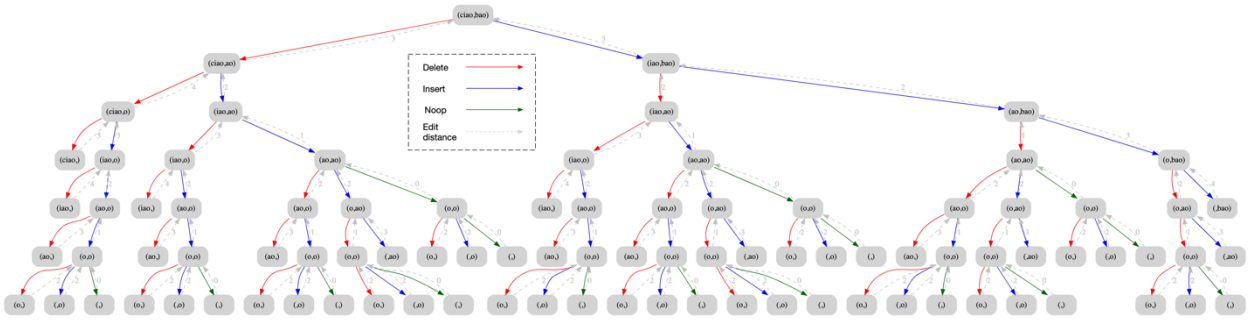
Considerazioni SECONDO ESERCIZIO

- Andrea Torella (matr. 912579)
- Agostino Messina (matr. 913813)

Il problema è quello di determinare la distanza di edit (operazioni) tra due stringhe (Edit distance) s1 e s2, non necessariamente della stessa lunghezza, ovvero determinare il minimo numero di operazioni necessarie per trasformare la stringa s2 in s1.
Si assume che le operazioni disponibili siano:

- cancellazione
- inserimento

Una prima versione dell'implementazione è quella che fa uso di un algoritmo ricorsivo Divide-et-Impera (DI). Il problema principale è che una programmazione di questo tipo è efficiente se i sotto problemi sono indipendenti tra di loro. Infatti, se non lo sono, come nel nostro caso, produce tempi esponenziali.



Possiamo vedere, analizzando l'albero delle computazioni corrispondente all'esecuzione di edit distance, come anche con due stringhe molto corte abbiamo delle chiamate ricorsive ripetute. Infatti, seguendo la prima implementazione dobbiamo considerare ad ogni passo il risultato di: inserimento, cancellazione e "non fare niente" (no_op), per poi prendere il minimo tra queste.
Il risultato è che la computazione sarà appesantita.

Si passa quindi ad una implementazione che adotta una strategia di programmazione dinamica (DP) e di memoization. Questa tecnica consiste nel costruire un array temporaneo che conserva il risultato dei sotto problemi, utilizzando comunque un approccio ricorsivo.

Si è deciso di procedere nel seguente modo:

1. Creazione di una matrice m x n, dove m e n corrispondono alle lunghezze delle due stringhe di cui si vuole calcolare l'edit distance, inizializzando in un primo momento tutte le celle a -1;

2. Ad ogni chiamata ricorsiva, viene inserito il valore in mat[m][n], in modo che se viene nuovamente richiamata edit distance con quei valori di m ed n, otteniamo il risultato in tempo O(1);

3. Viene effettuato un controllo a priori in mat[m][n] per controllare se abbiamo già precedentemente calcolato il risultato con questi parametri.

Grazie a questa tecnica la complessità dell’algoritmo da esponenziale si riduce a O(M * N).