamu akan berdosa jika menghalangi jalan saya.”

Si wanita tua yang menghadang jalan itu berkata, “Tiap hari merindukan saya, memuja saya, dan bertemu dengan saya lalu tidak kenal saya lagi.” Dalam hati si nenek tua itu berpikir: “Kamu perempuan tua yang sama sepertiku, saya memang rindu, dan memuja Dewi Kwan-im, mengapa harus merindukan dan memujamu?” Begitu ia merenung, lantas tidak tahu ke mana perginya si wanita tua yang menghadangnya itu, bayangannya sudah lenyap dan tak kelihatan lagi. Ia lalu mengingat kembali kata-kata itu: “Tiap hari merindukan saya, memuja saya, dan bertemu dengan saya lalu tidak kenal lagi, ya Tuhan, tadi adalah penjelmaan Dewi Kwan-im, melihat tapi tak kenal.” Saat itu, ia berlutut dan menangis sedih, melewatkan kesempatan dan nasib, bertemu tapi tidak berjodoh.

Sebagian besar orang di dunia sekarang ini seperti anak kecil yang kebingungan, tersesat, mengembara ke mana-mana, menangis dan berteriak mencari sang ibu. Buddha utama persis bagaikan ibunda yang bijak, demi anak-anaknya yang hilang entah kemana, sang ibu terus mencari ke mana-mama, dari tempat yang

sangat… sangat jauh, mendaki gunung dan menyeberangi sungai, mengalami berbagai macam kesulitan dan penderitaan mencari anak kandung sendiri. Namun ketika bertemu, anak-anaknya tidak ingat lagi secara jelas senyuman kasih sang ibu, juga tidak kenal lagi siapa sang ibu itu. Anak-anak itu lantas mendengar dan percaya dengan hasutan kejahatan, menganggap ibunda yang paling bijak sebagai orang jahat yang paling mengerikan di dunia, juga bersekongkol dengan kejahatan menganggapnya sebagai musuh, menodai dan memfitnah ibunda yang belas kasih.

Setiap hari penganut Buddha memohon penyelamatan, berharap Buddha menetap di dunia, tidak tahu sudah berapa kehidupan memohon, berharap Maitreya turun ke dunia manusia menyelamatkan mereka. Kini, "Buddha Utama" yang sesungguhnya turun ke dunia, namun kita malah ditipu oleh dusta-dusta jahat, juga secara besar-besaran mengikuti iblis memfitnah dan mencemari, oh…. betapa cerobohnya! Kalau begitu, bagaimana anak-anak itu bisa kembali kepangkuan ibunya.

(Sumber: Dajiyuan)

8. File 8

hhthththrhrthtrhrthbmnmnxcb,nzxbv,mvksjewE:g'b.df'bSFFJHSDHSGFKDJKJhh%%%%%%%%%%&&&&&&^$$%$%^$^$%^%^^546yzczczcbcxnvcmvmadahfsjdkglkgl[pu[yutyrreq21243547665676099

9. File 9

Departemen Teknik Informatika

Institut Teknologi Bandung

Tugas I IF2251 Strategi Algoritmik

Kompresi dan Dekompresi File dengan

Kode Huffman (Aplikasi Algoritma Greedy)

Batas pengumpulan : 12 Maret 2004

Tempat pengumpulan : - Laporan : Loker tujuh Lab IRK

- Source dan Exe : - email if11008@students.if.itb.ac.id

- email if11048@students.if.itb.ac.id

Arsip pengumpulan : - Source dan Exe program disertai readme.txt

- Laporan

Deskripsi tugas :

Dalam komunikasi data, pesan yang dikirim seringkali ukurannya sangat besar sehingga waktu pengirimannya lama. Begitu juga dengan penyimpanan data, arsip yang berukuran besar membutuhkan ruang penyimpanan yang besar. Kedua masalah ini dapat diatasi dengan mengkodekan pesan atau isi arsip sesingkat mungkin, sehingga waktu pengiriman pesan relatif cepat dan ruang penyimpanan yang dibutuhkan juga sedikit. Cara pengkodean seperti ini disebut kompresi (pemampatan) data dan pemulihan data tersebut kembali seperti aslinya disebut dekompresi. Salah satu cara kompresi dan dekompresi data ini adalah dengan menggunakan kode Huffman.

Pada tugas pertama Strategi Algoritmik ini, anda diminta mengimplementasikan kode Huffman dalam kompresi dan dekompresi data. Kode Huffman dibangkitkan dengan algoritma Huffman. Algoritma Huffman menggunakan prinsip greedy dalam pembentukan kode Huffman. Anda harus dapat mengompresi semua jenis data baik berupa teks, gambar, suara, dan video dan Anda harus mampu mengembalikan data yang sudah dikompres tersebut ke bentuk asalnya (dekompresi). Sebagai contoh, jika executable file dimampatkan (misalnya notepad.exe), maka program notepade.exe tersebut harus dapat dijalankan kembali

Program yang Anda buat selain mampu mengompresi dan mendekompresi data harus dapat menunjukkan perubahan hasil kompresi tersebut melalui rasio

pemampatannya. Rasio pemampatan dhitung dengan rumus:

P = Ukuran file hasil pemampatan/Ukuran file sebelum dimampatkan ? 100%

Yang berarti ukuran file menjadi P (dalam persentase) dari ukuran semula.

Spesifikasi program :

1. Program mampu mengkompresi data berjenis apapun secara tepat dan benar.

2. Program mampu mendekompresi data yang sebelumnya telah dikompres dengan tepat dan benar.

3. Program mampu memberikan rasio kompresi data.

4. Program juga harus mampu memberikan waktu proses kompresi dan dekompresi data.

5. Program harus mampu menampilkan proses kompresi dan dekompresi tersebut (misal dalam progress bar).

Lain – lain :

1. Anda dapat menambahkan feature-feature lain yang menunjang program yang anda buat (unsur kreatifitas).

2. Anda harus membuat file contoh yang belum dikompres dan file yang sudah dikompres.

3. Program ini harus Anda buat dalam format GUI, boleh menggunakan Visual C, Borland C++ Bulider, Delphi, atau VB, namun tidak dianjurkan Java (karena beda orientasi).

4. Tugas dikerjakan per kelompok dengan jumlah anggota tiga orang.

5. Program harus modular dan mengandung komentar yang jelas.

6. Pengumpulan adalah tanggal 12 Maret 2004. Untuk laporan dapat dimasukkan ke dalam loker nomor tujuh (7) Lab IRK sebelum pukul 17.00 WIB 12 Maret 2004. Sedangkan source dan executable program, dapat dikirim melalui email ke asisten hingga pukul 23.59.59 12 Maret 2004 waktu semeru. Keterlambatan akan mengurangi nilai.

7. Source, exe, dan readme.txt disimpan dalam folder StrAlgo1-xxxxx. Lima digit terakhir adalah NIM anggota terkecil. Hal ini berlaku juga sebagai subjek pengiriman. Readme.txt berisi anggota kelompok, NIM, dan keterangan lain yang dianggap perlu.

8. Semua pertanyaan menyangkut tugas ini harus dikomunikasikan melalui milis agar dapat dicermati oleh semua peserta kuliah IF2251.

9. Demo program akan dilaksanakan tanggal 13 Maret yaitu hari Sabtu kecuali ada pemberitahuan lebih lanjut dari asisten.

10. Urutan demo adalah First Come First Serve dari pengumpulan program.

11. Tiap anggota harus memahami proses pembuatan program, karena akan ada pertanyaan-pertanyaan yang harus dijawab per individu.

