APLIKASI ENKRIPSI DAN DEKRIPSI PESAN DENGAN ALGORITMA HUFFMAN MENGGUNAKAN PYTHON 2.7

Yusman

Yusman@staff.gunadarma.ac.id

ABSTRAK
Tujuan penulisan ini adalah untuk menjelaskan pembuatan aplikasi sederhana pengenalan kriptografi dengan menggunakan algoritma Huffman. Pertukaran informasi dapat mudah dilakukan dengan adanya media penyimpanan yang dapat dipindahkan antar komputer atau menggunakan surat elektronik. Terkadang ada informasi yang bersifat pribadi dan rahasia dan hanya orang tertentu yang boleh membaca informasi tersebut. Untuk itu dalam tulisan ini dibahas tentang perancangan program aplikasi yang bertujuan untuk mengenkripsi dan mendekripsi pesan menggunakan algoritma Huffman. Hasil akhir dari tulisan ini adalah suatu program aplikasi yang menghasilkan pesan acak yang sulit diterjemahkan jika tidak mengetahui cara untuk membukanya dan terbatas pada karakter alphabet (ASCII)

Kata-kunci : kriptografi, enkripsi, dekripsi, Huffman.

PENDAHULUAN

Kriptografi

Kriptografi merupakan seni dan ilmu menyembunyikan informasi dari penerima yang tidak berhak. Kata cryptography berasal dari kata Yunani kryptos (tersembunyi) dan graphein (menulis).
Kriptografi mempunyai tujuan antara lain :
a. Kerahasiaan Data (Confidentiality). Untuk melindungi identitas pemakai atau isi pesan agar tidak dapat dibaca oleh orang lain yang tidak berhak.
b. Integritas Data (Data Integrity). Untuk melindungi pesan agar tidak diubah oleh orang lain.
c. Autentikasi (Authentication). Untuk menjamin keaslian pesan.
d. Pemampatan File yaitu ukuran file menjadi lebih kecil dari file aslinya

Proses yang digunakan untuk menyamarkan atau menyembunyikan plaintext tersebut disebut dengan enkripsi. Teks yang sudah disamarkan atau disembunyikan pada proses enkripsi berisi informasi yang tidak dapat atau tidak mudah dibaca dan dimengerti dengan jelas. Teks hasil enkripsi ini disebut dengan chipertext.
Proses kebalikan enkripsi, yaitu mengubah chipertext menjadi plaintext disebut dengan proses dekripsi. Diperlukan kunci yaitu kode untuk melakukan enkripsi dan dekripsi.

Algoritma Huffman adalah salah satu teknik dalam kompresi data dari karakterASCII yang menggunakan struktur data binary tree. Tujuan utama Algoritma Huffman adalah bagaimana memampatkan / mengecilkan ukuran file tanpa mengubah isinya. Artinya ketika file yang diterima akan digunakan , isinya dapat dikembalikan kepada bentuk asliny. Dalam penulisan ini Algoritma Huffman digunakan untuk mengekripsi / menyembunyikan isi file asli sehingga tidak dapat dibaca bagi penerima yang tidak berhak.

METODE PENELITIAN

Algoritma Huffman:

1. Menghitung frekuensi masing-masing karakter (karakter,frekuensi)
2. Frekuensi tersebut diurutkan dari terendah ke tertinggi(Menaik)
3. Buat node baru yang berisi gabungan dari node dengan frekuensi terkecil yang terletak di sebelah kiri kemudian diurutkan kembali frekuensi dari terendah ke tertinggi (frekuensi karakter sebelah kiri selalu yang terendah)
4. Ulangi langkah 1, 2, dan 3 hingga hanya ada satu node(node akar)

Contoh data yang akan dikirim ABCACAD

Karakter	Frekuensi	Kode ASCII
A	3	01000001
B	1	01000010
C	2	01000011
D	1	01000100

Maka ABCACAD akan di ubah menjadi rangkaian bit string sebanyak 56 bit yaitu
01000001 01000010 01000011 01000001 01000011 01000001 01000100
A B C A C A D

Dimana jika string tersebut diencoding dengan algoritma huffman, akan memakan memori yang lebih kecil dari itu. Karena jumlah bit yang merepresentasikan suatu character dalam algoritma huffman bersifat dinamis, sehigga satu character dengan lainnya akan berbeda dalam jumlah bitnya.

Algoritma Huffman diaplikasikan pada ABCACAD adalah sebagai berikut

1. Menghitung frekuensi masing-masing karakter A, B, C, dan D yang akan di kirim
   A = 3, B = 1, C = 2, dan D = 1

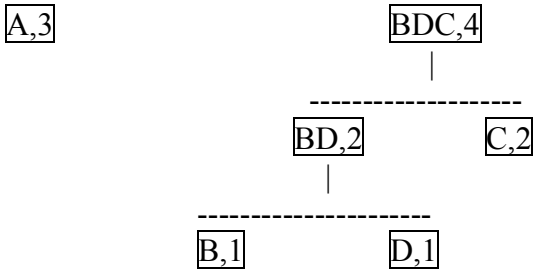
2. Frekuensi tersebut di urutkan makin naik sebagai berikut
   B D C A
   1 1 2 3

3.Pohon Huffman dibuat dengan mengikuti langkah-langkah sebagai berikut

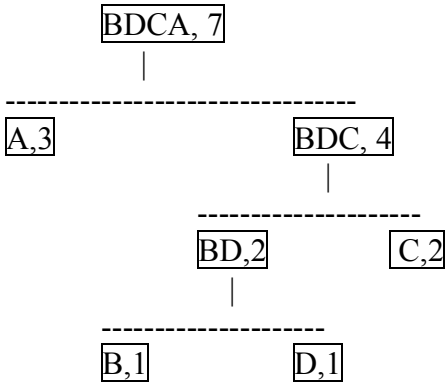
3.1 Buat 4 node untuk masing-masing pasangan yaitu Karakter, Frekuensi

3.2. Pilihlah dua node dengan frekuensi terkecil(node paling kiri / merupakan tempat dengan frekuensi paling rendah) dan buatlah node baru yang isinya merupakan gabungan dua node terkecil tersebut isinya merupakan gabungan dua node terkecil tersebut. Node baru tersebut menjadi bapak dari dua node terkecil tersebut. Node(BD,2) menjadi bapak dari node(B,1) dan node(D,1) serta diurutkan bersama sama node yang lain. (node(C,2) dan node(A,3).

