Lab. 7 Efficient Array Sorting

Do the exercises below in the Octave IDE. Make sure the files and the programs are in the same working directory.

1. Adapt Merge Sort

Adapt the implementation of Merge Sort presented in the slides of class 7, by adding to the results the number **c** of comparisons between elements of the vectors, and the extra space x, used to create new vectors. Use signature

```
function [S, c, x] = merge_sort(V)
```

2. Adapt Quick Sort

Adapt the implementation of Quick Sort presented in the slides of class 7, by adding to the results a) the number a of accesses to the elements of the vector that were considered, and b) the number of swaps s that were made in elements of the array. Use the signature

```
function [S, a, s] = quick_sort(V, lo, hi)
```

3. Assess efficiency of Quick Sort and Merge Sort

Check the efficiency (and correctness) of your implementation of the previous items, in the following vectors:

- V1 – a vector with values from 1 to n
- V2 – a vector with values from n down to 1
- V3 – a vector with elements from 1 to n, arbitrarily sorted.
- V4 – a vector with n elements with n/2 distinct values, sorted in increase order
- V5 – a vector similar to V4, but sorted in decreasing order;
- V6 – a vector similar to V4, but arbitrarily sorted.

4. Compare efficiency of Quick Sort and Merge Sort

Compare the results obtained in the previous item, with those obtained with Bubble Sort, as done in the previous lab class.