12. Pada saat demo, asisten akan memanggil per kelompok. Kelompok yang tidak berkepentingan dilarang masuk. Demo dilakukan di Lab IRK tepat pukul 09.00.

Isi laporan :

1. Deskripsi masalah dan dasar teori (maksimum satu halaman).

2. Strategi penyelesaian masalah.

3. Struktur data dan spesifikasi program.

4. Analisis hasil (hasil uji terhadap beberapa jenis file dari berbagai ukuran, baik dari segi rasio kompresi, waktu kompresi, dan waktu dekompresi).

5. Kesimpulan dan saran.

6. Referensi.

Keterangan laporan :

1. Laporan ditulis dalam bahasa Indonesia yang baik dan benar, tidak perlu panjang tetapi tepat sasaran dan jelas.

2. Laporan tidak perlu memakai cover mika dan dijilid. Cukup dibuat agar laporan tidak akan tercecer bila dibaca.

3. Laporan boleh menggunakan kertas rius, boleh bolak-balik, boleh dalam satu halaman kertas terdapat dua halaman tulisan asalkan masih terbaca.

4. Identitas per halaman harus jelas (misalnya : halaman, kode kuliah).

Penilaian :

- **Bagaimana cara kerja kompresi dan dekompresi citra?**
- **Bagaimana mengimplementasi kompresi pada citra?**
- **Bagaimana mengimplementasi dekompresi pada citra?**

III. Perumusan Masalah

Perumusan masalah yang dibahas adalah kompresi dan dekompresi pada citra dengan menggunakan algoritma Elias Gamma Coding.

IV. Tujuan

Dalam tugas akhir ini, penyusun akan mengimplementasikan kompresi citra menggunakan algoritma Elias Gamma Coding kemudian melakukan dekompresi pada citra serta mencari rasio algoritma tersebut dalam pengkompresian data.

V. Pembatasan Masalah

1. Implementasi kompresi dalam bentuk program

2. Jenis kompresi adalah lossy compression

3. Media yang dikompresi berupa sebuah image

4. Kompresi dilakukan secara offline

5. Algoritma yang digunakan adalah algoritma elias gamma coding

6. Melakukan dekompresi

VI. Spesifikasi Program

Program untuk implementasi kompesi data menggunakan algoritma Elias Gamma Coding dapat bermacam- macam seperti: C++, Microsoft Visual Basic, Matlab, Delphi. Saat ini penyusun sedang mengkaji program yang akan digunakan.

VII. Metodologi

1. Mempelajari referensi tentang kompresi data khususnya dengan algoritma Elias Gamma Coding

2. Perancangan program

3. Pembuatan program

4. Pengujian program

5. Analisis data

6. Pembuatan laporan

VIII. Diagram blok

IX. Daftar Pustaka

Bell, Timothy C., Alistair Moffat & Ian H. Witten. Departments of Computer Science,University of Calgary, Canada, and University of Waikato, New Zealand, ian@cpsc.ucalgary.ca.

Gonzales, Rafael C.1977. Digital Image Processing. Addison- Wesley Publishing Company.

Nelson, Mark. 1996. The Data Compression Book 2nd-ed. BPB Publications.

Jain, Anil K. 1989. Fundamentals of Digital Image Processing. Prentice-Hall.

Darmawan, Aan. Maret 2007. Diktat Kuliah Pengolahan Citra Dijital.

Memecahkan kode. Golomb mengkode dirancang di dalam cara khusus ini untuk memudahkan mereka pengawasandian. Kita pertama menunjukkan pengawasandian untuk seribu kasus yang sederhana = 16 (seribu adalah suatu daya dari 2). Untuk memecahkan kode, mulai pada yang ditinggalkan akhir dari kode dan menghitung nomor A dari 1's terdahulu terlebih dulu 0.Panjang kode itu adalah A + c +1 pahat (untuk seribu =16, ini adalah A +5 pahat). Jika kita menandakan rightmost lima pahat dari kode oleh R, lalu nilai dari kode itu adalah 16A + R.Pengawasandian sederhana ini mencerminkan cara kode dibangun. Untuk menyandi n dengan seribu =16, mulai dengan itu pembagian oleh 16 untuk mendapat n = 16A +R, lalu menulis A 1's diikuti oleh suatu kosong, yang diikuti oleh matriks S 4-bit R.

Menyandi. Golomb mengkode karena bilangan bulat taknegatif n bergantung pada pilihan dari suatu parameter m (we akan melihat kemudiannya bahwa untuk RLE, seribu perlu bergantung pada kemungkinan p dan di angka median dari menjalankan panjangnya-panjangnya). Jadi; Dengan demikian, [ini] merupakan suatu kode awalan yang parametrized, yang buat nya terutama bermanfaat jika di mana nilai-nilai yang baik untuk parameter itu dapat dihitung atau diperkirakan (lihat, sebagai contoh, Bagian 422). Pertama masuk menghitung Golomb

kode dari bilangan bulat taknegatif n untuk menghitung ke tiga jumlah q (kuosien),

r (-sisa), dan c oleh

q =n m, r= n -qm, dan c =log2m,

berikut yang mana kode dibangun dalam dua suku cadang; pertama adalah nilai dari q, coded di dalam unary (Berlatih 23), dan yang kedua adalah nilai yang biner r coded di suatu cara yang khusus.2c pertama -nilai-nilai seribu dari r bersifat coded, sebagai integer-integer yang tidak ditandatangani, di c -1 pahat masing-masing dan sisanya bersifat coded di c menggigit masing-masing (berakhir dengan c yang paling besar menggigit nomor, yang terdiri dari c 1's). Kasus di mana seribu adalah suatu daya dari 2 (m =2c) khusus karena itu memerlukan tidak (c -1)-bit kode. Kita mengetahui bahwa n = r +qm; maka sekali se kode Golomb dikodekan, nilai-nilai dari q dan r dapat digunakan untuk dengan mudah merekonstruksi n.

Ada juga kasus umum di mana p tidak dikenal dan tidak bisa dihitung (atau bahkan diperkirakan) terlebih dahulu, karena dawai itu adalah sangat panjang atau karena itu harus dimampatkan di dalam waktu nyata selagi itu tiba dari sumber nya. Dalam kasus yang demikian, satu algoritma yang adaptip itu bervariasi seribu menurut input-so-far itu adalah pilihan terbaik. Algoritma seperti itu, yang disebut Goladap [Langdon 83a], digambarkan di sini.

Mereka membuktikan bahwa kode Golomb adalah awalan terbaik mengkode ketika seribu terpilih oleh mereka ketidaksamaan. Kita pertanda pertama bahwa karena suatu yang diberi p, ketidaksamaan (23) hanya mempunyai satu solusi m. Kita mengolah ketidaksamaan ini dalam empat langkah-langkah sebagai berikut:

Tiga contoh diperkenalkan di sini untuk menggambarkan penampilan dari Golomb mengkode

di dalam memampatkan menjalankan panjangnya-panjangnya. Contoh yang pertama adalah dawai yang biner (21), yang mempunyai 41

nol dan 18 mereka. Kemungkinan suatu kosong kemudian 41/(41 + 18) ˜07, hasil?penyerahan

m = -bukukan 17/ batang kayu 07 = 1487 = 2.Urutan dari menjalankan panjangnya-panjangnya 5, 2, 0, 3, 1, 6, 0, 0,

1, 3, 5, 3, 2, 3, 0, 1, 4, dan 2 kaleng oleh karena itu disandikan dengan Golomb mengkode untuk seribu =2

ke dalam dawai dari 18 mengkode

Banyak metoda kompresi didasarkan pada menjalankan pengkodean panjangnya (RLE). Bayangkan a dawai biner di mana suatu kosong muncul dengan kemungkinan p dan yang muncul dengan kemungkinan 1-p. Jika p besar, akan ada berlari tentang kosong-kosong, mengusulkan pemakaian RLE untuk memampatkan dawai. Kemungkinan suatu lari dari n kosong-kosong adalah pn dan kemungkinan suatu lari dari n nol yang diikuti oleh suatu 1 adalah pn(1 -p), menunjukkan bahwa menjalankan panjangnya-panjangnya dibagi-bagikan secara geometris.

