SORTĂRI ÎN ECHIPĂ

Proiect realizat de Duzi Mihai-Nicolae și Cioancă Vlad-Antoniu

ALGORITMI TESTAȚI

În cadrul acestui proiect, am testat 8 algoritmi de sortare pentru vectori

1. Bubble Sort — O(n^2)
2. Merge Sort — O(nlogn)
3. Quick Sort — O(nlogn)
4. Shell Sort — O(nlogn)
5. Heap Sort — O(nlogn)
6. Radix Sort baza10 — O(n+max)
7. Radix Sort baza2^16 — O(n+max)
8. STL sort din C++

TESTE FOLOSITE

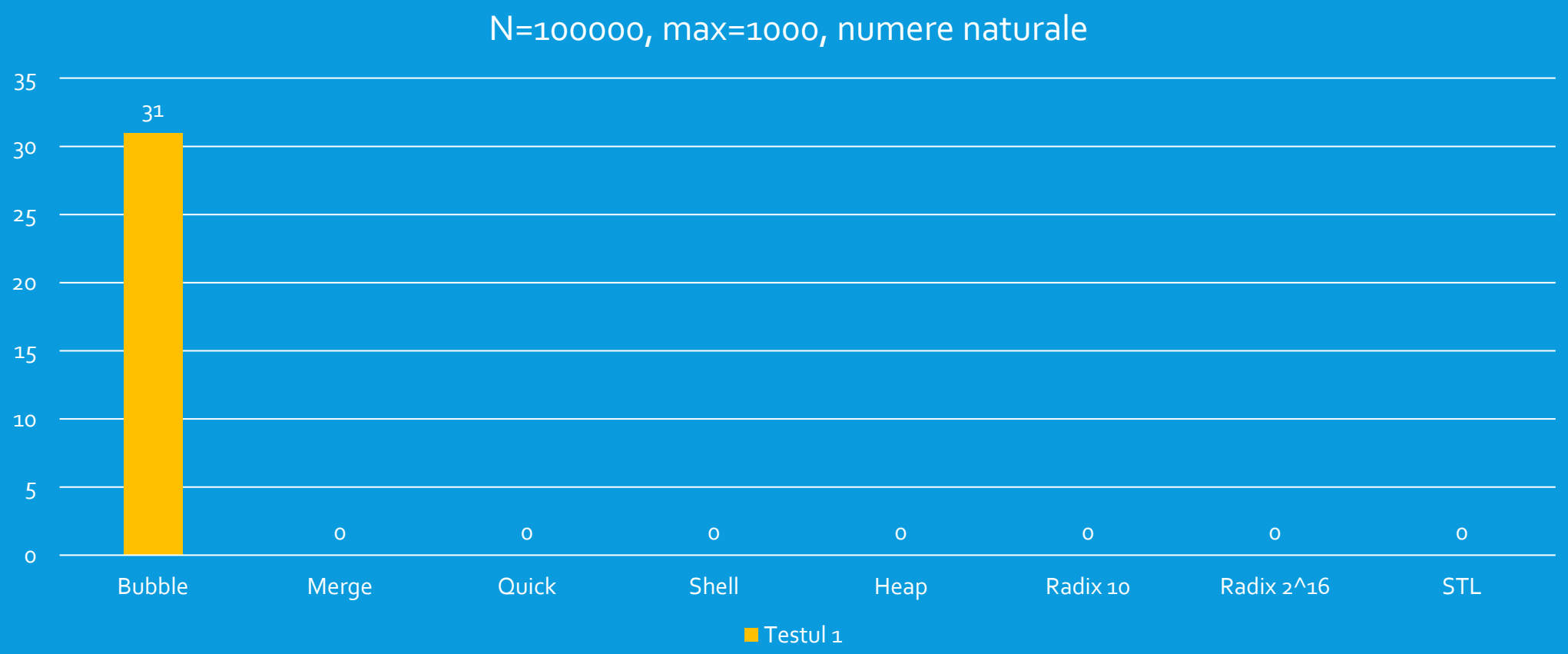
Am creat 10 teste de dimensiuni variate, astfel:

