

Corso: Algoritmi e strutture dati
Studente: Irene M. Gironacci
Titolo progetto: Huffman code
Anno: primo anno di laurea triennale

Breve descrizione del problema trattato

La codifica di Huffman usa un metodo specifico per scegliere la rappresentazione di ciascun simbolo, risultando in un codice senza prefissi (cioè in cui nessuna stringa binaria di nessun simbolo è il prefisso della stringa binaria di nessun altro simbolo) che esprime il carattere più frequente nella maniera più breve possibile.

Il progetto mostra, usando il metodo di Huffman, la possibilità di comprimere stringhe di testo, avvalendosi di funzioni per la costruzione dell'albero, per la codifica e la decodifica.

Descrizione funzionale del programma realizzato e delle eventuali interazioni con l'utente (dati di input che devono essere forniti, risultati attesi, ...)

Il programma è in grado di eseguire le seguenti funzioni, divise per menù:

- visualizzazione codici ricorrenze: visualizza tutti i codici delle ricorrenze delle lettere dell' alfabeto inglese (26 lettere): ad ogni lettera dell'alfabeto è associato un codice tramite una corrispondenza biunivoca (in modo da evitare che un codice sia prefisso di un altro). Anche allo spazio è assegnato un codice particolare (per consentire poi l'immissione di una stringa per la codifica – nella seconda opzione del menu – contenente spazi).
- immissione stringa da input: richiede di immettere una stringa da input (eventualmente contenente spazi), successivamente la codifica in una stringa di bit secondo il codice di huffman (quindi avente un numero di bit compresso) e mostra in output il risultato (ed la corrispondente differenza di dimensioni in bit). Infine esegue un controllo della korrettezza della codifica: prende la stringa in bit e la decodifica nella stringa originale (secondo le korrispondenze alfabeto – codici), se le due stringhe sono uguali la codifica è stata eseguita con successo.
- lettura stringa di bit da file: dopo aver richiesto il nome del file sorgente (contenente una stringa di bit generata secondo un alfabeto prefissato e non calcolato a run time - a differenza di quello delle prime due funzioni), e il nome del file contenente il dato alfabeto (e i relativi codici prefissati anch'essi), passa alla codifica della stringa di bit, restituendo in output uno schema con sottostringhe di bit e relative lettere associate.
n.b.: Il sistema di decodifica si avvale della costruzione di un albero di huffman grazie al file contenente i codici.
- visualizzazione albero ricorrenze: una volta richiesto il nome del file contenente alfabeto e i rispettivi codici (prestabiliti) viene costruito e visualizzato l'albero delle ricorrenze prodotto.
- uscita dal programma: consente l'immediata uscita dal programma.

Descrizione della struttura del programma realizzato: principali classi e strutture dati utilizzate, principali sottoprogrammi (funzione svolta e parametri passati), ecc.

La struttura del programma è costituita dalle seguenti strutture dati: classi, vettori, strutture di supporto (costanti, interi, stringhe etc.)

- classi principali: **Huffman class**
contenente nella parte privata la classe nodo dell'albero (*class node*, con puntatori ai figli *sx* e *dx*, e relativa informazione associata), e la radice dell'albero (definita come nodo). Nella parte pubblica invece sono presenti delle funzioni di base, quali per esempio la costruzione dell'albero elle

ricorrenze (*build_decoding_tree function*), funzioni di gestione dell'albero tra cui l'inserimento di un nodo (*insert function*), e la decodifica di una stringa (*decode function*), e altre funzioni per l'i/o (*write function*, *print_tree function*, *display_decoding_tree function*).

Hufftree class

classe template contenente le funzioni encode e decode. Permette la costruzione dell'albero, della mappa delle frequenze, la codificazione (tramite funzione *encoding*) e decodificazione (tramite funzione *decoding*).

