Progetto per "Algoritmi e strutture dati"
textitBWT + Huffman canonico

Bonesana Claudio, Koefer Nathan

22 gennaio 2012

1 Suddivisione dei compiti

Bonesana Claudio: *Mi sono occupato dell'algoritmo di Huffman canonico e della sua implementazione in c. Ho aiutato il mio compagno nell'implementazione dell'algoritmo Move to Front (MTF) nelle fasi finali ed ho eseguito l'unione di tutti gli algoritmi in un programma funzionante*

Koefer Nathan: *Mi sono occupato della Trasformata di Burrows-Wheeler (BWT) e della sua implementazione in c. Ho iniziato il lavoro sul Move to Front e ho completato con successo tutte le fasi di test per gli algoritmi BWT e MTF*

2 Descrizione del programma

Il codice scritto in linguaggio c è suddiviso per algoritmo. Il file principale è *src/main.c*. Mediante una serie di controlli, il programma lancia in successione i tre algoritmi: BWT - MTF - Huffman per la compressione, Huffman . MTF - BWT per la decompressione.

L'algoritmo BWT utilizza i sorgenti contenuti nella cartella *src/bwt/*. Il file *src/bwt/config.h* può essere modificato per indicare il numero di byte utilizzati come blocco per l'esecuzione dell'algoritmo. Da notare che un numero elevato influisce negativamente sul tempo di compressione: valori superiori a 10'000 sono da evitare.
Una nota sull'implementazione di un sort più efficace per il BWT. Purtroppo gli algoritmi di ordinamento a nostra disposizione riescono ad ordinare i caratteri in una stringa ma non le stringe in un array di stringhe. Per quanto efficiente possa essere un sort, la comparazione di due stringhe risulta comunque un'operazione lunga e pesante da eseguire. L?implementazione di un *quick sort* al posto del sort attuale è stata abbandonata in seguito agli scarsi risultati ottenuti.
L'utilizzo di un algoritmo come la *conta distribuita*, invece, obbliga a conoscere a priori il numero di chiavi esistenti. Purtroppo, con stringhe lunghe, questo numero diventa ingestibile rendendo inutile usare tale algoritmo.

L'algoritmo Move To Front utilizza i sorgenti contenuti nella cartella */src/mtf/*. Questo algoritmo implementa una lista dinamica e non presenta alcun file di configurazione.

L'algoritmo di Huffman canonico utilizza quattro file contenuti nella cartella */src/* e tutti quelli contenuti nella cartella */src/lib/*. Il file di configurazione */src/lib/config.h* contenuto in */src/lib/* non presenta particolari campi modificabili. Viene utilizzato unicamente per delle definizioni globali utilizzate in più parti del codice. I file che iniziano con *f_* sono dei file che contengono funzioni particolari richiamate dal codice principale. Il codice principale per la codifica canonica è contenuto nel file */src/lib/f_codifica.c*

mentre gli algoritmi per la codifica e decodifica sono contenuti rispettivamente nei file */src/compressore.c* e */src/decompressore.c*.
Un appunto sul file */src/f_array.c*. Le funzioni **shift_right** e **shift_right** sono state ottimizzate a tal punto da diventare di difficile lettura per un umano. Sotto le rispettive funzioni si trova commentato il codice originario non ottimizzato.

Il programma da linea di comando viene lanciato con il codice *./compressore −parametro input output*. I parametri da linea di comando sono **-c** per la compressione, **-d** per la decompressione e **-m** per il manuale.

3 Procedure di test

Gli algoritmi sono stati testati dai rispettivi autori separatamente con file di diversi formati e dimensioni. L'obiettivo primario è stato la creazione di un codice funzionante e quindi la corretta compressione e decompressione di un file di test. Una volta uniti gli algoritmi l'operazione di test è stata ripetuta.

Per eseguire i test è stato utilizzato il comando *time* di Unix su linea di comando prima del programma o dell'algoritmo. Per verificare l'esattezza dei file, il file originale è stato comparato mediante il comando $diff$ con il file elaborato: in caso di esito positivo tale comando non fornisce commenti o errori. In alternativa è stato utilizzato il comando $hexdump - C$ sui file per capire dove si trovava l'errore e per vedere se l'andamento della compressione è andato a buon fine secondo le aspettative.

Per verificare la velocità di alcuni codici, si è proceduto con l'analisi dettagliata delle singole funzioni all'interno di un programma. Tali modifiche sono state tolte nell'ultima versione dei codici in quanto non più utili.