1. N=100000, max=1000, numere naturale
2. N=100000, max=1000, numere reale
3. N=1000000, max=1000, numere naturale
4. N=1000000, max=1000, numere reale
5. N=3000000, max=1000, numere naturale
6. N=3000000, max=1000, numere reale
7. N=6000000, max=1000, numere naturale
8. N=6000000, max=1000, numere reale
9. N=10000000, max=1000, numere naturale
10. N=10000000, max=1000, numere naturale, v[3]=5, v[4]=4, v[i]=i+1 pentru orice alt i

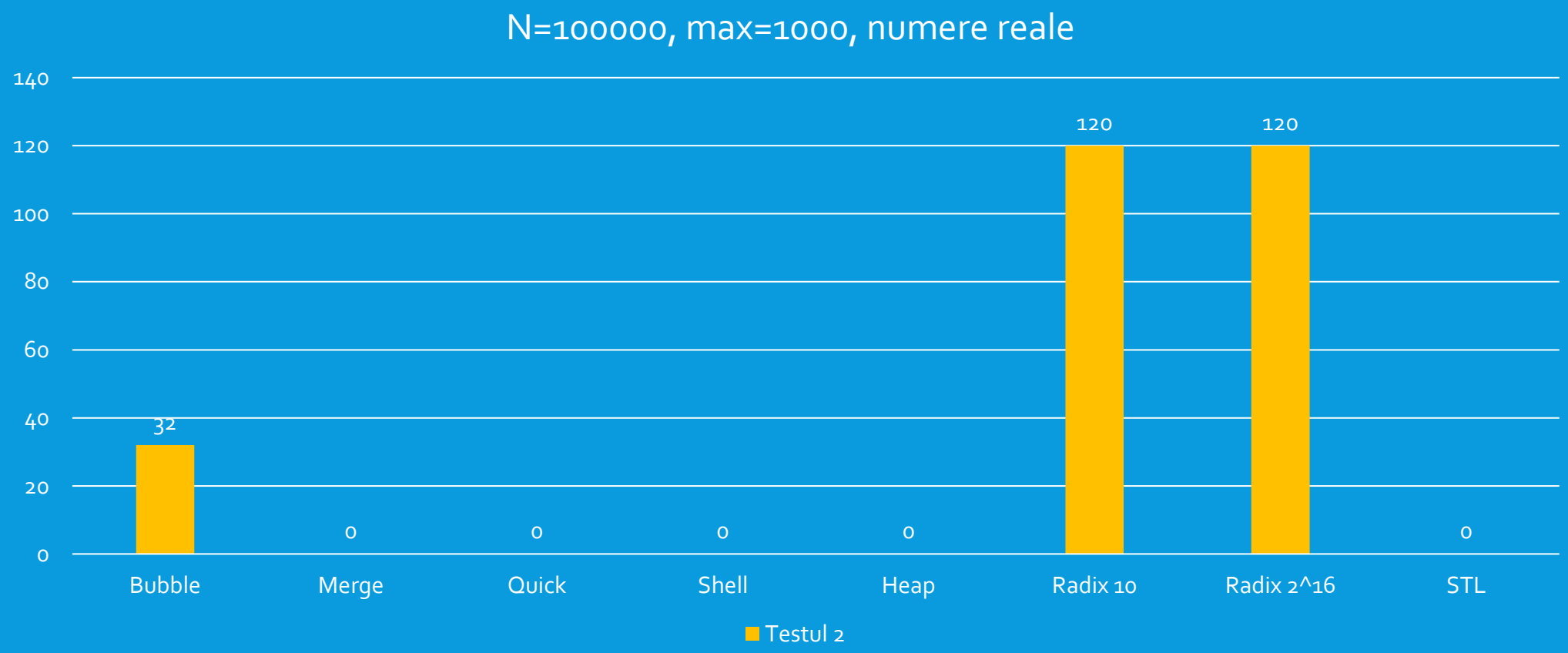
OBSERVAȚII

- Timpul de pe grafice este exprimat în secunde
- Numărul 120 înseamnă că vectorul nu a putut fi sortat în mai puțin de 2 minute cu algoritmul respectiv sau în cazul radix sort 10 și 2^16 că vectorul avea valori reale(am testat radix sort numai pe numere naturale)

