Algoritmi di compressione

Burrows-Wheeler Transform con Huffman Canonico

Il ritorno

Bonesana Claudio

Nathan Koefer

SUPSI DTI

19.01.2012

Introduzione

- Burrows-Wheeler Transform
  - Codifica in sequenze di simboli uguali
- Algoritmo di Huffman Canonico
  - Vero algoritmo di compressione
- Move to Front
  - Algoritmo di codifica

BURROWS-WHEELER TRANSFORM

- Problemi
  - Utilizza una matrice quadrata di ordine N
    - (dove N è il numero di byte letti)
  - Occorre ordinarla
    - Molte operazioni da compiere => lento
  - Occupa molta memoria
    - Occorre lavorare a blocchi => efficienza ridotta
    - Occorre lavorare con tabelle virtuali
  - Non è stato facile da implementare
    - Soluzioni contorte
    - Pesantezza del codice
    - Più sistemi per implementarlo

Burrows-Wheeler Transform

- Implementazione
  - Codifica
    - Tabella virtuale creata con array di puntatori
    - Creazione di un compare ad hoc
    - Creazione di un sort stabile e con un caso peggiore decente
      - (simile a un Bubble Sort)
  - Decodifica
    - Sfruttato un algoritmo basato sulle corrispondenze
      - (simile a Conta Distribuita)
    - Lettura a blocchi misti
    - Occorre conoscere un indice
  - Prepara l'input di MTF

Huffman Canonico

- Problemi incontrati
  - Lentezza nella decompressione
  - Occorre eseguire numerose operazioni sui bit
  - Occorre ottimizzare le operazioni sui bit
  - La compressione dipende notevolmente dal tipo di file
    - (varia dal 20% al 90% del file originale)

Huffman Canonico

- Implementazione
  - Operazioni bit-a-bit su **array di char** (byte)
  - **Shift left** e **right** ottimizzati per lavorare con grandi quantità di byte
  - Legge l'output di MTF

Huffman Canonico e BWT

- Possibili miglioramenti
  - Raffinare il codice
  - Huffman:
    - Implementare una ricerca binaria per la decompressione
    - Migliorare dei cicli *for*
  - BWT:
    - Trovare un algoritmo di ordinamento efficiente
    - Trovare una codifica veloce
    - Ampliare i blocchi

Move To Front

- L'anello mancante per unire il BWT a Huffman
- Sfrutta le sequenze di BWT inserendo molti zeri che vengono sfruttati dall'algoritmo di Huffman per creare una codifica migliore

Move To Front

- Implementazione
  - Uso di una **lista dinamica** (inserimento in testa)
  - Lettura di byte
  - Lettura dell'output di BWT
  - Scrittura dell'input di Huffman
  - Velocità lineare

Dimostrazione pratica