Michele Benedetti
24 giugno 2020

FILEDISTANCE

Il progetto **filedistance** è stato realizzato in **C** per il corso di laurea **L-31** presso **Unicam**, nell'anno accademico 2019/2020, dallo studente Benedetti Michele, per l'esame di **Sistemi Operativi Laboratorio.**

1.Descrizione Progetto:

Il software in questione permette, grazie all'algoritmo di Levenshtein, di calcolare la distanza di edit fra due stringhe.

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 stringa nell'altra. Le operazioni sono:

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

Sulla base di queste operazioni sarà possibile, attraverso il comando ***apply***, applicare le modifiche al file denominato *Source*, affinché abbia distanza 0 dal file denominato *Target.*

Inoltre, attraverso i comandi ***search*** e ***searchall***, sará possibile, dato un file *inputfile* e una directory *dir*, ricercare e stampare nello standard output, rispettivamente l'insieme dei file contenuti in *dir* (e nelle sue sottodirectory) che hanno distanza minima dal file *inputfile*, mentre nel secondo caso, ricercare e stampare nello standard output la distanza e il path assoluto dei file presenti in *dir* (e nelle sue sottodirectory) che hanno distanza inferiore alla variabile *limit*, in ordine crescente.

2. Strutture Dati

2.1 Struct *Lista* per salvataggio istruzioni

Le struttura dati utilizzata per memorizzare le istruzioni da applicare ad un file per modificarlo, é una struct con al suo interno un puntatore alla struct successiva, chiamata appunto **Lista**, che permette di memorizzare:

- ***type***: Enum che identifica il tipo di modifica da eseguire (enum).
- **pos**: Posizione nella stringa dove applicare l'istruzione (unsigned int).
- **character**: Il carattere da sostituire o da aggiungere nel caso di ADD o SET. (char).
- **next**: Il puntatore alla struct successiva. (struct list *).

```
typedef struct list{
    edit_type type;
    unsigned int pos;
    char character;
    struct list *next;
}Lista;
```

2.2 Struct *RecursiveList* per salvataggio dei path e distanza

La struttura dati utilizzata é la medesima di quella appena descritta, ma in questo caso viene utilizzata per memorizzare i relativi path e la distanza di edit dei file trovati nella directory e nelle sue sottodirectory. Questa struttura prende il nome di **RecursiveList**, visto e considerato che andrà a contenere il risultato di uno Scan ricorsivo di una Directory.

Questa struttura contiene quindi:

- ***path***: Puntatore a char, dove viene salvato il path relativo del file trovato, che verrà poi modi stampato come path assoluto (char *).
- ***distance***: La distanza di quel determinato file, dal file passato come input (int) .
- ***next***: Il puntatore alla struct successiva (struct pathFile*).

```
typedef struct pathFile{
    char *path;
    int distance;
    struct pathFile *next;
}RecursiveList;
```

3. Librerie

3.1 Librerie Standard:

Le librerie standard utilizzate all'interno del codice sono le seguenti:

- **stdio.h** : Per la gestione dell'input/output.
- **stdilb.h** : Per la gestione della memoria, dei processi e altre funzioni generali.
- **string.h** : Per la gestione e la manipolazione delle stringhe
- **dirent.h** : Per la gestione delle directory
- **time.h** : Per la gestione del tempo

3.2 Librerie Personalizzate

Gli Headers file creati sono in totale 6, ed hanno rispettivamente i seguenti compiti:

- **filedistance.h**
- **modfile.h**
- **recursive.h**
- **levenshtein.h**
- **time.h**
- **linked_list.h**

Filedistance.h

Descrizione:

Header file contenente le funzionalità principali del programma quali:

- **void** distance(**char** *file1, **char** *file2);
- **void** distanceOutput(**char** *file1, **char** *file2, **char** *outputfile);
- **void** apply(**char** *inputfile, **char** *filem, **char** *outputfile);
- **void** search(**char** *inputfile, **char** *dir);
- **void** searchAll(**char** *inputfile, **char** *dir, **int** limit);

distance: Funzione che effettua il calcolo della distanza di edit tra file1, file2 e stampa sullo standard output la distanza e il tempo impiegato in secondi.

distanceOutput: Funzione per generare il file .bin con all'interno le istruzioni per la modifica del "file1" affinchè abbia distanza 0 da “file2".

apply: Applica a inputfile le modifiche contenute nel file filem e salva il risultato nel file outputfile.

search: Restituisce in output i file contenuti in dir (e nelle sue sottodirectory) che hanno minima distanza da inputfile. Il path assoluto di ogni file viene presentato in una riga dello standard output.

searchAll: 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).

Modfile.h

Descrizione:

Header file contenente le funzionalità per la lettura e il caricamento in struttura delle istruzioni presenti in un file .bin. Successivamente troviamo le funzioni per applicare le istruzioni alla stringa e scriverla in un file, che viene creato qualora non esistesse.