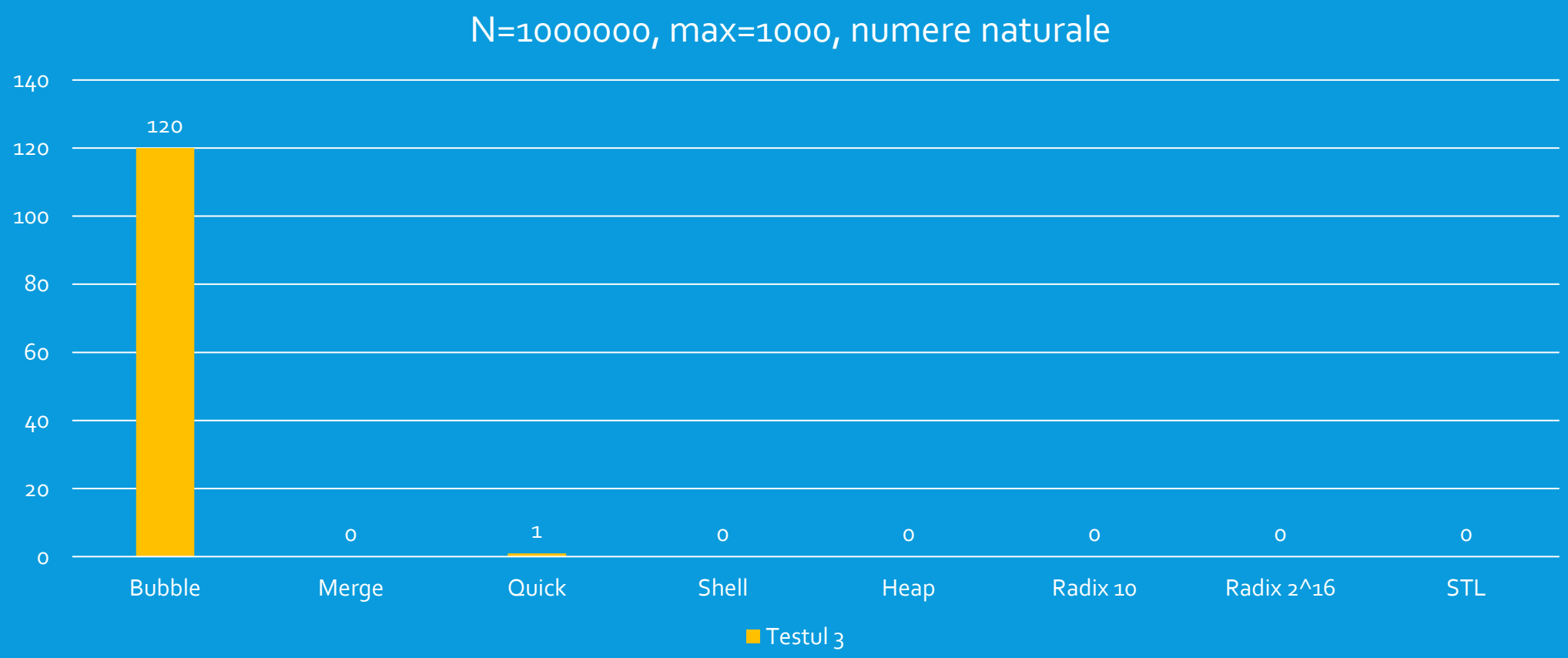
TESTUL 1


N=100000, max=1000, numere naturale
35
30
25
20
15
10
5
0
31
0
0
0
0
0
0
Bubble
Merge
Quick
Shell
Heap
Radix 10
Radix 2^16
STL
Testul 1