- vettori principali:

encode vector

vettore utilizzato per codificare una stringa: presa in input una stringa in caratteri (caratteri corrispondenti alle lettere dell'alfabeto), viene memorizzata in tale vettore e, tramite apposite funzioni viene codificata nella corrispondente stringa in bit, compressa (secondo la corrispondenza biunivoca tra alfabeto e codici calcolati a runtime).

decode vector

vettore utilizzato per decodificare una stringa: presa in input una stringa di bit, e memorizzata in tale vettore, tramite apposite funzioni viene decodificata nella corrispondente stringa in caratteri (secondo la corrispondenza biunivoca tra alfabeto e codici calcolati a runtime).

encode_map vector

vettore utilizzato per la memorizzazione delle corrispondenze biunivoche tra alfabeto e relativi codici.

- strutture di supporto:

Es: contatori, variabili booleane, costanti, etc.

Eventuale (breve) descrizione dettagliata di alcuni sottoprogrammi particolarmente significativi

- Insert function: muovendosi nel cammino dalla radice dell'albero, tramite un set di confronti sul valore del bit (0 o 1) trova la prima posizione libera di una foglia dell'albero. Se nel cammino vengono riscontrati caratteri diversi da 0 e 1, viene visualizzato apposito messaggio di errore (*illegal character in code*). Una volta trovata la posizione libera viene ivi memorizzato il dato da inserire (passato come parametro insieme al codice utilizzato per la costruzione dell'albero).
- encoding function: calcola la frequenza definita nella mappa frequency, il valore in bit del carattere. Inoltre consente anche di mettere in ordine crescente a seconda del valore del carattere. Dopo l'encoding viene anche calcolato l'albero in base ai valori delle frequenze.
- decoding function: calcola l'albero e la mappa dell'input definito (da utente) e fa il decoding in carattere ascii tramite l'uso di un vettore ausiliario di booleani.
n.b.: Encoding e Decoding usano lo stesso codice per il calcolo della mappa e per il calcolo dell'albero in base ai valori delle frequenze.
- decode and build_decoding_tree functions: Il programma consente inoltre di fare un decode, alternativo al precedente, utilizzato poi nella 3° opzione. Invocata successivamente alla costruzione dell'albero con le relazioni bit-lettere predefinite (contenute nel file sorgente *code.txt*), tramite la funzione *build_decoding_tree*, la funzione *decode* legge da un file (predefinito) una sequenza di bit, crea l'albero in base ad un confronto di bit (0 o 1) e crea un file di output in cui scrive i valori riconosciuti (delle corrispondenze bit-lettere) e i relativi bit (relazione anche visualizzata in output).

Esempi e casi di prova (dati di input e risultati prodotti)

- input e relative dinamiche run-time:

Exit [0° opzione]:

selezione opzione: 0 → "- produced by.. *3DstyleMaster* - premere un tasto per continuare . . ."

Show code [1° opzione]:

selezione opzione: 1 → *output generato*: codetable for encoding/decoding

A:	1101
B:	1110001
C:	01010
D:	10100
E:	001
F:	00000
G:	101010
H:	01011
I:	1100
J:	1110000111
K:	11100000
L:	11101
M:	00010
N:	1011
O:	1001
P:	111001
Q:	11100001101
R:	0100
S:	1000
T:	1111
U:	00011
V:	000010
W:	000011
X:	111000010
Y:	101011
Z:	11100001100
Space	011
:	

Encode and decode [2° opzione]:

1. input: "hi world" → *dimensione stringa (in bits): 64, stringa codificata:*
010111100011000011100101001110110100,
dimensione (in bits) della stringa codificata: 36,
stringa di bit decodificata: hi world.
2. input: "bye" → *dimensione stringa(in bits): 24, stringa codificata: 1110001101011001, dimensione (in bits) della stringa codificata: 16, stringa di bit decodificata: bye.*

Decoding from file [3° opzione]:

1. input: "msg.txt" → *code in file: 11110111011001011011101, after decoding:*

111101 ← P
1101 ← A
1001 ← R
01101 ← M
1101 ← A

2. input: "test.txt" → code in file:
00001001000010110000101110100010101010
00, after decoding:

000 ← T
0100 ← H
1000 ← I
0101 ← S

1000 ← I
0101 ← S

1101 ← A

000 ← T
101 ← E
0101 ← S
000 ← T

3. input: "test2.txt" → code in file:
00001001000010110000101110100010101010
003, after decoding:

000 ← T
0100 ← H
1000 ← I
0101 ← S

1000 ← I
0101 ← S

1101 ← A

000 ← T
101 ← E
0101 ← S
000 ← T

3- illegal bit: 3 – ignored.