Suatu pendekatan yang naif kepada memampatkan dawai seperti itu untuk menghitung kemungkinan dari tiap menjalankan panjangnya dan menerapkan metoda Huffman (Bagian 28) untuk memperoleh awalan terbaik mengkode karena menjalankan panjangnya-panjangnya. Dalam praktek, bagaimanapun, mungkin ada sejumlah besar menjalankan panjangnya-panjangnya

dan nomor ini tidak akan dikenal terlebih dahulu. Suatu pendekatan yang lebih baik untuk membangun satu keluarga yang tanpa batas dari awalan optimal mengkode, seperti bahwa tak peduli bagaimana merindukan suatu lari adalah, di sana akan merupakan suatu kode di dalam keluarga itu untuk menyandi nya. Mengkode di dalam keluarga itu harus bergantung pada kemungkinan p, jadi kita sedang mencari satu himpunan takhingga dari awalan yang parametrized mengkode. Golomb mengkode digambarkan di sini [Golomb 66] adalah seperti itu kode dan mereka adalah yang terbaik untuk kompresi data item yang dibagi-bagikan secara geometris.

COMPRESSING LANGUAGE MODELS WITH GOLOMB CODING

BACKGROUND The discussion below is merely provided for general background information and is not intended to be used as an aid in determining the scope of the claimed subject matter.

Language models are used in a variety of applications including noisy channel applications such as natural language processing, spell checking, and the like. In natural language applications, a speech recognizer typically works by combining acoustic evidence (channel model) with expectations about what the user is likely to say (language model) . One common form of language models is referred to as a tri-gram.

In general, a n-gram is a subsequence of n tokens

(words) . A tri-gram is a subsequence of 3 tokens. For example, from the phrase "to be or not to be", 8 tri-grams can be generated: "$ $ to", "$ to be", "to be or", "be or not", "or not to", "not to be," "to be $" and "be $ $," where the input string is padded with two special tokens denoted at: "$." Statistics can be applied to such n-grams to estimate a likelihood that a user intended a particular input.

Though a billion words of text used to be considered large, training sets for speech recognition routinely train on ten billion words of text. In general, large language models work well (meaning they have low entropy) ; however, memory capacity is often limited, especially in mobile devices such as cell phones, personal digital assistants (PDAs), electronic planners, and the like. One technique for addressing the memory situation involves trimming the language model, by removing infrequently used words and uncommon variants. However, removal of such terms reduces the overall effectiveness of the language model, leading to more semantic errors due to inability to match input to words in the trimmed model .

SUMMARY

This summary is provided to introduce in a simplified form some concepts, which are described below in the Detailed Description. This Summary is not intended to identify key features or essential features of the claimed subject matter, nor is it intended to be used as an aid in determining the scope of the claimed subject matter.

In one embodiment, a language model is compressed using Golomb encoding techniques . A list of values is generated from elements of

the language model. The list of integer values is sorted, and for each element, a difference between adjacent integer values in the list is calculated. Each calculated difference is encoded using a Golomb code.

In another embodiment, a system for processing user inputs has a user interface, a memory, a Golomb encoder/decoder, and a processor. The user interface is adapted to receive user inputs . The memory is adapted to store information and to store a Golomb compressed language model. The Golomb encoder/decoder is adapted to encode user input and to decode elements of the Golomb compressed language model. The processor is adapted to compare encoded user input against elements of the Golomb compressed language model to identify probable matches .

In another embodiment, a method of decoding user inputs using a Golomb-encoded language model is provided. A user input is divided into a plurality of- elements, each of which is encoded using a hash technique. Each encoded element is compared to elements of a Golomb-encoded language model to identify possible matches.

- Algoritma Boruvka : finds a minimum spanning tree for a graph

- Algoritma Ford-Fulkerson : computes the maximum flow problem in a graph

- Algoritma Edmonds-Karp : implementation of Ford-Fulkerson

- Nonblocking Minimal Spanning Switch say, for a telephone exchange

- Spring based algorithm : algorithm for graph drawing

- Topological sorting

- Algoritma Hungaria : algorithm for finding a perfect matching

Algoritma pencarian

- Pencarian linear : mencari sebuah item pada sebuah list tak berurut

- Algoritma seleksi : mencari item ke-k pada sebuah list

- Pencarian biner : menemukan sebuah item pada sebuah list terurut

- Pohon Pencarian Biner

- Pencarian Breadth-first : menelusuri sebuah graf tingkatan demi tingkatan

- Pencarian Depth-first : menelusuri sebuah graf cabang demi cabang

- Pencarian Best-first : menelusuri sebuah graf dengan urutan sesuai kepentingan dengan menggunakan antrian prioritas

- Algoritma Pencarian A Bintang : kasus khusus dari pencarian best-first

- Pencarian Interpolasi : pencarian mirip biner dengan faktor pada magnitudo (matematika) dari syarat pencarian terhadap nilai atas dan bawah dalam pencarian. Kadang-kadang disebut pencarian kamus atau pencarian interpolasi.

- Tabel Hash : mencari sebuah item dalam sebuah kumpulan tak berurut dalam waktu O(1).

String algorithms

String searching algorithm

- Algoritma Aho-Corasick

- Algoritma Bitap

- Boyer-Moore string search algorithm

- Knuth-Morris-Pratt algorithm

- Rabin-Karp string search algorithm

Approximate matching

- Levenshtein_distance

Algoritma penyusunan

- Binary search tree

- Bogosort

- Bubble sort : for each pair of indices, swap the items if out of order

- Bucket sort

- Comb sort

- Cocktail sort

- Counting sort

- Gnome sort

- Heapsort : convert the list into a heap, keep removing the largest element from the heap and adding it to the end of the list

- Insertion sort : determine where the current item belongs in the list of sorted ones, and insert it there

- Merge sort : sort the first and second half of the list separately, then merge the sorted lists

- Pancake sorting

- Pigeonhole sort

- Quicksort : divide list into two, with all items on the first list coming before all items on the second list.; then sort the two lists. Often the method of choice

- Radix sort : sorts strings letter by letter

- Selection sort : pick the smallest of the remaining elements, add it to the end of the sorted list

- Shell sort : an attempt to improve insertion sort

- Smoothsort

- Stupid sort

- Topological sorting

Data compression

:Category:Lossless compression algorithms

- Burrows-Wheeler transform : preprocessing useful for improving lossless compression

- DEFLATE (algorithm) : lossless data compression

- Delta encoding : aid to compression of data in which sequential data occurs frequently

- Incremental encoding : delta encoding applied to sequences of strings

- LZW : lossless data compression (Lempel-Ziv-Welch)

