Relazione Esercizio 2

martedì 9 novembre 2021 16:05

▼ **Comparazione su correctme.txt**

Programmazione dinamica

Andando a misurare le tempistiche di esecuzione del programma sull'intero file "*correctme.txt*" otteniamo i seguenti risultati:

Prova	Tempo
1	58.125
2	58.655
3	57.391

Programmazione non dinamica

Non sono in grado di fornire tempistiche per quanto riguarda l'esecuzione del programma sull'intero file '*correctme.txt*' in quanto i tempi risultano molto lunghi per ogni singola parola.

Decidiamo quindi di eseguire la comparazione delle 2 funzioni (*edit_distance_dyn_wrap* & *edit_distance_wrap*) su un file contenente una unica parola.

La parola in questione sarà "cinqve".

▼ **Comparazione su singola parola "cinqve"**

Programmazione dinamica

Prova	Tempo
1	3.125
2	3.156
3	3.265

Programmazione non dinamica

Prova	Tempo
1	619.125
2	622.138
3	624.873

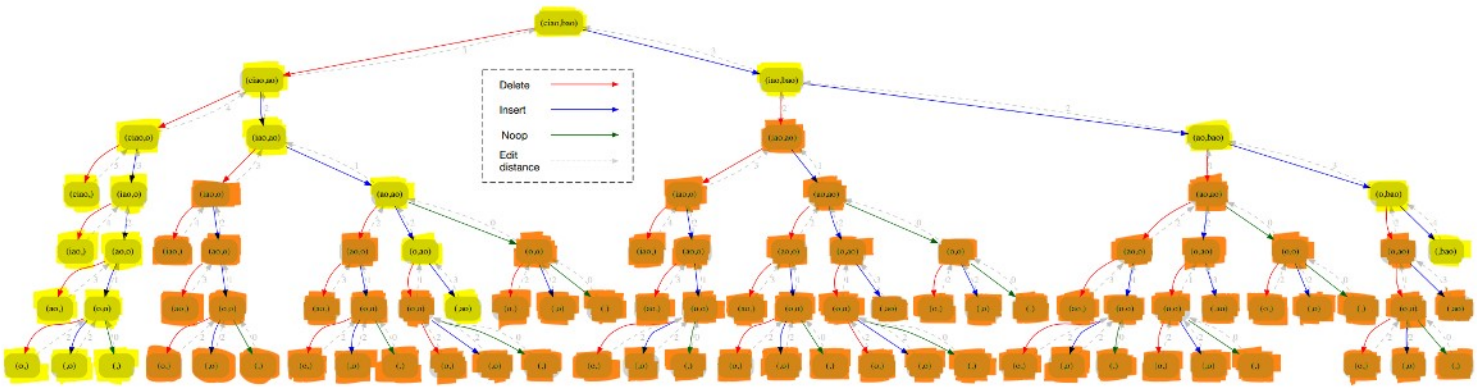
Sono visibilmente dei tempi enormi in confronto all'algoritmo che utilizza la programmazione dinamica.

Questo perché per ogni chiamata ad edit distance l'algoritmo andrà a creare e percorrerà una grande quantità di rami.

Inoltre il processo andrà ripetuto per ogni vocabolo di dictionary.txt, ovvero 661563 volte.

Dai risultati ottenuti si nota quanto sia efficace la programmazione dinamica per questo tipo di algoritmo.

Vediamo di seguito un esempio grafico che ne spiega il perché.



In giallo abbiamo tutti i rami che vengono effettivamente creati, sono i caratteri su cui richiamiamo la funzione di edit distance.

In arancione abbiamo tutti i rami che non vengono creati con l'utilizzo della programmazione dinamica perché abbiamo già memorizzato l'edit distance su quei caratteri.

Dall'enorme quantità di rami non creati riusciamo a spiegarci l'enorme divario di tempo che ha caratterizzato i nostri test