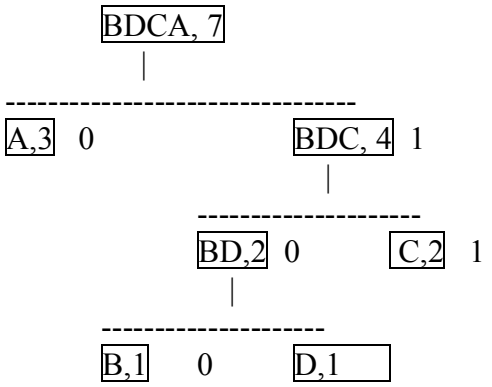
3.3 Ulangi langkah 3.2 yaitu pilih dua node terkecil dan gabungkan keduanya. Node(BD,2) dan node(C,2) adalah node yang terkecil maka di gabungkan menjadi node(BDC,4) kemudian diurutkan bersama node yang lain



3.4 Ulangi langkah diatas sampai terbentuk node akar



3.5 Beri bit 0 pada node yang merupakan anak kiri dan bit 1 pada node yang merupakan anak kanan. Hal ini dilakukan terhadap semua node kecuali terhadap node akar. Maka akan diperoleh tree berikut ini:



Jika dilihat dari pohon diatas makan encoding String “ABCACAD” dengan aturan jika berjalan ke kiri = 0 dan kekanan = 1, maka akan menjadi seperti ini

A = 0 B = 100 C = 11 D = 101

Algoritma Huffman diatas mengakibatkan data yang kirim menjadi kecil, yaitu :

0	100	11	0	11	0	101	= 13 bit
A	B	C	A	C	A	D	

Dan isi dari fle tersebut sudah di enkripsi menjadi
4
njh
A1: C2:¬ B3:⁻D3:
i
5

HASIL DAN PEMBAHASAN

Proses enkripsi dan deskripsi suatu file dengan algoritma Huffman di implementasikan dalam bahasa python versi 2.7.

Algoritma program untuk enkripsi secara garis besar sebagai berikut:

- Baca karakter apa saja yang di gunakan dalam file yang akan di mampatkan dan berapa kali masing-masing karakter digunakan (Frekuensi). Kemudian buat node-node untuk masing-masing pasangan(karakter, frekuensi)

- Pohon Huffman di buat

- Berilah tanda 0 atau 1untuk pengkodeannya

- Baca ulang file tersebut dan terjemahkan dalam kode yang telah dibuat, hasil terjemahan tersebut disimpan dalam file. File ini adalah file yang telah dimampatkan.

Algoritma program untuk membaca kembali file yang telah dienkripsi

- Baca tanda pengenal dan table kode
- Huffman Tree dibuat dari table kode tersebut
- Baca data dan langsung diterjemahkan ke bentuk aslinya, hasil terjemahan tersebut disimpan dalam file

Program asli diambil dari Nick Jones dan Josh Davis's CS 201 course at Carleton College, November 2010. dan di tulis ulang untuk penulisan ini.

Nama dan manfaat masing-masing fungsi utama dalam Huffman.py

1. Fungsi main

Fungsi utama dalam Huffman.py. Untuk menjalankan aplikasi ini digunakan perintah python huffman.py [compress/decompress] [filename] dimana untuk argumen “compress” untuk mengenkripsi dan memampatkan file dan argumen decompress untuk membuka kembali menjadi file asal..

```
if __name__ == "__main__"
        if sys.argv[1] == 'compress':
                encode(sys.argv[2])
        elif sys.argv[1] =='decompress':
                decode(sys.argv[2])
        else:
```

```
        print 'Salah Memasukkan data.: huffman.py [decode/encode] [filename]'
```

2. Fungsi ENCODE

Berguna untuk mengenkripsi file dan menghitung frekuensi untuk setiap karakter yang dibaca serta membuat pohon Huffman. Hasil file yang di enkripsi adalah Namafile+'Compressed.txt'

```
def encode(fileName):

    try:
        wholeDoc = open(fileName, 'r')
    except IOError:
        print 'File Salah'
        return
    docStr = wholeDoc.read()

    allNodes = createHuffmanNodes(docStr)
    tree = createTree(allNodes)
    tree.parseCodeWords(tree.getRoot())
    codeWordDict = tree.getCodeWords()

    (chars, charsBin) = tree.writeFile(docStr)
    compressedSize = len(chars)/1024.0
    originalSize = tree.getRoot().getWeight()/1024.0

    # write the file:
    outFileName = fileName[:len(fileName) - 4] + 'Compressed.txt'

    f = open(outFileName, 'w')
    f.write(chars)
    print Selesai. Nama File hasil adalah :', outFileName
    print 'Original:', originalSize, 'KB;', 'Compressed:', compressedSize, 'KB'
    print 'Compression rate of', compressedSize/originalSize
```

3. Fungsi DECODE

Berguna untuk mengubah ke bentuk asal dari file. Hasil file adalah NamaFile + 'decompressed.txt'.

```
def decode(fileName):
    (rebuiltDict, truncatedFileStr) = importDict(fileName)
    if (rebuiltDict, truncatedFileStr) != (None, None):
        tree = HuffmanTree()
```

```
        tree.setCodeWords(rebuiltDict)
        reconstructed = tree.decodeFile(truncatedFileStr)
        outFileName = fileName[:fileName.index('Compressed.txt')]
        outFileName += 'Decompressed.txt'
        f = open(outFileName, 'w')
        f.write(reconstructed)

        print 'Done. Your decompressed file is saved as:', outFileName
```

Uji Coba Aplikasi

```
Nama file : uji.txt

Isi dari file uji.txt
        Algoritma Huffman

C:\Python27>python huffman.py compress uji1.txt
Isi dari file uji1Compressed.txt
15
njh
A4:┐ 4:┘g4:└f4:
```

```
i4:H4:−
4:m3: l4:
o4:  n4:  r4:
u4:
t4a4:
'8¥Á¨×þ6{┐2

C:\Python27>python huffman.py decompress uji1Compressed.txt

Directory of C:\Python27

09/25/2013  05:07 PM                21 Uji1.txt
09/25/2013  05:07 PM                82 uji1Compressed.txt
09/25/2013  05:08 PM                21 uji1Decompressed.txt
               3 File(s)            124 bytes
               0 Dir(s)   2,019,008,512 bytes free
```

```
Nama File  LICENSE.txt
Isi dari File : lihat lampiran

C:\Python27>python huffman.py compress LICENSE.txt
Isi dari file LICENSECompressed.txt { Lihat Lampiran )
85
njh
15:
6:2 2: "9:‰ $14:`

Directory of C:\Python27

05/15/2013  10:53 PM            40,098 LICENSE.txt
09/25/2013  05:12 PM            25,988 LICENSECompressed.txt
               2 File(s)         66,086 bytes
               0 Dir(s)   2,018,983,936 bytes free

C:\Python27>python huffman.py decompress LICENSECompressed.txt
Traceback (most recent call last):
  File "huffman.py", line 150, in <module>
    decode(sys.argv[2])
  File "huffman.py", line 42, in decode
    reconstructed = tree.decodeFile(truncatedFileStr)
  File "C:\Python27\huffmantree.py", line 85, in decodeFile
    lastCharBin = '0'*(int(lastTwoChars[1]) - len(lastCharBin)) + lastCharBin #
lastTwoChars[1] tells how long bin. rep. of lastChar should be.
ValueError: invalid literal for int() with base 10: 'o'
```