- LZ77 (algorithm) : LZ77 and LZ78 are the names for the two lossless data compression algorithms

- LZMA : short for Lempel-Ziv-Markov chain-Algorithm

- LZO : data compression algorithm that is focused on speed

- PPM compression algorithm

- Shannon-Fano coding

- Truncated binary encoding

- Run-length encoding : lossless data compression taking advantage of strings of repeated characters

- SEQUITUR algorithm : lossless compression by incremental grammar inference on a string

- Embedded Zerotree Wavelet

- Entropy encoding : coding scheme that assigns codes to symbols so as to match code lengths with the probabilities of the symbols

- Huffman coding : simple lossless compression taking advantage of relative character frequencies

- Adaptive Huffman coding : adaptive coding technique based on Huffman coding

- Arithmetic coding : advanced entropy coding

- Range encoding : data compression method that is believed to approach the compression ratio of arithmetic coding

- Entropy encoding

- Unary coding : code that represents a number n with n ones followed by a zero

- Elias Elias delta coding | Elias gamma coding | Elias omega coding coding: universal code encoding the positive integers

- Fibonacci coding : universal code which encodes positive integers into binary code words

- Golomb coding : form of entropy coding that is optimal for alphabets following geometric distributions

- Rice coding : form of entropy coding that is optimal for alphabets following geometric distributions

:Category:Lossy compression algorithms

- Linear predictive coding : lossy compression by representing the spectral envelope of a digital signal of speech in compressed form

- A-law algorithm : standard companding algorithm

- Mu-law algorithm : standard analog signal compression or companding algorithm

- Fractal compression : method used to compress images using fractals

- Transform coding : type of data compression for "natural" data like audio signals or photographic images

- Vector quantization : technique often used in lossy data compression

- Wavelet compression : form of data compression well suited for image compression (sometimes also video compression and audio compression)

Computational geometry

- Gift wrapping algorithm : determining the convex hull of a set of points

- Graham scan determining the convex hull of a set of points in the plane

- Point in polygon : tests whether a given point lies within a given polygon

Grafik komputer

- Bresenham's line algorithm : plots points of a 2-dimensional array to form a straight line between 2 specified points (uses decision variables)

- DDA line algorithm : plots points of a 2-dimensional array to form a straight line between 2 specified points (uses floating-point math)

- Flood fill : fills a connected region of a multi-dimensional array with a specified symbol

- Painter's algorithm : detects visible parts of a 3-dimensional scenery

- Ray tracing : realistic image rendering (computer graphics)

Algoritma Kriptografi

Lihat juga Topik dalam kriptografi

- symmetric key algorithm :

- Advanced Encryption Standard (AES), winner of NIST competition

- Blowfish (cipher)

- Data Encryption Standard (DES), sometimes DE Algorithm, winner of NBS selection competition, replaced by AES for most purposes

- International Data Encryption Algorithm

- RC4 (cipher)

- Asymmetric key algorithm or digital signature :

- Digital Signature Algorithm

- ElGamal encryption

- RSA

- Diffie-Hellman key exchange

- NTRUEncrypt

- Cryptographic Message digest functions:

- MD5 – Note that there is now a method of generating collisions for MD5

- RIPEMD-160

- SHA-1

- keyed-hash message authentication code : keyed-hash message authentication

- Cryptographically secure pseudo-random number generator s

- Blum Blum Shub - based on the hardness of integer factorization

- Yarrow algorithm

- Fortuna (PRNG) , allegedly an improvement on Yarrow

- Other

- Diffie-Hellman : key exchange

Algoritma Distributed systems

- File 10

Departemen Teknik Informatika

Institut Teknologi Bandung

Tugas I IF2251 Strategi Algoritmik

Kompresi dan Dekompresi File dengan

Kode Huffman (Aplikasi Algoritma Greedy)

Batas pengumpulan : 12 Maret 2004

Tempat pengumpulan : - Laporan : Loker tujuh Lab IRK

- Source dan Exe : - email if11008@students.if.itb.ac.id

- email if11048@students.if.itb.ac.id

Arsip pengumpulan : - Source dan Exe program disertai readme.txt

- Laporan

Deskripsi tugas :

Dalam komunikasi data, pesan yang dikirim seringkali ukurannya sangat besar sehingga waktu pengirimannya lama. Begitu juga dengan penyimpanan data, arsip yang berukuran besar membutuhkan ruang penyimpanan yang besar. Kedua masalah ini dapat diatasi dengan mengkodekan pesan atau isi arsip sesingkat mungkin, sehingga waktu pengiriman pesan relatif cepat dan ruang penyimpanan yang dibutuhkan juga sedikit. Cara pengkodean seperti ini disebut kompresi (pemampatan) data dan pemulihan data tersebut kembali seperti aslinya disebut dekompresi. Salah satu cara kompresi dan dekompresi data ini adalah dengan menggunakan kode Huffman.

Pada tugas pertama Strategi Algoritmik ini, anda diminta mengimplementasikan kode Huffman dalam kompresi dan dekompresi data. Kode Huffman dibangkitkan dengan algoritma Huffman. Algoritma Huffman menggunakan prinsip greedy dalam pembentukan kode Huffman. Anda harus dapat mengompresi semua jenis data baik berupa teks, gambar, suara, dan video dan Anda harus mampu mengembalikan data yang sudah dikompres tersebut ke bentuk asalnya (dekompresi). Sebagai contoh, jika executable file dimampatkan (misalnya notepad.exe), maka program notepade.exe tersebut harus dapat dijalankan kembali

Program yang Anda buat selain mampu mengompresi dan mendekompresi data harus dapat menunjukkan perubahan hasil kompresi tersebut melalui rasio pemampatannya. Rasio pemampatan dhitung dengan rumus:

P = Ukuran file hasil pemampatan/Ukuran file sebelum dimampatkan ? 100%

Yang berarti ukuran file menjadi P (dalam persentase) dari ukuran semula.

Spesifikasi program :

1. Program mampu mengkompresi data berjenis apapun secara tepat dan benar.

2. Program mampu mendekompresi data yang sebelumnya telah dikompres dengan tepat dan benar.

3. Program mampu memberikan rasio kompresi data.

4. Program juga harus mampu memberikan waktu proses kompresi dan dekompresi data.

5. Program harus mampu menampilkan proses kompresi dan dekompresi tersebut (misal dalam progress bar).

Lain – lain :

1. Anda dapat menambahkan feature-feature lain yang menunjang program yang anda buat (unsur kreatifitas).

2. Anda harus membuat file contoh yang belum dikompres dan file yang sudah dikompres.

3. Program ini harus Anda buat dalam format GUI, boleh menggunakan Visual C, Borland C++ Bulider, Delphi, atau VB, namun tidak dianjurkan Java (karena beda orientasi).

4. Tugas dikerjakan per kelompok dengan jumlah anggota tiga orang.

5. Program harus modular dan mengandung komentar yang jelas.

6. Pengumpulan adalah tanggal 12 Maret 2004. Untuk laporan dapat dimasukkan ke dalam loker nomor tujuh (7) Lab IRK sebelum pukul 17.00 WIB 12 Maret 2004. Sedangkan source dan executable program, dapat dikirim melalui email ke asisten hingga pukul 23.59.59 12 Maret 2004 waktu semeru. Keterlambatan akan mengurangi nilai.