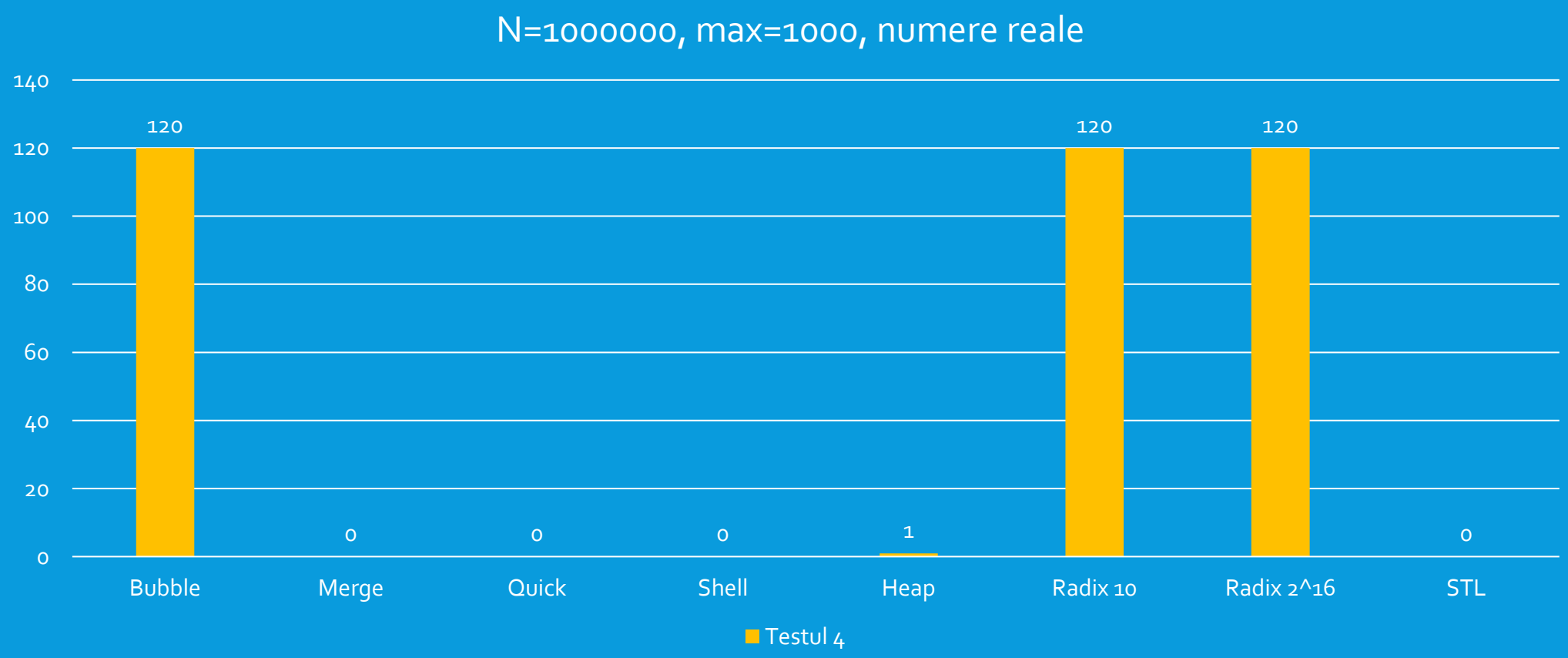
TESTUL 2


N=100000, max=1000, numere reale
140
120
100
80
60
40
20
0
32
0
0
0
0
120
120
0
Bubble
Merge
Quick
Shell
Heap
Radix 10
Radix 2^16
STL
Testul 2