KESIMPULAN

Aplikasi ini dibuat untuk pembelajaran tentang kriptografi yang sederhana. Dalam uji coba dibuat kesimpulan

File yang isinya mempunyai karakter berfrekuensi satu(tidak ada karakter yang sama) atau isinya lebih kecil dari pada tabel kode yang dibuat hasil file yang dimampatkan lebih besar dari file asli.

File hasil enkripsi yang lebih besar dari 1KB maka tidak dapat di deskripsi /dikembalikan ke bentuk aslinya.

Tidak adanya kunci (key) untuk deskripsi file membuat isi file mudah untuk dibaca bagi yang tidak berhak jika mempunyai program algoritma Huffman.

File asli tidak di hapus karena aplikasi ini untuk pembelajaran tentang kriptografi

Isi File telah dapat di samarkan / di kodekan sehingga tidak mudah dibaca

DAFTAR PUSTAKA

Ariyus, Dony., 2008, *Pengantar Ilmu Kriptografi: Teori, Analisis, dan Implementasi*. Penerbit Andi, Yogyakarta.

Munir, Rinaldi., 2006, *Kriptografi*. Penerbit Informatika, Bandung.

Suryanto, Tony., 1995, Pemampatan File dengan Algoritma Huffman. Penerbit Dinastindo, Jakarta.

```
# HUFFMAN.PY.

from huffmantreenode import HuffmanTreeNode
from queue import Queue
from huffmantree import HuffmanTree
import sys


def encode(fileName):
	"""Takes in a fileName, creates a HuffmanTree and compresses the original file based on this tree."""
	try:
		wholeDoc = open(fileName, 'r')
	except IOError:
		print 'Invalid filename. Make sure your file is in same directory as this program.'
		return
	docStr = wholeDoc.read()

	allNodes = createHuffmanNodes(docStr)
	tree = createTree(allNodes)
	tree.parseCodeWords(tree.getRoot())
	codeWordDict = tree.getCodeWords()

	(chars, charsBin) = tree.writeFile(docStr)
	compressedSize = len(chars)/1024.0
	originalSize = tree.getRoot().getWeight()/1024.0

	# write the file:
	outFileName = fileName[:len(fileName) - 4] + 'Compressed.txt'

	f = open(outFileName, 'w')
	f.write(chars)
	print 'Done. Your compressed file is saved as:', outFileName
	print 'Original:', originalSize, 'KB;', 'Compressed:', compressedSize, 'KB'
	print 'Compression rate of', compressedSize/originalSize

def decode(fileName):
	"""Takes in a file name, rebuilds dictionary and writes the decompressed file."""
	(rebuiltDict, truncatedFileStr) = importDict(fileName)
	if (rebuiltDict, truncatedFileStr) != (None, None):
		tree = HuffmanTree()
		tree.setCodeWords(rebuiltDict)
		reconstructed = tree.decodeFile(truncatedFileStr)
		outFileName = fileName[:fileName.index('Compressed.txt')]
		outFileName += 'Decompressed.txt'
		f = open(outFileName, 'w')
		f.write(reconstructed)

		print 'Done. Your decompressed file is saved as:', outFileName

def createHuffmanNodes(docStr):
	"""Reads string of entire document (docStr) and returns a dictionary of form: individual character -> HuffmanNode for that
char"""
	results = {}
	for x in docStr: # increment weight, else create a new node with weight 1
		try:
			temp = results[x]
			temp.setWeight(temp.getWeight() + 1)
		except KeyError:
			results[x] = HuffmanTreeNode(1, x)
	return results

def createTree(allNodes):
	"""Takes in dictionary of Huffman nodes. Creates a Huffman Tree based on each node's frequencies."""
	listNodes = allNodes.values()
	listNodes.sort(compareHuffNodes)

	singleNodes = Queue() # will contain all HTNodes at first
```

```
	comboNodes = Queue() # will contain trees - nodes with others hanging off
	while len(listNodes) > 0:
		singleNodes.enqueue(listNodes.pop(0)) # these will be sorted with least weight at front


	while len(singleNodes) + len(comboNodes) > 1:
		i = 0
		temp = [None, None]
		for i in range(2): # tiny for loop saves code!

			singleNodeWeight = None
			comboNodeWeight = None
			if singleNodes.peek() != None:
				singleNodeWeight = singleNodes.peek().getWeight()
			if comboNodes.peek() != None:
				comboNodeWeight = comboNodes.peek().getWeight()

			if singleNodeWeight == None:
				temp[i] = comboNodes.dequeue()
			elif comboNodeWeight < singleNodeWeight and comboNodeWeight != None:
				temp[i] = comboNodes.dequeue()
			else: # they can't both be None, so no need to check if singleNodeWeight is None
				temp[i] = singleNodes.dequeue()

		parent = HuffmanTreeNode()
		parent.setLeftChild(temp[0])
		parent.setRightChild(temp[1])
		comboNodes.enqueue(parent)

	root = comboNodes.dequeue() # should only be one left

	tree = HuffmanTree(root)
	return tree

def importDict(fileName):
	"""Rewrites the dictionary of key value pairs based on the first few lines of compressed file."""
	rebuiltDict = {}
	try:
		f = open(fileName, 'r')
	except IOError:
		print 'Invalid filename. Make sure your file is in same directory as this program.'
		return (None, None)
	numEntries = int(f.readline())

	if f.readline() != 'njh\n':
		print 'maybe this isn\'t a file comprresed by NJ Huffman'
	entryNum = 0
	while entryNum < numEntries:

		tempchar = f.read(1)
		temp = f.read(1) # start of length of codeword
		length = ''
		while temp != ':':
			length += temp
			temp = f.read(1)

		length = int(length)
		binStr = ''
		i = 0
		for i in range(length/8):
			encodedChar = f.read(1)
			tempBinStr = bin(ord(encodedChar))[2:] # don't want the '0b'
			binStr += tempBinStr.zfill(8) # char => binary => add zeros to make length 8
		if length % 8 != 0: # there's an extra char left to read
			encodedChar = f.read(1)
			tempBinStr = bin(ord(encodedChar))[2:] # don't want the '0b'
			binStr += tempBinStr.zfill(length % 8) # char => binary => add zeros to make length = length % 8

		rebuiltDict[tempchar] = binStr
		entryNum += 1
```

```
	return (rebuiltDict, f.read())

def compareHuffNodes(node1, node2):
	"""Compares two Huffman nodes by weight"""
	if node1.getWeight() < node2.getWeight():
		return -1
	else:
		return 1

if __name__ == "__main__":
	if sys.argv[1] == 'compress':
		encode(sys.argv[2])
	elif sys.argv[1] =='decompress':
		decode(sys.argv[2])
	else:
		print 'Invalid input. Please enter in the form: huffman.py [decode/encode] [filename]'



*-----------------------------------------------------------------------------
#HuffmannTree.py

from huffmantreenode import HuffmanTreeNode

class HuffmanTree(object):

	def __init__(self, root = None):
		"""Creates a new instance of a Huffman tree. The root is the HuffmanTreeNode at the top of the entire tree. """
		self.root = root
		self.codeWords = {} # of form char => binary string

	def parseCodeWords(self, currentNode, currentCode = ''):
