

Homework 04

General comments:

- For test cases use vector x generated with `rand(1,N)` where N number of elements in test vector. See more help about `rand`, but in short it fills matrix of specified size with random numbers in the range of 0 to 1. In our case this matrix is a vector since we specified its size as $1 \times N$.
- Try your sort algorithms with reasonably small N (less than 10) at first. Then you can check that output is fine by yourself.
- All sorting algorithm should sort in ascending order unless mentioned otherwise.
- Do not forget to run some test cases.

Problem 1 (5 points)

Write your own implementation of the bubble sort algorithm. Call it 'bubblesort'.

Problem 2 (5 points)

Base on provided `qsort` implementation, write your own implementation of the `qsort` sort algorithm but the one which sort vector in the descending order. Call this function 'qsortDesc'.

Problem 3 (5 points)

Write your own implementation of the heap sort algorithm. Call it 'heapsort'.

Problem 4 (5 points)

For your algorithms 'bubblesort' and 'heapsort', 'qsortDesc', and Matlab built in 'sort'. Plot (on the same figure) their time of execution vs number (N) of the elements of the input test vector. Do not forget to label each curve, see `legend` command.

- N should span from 1 to 10000 (at least 10 points).
- You may like `loglog` plot presentation better.
- Hint. To find the execution time use `tic` and `toc`, see more help for them. For example `tic; qsort(xtest); toc`

Which algorithm is better for a small N and which is for a large?