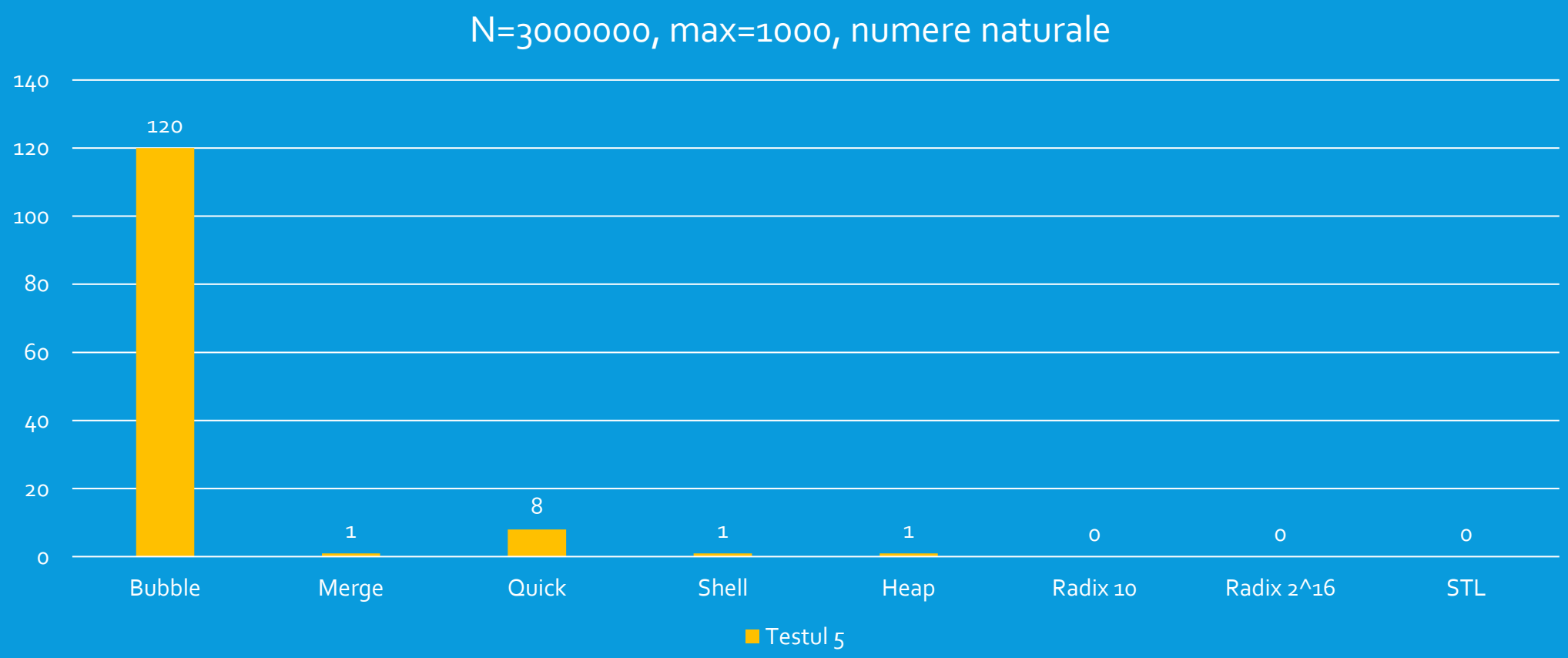
TESTUL 3


N=1000000, max=1000, numere naturale
140
120
100
80
60
40
20
0
120
0
1
0
0
0
0
Bubble
Merge
Quick
Shell
Heap
Radix 10
Radix 2^16
STL
Testul 3