		"""Traverses the tree, populating the dictionary with characters and their cooresponding codewords."""
		if currentNode == None: # i.e. parent was a leaf node
			return
		else:
			if currentNode.getValue() != None: # we don't want branch nodes in our dictionary
				self.codeWords[currentNode.getValue()] = currentCode

		self.parseCodeWords(currentNode.getLeftChild(), currentCode + '0') # set left to be currentCode + '0'
		self.parseCodeWords(currentNode.getRightChild(), currentCode + '1') # set right to be currentCode + '1'

	def getCodeWords(self):
		"""Returns dictionary with character keys and codeword values."""
		return self.codeWords

	def setCodeWords(self, dict):
		"""For rebuilding. Program will read in representation of codeWords dictionary, set codeWords equal to it."""
		self.codeWords = dict

	def getRoot(self):
		"""Returns root of this HuffmanTree"""
		return self.root

	def writeFile(self, doc):
		"""Returns a string in binary form of the inputted string doc, and the ASCII equivalent of the binary string,
which will be written to the compressed file. Uses the codeWord dictionary to convert from characters to binary digits"""

		binStr = self.convertToBinary(doc)
		encodedStr = self.getDictRep() # this will add the dictionary to start of compressed file
		i = 0
		while i < len(binStr) - 8: # change binary reps. to characters. Must stop before the end, because string length
isn't necessarily divisble by 8. Last chunk will probably not be 8 long.
			temp = chr(int(binStr[i:i+8], 2))
			encodedStr += temp
			i += 8

		lastLength = len(binStr[i:]) # how many left over are there?
		encodedStr += chr(int(binStr[i:], 2)) + str(lastLength) # add the ascii of last few bits, and how many bits that is
		return (encodedStr, binStr)
```

```
	def getDictRep(self):
		"""Returns a string representation of the codeWords dictionary to be written to compressed file."""
		fullStr = str(len(self.codeWords)) + '\nnjh\n'
		for char in self.codeWords:
			binStr = self.codeWords[char]
			encodedStr = char + str(len(binStr)) + ':'
			i = 0
			while i < len(binStr) - 8: # change binary reps. to characters. Must stop before the end, because string length isn't necessarily divisble by 8. Last chunk will probably not be 8 long.
				temp = chr(int(binStr[i:i+8], 2))
				encodedStr += temp
				i += 8

		encodedStr += chr(int(binStr[i:], 2)) # add the ascii of last few bits, and how many bits whole string is
			fullStr += encodedStr
		return fullStr

	def convertToBinary(self, originalDoc):
		"""Converts originalDoc to binary string based on HuffmanTree"""
		binStr = ''
		for x in originalDoc:
			binStr += self.codeWords[x]
		return binStr

	def decodeFile(self, doc):
		"""Takes in the twice encoded file (i.e. the ascii characters of the orignial binary code). Decodes to binary and eventually to original text. Should only be called once the tree has been rebuilt."""
		binStr = ''
		lastTwoChars = doc[len(doc) - 2:] # Last two characters are important. Last one tells us how many bin. digits the second to last one should have when changed to binary representation.
		adjustedDoc = doc[:len(doc) - 2]
		for character in adjustedDoc:
			temp = bin(ord(character))[2:] # strip '0b'
			extraSpace = 8 - len(temp) # this number indicates how many zeros should be added at front of temp (so that it is of length 7)
			zeros = '0'*extraSpace
			temp = zeros + temp
			binStr += temp
		lastCharBin = bin(ord(lastTwoChars[0]))[2:] # just like for all the others
		lastCharBin = '0'*(int(lastTwoChars[1]) - len(lastCharBin)) + lastCharBin # lastTwoChars[1] tells how long bin. rep. of lastChar should be.
		binStr += lastCharBin

		toReturn = self.binToOriginal(binStr)
		return toReturn

	def binToOriginal(self, binStr):
		"""Takes in binary string of digits, converts them to text based on the codeWords dictionary"""
		reverseDir = {}
		for x in self.codeWords:
			reverseDir[self.codeWords[x]] = x
		begIndex = 0
		curIndex = 0
		rebuiltStr = ''
		while curIndex < len(binStr) + 1:
			try:
				rebuiltStr += reverseDir[str(binStr[begIndex:curIndex])] # try looking up current digit string in dict
				begIndex = curIndex # if it worked, move the beginning index to the end of last chunk
			except KeyError:
				curIndex += 1 # move along if current digit string was not found in dictionary
		return rebuiltStr
```

```
*---------------------------------------------------------------
# HuffmanTreeNode.py

from binarytreenode import BinaryTreeNode

class HuffmanTreeNode(BinaryTreeNode):
	"""A subclass of BinaryTreeNode. Acts in exact same manner as BTN except has an extra instance variable, weight, which stores the frequency of the instance variable value. Interior nodes' weight is the sum of all of its children. Nodes DO NOT KNOW THEIR OWN CODE! This is instead dealt with by another class, HuffmanTree, which contains a pointer to the root node and deals with the codewords."""

	def __init__(self, weight=0, value=None):
		"""Initializes a new Huffman tree node with given weight and value (though interior nodes will have value None) """
		BinaryTreeNode.__init__(self,value)
		self.weight = weight

	def setWeight(self, newWeight):
		"""Sets this node's weight to newWeight. Useful for interior nodes."""
		self.weight = newWeight

	def getWeight(self):
		"""Returns this node's weight """
		return self.weight

	def setLeftChild(self, node):
		"""Sets left child to given node, which will presumably be a HTN."""
		BinaryTreeNode.setLeftChild(self, node)
		self.weight += node.getWeight()

	def setRightChild(self, node):
		"""Sets right child to given node, which will presumably be a HTN."""
		BinaryTreeNode.setRightChild(self, node)
		self.weight += node.getWeight()

	def __str__(self):
		"""Returns a string representation - empty child is []. Easier to rewrite this method than to call it from superclass and alter it then."""
		if self.left == None:
			leftStr = '[]'
		else:
			leftStr = str(self.left)

		if self.right == None:
			rightStr = '[]'
		else:
			rightStr = str(self.right)

		if self.value == None:
			valStr = 'None'
		else:
			valStr = str(self.value)
		return '[(' + valStr + ', ' + str(self.weight) + '),' + leftStr + ', ' + rightStr + ']'

if __name__ == "__main__":
	n = HuffmanTreeNode(12, 'a')
	m = HuffmanTreeNode(13, 'd')
	parent = HuffmanTreeNode()
	parent.setLeftChild(n)
	parent.setRightChild(m)
	print 'm: ' , m
	print 'n: ' , n
	print 'parent:', parent
```

