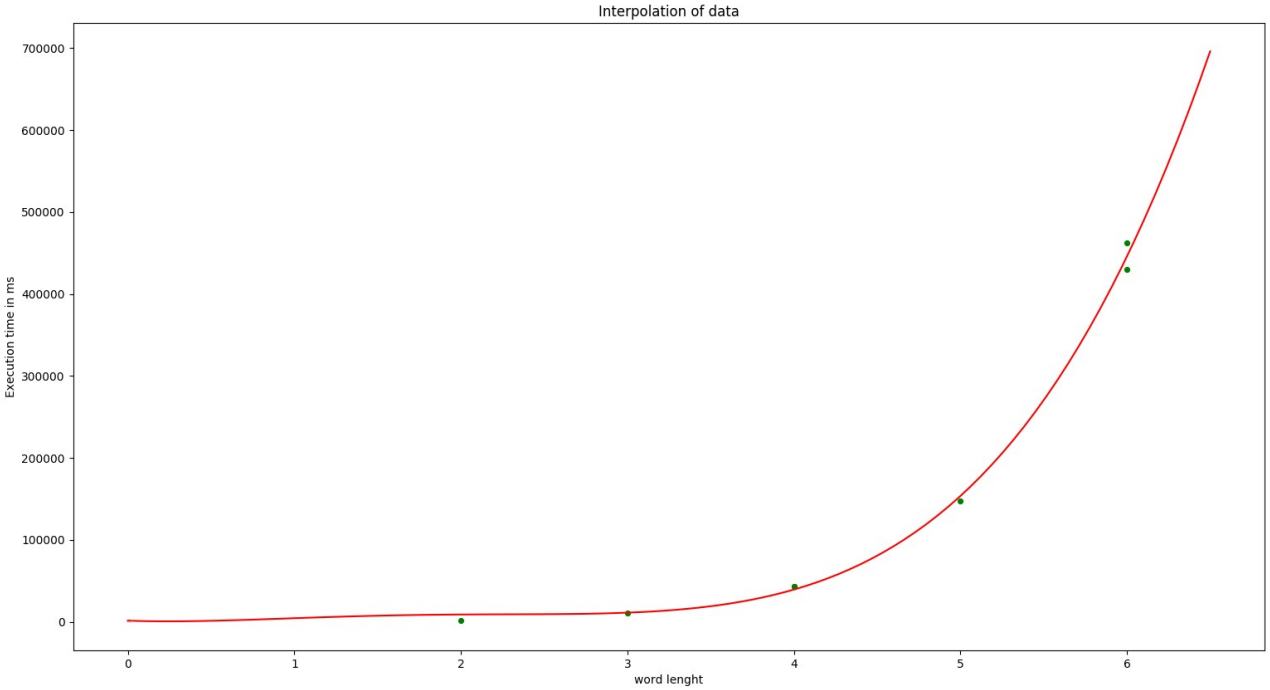
Esercizio2

La classe exampleApp avvia il programma creando un istanza di se stessa, il costruttore richiede come parametri i due file, file da correggere e file dizionario, e li salva in delle variabili ArrayList private.
L'unico metodo pubblico è il metodo getCorrection che ritorna una mappa di coppie <Parola da correggere, lista di correzioni>, questa mappa viene infine filtrata per escludere le parole con edit distance uguale a zero e quindi corrette e poi viene stampata a video.
La prima fallacia che si può notare sta nel fatto che le correzioni dipendono fortemente dal dizionario, con eventuale presenza di falsi negativi.
Ad esempio le parole "selice" (correzione felice) e "made" (correzione madre) sono ovviamente sbagliate nel contesto della frase ma sono parole presenti nel dizionario italiano ed in quanto tali non risultano errate nel mio programma.
Un altro problema sta nel fatto che secondo la consegna le uniche operazioni disponibili sono eliminazione ed inserimento.
Non essendo considerata la sostituzione, al posto di alcune parole errate, ad esempio "squola" e "corpito", vengono proposte correzioni altresì sbagliate.
Ad esempio come sostituzione di "squola" viene proposto "suola", al posto che "scuola" in quanto quest'ultima ha un edit distance pari a 2 quinid inferiore alla edit distance di "suola".

Variante ricorsiva

La variante ricorsiva di questo programma è troppo lenta per essere considerata accettabile, il tempo di esecuzione aumenta esponenzialmente a seconda della lunghezza della parola da correggere.
Qui sotto la tabella ed il grafico dei dati ricavati, purtroppo pochi in quanto già una parola di 9 lettere (ad esempio "perpeteva") richiederebbe 35 ore di esecuzione.
Per evidenti motivi ho dovuto interrompere l'esecuzione ed accontentarmi dei dati raccolti.
In rosso sul grafico una possibile curva, interpolazione dei dati, per rendere più evidente la variazione.
I dati raccolti sono invece rappresentati dai punti verdi

lunghezza	Execuzione (ms)
6	430261
5	147782
6	462559
4	43155
3	10915
4	42821
2	2050



Programmazione dinamica

Questa variante è notevolmente più veloce, il file correctme.txt viene analizzato in 17 secondi e la parola più lunga, 10 caratteri, viene analizzata in 1,8 secondi. (usando il comando Time di linux)
Questa versione risulta quindi più efficiente sia dal punto di vista temporale che dal punto di vista della crescita di complessità.
Infatti si può notare che il tempo di esecuzione cresce quasi linearmente e non esponenzialmente come nel caso della ricorsione.
Qui i dati rappresentati graficamente, non inserisco la tabella perché a differenza di prima i dati sono molti e un grafico è sufficiente a notare la crescita quasi lineare.