Report exercise 2

Daniele Liberatore, Sandro Massa, Matteo Palazzo

June 11, 2019

Edit-distance's functioning

For the dynamic programming implementation, we save the carried out operations to avoid redundancies. Those are due to the sub-problems not being independent from one another, making this technique, known as *memoization*, very effective.
The length of the strings X and Y are the dimensions of the matrix c containing the results. The cell c_{ij} stores the edit-distance from the prefix X_i to Y_j. Initially all cells are empty; they will store the result of the first time the corresponding operation is called. Without *memoization* the algorithm equals its recursive version (see slides for reference).
Below are some comparisons between the two versions:

X	*Y*	*Edit-distance*	*Recursive*	*Dynamic*
Calla	Palla	1	0.100s	0.100s
Bcamap	Palla	5	0.100s	0.100s
Lonpirante	Lungimirante	4	0.800s	0.100s
Antocartistuzinale	Anticostituzionale	5	$>$ 30m	0.100s

Checker's functioning

Checker compares each word from *target* (correctme.txt) with each one in *dict* (dictionary.txt), associating to each word of the former a list of words of the latter with minimum edit-distance. If a *dict* word's edit-distance is higher than the one already calculated, the former is ignored.
Checker_Test terminates in an average of 5 seconds; performances increase if the *dict* is ordered by increasing length of its words. Complexity however changes only by a constant factor.