Università degli Studi di Torino
Corso di Laurea Magistrale in Informatica
Algoritmi e Strutture dati

Relazione esercitazione 2

Alessandro Clocchiatti
Matricola 909105

Consegna
Si consideri il problema di determinare la distanza di edit tra due stringhe (Edit distance): date due stringhe s1 e s2, non necessariamente della stessa lunghezza, determinare il minimo numero di operazioni necessarie per trasformare la stringa s2 in s1. Si assuma che le operazioni disponibili siano: cancellazione e inserimento.

Esempi:
- "casa" e "cassa" hanno edit distance pari a 1 (1 cancellazione);
- "casa e "cara" hanno edit distance pari a 2 (1 cancellazione + 1 inserimento);
- "vinaio" e "vino" hanno edit distance uguale a 2 (2 inserimenti);
- "tassa" e "passato" hanno edit distance pari a 4 (3 cancellazioni + 1 inserimento);
- "pioppo" e "pioppo" hanno edit distance pari a 0.

Si implementi una versione ricorsiva della funzione `edit_distance` basata sulle seguenti osservazioni (indichiamo con $|s|$ la lunghezza di s e con $rest(s)$ la sottostringa di s ottenuta ignorando il primo carattere di s):
- se $|s1| = 0$, allora $edit\_distance(s1, s2) = |s2|$;
- se $|s2| = 0$, allora $edit\_distance(s1, s2) = |s1|$;
- altrimenti, siano:
  - $d_{no-op} = \begin{cases} edit\_distance\big(rest(s1), rest(s2)\big) & \text{se } s1[0]=s2[0] \\ \infty & altrimetni \end{cases}$
  - $d_{canc} = 1 + edit\_distance\big(s1, rest(s2)\big)$
  - $d_{ins} = 1 + edit\_distance(rest(s1), s2)$

Si ha $edit_{distance(s1,s2)} = \min\{d_{no-op}, d_{canc}, d_{ins}\}$

Si implementi una seconda versione `edit_distance_dyn` della funzione, adottando una strategia di programmazione dinamica. Tale versione deve essere anch'essa ricorsiva (in particolare, essa può essere facilmente ottenuta a partire dall'implementazione richiesta al punto precedente).
Nota: Le definizioni sopra riportate non corrispondono al modo usuale di definire la distanza di edit. Sono del tutto sufficienti però per risolvere l'esercizio e sono quelle su cui dovrà essere basato il codice prodotto.

Unit Testing
Implementare gli unit-test degli algoritmi secondo le indicazioni suggerite nel documento Unit Testing.

Uso delle funzioni implementate
Il file *dictionary.txt* contiene l'elenco (di una parte significativa) delle parole italiane. Le parole sono scritte di seguito, ciascuna su una riga. Il file *correctme.txt* contiene una citazione di John Lennon. La citazione presenta alcuni errori di battitura.
Si implementi un'applicazione che usa la funzione `edit_distance_dyn` per determinare, per ogni parola `w` in *correctme.txt*, la lista di parole in *dictionary.txt* con edit distance minima da `w`. Si sperimenti il funzionamento dell'applicazione e si riporti in una breve relazione (circa una pagina) i risultati degli esperimenti.

---

1. Risultati

L'implementazione delle due versioni della funzione edit distance ha permesso di comprendere a pieno l'utilità della programmazione dinamica attraverso l'utilizzo della tecnica di memoization.

La versione statica dell'edit distance prevede delle chiamate ricorsive su sottostringhe delle due parole analizzate. Da un'analisi dell'esecuzione della versione statica, si nota come alcune operazioni vengano ripetute più volte.
La versione dinamica prevede la presenza di una struttura dati (una matrice) in cui vengono memorizzati i risultati dei sotto problemi, in modo tale da non ripetere inutilmente operazioni già effettuate in precedenza.
In questo modo si passa ad una complessità esponenziale (edit distance statico) ad una complessità lineare $O(n \cdot m)$ dove n e m sono le lunghezze delle stringhe analizzate.

L'applicazione implementata effettua una rimozione della punteggiatura nel file *correctme.txt*, in modo tale da calcolare l'edit distance solamente sulle parole.

L'efficienza della versione dinamica è confermata dall'analisi dei tempi di esecuzione:

	word - word	*word - dictionary*
Edit distance static	TODO	$\sim 600\ sec$
Edit distance dynamic	$[3.19 \cdot 10^{-7}, 0.03]\ sec$	$\sim 1.4\ sec$

La durata dell'applicazione può essere ridotta se, durante l'esecuzione, si evita di calcolare la edit distance (statica o dinamica) tra la coppia di stringhe la cui differenza di lunghezza è maggiore dell'edit distance minima trovata fino a quel punto. In questo caso, l'edit distance sarà maggiore o uguale alla differenza di lunghezza delle due stringhe e quindi non verrebbe un valore rilevante.
Questa implementazione riduce le tempistiche di esecuzione, indipendentemente dal tipo di algoritmo di edit distance utilizzato.

Analizzando i risultati ottenuti è emerso come in alcuni casi l'algoritmo ritorni la parola del dizionario con edit distance minore, anche se non corrisponde alla parola sintatticamente e semanticamente corretta.

Word	**Edit distance word**	**Correct word**
made	made	madre
squola	suola	scuola

Siccome l'algoritmo implementato non prevede l'operazione di sostituzione, in alcuni casi ci sono parole nel dizionario che ottengono un'edit distance minore rispetto a quella corretta. Per risolvere il problema sarebbe necessario introdurre dei sistemi di analisi del contesto, non previsti in questa esercitazione.