7. Source, exe, dan readme.txt disimpan dalam folder StrAlgo1-xxxxx. Lima digit terakhir adalah NIM anggota terkecil. Hal ini berlaku juga sebagai subjek pengiriman. Readme.txt berisi anggota kelompok, NIM, dan keterangan lain yang dianggap perlu.

8. Semua pertanyaan menyangkut tugas ini harus dikomunikasikan melalui milis agar dapat dicermati oleh semua peserta kuliah IF2251.

9. Demo program akan dilaksanakan tanggal 13 Maret yaitu hari Sabtu kecuali ada pemberitahuan lebih lanjut dari asisten.

10. Urutan demo adalah First Come First Serve dari pengumpulan program.

11. Tiap anggota harus memahami proses pembuatan program, karena akan ada pertanyaan-pertanyaan yang harus dijawab per individu.

12. Pada saat demo, asisten akan memanggil per kelompok. Kelompok yang tidak berkepentingan dilarang masuk. Demo dilakukan di Lab IRK tepat pukul 09.00.

Isi laporan :

1. Deskripsi masalah dan dasar teori (maksimum satu halaman).

2. Strategi penyelesaian masalah.

3. Struktur data dan spesifikasi program.

4. Analisis hasil (hasil uji terhadap beberapa jenis file dari berbagai ukuran, baik dari segi rasio kompresi, waktu kompresi, dan waktu dekompresi).

5. Kesimpulan dan saran.

6. Referensi.

Keterangan laporan :

1. Laporan ditulis dalam bahasa Indonesia yang baik dan benar, tidak perlu panjang tetapi tepat sasaran dan jelas.

2. Laporan tidak perlu memakai cover mika dan dijilid. Cukup dibuat agar laporan tidak akan tercecer bila dibaca.

3. Laporan boleh menggunakan kertas rius, boleh bolak-balik, boleh dalam satu halaman kertas terdapat dua halaman tulisan asalkan masih terbaca.

4. Identitas per halaman harus jelas (misalnya : halaman, kode kuliah).

Penilaian :

1. Kebenaran program (50%) : program mampu memroses data yang sudah disediakan dan data dari asisten.

2. Demo – pemahaman Anda dalam pembuatan program (30%)

3. Laporan (20%)

4. Interface, feature-feature program, dan unsur kreativitas (10%)

-selamat mengerjakan-

PROPOSAL PENGAJUAN TUGAS AKHIR

Kompresi Data dengan Algoritma Elias Gamma Coding

Pembimbing:

I. Latar belakang

Seperti yang kita ketahui semakin tinggi resolusi suatu citra maka ukuran data yang dihasilkan semakin besar. Ukuran data yang besar membutuhkan tempat penyimpanan (memori) yang besar pula. Hal ini dapat diatasi dengan program kompresi data agar dapat menghemat memori penyimpanan.

Kompresi adalah proses pengubahan sekumpulan data menjadi bentuk kode dengan tujuan untuk menghemat kebutuhan tempat penyimpanan. Ukuran kompresi yang baik adalah yang dapat menghasilkan rasio besar.

Algoritma yang akan dibahas dalam tugas akhir ini adalah algoritma Elias Gamma Coding. Algoritma ini termasuk jenis algoritma lossy compression yang berarti media yang dikompresi berupa image. Algoritma ini tergolong baru berkembang dalam jangka waktu 5 tahun terakhir maka penyusun mencoba mengimplementasi program kompresi data

kode dari bilangan bulat taknegatif n untuk menghitung ke tiga jumlah q (kuosien),

r (-sisa), dan c oleh

q =n m, r= n -qm, dan c =log2m,

berikut yang mana kode dibangun dalam dua suku cadang; pertama adalah nilai dari q, coded di dalam unary (Berlatih 23), dan yang kedua adalah nilai yang biner r coded di suatu cara yang khusus.2c pertama -nilai-nilai seribu dari r bersifat coded, sebagai integer-integer yang tidak ditandatangani, di c -1 pahat masing-masing dan sisanya bersifat coded di c menggigit masing-masing (berakhir dengan c yang paling besar menggigit nomor, yang terdiri dari c 1's). Kasus di mana seribu adalah suatu daya dari 2 (m =2c) khusus karena itu memerlukan tidak (c -1)-bit kode. Kita mengetahui bahwa n = r +qm; maka sekali se kode Golomb dikodekan, nilai-nilai dari q dan r dapat digunakan untuk dengan mudah merekonstruksi n.

Ada juga kasus umum di mana p tidak dikenal dan tidak bisa dihitung (atau bahkan diperkirakan) terlebih dahulu, karena dawai itu adalah sangat panjang atau karena itu harus dimampatkan di dalam waktu nyata selagi itu tiba dari sumber nya. Dalam kasus yang demikian, satu algoritma yang adaptip itu bervariasi seribu menurut input-so-far itu adalah pilihan terbaik. Algoritma seperti itu, yang

disebut Goladap [Langdon 83a], digambarkan di sini.

Mereka memb

ktikan bahwa kode Golomb adalah awalan terbaik mengkode ketika seribu terpilih oleh mereka ketidaksamaan. Kita pertanda pertama bahwa karena suatu yang diberi p, ketidaksamaan (23) hanya mempunyai satu solusi m. Kita mengolah ketidaksamaan ini dalam empat langkah-langkah sebagai berikut:

Tiga contoh diperkenalkan di sini untuk menggambarkan penampilan dari Golomb mengkode

di dalam memampatkan menjalankan panjangnya-panjangnya. Contoh yang pertama adalah dawai yang biner (21), yang mempunyai 41

nol dan 18 mereka. Kemungkinan suatu kosong kemudian 41/(41 + 18) ˜07, hasil?penyerahan

m = -bukukan 17/ batang kayu 07 = 1487 = 2.Urutan dari menjalankan panjangnya-panjangnya 5, 2 0, 3, 1, 6, 0, 0,

1, 3, 5, 3, 2, 3, 0, 1, 4, dan 2 kaleng oleh karena itu disandikan dengan Golomb mengkode untuk seribu =2

ke dalam dawai dari 18 mengkode

Banyak metoda kompresi didasarkan pada menjalankan pengkodean panjangnya (RLE). Bayangkan a dawai biner di mana suatu kosong muncul dengan kemungkinan p dan yang muncul dengan kemungkinan 1-p. Jika p besar, akan ada berlari tentang kosong-kosong, mengusulkan pemakaian RLE untuk memampatkan dawai. Kemungkinan suatu lari dari n kosong-kosong adalah pn dan kemungkinan suatu lari dari n nol yang diikuti oleh suatu 1 adalah pn(1 -p), menunjukkan bahwa menjalankan panjangnya-panjangnya dibagi-bagikan secara geometris.

Suatu pendekatan yang naif kepada memampatkan dawai seperti itu untuk menghitung kemungkinan dari tiap menjalankan panjangnya dan menerapkan metoda Huffman (Bagian 28) untuk memperoleh awalan terbaik mengkode karena menjalankan panjangnya-panjangnya. Dalam praktek, bagaimanapun, mungkin ada sejumlah besar menjalankan panjangnya-panjangnya

dan nomor ini tidak akan dikenal terlebih dahulu. Suatu pendekatan yang lebih baik untuk membangun satu keluarga yang tanpa batas dari awalan optimal mengkode, seperti bahwa tak peduli bagaimana merindukan suatu lari adalah, di sana akan merupakan suatu kode di dalam keluarga itu untuk menyandi nya. Mengkode di dalam keluarga itu harus bergantung pada kemungkinan p, jadi kita sedang mencari satu himpunan takhingga dari awalan yang parametrized mengkode. Golomb mengkode digambarkan di sini [Golomb 66] adalah seperti itu kode dan mereka adalah yang terbaik untuk kompresi data item yang dibagi-bagikan secara geometris.