*---
# LinkedListNode.py
class LinkedListNode(object):
	"""A basic linked list node class, for use in implementing other more interesting classes."""

	def __init__(self, data):
		"""Initializes a new node with the given data and no next node."""
		self.data = data
		self.next = None

	def setData(self, data):
		"""Sets the node's data, which is set for the first time in __init__()."""
		self.data = data

	def getData(self):
		"""Returns the data currently stored in the node."""
		return self.data

	def setNext(self, next):
		"""Sets the next node, which is None by default."""
		self.next = next

	def getNext(self):
		"""Returns the next node, or None."""
		return self.next

	def __str__(self):
		"""Returns data as string"""
		return self.data

if __name__ == "__main__":
	node = LinkedListNode('hello')
	node2 = LinkedListNode('bye')
	node3 = LinkedListNode('later')
	node.setNext(node2)
	node2.setNext(node3)

	temp = node
	while temp != None:
		print str(temp)
		temp = temp.getNext()
*---
queue.py

from linkedlistnode import LinkedListNode

class Queue(object):
	"""A queue, similar to the one in our Miller and Ranum textbook."""

	def __init__(self):
		"""Initializes an empty queue."""
		self.front = None
		self.back = None
		self.count = 0

	def enqueue(self, x):
		"""Puts the given object into the queue, at the back."""
		if self.count == 0:
			self.front = LinkedListNode(x)
			self.back = self.front
		else:
			temp = LinkedListNode(x)
			self.back.setNext(temp)
			self.back = temp
		self.count += 1

```
	def dequeue(self):
		"""If the queue is not empty, then removes the front object and returns it. If the queue is empty, then does nothing and returns None."""
		if self.count == 0:
			return None
		else:
			temp = self.front.getData()
			self.front = self.front.getNext()
			if self.count == 1:
				self.back = None
			self.count -= 1
			return temp

	def peek(self):
		"""If the queue is not empty, then returns the front object, without removing it from the queue. If the queue is empty, then returns None."""
		if self.front == None:
			return None
		else:
			return self.front.getData()

	def __len__(self):
		"""Returns the number of objects in the queue."""
		return self.count

	def __str__(self):
		"""Returns a human-readable string representation of the queue, from front to back, assuming that the objects in the queue themselves respond to the __str__ method."""
		if self.count == 0:
			return "()"
		elif self.count == 1:
			return "(" + str(self.front.getData()) + ")"
		else:
			# There are at least two objects; commas are needed.
			string = "(" + str(self.front.getData())
			current = self.front.getNext()
			while current != None:
				string += ", " + str(current.getData())
				current = current.getNext()
			string += ")"
			return string

# If this file was executed (rather than imported), then run this demo.
if __name__ == "__main__":
	myQueue = Queue()
	print myQueue
	myQueue.enqueue("Jill")
	print myQueue
	myQueue.enqueue("John")
	print myQueue
	print myQueue.getSize()
	print myQueue.dequeue()
	print myQueue
	myQueue.enqueue(2.0 * 3.14159)
	print myQueue
	myQueue.enqueue("Jenny")
	print myQueue
	print myQueue.dequeue()
	print myQueue
	print myQueue.dequeue()
	print myQueue
	print myQueue.dequeue()
	print myQueue
	print myQueue.dequeue()
	print myQueue
```

FILE LICENCE.TXT

A. HISTORY OF THE SOFTWARE
==========

Python was created in the early 1990s by Guido van Rossum at Stichting Mathematisch Centrum (CWI, see http://www.cwi.nl) in the Netherlands as a successor of a language called ABC. Guido remains Python's principal author, although it includes many contributions from others.

In 1995, Guido continued his work on Python at the Corporation for National Research Initiatives (CNRI, see http://www.cnri.reston.va.us) in Reston, Virginia where he released several versions of the software.

In May 2000, Guido and the Python core development team moved to BeOpen.com to form the BeOpen PythonLabs team. In October of the same year, the PythonLabs team moved to Digital Creations (now Zope Corporation, see http://www.zope.com). In 2001, the Python Software Foundation (PSF, see http://www.python.org/psf/) was formed, a non-profit organization created specifically to own Python-related Intellectual Property. Zope Corporation is a sponsoring member of the PSF.

All Python releases are Open Source (see http://www.opensource.org for the Open Source Definition). Historically, most, but not all, Python releases have also been GPL-compatible; the table below summarizes the various releases.

Release	Derived from	Year	Owner	GPL-compatible? (1)
0.9.0 thru 1.2		1991-1995	CWI	yes
1.3 thru 1.5.2	1.2	1995-1999	CNRI	yes
1.6	1.5.2	2000	CNRI	no
2.0	1.6	2000	BeOpen.com	no
1.6.1	1.6	2001	CNRI	yes (2)
2.1	2.0+1.6.1	2001	PSF	no
2.0.1	2.0+1.6.1	2001	PSF	yes
2.1.1	2.1+2.0.1	2001	PSF	yes
2.2	2.1.1	2001	PSF	yes
2.1.2	2.1.1	2002	PSF	yes
2.1.3	2.1.2	2002	PSF	yes

