Algoritmi e Strutture Dati

Consegna 2

# Organizzazione del codice

Il codice è organizzato come da standard nelle cartelle bin, build, src, contenenti rispettivamente i file binari, oggetto e sorgenti. Oltre l’implementazione del makefile, è stata sviluppata anche la unit test, con utilizzo di Unity.

Le istruzioni per l’esecuzione sono presenti nel file app\_main.c e vengono stampate a schermo in caso di errore nel passaggio dei parametri richiesti. Nei parametri, è possibile scegliere le directory di input, output e il numero massimo di correzioni per parola.

Il codice è suddiviso in due librerie e un’applicazione: nelle librerie troviamo la edit\_distance\_lib, contenente la funzione di edit\_distance con variazione in programmazione dinamica, e la struttura dati del dizionario.

# Struttura dati del dizionario

Nel file dictionary.c è presente l’implementazione della struttura dati utilizzata per contenere il dizionario. La struttura è molto semplice e si occupa di caricare tutte le parole, grazie al file di input già ordinato. Questo permette alla libreria di non doversi preoccupare dell’ordine, già definito dal file, e quindi essere molto rapida. Inoltre l’utilizzo di una struttura ordinata, permette di usare algoritmi più rapidi, come la ricerca binaria.

Per evitare di ricercare la correzione di parole non errate infatti, la libreria utilizza un algoritmo di ricerca binaria, richiamato dall’applicazione per ogni parola del file da correggere, che controlla rapidamente se la parola passata sia presente nel dizionario; nel caso di un riscontro positivo la parola è quindi corretta e perciò si evita di scorrere l’intera struttura e valutare tutte le edit distance.

# Edit distance dinamica e memoization: risultati

Edit distance scritta utilizzando i concetti di programmazione dinamica si conferma decisamente più rapida della sua controparte “standard”. L’utilizzo di una matrice di interi nella quale vengono salvate le edit\_distance già calcolate, per ogni coppia di parole, permette infatti di recuperare rapidamente i valori utili alla funzione, invece di ricalcolarli di volta in volta, sfruttando quindi il concetto di memoization per ridurre i tempi di esecuzione.

Dal lato pratico la differenza è notevole: si passa da diverse decine di minuti con l’utilizzo della edit distance “standard” (l’esecuzione è stata interrotta poiché non portava risultati), a meno di 2 secondi di esecuzione completa con la versione dinamica.

Da notare però che i risultati dell’applicazione non saranno perfetti: il file di output corretto potrà non coincidere con le correzioni desiderate, poiché nelle specifiche della consegna le operazioni atomiche fornite sono inserimento e cancellazione, diversamente ad esempio dall’algoritmo della distanza di Levenshtein, che prevede anche la sostituzione, oppure alla distanza di Damerau-Levenshtein che prevede anche lo scambio di due caratteri adiacenti.