COMPRESSING LANGUAGE MODELS WITH GOLOMB CODING

BACKGROUND The discussion below is merely provided for general background information and is not intended to be used as an aid in determining the scope of the claimed subject matter.

Language models are used in a variety of applications including noisy channel applications such as natural language processing, spell checking, and the like. In natural language applications, a speech recognizer typically works by combining acoustic evidence (channel model) with expectations about what the user is likely to say (language model) . One common form of language models is referred to as a tri-gram.

In general, a n-gram is a subsequence of n tokens

(words) . A tri-gram is a subsequence of 3 tokens. For example, from the phrase "to be or not to be", 8 tri-grams can be generated: "$ $ to", "$ to be", "to be or", "be or not", "or not to",

"not to be," "to be $" and "be $ $," where the input string is padded with two special tokens denoted at: "$." Statistics can be applied to such n-grams to estimate a likelihood that a user intended a particular input.

Though a billion words of text used to be considered large, training sets for speech recognition routinely train on ten billion words of text. In general, large language models work well (meaning they have low entropy) ; however, memory capacity is often limited, especially in mobile devices such as cell phones, personal digital assistants (PDAs), electronic planners, and the like. One technique for addressing the memory situation involves trimming the language model, by removing infrequently used words and uncommon variants. However, removal of such terms reduces the overall effectiveness of the language model, leading to more semantic errors due to inability to match input to words in the trimmed model .

SUMMARY

This summary is provided to introduce in a simplified form some concepts, which are described below in the Detailed Description. This Summary is not intended to identify key features or essential features of the claimed subject matter, nor is it intended to be used as an aid in determining the scope of the claimed subject matter.

In one embodiment, a language model is compressed using Golomb encoding techniques . A list of values is generated from elements of the language model. The list of integer values is sorted, and for each element, a difference between adjacent integer values in the list is calculated. Each calculated difference is encoded using a Golomb code.

In another embodiment, a system for processing user inputs has a user interface, a memory, a Golomb encoder/decoder, and a processor. The user interface is adapted to receive user inputs . The memory is adapted to store information and to store a Golomb compressed language model. The Golomb encoder/decoder is adapted to encode user input and to decode elements of the Golomb compressed language model. The processor is adapted to compare encoded user input against elements of the Golomb compressed language model to identify probable matches .

In another embodiment, a method of decoding user inputs using a Golomb-encoded language model is provided. A user input is divided into a plurality of- elements, each of which is encoded using a hash technique. Each encoded element is compared to elements of a Golomb-encoded language model to identify possible matches. Possible matches are analyzed statistically to estimate a likelihood that a possible match is a correct mapping of the user input to the Golomb-encoded language model.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram of one computing environment in which embodiments may be practiced.

FIG. 2 is a block diagram of an alternative computing environment in which embodiments may be practiced.

FIG. 3 is a simplified flow diagram of an embodiment of a process for compressing a language model for use in computing devices .

FIG. 4 is a simplified flow diagram, of a process for Golomb coding differences between hash values calculated according to the process of FIG. 3.

FIG. 5 is a simplified block diagram of a Huffman tree illustrating a unary code..

FIG. 6 is a simplified flow diagram of an embodiment of a process for decoding a Golomb coded first difference.

FIG. 7 is a simplified block diagram of an embodiment of a system adapted for using a language model compressed with Golomb coding techniques.

FIG. 8 is a simplified flow diagram of an embodiment of a process for decoding user input against a Golomb-encoded language model.

DETAILED DESCRIPTION

Language models are utilized in speech recognition systems, in context sensitive spelling correction systems, in interfaces used to enter Asian characters into computers, and the like. Golomb compression techniques can be applied to user inputs, such as uniform resource locator (URL) data for navigating global computer networks, such as the Internet. Since memory is often limited in practice, especially in mobile platforms such as cell phones, personal digital assistants (PDAs) , and the like, compression of the language model can be quite useful, and Golomb coding techniques can be used both to compress a language model and to decode results.

FIG. 1 illustrates an example of a suitable computing system environment 100 on which embodiments language model compression techniques may be implemented. The

computing system environment 100 is only one example of a suitable computing environment and is not intended to suggest any limitation as to the scope of use or functionality of the invention. Neither should the computing environment 100 be interpreted as having any dependency or requirement relating to any one or combination of components illustrated in the exemplary operating environment 100.

Embodiments of the invention are operational with numerous other general purpose or special purpose computing system environments or configurations. Examples of well-known computing systems, environments, and/or configurations that may be suitable for use with embodiments of the invention include, but are not limited to, personal computers, server computers, hand-held or laptop devices, multiprocessor systems, microprocessor-based systems, set top boxes, programmable consumer electronics, network PCs, minicomputers, mainframe computers, telephony systems, distributed computing environments that include any of the above systems or devices, and the like. Embodiments may be described in the general context of computer-executable instructions, such as program modules, being executed by a computer. Generally, program modules include routines, programs, objects, components, data structures, etc. that perform particular tasks or implement particular abstract data types. The invention is designed to be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules are located in both local and remote computer storage media including memory storage devices .

With reference to FIG. 1, an exemplary system for implementing an embodiment includes a general-purpose computing device in the form of a computer 110. Components of computer 110 may include, but are not limited to, a processing unit 120, a system memory 130, and a system bus 121 that

couples various system components including the system memory to the processing unit 120. The system bus 121 may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. By way of example, and not limitation, such architectures include Industry Standard Architecture (ISA) bus, Micro Channel Architecture (MCA) bus, Enhanced ISA (EISA) bus, Video Electronics Standards Association (VESA) local bus, and Peripheral Component Interconnect (PCI) bus also known as Mezzanine bus.

1000 5, 888 55,701,699 29 21,383

Table 2 illustrates memory usage for one embodiment of Golomb coded hash values for URLs. Memory depends on both the number of URLs (N) and average delta (P/N) . The table illustrates 3 settings of average delta, corresponding to 14-29 bits per URL. This type of compression makes it possible to incorporate large language models in portable devices.

In general, the language model can be used to assist a user in accessing information. For example, in a search engine, Golomb coding techniques can be employed to test alternate

spellings of search terms provided by the user. In the context of a web browser on a portable device, Golomb compressions (coding/decoding) techniques can be adapted to test alternative URL values in order to correct for mistyped URLs.

FIG. 8 is a simplified flow diagram of an embodiment for decoding a user input relative to a Golomb-coded language model. A user input is received or read, for example, symbol by symbol from a data stream, a file, or an input device (step

800) . The user input is divided into a plurality of n-grams

(step 802) . Each of the plurality of n-gramsi are encoded using a hash technique (step 804) . Each encoded n-gram is compared to the Golomb-coded language model to identify possible matches (step 806) . A likelihood is estimated statistically for each possible match that the possible match is a correct mapping of the received user input to an element within the Golomb-coded language model (step 808) . Any number

of statistical algorithms can be applied to estimate the likelihood that a given match is correct. In general, each n- gram can be encoded using Golomb-encoding technique, such as that described above with respect to FIG. 4.

Although the present invention has been described with reference to particular embodiments, workers skilled in the art will recognize that changes may be made in form and detail without departing from the spirit and scope of the invention .

