RELAZIONE LABORATORIO ALGORITMI 2017/2018

ESERCIZIO 2:

Abbiamo implementato due versioni di un algoritmo che misura l'edit distance tra due stringhe, una ricorsiva e una dinamica, sempre ricorsiva. Lo scopo è stato quello di valutare i tempi di esecuzione delle due versioni dell'algoritmo.

Distance:

Nel package distance sono presenti 4 file: la libreria Distance.java, DistanceExeption.java per le eccezioni, e due test che testano uno la versione non dinamica, e l'altro quella dinamica.

Distance Runner:

Nel package distancerunner invece abbiamo sviluppato le applicazioni per eseguire i dataset forniti. Nello specifico DistanceRunner.java esegue il file UsageDistance.java.

Considerazioni sulla complessità:

La funzione non dinamica esegue un gran numero di chiamate ricorsive. Anzi, se nessuna delle due condizioni dell'if è vera, vengono fatte due chiamate ricorsive, che a loro volta potrebbero fare altre due chiamate ricorsive.

La complessità della versione non dinamica risulta quindi essere m*n, dove m e n sono rispettivamente la lunghezza della prima e della seconda stringa.

Ovviamente questo su una stringa molto lunga si traduce in tempi di esecuzione lunghissimi, che non sono giustificati perchè molte volte si va a fare un'operazione di controllo inutile. Nello specifico andiamo a

confrontare due caratteri che abbiamo già confrontato in precedenza.

Per migliorare l'algoritmo ed eliminare questi controlli "inutili" abbiamo sfruttato la programmazione dinamica. L'algoritmo è sempre ricorsivo, ma i tempi di esecuzione di questa versione sono decisamente più contenuti.

Per sviluppare la versione dinamica abbiamo inizializzato tutte le posizioni di una matrice n*m a Integer.MAX_VALUE. L'algoritmo dinamico risulta avere come complessità min(n,m) con n lunghezza della prima stringa e m lunghezza della seconda.
Dopo di che andiamo ad eseguire i controlli, le prime due condizioni dell'if gestiscono il caso in cui non si trovano due caratteri uguali, e quindi arrivati alla fine della stringa ritorna la lunghezza dell'altra stringa, che corrisponde al valore di editDistance effettivo.
Altrimenti entriamo nel caso in cui effettuiamo le 3 chiamate ricorsive, come nell'algoritmo non dinamico, la differenza però è che nella versione dinamica andiamo ad effettuare un ulteriore controllo (array[i][j] == Integer.MAX_VALUE) così da verificare se la posizione è già stata controllata o no. Se non è stata ancora controllata avrà Integer.MAX_VALUE come valore e quindi esegue le chiamate ricorsive, altrimenti ritorna l'elemento in quella posizione, che rappresenta il valore di editDistance calcolato fino a quel momento.

Questa semplice operazione rende molto più efficiente l'algoritmo in quanto ci evita molte chiamate ricorsive che sono molto costose, e riduce la complessità dell'algoritmo a min(length(s1),length(s2)).