Huffman tree [4° opzione]:

1. input: "code.txt" → contenuto del file:

A: 1101
B: 001101
C: 01100
D: 0010
E: 101
F: 111100
G: 001110
H: 0100
I: 1000
J: 11111100
K: 11111101
L: 01111
M: 01101
N: 1100
O: 1110
P: 111101

Q: 111111100
R: 1001
S: 0101
T: 000
U: 01110
V: 001100
W: 001111
X: 111111101
Y: 111110
Z: 11111111

output -> visualizzato in figura 0.5.

Immagini guida

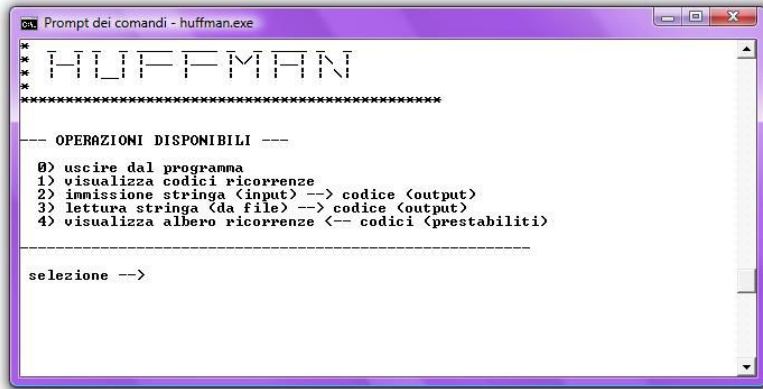


Figura 0.1

Interfaccia grafica e schermata principale delle varie opzioni disponibili nel programma