Algoritma

Pendahuluan

Kata Algoritma berasal dari nama seorang ahli matematika dari persia Al Khawarizmi . Pada awalnya kata algorism adalah istilah yang merujuk kepada aturan-aturan aritmetis untuk menyelesaikan persoalan dengan menggunakan bilangan numerik arab. Pada abad 18, istilah ini berkembang menjadi algoritma , yang mencakup semua prosedur atau urutan langkah yang jelas dan diperlukan untuk menyelesaikan suatu permasalahan. Jika dijelaskan lebih lanjut, algoritma (Inggris: algorithm) merupakan kumpulan perintah untuk menyelesaikan suatu masalah. Perintah-perintah ini dapat diterjemahkan secara bertahap dari awal hingga akhir. Masalah tersebut dapat berupa apa saja, dengan catatan untuk setiap masalah, ada kriteria kondisi awal yang harus dipenuhi sebelum menjalankan algoritma. Algoritma akan dapat selalu berakhir untuk semua kondisi awal yang memenuhi kriteria, berbeda dengan heuristik . Algoritma sering mempunyai langkah pengulangan (iterasi) atau memerlukan keputusan (Aljabar Boolean dan pertidaksamaan) sampai tugas selesai. Desain dan analisis algorithma adalah suatu cabang khusus dalam Ilmu Komputer yang mempelajari karakteristik dan performa dari suatu algoritma dalam menyelesaikan masalah, terlepas dari implementasi algoritma tersebut. Dalam cabang disiplin ini algoritma dipelajari secara abstrak, terlepas dari sistem komputer atau bahasa pemrograman yang digunakan. Algoritma yang berbeda dapat diterapkan pada suatu masalah dengan kriteria yang sama. Kompleksitas dari suatu algoritma merupakan ukuran seberapa banyak komputasi yang dibutuhkan algoritma tersebut untuk menyelesaikan masalah. Secara informal, algoritma yang dapat menyelesaikan suatu permasalahan dalam waktu yang singkat memiliki kompleksitas yang rendah, sementara algoritma yang membutuhkan waktu lama untuk menyelesaikan suatu masalah membutuhkan kompleksitas yang tinggi.

Jenis-jenis Algoritma

Terdapat beragam klasifikasi algoritma dan setiap klasifikasi mempunyai alasan tersendiri. Salah satu cara untuk melakukan klasifikasi jenis-jenis algoritma adalah dengan memperhatikan paradigma dan metode yang digunakan untuk mendesain algoritma tersebut. Beberapa paradigma yang digunakan dalam menyusun suatu algoritma akan dipaparkan dibagian ini. Masing-masing paradigma dapat digunakan dalam banyak algoritma yang berbeda.

- Divide and Conquer , paradigma untuk membagi suatu permasalahan besar menjadi permasalahan-permasalahan yang lebih kecil. Pembagian masalah ini dilakukan terus menerus sampai ditemukan bagian masalah kecil yang mudah untuk dipecahkan. Singkatnya menyelesaikan keseluruhan masalah dengan membagi masalah besar dan kemudian memecahkan permasalahan-permasalahan kecil yang terbentuk.

- Dynamic programming , paradigma pemrograman dinamik akan sesuai jika digunakan pada suatu masalah yang mengandung sub-struktur yang optimal (, dan mengandung beberapa bagian permasalahan yang tumpang tindih . Paradigma ini sekilas terlihat mirip dengan paradigma Divide and Conquer , sama-sama mencoba untuk membagi permasalahan menjadi sub permasalahan yang lebih kecil, tapi secara intrinsik ada perbedaan dari karakter permasalahan yang dihadapi.

- Metode serakah. Sebuah algoritma serakah mirip dengan sebuah Pemrograman dinamik , bedanya jawaban dari submasalah tidak perlu diketahui dalam setiap tahap; dan menggunakan pilihan "serakah" apa yang dilihat terbaik pada saat itu.

Lihat pula

- Daftar algoritma

af:Algoritme

an:Algoritmo

Secara umum, Ilmu komputer (Ilkom), atau yang dalam bahasa Inggrisnya disebut Computer Science (CS), adalah ilmu yang mempelajari tentang komputasi , baik perangkat keras (hardware) maupun perangkat lunak (software). Ilmu komputer mencakup beragam topik berkaitan dengan komputer , dari analisa abstrak algoritma sampai subyek yang lebih konkret seperti bahasa pemrograman , perangkat lunak , dan perangkat keras . Sebagai suatu disiplin ilmu, Ilmu Komputer berbeda dengan pemrograman komputer , rekayasa perangkat lunak dan teknik komputer , sekalipun ketiga istilah tersebut sering disalahartikan. Tesis Church-Turing menyatakan bahwa semua alat komputasi yang telah umum diketahui sebenarnya sama dalam hal apa yang bisa mereka lakukan, sekalipun dengan efisiensi yang berbeda. Tesis ini terkadang dianggap sebagai prinsip dasar dari ilmu komputer. Para ahli ilmu komputer biasanya menekankan mesin von Neumann atau mesin Turing (komputer yang mengerjakan tugas yang kecil dan deterministik pada suatu waktu tertentu), karena seperti itulah kebanyakan komputer digunakan sekarang ini. Para ahli ilmu komputer juga mempelajari jenis mesin yang lain, beberapa diantaranya praktikal (seperti paralel komputer dan Kuantum komputer) dan beberapa diantaranya cukup teoritis (seperti Random komputer and Oracle komputer). Ilmu Komputer mempelajari apa yang bisa dilakukan oleh program, dan apa yang tidak (komputabilitas dan intelegensia buatan), bagaimana program harus mengevaluasi suatu hasil (algoritma), bagaimana program harus menyimpan dan mengambil bit tertentu dari suatu informasi (struktur data), dan bagaimana program dan pengguna berkomunikasi (antarmuka pengguna dan bahasa pemrograman). Ilmu komputer berakar dari elektronika , matematika dan linguistik . Dalam tiga dekade terakhir dari abad 20 , ilmu komputer telah menjadi suatu disiplin ilmu baru dan telah mengembangkan metode dan istilah sendiri. Departemen ilmu komputer pertama didirikan di Universitas Purdue pada tahun 1962 . Hampir semua universitas sekarang mempunyai departemen ilmu komputer. Penghargaan tertinggi dalam ilmu komputer adalah Turing Award , pemenang penghargaan ini adalah semua pionir di bidangnya. Edsger Dijkstra mengatakan: :Ilmu komputer bukan tentang komputer sebagaimana astronomi bukan tentang teleskop Fisikawan ternama Richard Feynman mengatakan: :Ilmu komputer umurnya tidak setua fisika; lebih muda beberapa ratus tahun. Walaupun begitu, ini tidak berarti bahwa "hidangan" ilmuwan komputer jauh lebih sedikit dibanding fisikawan. Memang lebih muda, tapi dibesarkan secara jauh lebih intensif!

Catatan tentang istilah 'Informatika' dan 'Ilmu komputer'

Dalam bahasa Indonesia , istilah Informatika diturunkan dari bahasa Perancis informatique,

yang dalam bahasa Jerman disebut Informatik. Sebenarnya, kata ini identik dengan istilah computer science di Amerika Serikat dan computing science di Inggris. Namun, istilah informatics dalam bahasa Inggris memiliki makna yang sedikit berbeda, yaitu lebih menekankan pada aspek pengolahan informasi secara sistematis dan rasional.