2.2.1	2.2	2002	PSF	yes
2.2.2	2.2.1	2002	PSF	yes
2.2.3	2.2.2	2003	PSF	yes
2.3	2.2.2	2002-2003	PSF	yes
2.3.1	2.3	2002-2003	PSF	yes
2.3.2	2.3.1	2002-2003	PSF	yes
2.3.3	2.3.2	2002-2003	PSF	yes
2.3.4	2.3.3	2004	PSF	yes
2.3.5	2.3.4	2005	PSF	yes
2.4	2.3	2004	PSF	yes
2.4.1	2.4	2005	PSF	yes
2.4.2	2.4.1	2005	PSF	yes
2.4.3	2.4.2	2006	PSF	yes
2.4.4	2.4.3	2006	PSF	yes
2.5	2.4	2006	PSF	yes
2.5.1	2.5	2007	PSF	yes
2.5.2	2.5.1	2008	PSF	yes
2.5.3	2.5.2	2008	PSF	yes
2.6	2.5	2008	PSF	yes
2.6.1	2.6	2008	PSF	yes
2.6.2	2.6.1	2009	PSF	yes
2.6.3	2.6.2	2009	PSF	yes
2.6.4	2.6.3	2009	PSF	yes
2.6.5	2.6.4	2010	PSF	yes
2.7	2.6	2010	PSF	yes

Footnotes:

(1) GPL-compatible doesn't mean that we're distributing Python under the GPL. All Python licenses, unlike the GPL, let you distribute a modified version without making your changes open source. The GPL-compatible licenses make it possible to combine Python with other software that is released under the GPL; the others don't.

(2) According to Richard Stallman, 1.6.1 is not GPL-compatible, because its license has a choice of law clause. According to CNRI, however, Stallman's lawyer has told CNRI's lawyer that 1.6.1 is "not incompatible" with the GPL.

Thanks to the many outside volunteers who have worked under Guido's direction to make these releases possible.

B. TERMS AND CONDITIONS FOR ACCESSING OR OTHERWISE USING PYTHON

relationship of agency, partnership, or joint venture between PSF and Licensee. This License Agreement does not grant permission to use PSF trademarks or trade name in a trademark sense to endorse or promote products or services of Licensee, or any third party.

8. By copying, installing or otherwise using Python, Licensee agrees to be bound by the terms and conditions of this License Agreement.

BEOPEN.COM LICENSE AGREEMENT FOR PYTHON 2.0

BEOPEN PYTHON OPEN SOURCE LICENSE AGREEMENT VERSION 1

1. This LICENSE AGREEMENT is between BeOpen.com ("BeOpen"), having an office at 160 Saratoga Avenue, Santa Clara, CA 95051, and the Individual or Organization ("Licensee") accessing and otherwise using this software in source or binary form and its associated documentation ("the Software").

2. Subject to the terms and conditions of this BeOpen Python License Agreement, BeOpen hereby grants Licensee a non-exclusive, royalty-free, world-wide license to reproduce, analyze, test, perform and/or display publicly, prepare derivative works, distribute, and otherwise use the Software alone or in any derivative version, provided, however, that the BeOpen Python License is retained in the Software, alone or in any derivative version prepared by Licensee.

3. BeOpen is making the Software available to Licensee on an "AS IS" basis. BEOPEN MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR
IMPLIED. BY WAY OF EXAMPLE, BUT NOT LIMITATION, BEOPEN MAKES NO AND
DISCLAIMS ANY REPRESENTATION OR WARRANTY OF MERCHANTABILITY OR FITNESS
FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF THE SOFTWARE WILL NOT
INFRINGE ANY THIRD PARTY RIGHTS.

4. BEOPEN SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF THE
SOFTWARE FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS
AS A RESULT OF USING, MODIFYING OR DISTRIBUTING THE SOFTWARE, OR ANY

DERIVATIVE THEREOF, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.

5. This License Agreement will automatically terminate upon a material breach of its terms and conditions.

6. This License Agreement shall be governed by and interpreted in all respects by the law of the State of California, excluding conflict of law provisions. Nothing in this License Agreement shall be deemed to create any relationship of agency, partnership, or joint venture between BeOpen and Licensee. This License Agreement does not grant permission to use BeOpen trademarks or trade names in a trademark sense to endorse or promote products or services of Licensee, or any third party. As an exception, the "BeOpen Python" logos available at http://www.pythonlabs.com/logos.html may be used according to the permissions granted on that web page.

7. By copying, installing or otherwise using the software, Licensee agrees to be bound by the terms and conditions of this License Agreement.

CNRI LICENSE AGREEMENT FOR PYTHON 1.6.1
---

1. This LICENSE AGREEMENT is between the Corporation for National Research Initiatives, having an office at 1895 Preston White Drive, Reston, VA 20191 ("CNRI"), and the Individual or Organization ("Licensee") accessing and otherwise using Python 1.6.1 software in source or binary form and its associated documentation.

2. Subject to the terms and conditions of this License Agreement, CNRI hereby grants Licensee a nonexclusive, royalty-free, world-wide license to reproduce, analyze, test, perform and/or display publicly, prepare derivative works, distribute, and otherwise use Python 1.6.1 alone or in any derivative version, provided, however, that CNRI's License Agreement and CNRI's notice of copyright, i.e., "Copyright (c) 1995-2001 Corporation for National Research Initiatives; All Rights Reserved" are retained in Python 1.6.1 alone or in any derivative version prepared by Licensee. Alternately, in lieu of CNRI's License Agreement, Licensee may substitute the following text (omitting the quotes): "Python 1.6.1 is made available subject to the terms and conditions in CNRI's License Agreement. This Agreement together with Python 1.6.1 may be located on the Internet using the following unique, persistent identifier (known as a handle): 1895.22/1013. This Agreement may also be obtained from a proxy server on the Internet using the following URL: http://hdl.handle.net/1895.22/1013".

3. In the event Licensee prepares a derivative work that is based on or incorporates Python 1.6.1 or any part thereof, and wants to make the derivative work available to others as provided herein, then Licensee hereby agrees to include in any such work a brief summary of the changes made to Python 1.6.1.

4. CNRI is making Python 1.6.1 available to Licensee on an "AS IS" basis. CNRI MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED. BY WAY OF EXAMPLE, BUT NOT LIMITATION, CNRI MAKES NO AND
DISCLAIMS ANY REPRESENTATION OR WARRANTY OF MERCHANTABILITY OR FITNESS
FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF PYTHON 1.6.1 WILL NOT
INFRINGE ANY THIRD PARTY RIGHTS.

5. CNRI SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF PYTHON
1.6.1 FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS AS
A RESULT OF MODIFYING, DISTRIBUTING, OR OTHERWISE USING PYTHON 1.6.1,
OR ANY DERIVATIVE THEREOF, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.