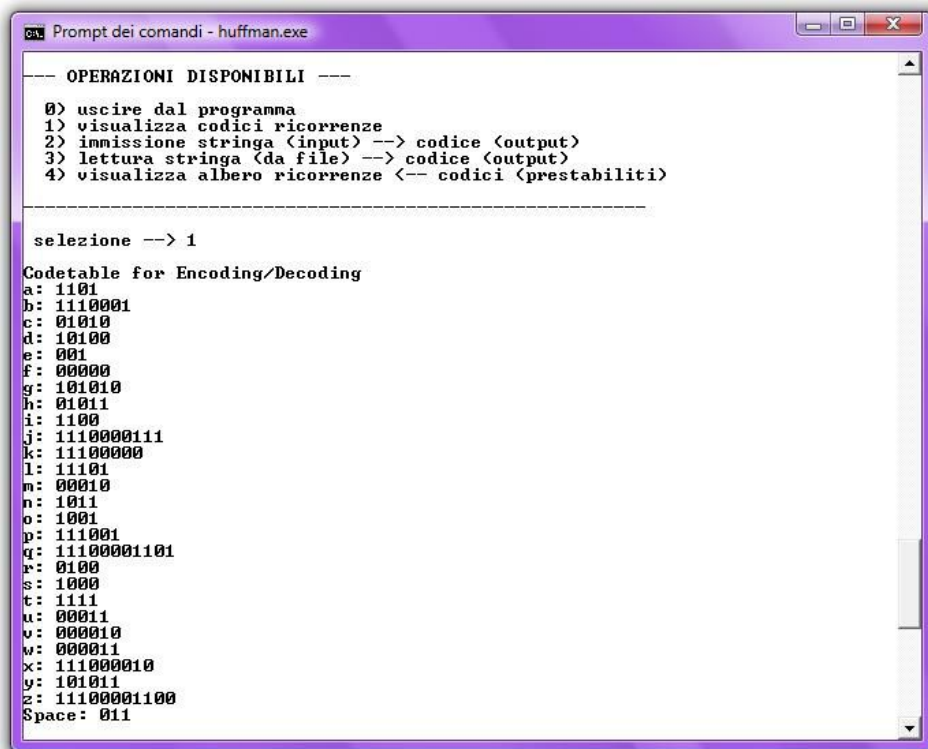


Figura 0.2

Dinamica run-time della 1° opzione: viene visualizzato l'elenco delle corrispondenze biunivoche tra alfabeto e codici relativi (calcolati secondi precise statistiche di ricorrenza di ciascuna lettera nella lingua inglese, si veda Alegato 0.6).

```

Prompt dei comandi - huffman.exe

--- OPERAZIONI DISPONIBILI ---
0) uscire dal programma
1) visualizza codici ricorrenze
2) immissione stringa <input> --> codice <output>
3) lettura stringa <da file> --> codice <output>
4) visualizza albero ricorrenze <-- codici <prestabiliti>

-----
selezione --> 2

--- INPUT AND DECODE MSG ACTIUVETED -----
please, enter your string: hello everybody
string to manipulate is :   hello everybody

size of input string <in bits>: 120
string encoded is :   0101100111101111011001011001000010001010010101111100011001
10100101011

size of encoded string <in bits>: 69

Let's decode the encoded string back ---
The decoded string is :   hello everybody

```

Figura 0.3

La figura mostra l'esecuzione della 2° opzione: immessa la stringa "hello everybody", questa viene codificata e successivamente decodificata.

```

Prompt dei comandi - huffman.exe

--- OPERAZIONI DISPONIBILI ---
0) uscire dal programma
1) visualizza codici ricorrenze
2) immissione stringa <input> --> codice <output>
3) lettura stringa <da file> --> codice <output>
4) visualizza albero ricorrenze <-- codici <prestabiliti>

-----
selezione --> 3

--- LOOKING FOR THE MESSAGE -----
name of message file: msg.txt
now enter name of code file: code.txt

--- DECODING MESSAGE ACTIUVETED -----
111101 <-- P
1101 <-- A
1001 <-- R
01101 <-- M
1101 <-- A

```

Figura 0.4

Esecuzione della 3° opzione: dato il file msg.txt, viene presa la stringa di bit contenuta in tale file (stringa deve rispettare la codifica) , e viene associata a ciascuna sottostringa di bit riconosciuta la lettera corrispondente.

```

Prompt dei comandi - huffman.exe

here is the huffman decoding tree:

                                     *
                                     Z
                                     *
                                     X
                                     *
                                     Q
                                     *
                                     K
                                     *
                                     J
                                     *
                                     Y
                                     *
                                     P
                                     *
                                     F
                                     *
                                     O
                                     *
                                     A
                                     *
                                     N
                                     *
                                     E
                                     *
                                     R
                                     *
                                     I
                                     *
                                     L
                                     *
                                     U
                                     *
                                     M
                                     *
                                     C
                                     *
                                     S
                                     *
                                     H
                                     *
                                     W
                                     *
                                     G
                                     *
                                     B
                                     *
                                     U
                                     *
                                     D
                                     *
                                     T

```

Figura 0.5

Mostra l'albero delle ricorrenze secondo l'alfabeto prefissato (si veda sopra, in *Huffman tree*, 4° opzione)

Analisi delle frequenze

[From Wikipedia, the free encyclopedia]