Hubungan Informatika dengan bidang lain

Ilmu komputer berkaitan erat dengan beberapa bidang lain. Bidang-bidang ini tidak benar-benar terpisah, sekalipun mempunyai perbedaan penting.

Ilmu Informasi

Ilmu Informasi adalah ilmu yang mempelajari data dan informasi, mencakup bagaimana menginterpretasi, menganalisa, menyimpan, dan mengambil kembali. Ilmu informasi dimulai sebagai dasar dari analisa komunikasi dan basis data .

Sistem Informasi

Sistem Informasi adalah aplikasi komputer untuk mendukung operasi dari suatu organisasi: operasi, instalasi, dan perawatan komputer, perangkat lunak, dan data. Sistem Informasi Manajemen adalah kunci dari bidang yang menekankan finansial dan personal manajemen. 'Sistem Informasi' dapat berupa gabungan dari beberapa elemen teknologi berbasis komputer yang saling berinteraksi dan bekerja sama berdasarkan suatu prosedur kerja (aturan kerja) yang telah ditetapkan, dimana memproses dan mengolah data menjadi suatu bentuk informasi yang dapat digunakan dalam mendukung keputusan.

Rekayasa perangkat lunak

Rekayasa Perangkat Lunak menekankan analisa, desain, dan konstruksi dari perangkat lunak menggunakan alat-alat dan cara kerja yang baru.

Rekayasa Komputer

Rekayasa Komputer adalah ilmu yang mempelajari analisa, desain, dan konstruksi dari perangkat keras komputer. Ilmu yang mempelajari segala aspek pembuatan, konstruksi, pemeliharaan perangkat lunak

Keamanan Informasi

Keamanan Informasi adalah ilmu yang mempelajari analisa dan implementasi dari keamanan sistem informasi (termasuk Kriptografi).

Cabang Ilmu Utama Informatika

Dasar Matematika

- Aljabar Boolean

- Matematika Diskrit

- Teori graf

- Teori Informasi

- Logika Simbolik

- Peluang and Statistik

Teori Ilmu Komputer

- Teori Informasi Algoritmik

- Kompilator

  - Analisis Leksikal

  - Penguraian

- Kriptografi

- Semantik Denotasional

- Komputasi (atau Ilmu Komputer Teoritis)

  - analisa dari algoritma dan Teori Kompleksitas Komputasi dari problem

  - logika dan arti dari program

  - logika matematika dan bahasa formal

- Teori Tipe

Perangkat Keras

(lihat juga elektronika)

- struktur kontrol dan Mikroprogram

- aritmetic dan struktur logika

- struktur memori

- masukan/keluaran dan komunikasi data

- media penyimpanan

- CD Media dan CD ROM

- desain logika

- sirkuit terpadu

- Very Large System Integration

- kinerja dan reliabilitas

Organisasi Sistem Komputer

(lihat juga elektronika)

- Arsitektur Komputer

- Jaringan Komputer

- Komputasi Terdistribusi

- Komputasi Grid

- Kinerja dari Sistem

- Implementasi dari Sistem Komputer

Perangkat Lunak

- Program Komputer and Pemrograman Komputer

- Pemrograman Paralel

- Spesifikasi Program

- Verifikasi Program

- Teknik Pemrograman

- Rekayasa Perangkat Lunak

- Optimisasi Perangkat Lunak

- Metrik Perangkat Lunak

- Pola Desain (Ilmu Komputer)

- Metode Pengembangan Perangkat Lunak

- Bahasa Pemrograman

- Sistem Operasi

Data dan Sistem Informasi

- Struktur Data

- Representasi penyimpan data

- Kriptografi

- Kompresi data

- Pengkodean dan Teori Informasi

- Berkas

- Format Berkas

- Sistem Informasi

- Basis Data

- Penyimpanan dan Pengambilan Informasi Informasi

- Antarmuka dan presentasi informasi

Metodologi Komputasi

- manipulasi simbolik dan aljabar

- Kecerdasan Buatan

- Grafik Komputer

- Pengolahan Citra dan Visi Komputer

- Pengenalan Pola

- Pengenalan Suara

- Simulasi dan Pemodelan

- Pengolahan dokumen dan teks

- Pengolahan Sinyal Digital

Aplikasi Komputer

- Pengolahan data administratif

- Perangkat lunak matematika

  - Analisis numerik

  - Pembukti teori otomatis

  - Aljabar komputer

- Ilmu dan teknik fisika

  - Kimia Komputasional

  - Fisika Komputasional

- Ilmu hayat dan medis

  - Bioinformatika

  - Biologi Komputasional

  - Informatika Medika

- Sosiologi

- Seni dan kemanusiaan

- rekayasa berbantuan komputer

- Robotik

- Interaksi manusia dan komputer

  - Sintesa suara

  - Rekayasa kedapatgunaan

- Hiburan

  - Permainan Komputer

Lingkungan Komputasi

- Industri Komputer

- Sejarah dari Perhitungan

- Komputer dan pendidikan

- Komputer dan masyarakat

  - Kerja Kooperatif Didukung Komputer

- Aspek hukum dari komputer

- manajemen dari komputasi dan sistem informasi

- personal komputer

- Keamanan Komputer dan Keamanan Informasi

Sejarah

- Sejarah dari Perhitungan

- Projek pemrograman awal

- Departemen Ilmu Komputer

- Garis Waktu dari Algoritma

Ahli Terkenal Ilmu Komputer

- John Backus Penemu FORTRAN, bahasa pemrograman tingkat tinggi pertama dan susunan Backus-Naur untuk mendeskripsikan bahasa formal sintaks .

- James Cooley dan John Tuckey Fourier Transform Cepat (Fast Fourier Transform) dan pengaruhnya pada riset keilmuan.

- Ole-Johan Dahl dan Kristen Nygaard , penemu bahasa berorientasi objek SIMULA.

- Edsger Dijkstra untuk algoritma, Goto .

- Kenneth Iverson Penemu APL, untuk kontribusinya di perhitungan interaktif.

- William Kahan untuk standard IEEE floating-point.

- Donald Knuth untuk Seni dari Pemrograman Komputer

- Ada Lovelace programer terkenal pertama di dunia

- John von Neumann yang telah mengembangkan arsitektur von Neumann .

- Claude E. Shannon untuk teori informasi

- Alan Turing untuk teori komputabilitas.

- James Wilkinson Teknik "analisa kesalahan dari belakang" dan kemajuan di bidang perhitungan matriks. Wilkinson adalah juga penggerak dalam pengembangan Pilot ACE, komputer di Inggris yang pertama, pada akhir 1940-an. (lihat Wilkinson pada biografi MacTutor.)

- Konrad Zuse Pembuat binari komputer yang pertama pada 1930-an, di mana dia menrencanakan bahasa pemrograman jauh sebelum waktunya. Lihat Daftar Ahli Ilmu Komputer untuk informasi lebih lanjut.

Lihat juga

- Bug

- Bahasa Pemrograman

- Perhitungan

- Sejarah dari Perhitungan

- Turing Award (ACM)

- Medali IEEE John von Neumann

- Hello world

- Istilah Komputer

- Berkas Istilah

- Topik utama Ilmu Komputer

- Analogi Perhitungan

- Internet

- Multimedia

- Akusisi data

- Tolok

- Jaringan Sensor

- Komputasi dan Algorithma Online ,

- Format Bilangan Komputer

Pranala Luar