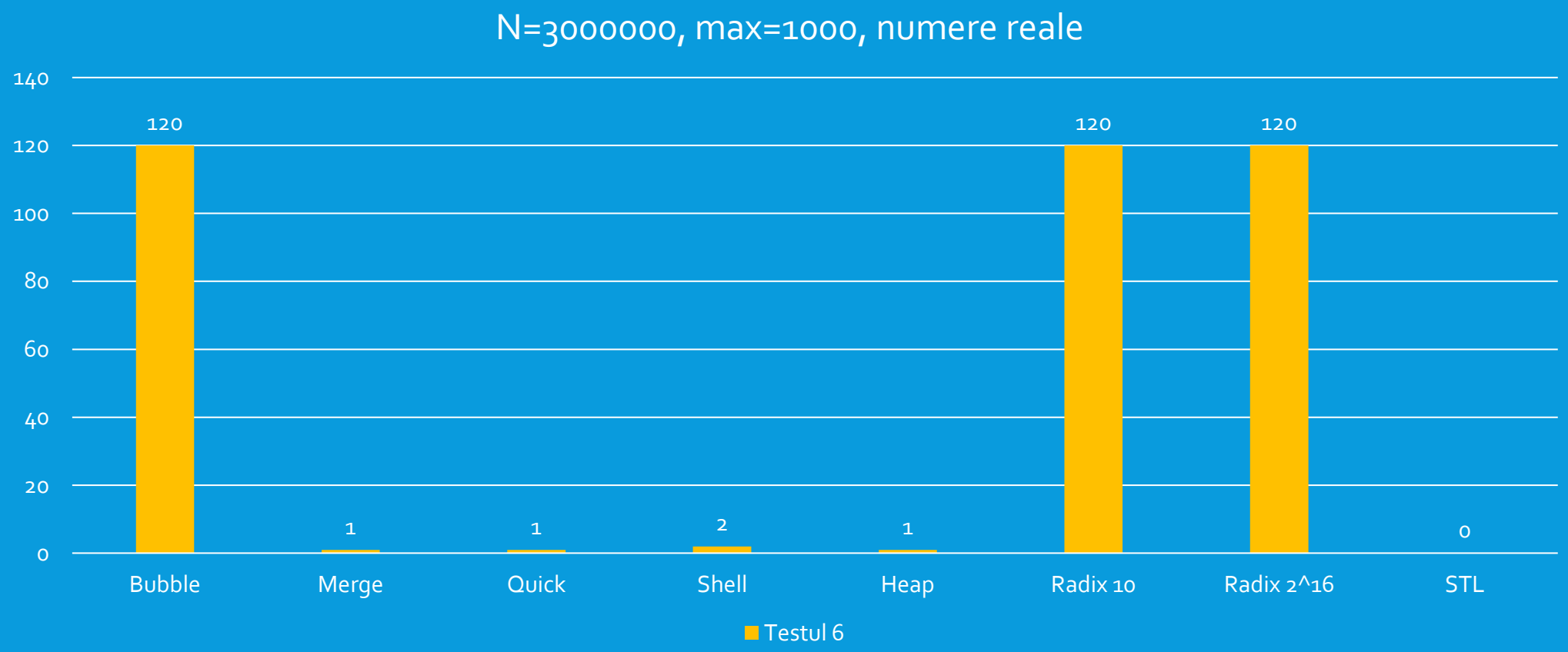
TESTUL 4


N=1000000, max=1000, numere reale
140
120
100
80
60
40
20
0
120
0
0
0
1
120
120
0
Bubble
Merge
Quick
Shell
Heap
Radix 10
Radix 2^16
STL
Testul 4