6. This License Agreement will automatically terminate upon a material breach of its terms and conditions.

7. This License Agreement shall be governed by the federal intellectual property law of the United States, including without limitation the federal copyright law, and, to the extent such U.S. federal law does not apply, by the law of the Commonwealth of Virginia, excluding Virginia's conflict of law provisions. Notwithstanding the foregoing, with regard to derivative works based on Python 1.6.1 that incorporate non-separable material that was previously distributed under the GNU General Public License (GPL), the law of the Commonwealth of Virginia shall govern this License Agreement only as to issues arising under or with respect to Paragraphs 4, 5, and 7 of this License Agreement. Nothing in this License Agreement shall be deemed to create any relationship of agency, partnership, or joint venture between CNRI and Licensee. This License Agreement does not grant permission to use CNRI trademarks or trade name in a trademark sense to endorse or promote products or services of Licensee, or any third party.

8. By clicking on the "ACCEPT" button where indicated, or by copying, installing or otherwise using Python 1.6.1, Licensee agrees to be bound by the terms and conditions of this License Agreement.

ACCEPT

CWI LICENSE AGREEMENT FOR PYTHON 0.9.0 THROUGH 1.2

Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of Stichting Mathematisch Centrum or CWI not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission.

STICHTING MATHEMATISCH CENTRUM DISCLAIMS ALL WARRANTIES WITH REGARD TO
THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND
FITNESS, IN NO EVENT SHALL STICHTING MATHEMATISCH CENTRUM BE LIABLE
FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT
OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Additional Conditions for this Windows binary build

This program is linked with and uses Microsoft Distributable Code, copyrighted by Microsoft Corporation. The Microsoft Distributable Code includes the following files:

msvcr90.dll

msvcp90.dll
msvcm90.dll

If you further distribute programs that include the Microsoft Distributable Code, you must comply with the restrictions on distribution specified by Microsoft. In particular, you must require distributors and external end users to agree to terms that protect the Microsoft Distributable Code at least as much as Microsoft's own requirements for the Distributable Code. See Microsoft's documentation (included in its developer tools and on its website at microsoft.com) for specific details.

Redistribution of the Windows binary build of the Python interpreter complies with this agreement, provided that you do not:

- alter any copyright, trademark or patent notice in Microsoft's Distributable Code;

- use Microsoft's trademarks in your programs' names or in a way that suggests your programs come from or are endorsed by Microsoft;

- distribute Microsoft's Distributable Code to run on a platform other than Microsoft operating systems, run-time technologies or application platforms; or

- include Microsoft Distributable Code in malicious, deceptive or unlawful programs.

These restrictions apply only to the Microsoft Distributable Code as defined above, not to Python itself or any programs running on the Python interpreter. The redistribution of the Python interpreter and libraries is governed by the Python Software License included with this file, or by other licenses as marked.

This copy of Python includes a copy of bzip2, which is licensed under the following terms:

This program, "bzip2", the associated library "libbzip2", and all documentation, are copyright (C) 1996-2010 Julian R Seward. All rights reserved.

Redistribution and use in source and binary forms, with or without

modification, are permitted provided that the following conditions
are met:

1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.

2. The origin of this software must not be misrepresented; you must
   not claim that you wrote the original software. If you use this
   software in a product, an acknowledgment in the product
   documentation would be appreciated but is not required.

3. Altered source versions must be plainly marked as such, and must
   not be misrepresented as being the original software.

4. The name of the author may not be used to endorse or promote
   products derived from this software without specific prior written
   permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS'' AND ANY EXPRESS
OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE
ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY
DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE
GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF
THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Julian Seward, jseward@bzip.org
bzip2/libbzip2 version 1.0.6 of 6 September 2010

This copy of Python includes a copy of Berkeley DB, which is licensed under the
following terms:

```
/*-
 * $Id: LICENSE,v 12.9 2008/02/07 17:12:17 mark Exp $
 */
```

The following is the license that applies to this copy of the Berkeley DB software. For a license to use the Berkeley DB software under conditions other than those described here, or to purchase support for this software, please contact Oracle at berkeleydb-info_us@oracle.com.

```
 */
/*
 * Copyright (c) 1990, 1993, 1994, 1995
 *	The Regents of the University of California.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS
``AS IS" AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED
TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR
CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
POSSIBILITY OF
 * SUCH DAMAGE.
 */
/*
 * Copyright (c) 1995, 1996
 *	The President and Fellows of Harvard University.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
```

This copy of Python includes a copy of openssl, which is licensed under the following terms:

```
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 *
================================================================
=====
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

 Original SSLeay License
 -----------------------

/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
```

```
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ``AS IS" AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
```

This copy of Python includes a copy of Tcl, which is licensed under the following terms:

```
This software is copyrighted by the Regents of the University of
California, Sun Microsystems, Inc., Scriptics Corporation, ActiveState
Corporation and other parties.  The following terms apply to all files
associated with the software unless explicitly disclaimed in
```

individual files.

The authors hereby grant permission to use, copy, modify, distribute,
and license this software and its documentation for any purpose, provided
that existing copyright notices are retained in all copies and that this
notice is included verbatim in any distributions. No written agreement,
license, or royalty fee is required for any of the authorized uses.
Modifications to this software may be copyrighted by their authors
and need not follow the licensing terms described here, provided that
the new terms are clearly indicated on the first page of each file where
they apply.

IN NO EVENT SHALL THE AUTHORS OR DISTRIBUTORS BE LIABLE TO ANY PARTY
FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES
ARISING OUT OF THE USE OF THIS SOFTWARE, ITS DOCUMENTATION, OR ANY
DERIVATIVES THEREOF, EVEN IF THE AUTHORS HAVE BEEN ADVISED OF THE
POSSIBILITY OF SUCH DAMAGE.

THE AUTHORS AND DISTRIBUTORS SPECIFICALLY DISCLAIM ANY WARRANTIES,
INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT. THIS SOFTWARE
IS PROVIDED ON AN "AS IS" BASIS, AND THE AUTHORS AND DISTRIBUTORS HAVE
NO OBLIGATION TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR
MODIFICATIONS.

GOVERNMENT USE: If you are acquiring this software on behalf of the
U.S. government, the Government shall have only "Restricted Rights"
in the software and related documentation as defined in the Federal
Acquisition Regulations (FARs) in Clause 52.227.19 (c) (2). If you
are acquiring the software on behalf of the Department of Defense, the
software shall be classified as "Commercial Computer Software" and the
Government shall have only "Restricted Rights" as defined in Clause
252.227-7013 (c) (1) of DFARs. Notwithstanding the foregoing, the
authors grant the U.S. Government and others acting in its behalf
permission to use and distribute the software in accordance with the
terms specified in this license.

