FILEDISTANCE

Greta Sorritelli [MAT: 104952]

La *distanza di edit* è una funzione che consente di verificare quanto due stringhe (o sequenze di byte) siano *lontane* una dall'altra. Questa distanza viene calcolata sulla base del numero di operazioni necessarie a trasformare **una data sequenza di byte nell'altra**.
Le operazioni sono:

- Aggiungere un byte;
- Eliminare un byte;
- Modificare un byte.

Per calcolare la distanza di edit è stato utilizzato l'algoritmo di Levenshtein.

La distanza di Levenshtein tra due stringhe A e B è il numero minimo di modifiche elementari che consentono di trasformare la A nella B. Per modifica elementare si intende:

- la cancellazione di un carattere
- la sostituzione di un carattere con un altro
- l'inserimento di un carattere

Un algoritmo usato comunemente per calcolare la distanza di Levenshtein richiede l'uso di una matrice di $(n + 1) \times (m + 1)$, dove n e m rappresentano le lunghezze delle due stringhe.

All'interno di tale matrice poi vengono aggiunti dei numeri (all'interno delle caselle date da riga e colonna) per ogni riga scorrendo da sinistra verso destra. Ognuno tra tali numeri aggiunti corrisponde al valore minore tra il numero sopra ([-1][ ], cioè DEL), quello in alto a sinistra([-1][-1], cioè SET) e quello a sinistra ([ ][-1], cioè ADD) rispetto quello preso in considerazione *più uno*. Questo avviene nei casi in cui gli elementi dei due file sono diversi. Nel caso in cui questi siano uguali non si aggiunge uno (procedimento descritto dalla tabella sottostante). Ripartendo dal basso (ultima posizione della matrice, che indica la distanza totale di edit), inoltre, si ricostruiscono le istruzioni da poter salvare in un file.

La distanza di Levenshtein ha alcuni semplici limiti superiori ed inferiori:

- è almeno la differenza fra le lunghezze delle due stringhe;
- è 0 se e solo se le due stringhe sono identiche;
- se le lunghezze delle due stringhe sono uguali, la distanza non supera la lunghezza delle stringhe;
- il limite superiore è pari alla lunghezza della stringa più lunga.

		Posizione X (del file2)
	SET[-1][-1]	DEL[-1][ ]
Posizione Y (del file1)	ADD[ ][-1]	min(SET, ADD, DEL) + 1 (se X != Y) + 0 (se X == Y)

Funzionalità

1. Calcolo della distanza tra due file:

filedistance distance file1 file2

dati due file *f1* e *f2*, il programma produce in output la distanza tra *f1* e *f2* ed il tempo di calcolo in secondi. Il calcolo viene effettuato tramite l'algoritmo ottimizzato di Levenshtein (non utilizza la matrice ma prende in calcolo solo due righe di quest'ultima).

filedistance distance file1 file2 output

Il programma genera la minima sequenza di operazioni necessarie per trasformare *f1* in f2 se viene specificato un file di output, in cui le istruzioni vengono scritte in ***binario*** in ordine ***crescente*** di indice della posizione. Tale sequenza viene rappresentata attraverso una sequenza di istruzioni:

a. ADD*nb*: n è un intero senza segno (unsigned int) rappresentato con 4 byte (byte più significativo all'inizio) mentre *b* è un singolo byte. Tale operazione indica che viene aggiunto il byte *b* nella posizione *n*.
b. DEL*n*: n è un intero senza segno (unsigned int). Tale operazione indica che viene eliminato il byte nella posizione *n*.
c. SET*nb*: n è un intero senza segno (unsigned int) rappresentato con 4 byte mentre *b* è un singolo byte. Tale operazione indica che il valore del byte in posizione *n* viene impostato a *b.*

2. Applicazione delle modifiche ad un file:

filedistance apply inputfile filem outputfile

dato un file *inputfile* da elaborare ed un *filem* contenente la lista di *modifiche da apportare* scritte in binario, produce in output un file *outputfile* ottenuto da *inputfile* applicando le modifiche indicate in *filem.*
Tali istruzioni vengono lette dal filem e salvate in una struttura (vengono memorizzati il tipo dell'operazione, tramite una stringa, la posizione dell'indice del file, tramite un intero, e il byte da aggiungere o settare, memorizzato come carattere).

3. Ricerca di file:

filedistance search inputfile dir

dato un file *inputfile* ed una directory *dir* viene restituito l'insieme dei file a distanza *minima* da *inputfile* contenuti in *dir* (e nelle sue sottodirectory). Il path assoluto di ogni file viene presentato in una riga dello *standard output.*

filedistance searchall inputfile dir limit

Vengono restituiti in output tutti i file contenuti in *dir* (e nelle sue sottodirectory) che hanno una distanza da *inputfile* minore o uguale di *limit* (che è un intero). I file vengono presentati nello standard output in ordine crescente secondo il formato: *distanza pathassoluto.*

Il progetto consiste dei seguenti moduli:

-**distance**: si occupa di calcolare la distanza di edit e salvare le istruzioni in binario in un file di output.

-**apply**: applica le istruzioni contenute in un file su di un altro file in input e ne produce un altro in output.

-**search**: calcola le distanze di edit tra un file in input e i file di una directory, fa uso del modulo distance, scorre tutte le directory e sottodirectory tramite una visita in profondità (la prima volta viene richiamata per calcolare il totale dei file, la seconda per inserire i path nella struttura apposita); per riordinare i path in ordine crescente di distanza viene utilizzato l'algoritmo di ordinamento di InsertionSort.

-**io**: contiene i metodi utili per effettuare la r/w su file in binario.

Strutture dati presenti nel progetto:

- **INSTRUCTIONS:** viene utilizzata per salvare le istruzioni, consiste di tre elementi:

  -*operazione* = tipo di operazione (DEL, ADD o SET) (char *)

  -*n* = indice della posizione nel file dove effettuare l'istruzione (unsigned int)

  -*b* = byte da aggiungere o impostare (char)

- **DISTANCE_PATH:** viene utilizzata per salvare i file presenti in una directory (o sottodirectory) con la relativa distanza da un file, consiste di:

  -*distance* = distanza da un file in ricerca (unsigned int)

  -*path* = percorso assoluto del file (char *)