Raissa Paganone matricola 842741

ESERCIZIO 1 – PRIMO UTILIZZO

The algorithms I've created are the merge and the insertion sort.

The first one consists on "Divide et Impera" because it divides the array in little parts. If the array has length 0 or 1 it's yet sorted, but, when the length is higher than 1, we split the array in 2 subsequences where the first one has one element more than the second. Every part is sorted using the algorithm recursively and then we extract the minimum for each part and we save it in the array. Now the array is ordered.

The second algorithm doesn't create another array where it saves the ordered array, but it set the element in order in the actual array every time.

By testing those algorithms on my portable computer with the file "integers.csv" for the ascending sorting I obtain these results:

- Mergesort : the sorting ended successfully in an average of about 2.3 seconds printing also the first 100 elements of the sequence.
- Insertionsort: the sorting doesn't end in less than 10 minutes, so it's a complete failure!

As a matter of fact the mergesort has complexity O(n).

The complexity of insertionsort, whereas, depends on the length of the array and it's a quadratic complexity.

ESERCIZIO 1 – SECONDO UTILIZZO

For this exercise I've implemented the methods: printArray, loadArray and hasTwoElements.

We need the first one to print the arrays.

The second one is used to load the elements of a file into an array.

The third algorithm is used to know if the elements of the file "sums.txt" are the sum of 2 elements of the file "integers.csv".

After having tested all my class prints an array of Boolean of length = number of the elements in "sums.txt".

In average it takes about 1 minute and it returns:

true
true
true
true
true
true
true
true
true
false
false

In this case it returns false because 100 is a number too little.
In this one it returns false because 40000000000 is a number too big.

ESERCIZIO 2 – USO DELLE FUNZIONI

For the second exercise I implement 2 algorithms.

The first one calculates the edit distance between 2 words using the recursion. The second one is dynamic and iterative. Because of this the second algorithm is more efficient and faster.

Furthermore, the second algorithm accept in input 2 strings as parameters and uses a matrix of number for searching the result without compare every single character of every single word.

For the usage of this algorithms I've created the class "SecondoUso2" where there are the following methods:

- loadCor: it loads the words of "dictionary.txt" into an ArrayList
- loadArr: it loads the words of "correctme.txt" into an ArrayList
- printArray: it's used for the print of the arrays
- correttore: it's the method which calls edit_distance_dyn for the comparison between the words of "dictionary" and those of "correctme" and returns an array of integers in which there are the results of the minimum edit distance between the words.

This implementation is efficient and fast even if when we control if everything is ok something looks wrong.

In fact the distance calculated takes into account characters like "!", "?", ".", ",", ";" ect. so where there are these symbols the distance will be 1 more than the real one.

Ex: "mia wita." → our methods returns 0, 3 even if the distance between "wita" and "vita" is 2. The reason why it works so is the fact that it considers "." like a character and in "dictionary" there aren't ".": in fact it calculates 1 cancellation ("w"), 1 insertion ("v") and another cancellation (".").

Another problem that must be named is that some words has edit distance minimum with a word which isn't correct according to the sense of the sentence.

Ex: "andavo a squola" → our method returns: 0, 0, 1 even if in order to correct "squola" into "scuola" we have to do 1 cancellation and 1 insertion, so the correct edit distance should be 2. The reason why it returns 1 instead of 2 is because of the existence of "suola" which is a word that differs from "squola" just for 1 cancellation without considering the sense of the sentence.