INFORMATICA MUSICALE

UNIVERSITA' DEGLI STUDI DI CATANIA
DIPARTIMENTO DI MATEMATICA E INFORMATICA
LAUREA TRIENNALE IN INFORMATICA
A.A. 2018/19
Prof. Filippo L.M. Milotta

ID PROGETTO: 1D

TITOLO PROGETTO: Huffman+Morse Encoding

AUTORE 1: Pennisi Matteo

Indice

1. Obiettivi del progetto

L'obbiettivo principale è applicare Huffman su una stringa di soli numeri (0-9) e successivamente usare la codifica binaria (0-1) prodotta per generare audio mediante codice Morse. Per raggiungere tale obbiettivo sarebbe sufficiente scrivere il codice in Python e poi eseguirlo, ottenendo direttamente il risultato finale. Un altro obbiettivo molto importante è anche la chiarezza e la riproducibilità dell'esperimento. Per tale ragione l'intero codice è stato scritto in un Notebook Jupyter. La piattaforma Jupyter Notebook integra perfettamente la shell di Python, assieme a strumenti per la gestione di documenti (Rich Text Format). La piattaforma supporta pertanto, molteplici tipologie di contenuti come: testi semplici (Markdown), esecuzione di blocchi di codice (Python), funzioni matematiche, grafici, immagini e altri contenuti multimediali visualizzabili direttamente su Browser.

2. Metodo Proposto

A) Analizzare la frequenza dei caratteri in una stringa casuale

Genero una stringa di soli numeri casuale con una lunghezza arbitraria di 50 caratteri. È importante non generare una stringa troppo lunga perché al crescere della lunghezza la frequenza di ripetizione diventerà pressoché uguale per ogni carattere. Il nostro obbiettivo è, invece, emulare la comunicazione reale in cui vi è un sottoinsieme di caratteri che si ripetono più di altri.

Esempio

```
INPUT = 02591044212997205022201227012147049425841134757152


Caratteri dal meno frequente al più frequente

6 --> 0.0%
3 --> 2.0%
8 --> 2.0%
9 --> 8.0%
5 --> 10.0%
7 --> 10.0%
0 --> 14.0%
4 --> 14.0%
1 --> 16.0%
2 --> 24.0%
```

B) Generare un dizionario di Huffman basandosi sulla stringa precedentemente realizzata

La compressione con Huffman inizia creando una lista dei caratteri in base alla frequenza (Punto A). Successivamente si crea per passaggi successivi un albero costituito da una serie di ramificazioni binarie.

L'algoritmo di compressione Huffman associa agli elementi più rari una codifica più lunga mentre a quelli più frequenti una codifica più breve. Una volta creato l'albero, bisogna associare ad ogni nodo un bit. È possibile associare lo 0 a tutti i nodi di sinistra e 1 a tutti quelli di destra. È possibile anche fare il contrario senza compromettere il risultato finale. La generazione del dizionario viene effettuata grazie alla libreria python *huffman* che riceve in ingresso la stringa casuale del punto A.

Esempio

```
0  |  110
2  |  01
5  |  001
9  |  1001
1  |  111
4  |  101
7  |  000
8  |  10001
3  |  10000
```

C) Codificare una stringa con il dizionario precedentemente trovato

Una volta generato il dizionario è possibile applicarlo per codificare una stringa. Il dizionario è stato salvato come *dizionario python*, un tipo di dato composto da coppie key-value. Iterando i caratteri della stringa è possibile usare il carattere come *key* e quindi ottenere il *value* corrispondente, cioè la sua codifica Huffman.

```
for character in test_string:
    char_encoded=codifica[character]
```

D) Generare la codifica sonora della stringa utilizzando il codice Morse

Per la codifica del suono ho definito una funzione che prende in ingresso tre parametri: frequenza, durata, ampiezza. La frequenza sarà fissa a 440 Hz per tutti i suoni. L'ampiezza è normalmente pari a 2, mentre è pari a 0 per generare le pause. Per quanto riguarda la durata dei suoni, ho stabilito un'unita di tempo arbitraria che, moltiplicata ai valori della tabella sottostante, costituirà la durata effettiva.

Tipo di informazione	*Durata*
Codifica 0	1 Unità di tempo
Codifica 1	3 Unità di tempo
Pausa tra ogni 0 o 1	1 Unità di tempo
Pausa tra ogni carattere	3 Unità di tempo

L'audio in python è gestito mediante un array inizialmente vuoto *output=[]*. Ogni suono generato è a sua volta un array e quindi per mettere tutto insieme basterà concatenare di volta in volta i nuovi suoni all'array di output. Per far ciò si può usare la funzione *concatenate* della libreria *numpy*.

```
output=np.concatenate((output,mytone))
```

3. Risultati Ottenuti

Eseguendo interamente il notebook è possibile ascoltare alla fine il risultato di tutto il processo appena descritto. La visualizzazione del player all'interno del Notebook è possibile grazie al modulo *Audio* e *display* di *IPython*.

```
display(Audio(output, rate=rate, autoplay=False))
```

```
Codifica di: 24545347890
```