Relazione progetto Algoritmi e Strutture Dati

IMPLEMENTAZIONE DIZIONARIO IN LINGUAGGIO C CON RED-BLACK TREE

31 Maggio 2017

Nicolas Farabegoli
Paolo Baldini
Università di Bologna
nicolas.farabegoli@studio.unibo.it
paolo.baldini@studio.unibo.it

Indice

1 Analisi

L'obiettivo del progetto è quello di costruire e gestire un vocabolario contenente le voci, ordinate, e le corrispondenti definizioni. Il dizionario può essere inizialmente generato a partire da un file di testo. In questo caso si vogliono memorizzare tutte le parole presenti in un file di testo in ordine lessico grafico.

Si utilizzi per rappresentare il dizionario la struttura dati che si ritiene più opportuna, dopo una attenta analisi. Si possono utilizzare le strutture dati astratte affrontate dal corso (alberi, hash table, code, pile, liste, grafi, ...).

Il file contiene un testo le cui parole non sono memorizzate in ordine lessicografico, e sono separate da spazio. I vocaboli sono inizialmente salvati senza la corrispondente definizione; la definizione, una stringa di dimensione sconosciuta, sarà inserita in un secondo momento, tramite la funzione apposita.

Ogni voce del dizionario contiene al massimo 20 caratteri, e minimo 2, e le definizioni inserite ne contengono al massimo 50.

1.1 Vincoli

Nel dizionario non devono essere inserite le parole di un solo carattere o con più di 10 caratteri. Inoltre non devono essere inserite parole che differiscono solo per caratteri minuscoli/minuscoli. Le parole e le definizioni inserite nel dizionario non devono presentare caratteri maiuscoli; nel caso di inserimento di una parola con un carattere, o più, maiuscoli, convertirli per inserirli nel modo corretto.

Si assume che nel file di testo e nel dizionario non sono presenti nomi proprio (di persona) e le voci nel dizionario possono ammettere caratteri accentati (è, à, etc...), ma non ammettono cifre e caratteri di punteggiatura.

1.2 Funzionalità richieste

Sono presenti metodi di gestione del dizionario come il conteggio delle voci del dizionario, l'inserimento o la cancellazione di un nuovo vocabolo, ...

Esistono due diverse funzioni di ricerca:

- la prima, di base, dato un termine restituisce, se presente, la sua definizione
- la seconda è una funzione di searchAdvance che verifica la presenza della parola da cercare e restituisce la tre parole più simile a quella che si vuole cercare. Per similarità si intende il minor numero di modifiche (sostituzioni, inserimenti) per trasformare una parola nell'altra.

Le specifiche della ricerca avanzata, e l'algoritmo da utilizzare, sono a scelta dello studente. Inoltre si vuole avere la possibilità di caricare e salvare il dizionario, a partire dalla struttura dati ottenuta, con il seguente formato, in un file testuale (**.txt**):

```
"che":  [(null)]
"classe:  [(null)]
"come":  [nc]
"determinato":  [ulz u]
"di":  [(null)]
```

Oltre al salvataggio classico è richiesto di sviluppare il salvataggio del file compresso, quindi con un ridotto quantitativo di spazio in memoria, utilizzando la tecnica di compressione di Huffman. Il vocabolario è compresso a partire dallo stesso formato del normale salvataggio ("che": [(null)] ...).

Oltre al salvataggio del file compresso, si vuole avere la possibilità di caricare il dizionario compresso da un file di test (decompressione), in modo tale da poterlo visualizzare e/o rielaborare successivamente. Il testo ottenuto dalla decompressione è dello stresso formato del dizionario salvato in memoria. La funzione deve produrre in uscita la struttura dati vocabolario scelta.

2 Aspetti del progetto

2.1 Componenti del gruppo

Il gruppo si compone di due persone: Paolo Baldini (`xxxxxxx`), Nicolas Farabegoli (`788928`).

2.2 Suddivisione del lavoro

Sono state assegnate 13 funzioni per lavorare sulla struttura dati; l'implementazione delle suddette funzioni è stato ripartita nel seguente modo:

- `createFormFile()` - Nicolas Farabegoli
- `printDictionary()` - Paolo Baldini
- `countWord()` - Paolo Baldini
- `insertWord()` - Nicolas Farabegoli
- `cancWord()` - Nicolas Farabegoli
- `getWordAt()` - Paolo Baldini
- `insertDef()` - Nicolas Farabegoli
- `searchDef()` - Nicolas Farabegoli
- `saveDictionary()` - Nicolas Farabegoli
- `importDictionary()` - Nicolas Farabegoli
- `searchAdvance()` - Paolo Baldini
- `compressHuffman()` - Paolo Baldini
- `decompressHuffman()` - Paolo Baldini

2.3 Tools di sviluppo

I tools di sviluppo che sono stati utilizzati sono: Visual Studio Enterprise 2015 (Editor + compilatore) per effettuare il test in ambiente Windows, mentre VIM (Editor) e GCC (compilatore) per il test in ambiente Linux.

I test del software in ambiente Windows sono stati effettuati su Windows 10 Pro, invece i test in ambiente linux sono stati effettuati utilizzando Arch Linux, Kernel 4.11.3-1 e GCC 7.1.1.

2.4 Deadline

Il progetto viene terminato per il primo appello di Giugno, si stima quindi come data di consegna del progetto (in accordo con le scadenze imposte dal docente) il 1 giugno 2017.