Introduzione

La teoria dei codici, nata negli anni ‘40, costituisce una branca dell’informatica teorica in continuo sviluppo. Ha suscitato grande interesse sia da un punto di vista teorico, per le ricche e suggestive strutture algebriche che in essa intervengono, sia da un punto di vista pratico, per le numerose e importanti sue applicazioni.

Un codice è un insieme di parole su un alfabeto finito che è accettabile nella trasmissione dell’informazione: giustapponendo tali parole si ottengono messaggi che possono essere univocamente decifrati.
L’univoca decifrabilità formalizza l’idea di non ambiguità di un messaggio codificato. Tipici esempi sono il codice Morse, il codice ASCII.

Nel 1950 Sardinas e Patterson fornirono una condizione necessaria e sufficiente perché un insieme sia un codice, usata successivamente da Capocelli e Hoffmann (1985) che idearono un algoritmo per testare l’univoca decifrabilità di un insieme finito di parole.

Nel 1982 Rodeh disegnò un algoritmo per testare l’univoca decifrabilità di un codice basato sull’algoritmo di McCreight per la costruzione dei suffix tree. La complessità di tale algoritmo, ogetto del lavoro di tesi, è $O(nm)$ dove n rappresenta il numero di parole di codice da processare e m la loro lunghezza totale.

L’algoritmo si basa sulla definizione di coda (particolari parole costruite a partire dall’insieme in input) e consente di trasportare l’univoca decifrabilità di un insieme finito sull’insieme delle sue code. Se nessuna coda è una parola di codice allora l'insieme è un codice (e viceversa).

Dunque l’algoritmo di Rodeh genera tutte le code possibili e le confronta con tutte le parole di codice terminando negativamente nel caso il confronto risulti positivo.

Cruciale è quindi la definizione dell’insieme delle code e la loro gestione in modo efficiente. Per tener traccia dell’insieme delle code, Rodeh usa il suffix tree così come implementato dall'algoritmo di McCreight [McC].

Il suffix tree (o albero dei suffissi) è una struttura dati che evidenzia la struttura interna di una stringa in modo da facilitare operazioni comuni come la ricerca di sottostringhe. In particolare è un albero associato ad una stringa *w* tale che le sue foglie sono associate ai suffissi di *w*. Un esempio d’uso potrebbe essere la ricerca di un testo in un dizionario: le parole presenti nel dizionario verrebbero rappresentate da un albero dei suffissi che agirebbe da indice per il testo da cercare. La particolarità è che la ricerca verrà effettuata in tempo lineare nella lunghezza del testo. Di nostro interesse è sicuramente la possibilità di verificare tramite un suffix tree se un linguaggio è un codice, ovvero se risulta univocamente decifrabile.

Esistono diversi algoritmi per la costruzione del suffix tree, ma l'algoritmo di Rodeh utilizza quello di McCreight che nel 1976 disegnò un algoritmo in tempo lineare più efficiente in spazio rispetto ai precedenti.

Descriviamo brevemente i passi dell'algoritmo di Rodeh.

Supponiamo che $C = \{a_1,...,a_n\}$ sia l'insieme di cui vogliamo testare l'univoca decifrabilità.

Inizialmente viene costruito il suffix tree T sul testo $w = a_1\$_1... a_n\$_n$ i cui \$ sono simboli speciali, diversi tra loro e da tutte le lettere

a_i, $1 \leq i \leq n$.

Tale suffix tree T è costruito usando l'algoritmo di McCreight come implementato in [G].

In tal modo si ottengono rappresentate nelle foglie di T i suffissi di w.

Oggetto di questo lavoro di tesi è stata la modifica di T consentendo successivamente l'implementazione dell'algoritmo di Rodeh, migliorando e proseguendo il lavoro iniziato in [V].

In particolare è stata aggiornata la struttura dei nodi dell'albero aggiungendo altre informazioni indispensabili alla riuscita dell'algoritmo di Rodeh.

È stato aggiunto un metodo di *pruning* per consentire la potatura di alcuni rami contenenti suffissi obsoleti creati con l'uso di separatori all'interno della stringa in input.

Un ulteriore metodo (*SuffixTreeCode*) è stato introdotto per rendere più semplice ed immediato l' accesso al suffix tree, quindi l' eliminazione di altri nodi rappresentanti informazione vuota. In tal modo è stato possibile determinare l'insieme delle code cui tratta l'algoritmo di Rodeh ed implementare l'algoritmo finale capace di verificare, data una lista di parole in input, l' univoca decifrabilità.

La prima parte della tesi è completamente dedicata alle definizioni di base in modo da consentire una maggiore comprensione dei capitoli successivi nonché alla presentazione dei suffix tree con un ampia descrizione ed esempi.

L'ultima parte è dedicata alla descrizione e all'implementazione dell'algoritmo di Rodeh nonché ai dettagli sulle strutture dati utilizzate.

L'implementazione è stata realizzata in Java, usando Eclipse come ambiente di sviluppo.

In particolare il primo capitolo presenta una panoramica sulla teoria dei codici evidenziando alcune nozioni di base essenziali alla comprensione del lavoro svolto: definizione di parole, linguaggi, codici ed una prima formalizzazione del test di Sardinas e Patterson.

Il secondo capitolo partendo dal test di Sardinas e Patterson (descritto nel capitolo precedente) tratta alcuni algoritmi in grado di verificare l'univoca decifrabilità di un codice soffermandosi su quello di Rodeh (oggetto di questo lavoro di tesi) basato sulla costruzione dell'albero dei suffissi di McCreight. Da qui la definizione e le proprietà dei suffix tree in generale e dei suffix tree secondo McCreight.

Il terzo capitolo descrive in modo dettagliato le modifiche e le strutture dati ausiliarie utili per l'implementazione dell'algoritmo di Rodeh.

Il quarto capitolo è il cuore del lavoro e riguarda l'implementazione dell'algoritmo di Rodeh: dato un codice in input, l'algoritmo sarà capace di verificarne l'UD stampandone il responso.

Bibliografia

[BP] J.Berstel and D.Perrin
"Theory of Codes", Academic Press (1985)

[D] L.D'Auria
"Implementazione in Java di un algoritmo di Capocelli e Hoffman per decidere l'univoca decifrabilità e altre proprietà di un insieme finito di parole". Tesi di Laurea in Informatica (A.A 2005-2006).

[G] C.Giordano
"Implementazione in Java dell'algoritmo di McCreight per la c ostruzione dei suffix tree". Tesi di laurea in informatica (A.A 2006-2007).

[GT] M.T.Goodrich and R.Tamassia
"Data structures and algorithms in Java", 3^Edition Wiley(2004).

[McC] E.M.McCreight
"A space-economical suffix tree construction algorithm"
Journal of the Association of Computing Machinery,
Vol 23 n°2, pp. 262-272.

[R] M.Rodeh
"A Fast Test for Unique Decipherability Based on Suffix Trees"
Ieee Transaction on Information Theory, Vol. IT-28 n°4
(1982), pp. 648-651.

[V] D. Vece
"Implementazione in Java dell'algoritmo di Rodeh per
decidere l'univoca decifrabilità di un insieme finito di
parole: uso del suffix tree e sue estensioni". Tesi di Laurea in
Informatica applicata (A.A 2009-2010).

[E] Eclipse
www.eclipse.org

[J] Java
www.java.com

[GT] M.T.Goodrich and R.Tamassia
"Data structures and algorithms in Java", 3^Edition
Wiley(2004).