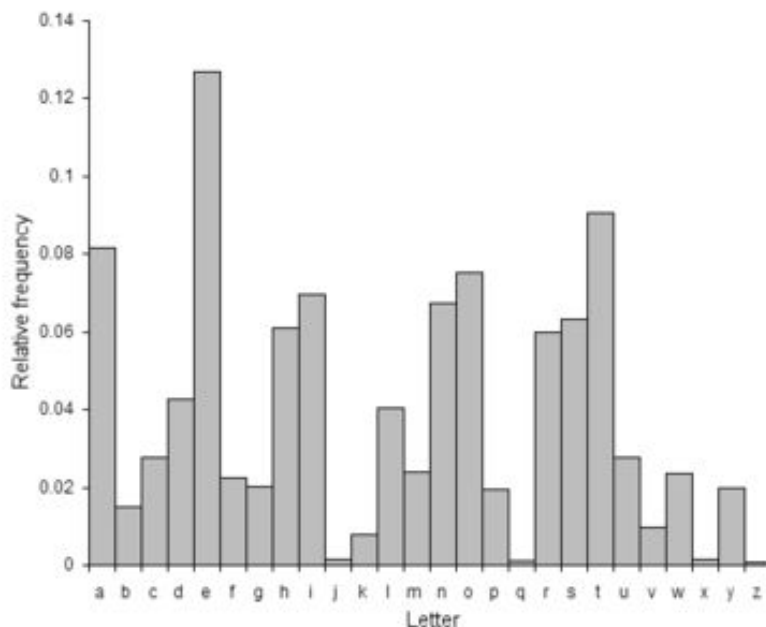
Le indagini quantitative sui testi si servono spesso di qualche forma di **analisi delle frequenze**.

Possono essere interessanti le analisi delle frequenze di caratteri, di parole, di gruppi di parole che si possono assegnare a lemmi o significati definiti; queste analisi possono riguardare un singolo testo (da un frammento epigrafico, a un'opera come la Divina commedia), un intero *corpus* letterario o un opportuno campione di un linguaggio specialistico o di un'intera lingua. In particolare un primo metodo che si adotta in attività di crittanalisi si basa sul fatto che in ogni lingua la frequenza di uso di ogni lettera è piuttosto determinata; questo è vero in modo rigoroso solo per testi lunghi, ma spesso testi anche corti hanno frequenze non molto diverse da quelle previste. Si può notare quanto le prime lettere di queste lingue siano presenti in quantità molto maggiore delle altre, ad esempio da un testo in cui un certo simbolo appare oltre il 12% delle volte si può facilmente intuire che quel simbolo corrisponde alla lettera E (in inglese la distanza della E dalle altre lettere è ancora più marcata).

Vediamo come riferimento le frequenze delle lettere più comuni di due lingue:

Italiano		Inglese	
E	11,79	E	12,31
A	11,74	T	9,59
I	11,28	A	8,05
O	9,83	O	7,94
N	6,88	N	7,19

Relative frequencies of letters in the English language



Allegato 0.6

Tratto da wikipedia, contiene informazioni per quanto riguarda lo studio delle frequenze delle lettere dell'alfabeto nella lingua inglese (frequenze poi usate per calcolare le corrispondenze nelle opzioni 2° e 3°).



http://it.wikipedia.org/wiki/Analisi_delle_frequenze





Relative frequencies of letters in text.

English letter frequencies:[\[1\]](#)

Italian letter frequencies:[\[1\]](#)

Letter 	Frequency 
a	8.167%
b	1.492%
c	2.782%
d	4.253%
e	12.702%
f	2.228%
g	2.015%
h	6.094%
i	6.966%
j	0.153%
k	0.772%
l	4.025%
m	2.406%
n	6.749%
o	7.507%
p	1.929%
q	0.095%
r	5.987%
s	6.327%
t	9.056%
u	2.758%
v	0.978%
w	2.360%
x	0.150%
y	1.974%
z	0.074%

Lettera 	Frequenza 
a	11.74%
b	0.92%
c	4.5%
d	3.73%
e	11.79%
f	0.95%
g	1.64%
h	1.54%
i	11.28%
l	6.51%
m	2.51%
n	6.88%
o	9.83%
p	3.05%
q	0.51%
r	6.37%
s	4.98%
t	5.62%
u	3.01%
v	2.10%
z	0.49%

© Irene M. Gironacci



Quest'opera di Irene M. Gironacci è distribuita con Licenza [Creative Commons](#)
[Attribuzione - Non commerciale 4.0 Internazionale](#).