Algorithm & complexity

Lab: qiuck sort

# Checking the implementation of the quick sort algorithm

Checking the implementation of the ***quick sort*** algorithm in the recursive and non-recursive version.

Verification for the worst case (sorting ordered collection) for different strategies for selecting pivot (a separating element): **middle**, **first**, **last**.

Please perform the tests for the 32 bit release version (Release x86).

**Complete the table below with the test results.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Pivot selection strategy** | **Recursive implementation**  sorting time for the biggest collection or error description | **No recursive implementation** | |
| sorting time for the biggest collection or error description | max. stack size |
| middle | 54542300 nano time | 56894100 nano time | 50000 |
| first | 1750177500 nano time for collection size 30000, for more size collection stack overflow. | 4853697100 nano time for collection size 50000,  Due to we used loop there is no stack overflow | 30000 |
| last | 464645700 nano time for collection size 30000,for more size collections stack overflow. | 1053375700 nano time for collection size 50000, Due to we used loop there is no stack overflow. | 3000 |

**Conclusions (compare recursive and no recursive implementation):**

|  |
| --- |
| In recursive implementation algorithm is fast and due to we use recursion it might cause sack overflow.  In other hand,  Due to we use loop algorithm will be slower than recursive implementation, and it will not cause to stack overflow because we did not use it. |