- **void** modifyFile(**char** *inputfile, **char** *filem, **char** *outputfile);

  Legge le modifiche presenti in "filem.bin" da applicare a "inputfile".Genera il file "outputfile" con le modifiche applicate.

- **Lista** *readFromBinFile(**char** *filem);

  Funzione per leggere le istruzioni da un file .bin e inserirle
  all'interno di una lista. Restituisce la lista di istruzioni

- **void** generateOutputFile(**char** *outputfile, **char** *string);

  Genero (se non esiste) e scrivo la stringa "string"
  nel file "outputfile"

- **char** *addChar(**char** *string, **unsigned int** pos, **char** character);

  Funzione per aggiungere un carattere "character" in una
  determinata posizione "pos" in una stringa "string"

- **char** *delChar(**char** *string, **unsigned int** pos);

  Funzione per cancellare un carattere dalla
  stringa "string" alla posizione "pos"

- **char** *setChar(**char** *string, **unsigned int** pos, **char** character);

  Funzione per cambiare un carattere "character"
  alla posizione "pos" della stringa “string”

Recursive.h

Descrizione:

Header contente la struttura dati per la memorizzazione dei path e delle relative distanze, la funzione getRecursive, che richiama a sua volta una funzione ricorsiva, la quale ad ogni ciclo, se trova un file, carica in struttura il path relativo e la distanza da inputfile.

- **void** getRecursive(**char** *inputfile, **char** *basepath, **int** limit);

  Scan ricorsivo della directory per popolare la struttura.
  Richiamo delle relarive funzioni per search || searchall

Levenshtein.h

Descrizione:

Header file contenente le funzioni per applicare l'algoritmo di Levenshtein alle stringe passate in input, e opzionalmente per la generazione (qualora non esistette già) del file .bin contenente le istruzioni per modificare file1 in file2. Vi sono quindi le relative funzioni per il la creazione e il calcolo della matrice, per deallocare la memoria allocata ala matrice una volta terminato il calcolo e per generare o sovrascrivere il file .bin.

- **void** deallocateMat(**int** size, **int**** mat);

  Funzione per deallocare la memoria della matrice

int **matGenerate(**char** *str1, **int** x, **char** *str2, **int** y);

Funzione per generare ed inizializzare la matrice per l'algoritmo di Levensthein.

Restituisco la matrice calcolata.

- **Lista** *matCalculate(**int** **mat, **char** *str1, **int** x, **char** *str2, **int** y);

  Funzione che prende in ingresso una matrice "mat", la stringa "str1" e la sua relativa lunghezza "x", la stringa "str2" e la sua relativa lunghezza "y", calcola le istruzioni da eseguire per modificare la stringa1 nella stringa 2, le aggiunge ad una lista e la restituisce.

- **void** generateBinaryFile(**char** *outputfile, Lista *list);

  Funzione per generare il file .bin data una lista di istruzioni.

- **int** levensthein_distance(**char** *file1, **char** *file2);

  Calcolo della distanza tra "file1" e "file2". Restituisce la distanza tra le 2 stringhe.

Time.h

Descrizione:

Header file contente le funzione per tenere traccia del tempo in secondi tra l'inizio e la fine di una funzione.

Linked_list.h

Descrizione:

Header file contenente la struttura dati utilizzata per la memorizzazione delle istruzioni da applicare ad un file sorgente per trasformarlo in un file target. Vengono quindi definite le funzioni per: popolare la lista (push), invertire la lista una volta popolata, getType che restituisce un puntatore a char contenente il relativo Type e la funzione createString, che apre il file passatogli, posizione il puntatore alla fine del file, legge la lunghezza e crea un puntatore a char della lunghezza letta. Una volta allocata abbastanza memoria viene lette il contenuto del file e scritto all'interno del buffer.

- **void** push(Lista** head_ref, edit_type type, **unsigned int** pos, **char** character);

  Funzione per aggiungere un nodo in testa alla lista.

- **void** reverse(Lista** head_ref);

  Funzione per effettuare il reverse della lista.

- **char** *getType(Lista *list);

  Funzione per ottenere la stringa associata all'edit_type presente nella lista.