This copy of Python includes a copy of Tk, which is licensed under the following terms:

Government shall have only "Restricted Rights" as defined in Clause
252.227-7013 (c) (1) of DFARs. Notwithstanding the foregoing, the
authors grant the U.S. Government and others acting in its behalf
permission to use and distribute the software in accordance with the
terms specified in this license.

This copy of Python includes a copy of Tix, which is licensed under the following terms:

MODIFICATIONS.

GOVERNMENT USE: If you are acquiring this software on behalf of the U.S. government, the Government shall have only "Restricted Rights" in the software and related documentation as defined in the Federal Acquisition Regulations (FARs) in Clause 52.227.19 (c) (2). If you are acquiring the software on behalf of the Department of Defense, the software shall be classified as "Commercial Computer Software" and the Government shall have only "Restricted Rights" as defined in Clause 252.227-7013 (c) (1) of DFARs. Notwithstanding the foregoing, the authors grant the U.S. Government and others acting in its behalf

permission to use and distribute the software in accordance with the terms specified in this license.

Parts of this software are based on the Tcl/Tk software copyrighted by the Regents of the University of California, Sun Microsystems, Inc., and other parties. The original license terms of the Tcl/Tk software distribution is included in the file docs/license.tcltk.

Parts of this software are based on the HTML Library software copyrighted by Sun Microsystems, Inc. The original license terms of the HTML Library software distribution is included in the file docs/license.html_lib.

FILE LICENCECOMPRESSED.TXT

85
njh
15:' '
6:2 2: "9:‰ $14:`
7:-6:├,7:w/10:ù .6:¶18:,08:*ƒ*310:' ' 28:ø510:‰└411:ù 711:`−610:Ø
910:ù└812:ù ;11:Ø┐:10:‰┐=7:1?15:`- A6:'@12:'
'┐C7:MB8:µE6:7D7:LG8:^F7:/I6::H7:+K12:`┐J13:' '─M8:·L7:mO6:(N6:#Q11:`
P7:@S6:0R6:,U7:.T6:4W9:Ø V9:' ' Y8:´X11:' ' [15:`Z14:`
]15:' '- _14:`a5:
`12:ù c6:1b7:Be4:
d6:3g7:Ef7:~i5:- h6:5k10:ù┐j11:`
m7:}l6:)o5:∟n5:┤q11:`┘p7:□s5:↕r5:┴u6:┬t4:┘w7:Cv8:Ùy7:vx11:Ø└z11:`◆ A_XiE–…
Í┴îaBý63Ù¾LX±bÅ‹┬,X±bÅ‹┬,X±bÅ‹┬,Xå”É¯.ClŒjíGÌõÄÕÇmiìA|ÿ<∟^]^…½Ÿ
–Ü³’’ßF¡,{↔S×‹•»SWúÔô±Ôš÷J-ôaMØõÜ“¹ª'ø›åùaÃ†¦0ü¥éaž,š¹←¦¯Z[|å“d4’Øâò—*sðÒ–
Þ*ÍŠæ-¦—ÌžÖj ┼·³áWúß^DÉ¯.ÀË/õõñ÷öÒ5d×•î6”×‹EÔò-¾4-›ÈúÛö1åÒ¾„²{—
‘ú¼úq5zÉL²êä┤ÏóÄýÈ…½ŸytõÖ|Í}"+ó∟¹└²kË A5ro+ÿ•j{—
?r¹FÔ÷.ÚK- NÚã©Õüžj{eä0¦Çg]É;š¢ ‰¾_┴8jczùJ¼“—Seª-
&- ^¹cÉ9~å·Õ‹ëù¤=^®jå^—lŸ2O²õm&ËÖ^åäsðš¾Rçä†Úº™eÕËvö> ààû‘
{>
¾bjäÉ¯.cÊ¹›ì½<ÿõî,;~Ÿy²ù‰Ì-^ ïu1çÑ8~å~‰«–¯G÷¹└²kËíl) íú urŒS ž§? «’o×Ë³'
'½ÄÕÈ↔“^_kaHGoÓï6_18Lô_#i&ÕÚžåä0¯Èc ¼þùMå ò-OrýÉ;š¢ ‰¾_–
∟8j0^□u1çÕ→ ÕÏˆ88.âjäÉ¯.aÏÉ
µ|—ñkæÔ÷.0 `¿'sTOñ7ËòÃ‡
Oý“^]NV/Ëþ_¾V├
²?r¿_>ãr¿.Ïúó÷ åb' 'þ`-Oræ5v£æKûãïÞÅ´ÓØN∟‡r
d×—
f¯KQóË«£ÓKâ‹6’yýê ”−└yýÉ¼¯þU©î\ô†’ÿËË•õâŸ_ì'©ÏÙ&®@Áz™e=4 ;&¼¹W¥Û'
Ú¹G÷¹‡┬®.0¤îj‰þ&ù~Xpá©Ïïyqjâêr±OÜ®I«” {˜qjâäÇýëù=Ë°Ô~’r¾Å´ÓÛ¹÷’Nä% ¿┼
m4÷ vMy|ªô»dò5nË ¦\!;Ü^ µœy÷ûSÐ©},,&®#aRä'§ É-÷ëkæ
òÉ5seµ÷┬‘W¥Û'’™d cÒí“€ ‼-¾Ù|À
Gm@ (‡w¨ ◀┼6³
êóè −<ûý©èT» á…┘ÃË 5>u ÄÖ«┼J| ┘┤ÏóÁ3óüñ8‼v= ↔žY A)l‰-V ”‰Ôø %>
/Ÿç‰Ù,ùþ œ ±ÙÐ
g–@┼Jl ┘↕‘:Ÿ |AÁÁ€ MŽÎ€ /Ì€>¶*ƒ* ”Ø

ˆ880
W£ûÝLyô/Ìє ”Ø¶, ┐ M€ *øƒƒ*, ›↔ vy/†-@
A
AÛ↔┘¦À¤┼┼pp@ ┼0^ _™ |)–¤┼
AÛ↔┘¦À¤┼┼pp@ ┼0^ vyd ð¤↕ @ |)lwÂ jA ñ

┘ └à
g–@
|
A)┘
Ä└┴┼ ┘┤€ ↔žY |)┘§À
A)┘
Ä└À ┘┤€ ↔žY |)┘¥²
Â J| ñ

ð └à
g–@
|)┘
ÂŸ
Ä└À ┘┤€ ↔žY |)ð§À
|)┘
Ä└À ┘┤€ ↔žY |)ð¥²
ÂŸ
| ñ

l€ @Áx Ùå
Â–È ð§ÂŸ |AÁü3ñ

l └à
g–@
[*A ð¥² |AÁü3ñ

l └à
g–@
[*| ð¥²¤┼ ┼p□

SISA dari file ini di hapus