Progetto Arithmetic Coding - LZ77

Alice Mariotti Nesurini - Enrico Bottani, I2AC

1 Struttura codice

1.1 Utilizzo

Compilazione

```
gcc src/main.c src/lz77/lz77.c src/lz77/lz77.h src/tools/file/file.c
src/tools/file/file.h src/tools/charcb/cb/cibuff.c src/tools/charcb/cb/cibuff.h
src/tools/cmdef.h src/tools/bytestr/bytestr.c src/tools/bytestr/bytestr.h
src/tools/bitfile/bitfilewriter.h src/tools/bitfile/bitfilewriter.c
src/tools/bitfile/bitfilereader.c src/tools/bitfile/bitfilereader.h
src/lz77/dec/dec.c src/lz77/dec/dec.h src/lz77/cmp/enc.c
src/lz77/cmp/enc.h src/lz77/dec/dlog.c src/lz77/cmp/clog.c src/lz77/cmp/clog.h
src/lz77/dec/dlog.h src/tools/utility.c src/tools/utility.h src/tools/kmp/kmp.c
src/tools/kmp/kmp.h src/tools/kmp/kmplog.c src/tools/kmp/kmplog.h
src/tools/scharcb/scharcb.c src/tools/scharcb/scharcb.h src/ac/ac_encoding.h
src/ac/ac_encoding.c src/ac/element.c src/ac/element.h src/ac/ac_decoding.c
src/ac/ac_decoding.h
```

Compressione
LZ77AC -c ../../LZ77AC/test/alice

Decompressione
LZ77AC -d ../../LZ77AC/test/alice.press

2 Tempi compressione/decompressione

2.1 Arithmetic Coding

Nome	Size originale	Size compresso	Compressione [s]	Decompressione [s]
Alice.txt	163.8 [kB]	95 [kB]	1.576	1.351
Immagine.tiff	3.4 [MB]	3.2 [MB]	66.32	31.51
32k_ff	32.8 [kB]	1 [kB]	0.11	0.091
32k_random	32.8 [kB]	33.8 [kB]	0.592	0.365
ff_ff_ff	3 [B]	1 [kB]	0.001	0.000

2.2 LZ77

Nome	Size originale	Size compresso	Compressione [s]	Decompressione [s]
Alice.txt	163.8 [kB]	122.7 [kB]	1.241	0.001
Immagine.tiff	3.4 [MB]	4.9 [MB]	45.92	0.190
32k_ff	32.8 [kB]	6.1 [kB]	0.001	0.000
32k_random	32.8 [kB]	48.4 [kB]	0.463	0.001
ff_ff_ff	3 [B]	4 [B]	0.000	0.000

2.3 Algoritmi combinati

Nome	Size originale	Size compresso	Compressione [s]	Decompressione [s]
Alice.txt	163.8 [kB]	121.8 [kB]	3.209	1.270
Immagine.tiff	3.4 [MB]	4.8 [MB]	124.0	49.71
32k_ff	32.8 [kB]	1.4 [kB]	0.070	0.042
32k_random	32.8 [kB]	44 [kB]	1.263	0.510
ff_ff_ff	3 [B]	1 [kB]	0.000	0.000

3 Problemi

Problemi sulla memoria, per risolvere questi problemi, che spesso erano allocazioni non ripulite correttamente, è stato molto utile valgrind (su linux) per debug.

Il passaggio del numero totale di caratteri codificati e dell'array delle frequenze (per l'arithmetic coding) è stata un'aggiunta relativamente recente, e con questa aggiunta la dimensione dei file compressi aumenta notevolmente, ma visto che non è stato sviluppato un algoritmo ac addattivo, nel file di output questi dati devono essere presenti, essendo che il compressore non ha modo di avere queste informazioni in altri modi.

Per migliorare i tempi di compressione abbiamo usato Instrument time profiling (su mac) e callgrind con KChacegrind su linux. Ci è stato utile per trovare quali funzioni impiegavano più tempo e applicare delle migliorie a tali funzioni e passaggi.

4 Sviluppi futuri

4.1 Consistenza dati

Al momento il nostro programma comprime prima con l'algoritmo LZ77 ed in seguito applica l'Arithmetic Coding sul risultato della prima codifica. Questo viene fatto leggendo il file compresso da LZ77 carattere per carattere incurante del significato dei bit processati. Una codifica sicuramente più efficiente avrebbe creato tabelle di frequenza separate per il dictionary, l'offset e il nuovo carattere mantenendo inoltre la consistenza dei dati passati. L'Arithmetic Coding leggendo infatti ogni 8 bit senza alcun tipo di parsing o cura del vaore di quest'ultima ha come come risultato di leggere caratteri molto casuali e quindi difficili da comprimere per un algoritmo entropico. Per motivi di tempo non è stato possibile implementare questa importante feature.