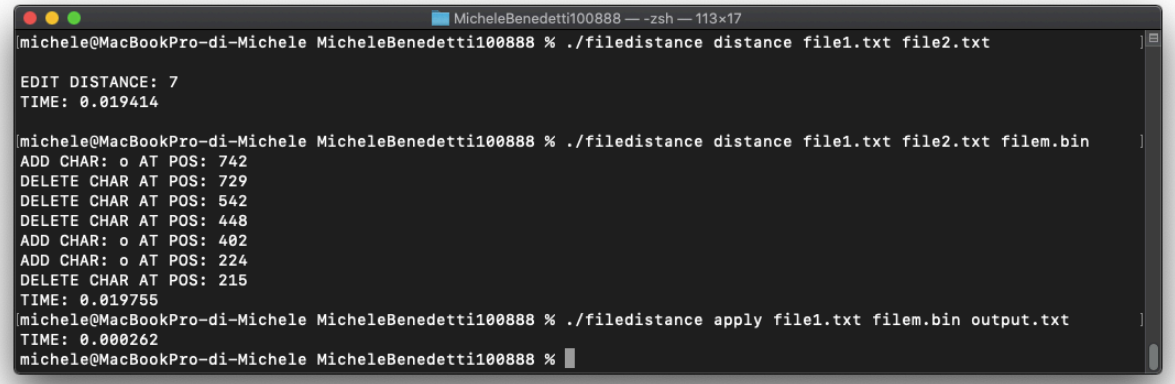
- **char** *createString(**char** *file);

  Funzione per inserire il contenuto di un file in un buffer.

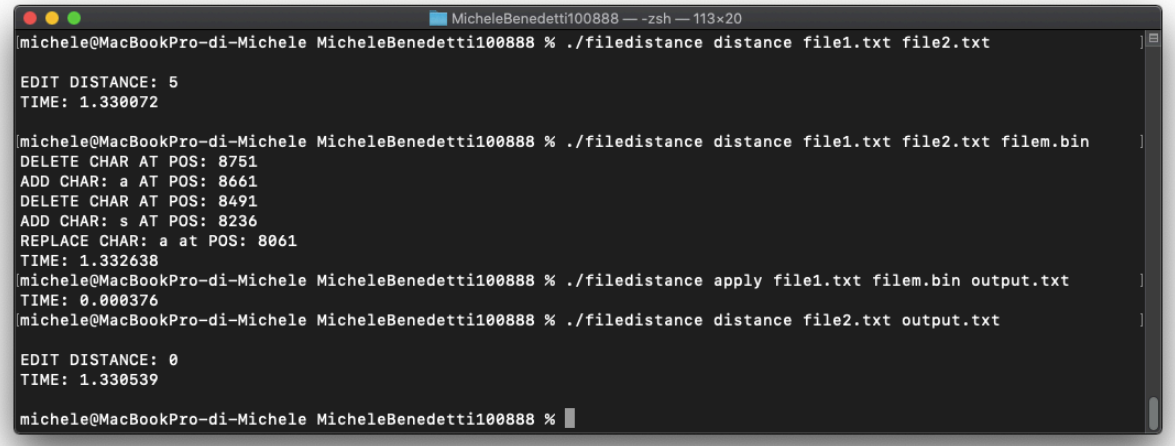
4. Test

Test distance, distanceOutput e apply con file da 100 byte ciascuno, e relativi tempi:

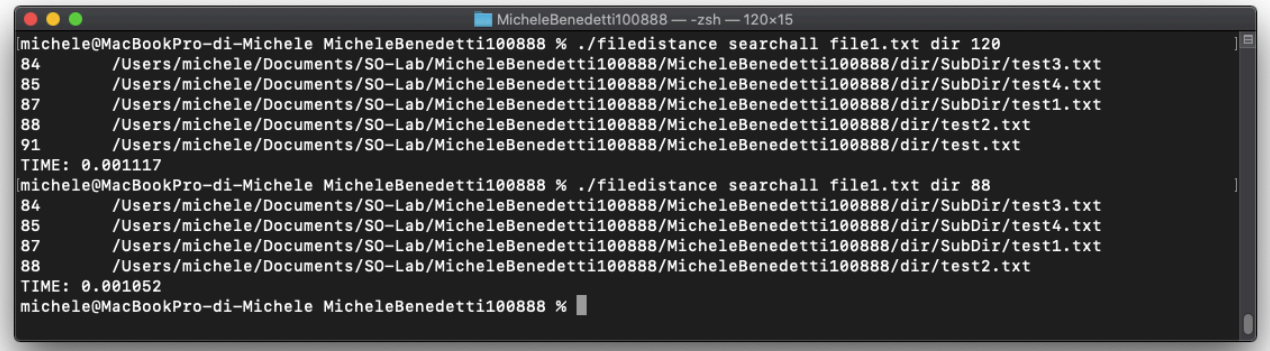
Test distance, distanceOutput e apply con file da 1000 byte ciascuno, e relativi tempi:



Test distance, distanceOutput e apply con file da 10000 byte ciascuno, e relativi tempi:



Test searchAll con 5 file totali, ogni uno con distanza diversa dal file in input. Vengono restituiti solo i file con distanza inferiore al parametro passato e in ordine crescente.



Test search, dove viene restituita la distanza e i path assoluti dei file con minor distanza da inputFile. In Questo caso i file con minor distanza erano 2, e vengono restituiti entrambi.