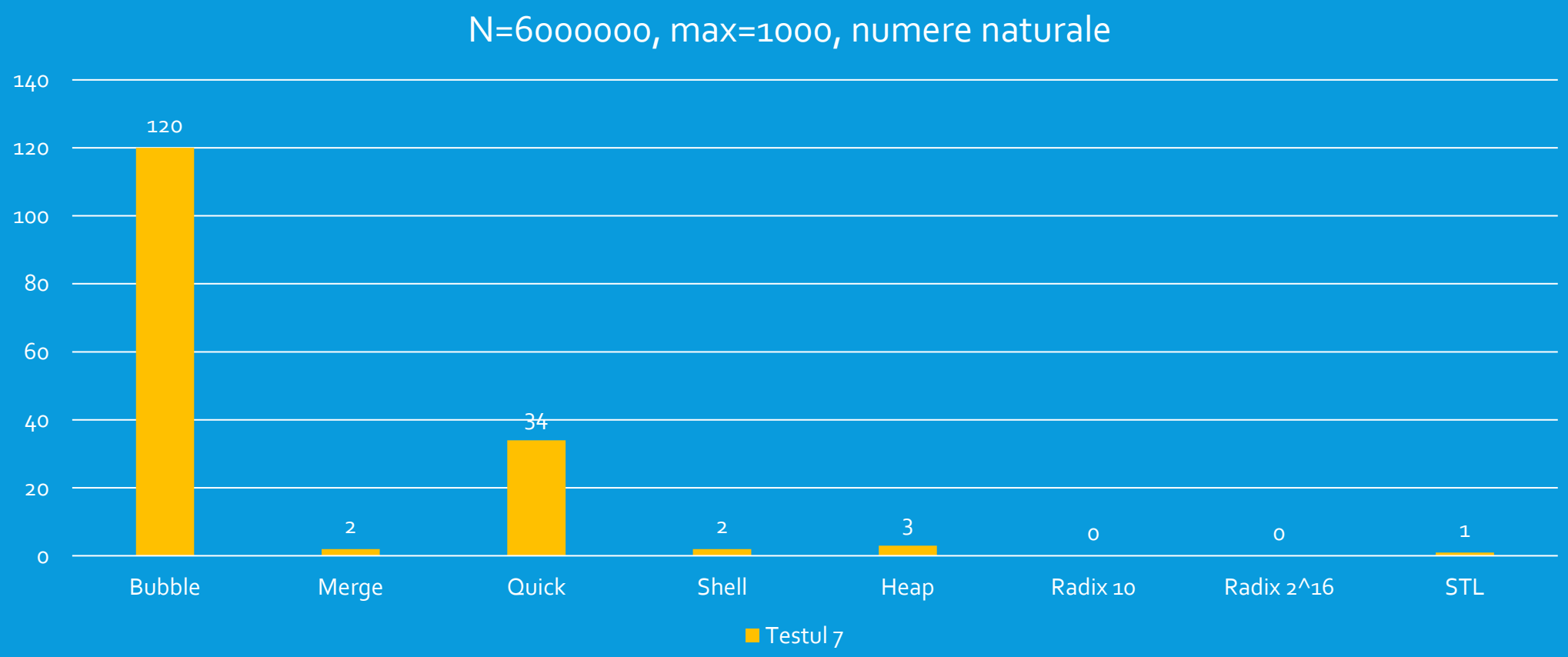
TESTUL 5


N=3000000, max=1000, numere naturale
140
120
100
80
60
40
20
0
120
1
8
1
1
0
0
0
Bubble
Merge
Quick
Shell
Heap
Radix 10
Radix 2^16
STL
Testul 5

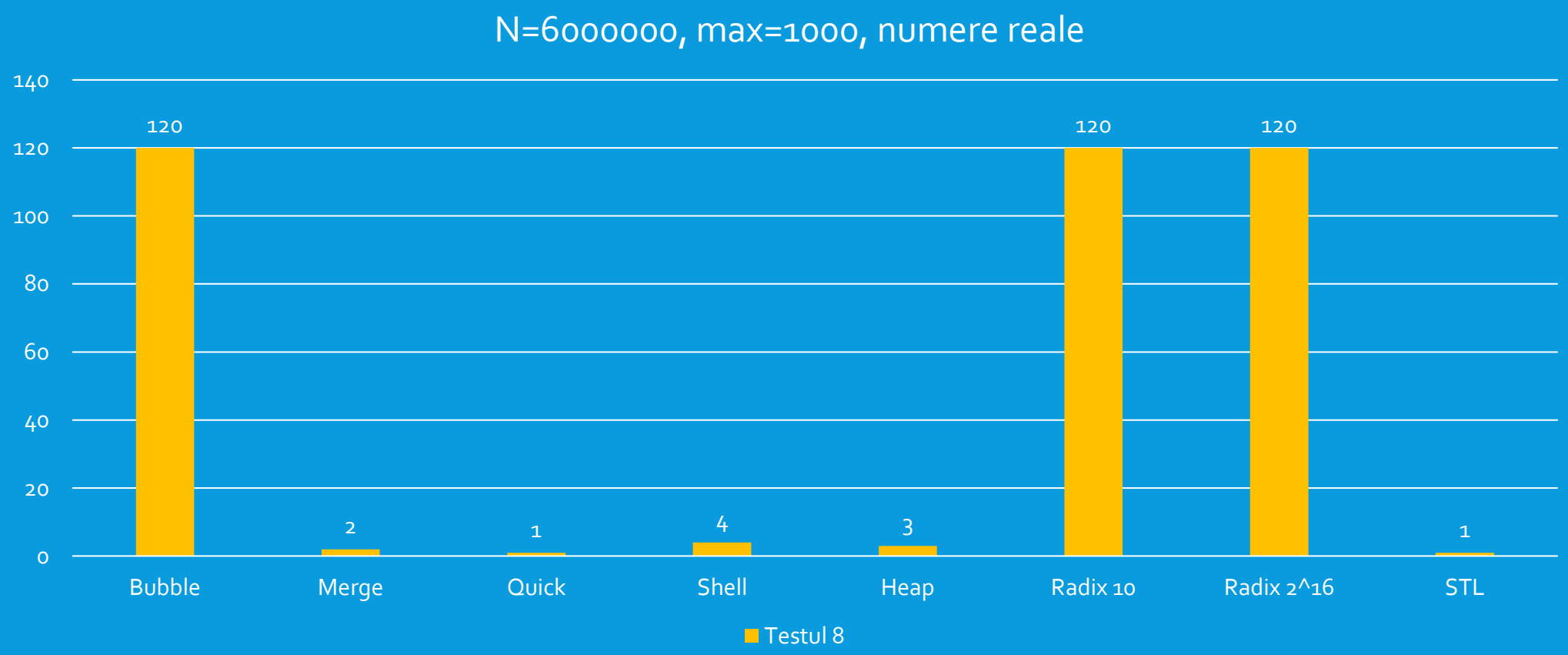
TESTUL 6


N=3000000, max=1000, numere reale
140
120
100
80
60
40
20
0
120
1
1
2
1
120
120
0
Bubble
Merge
Quick
Shell
Heap
Radix 10
Radix 2^16
STL
Testul 6

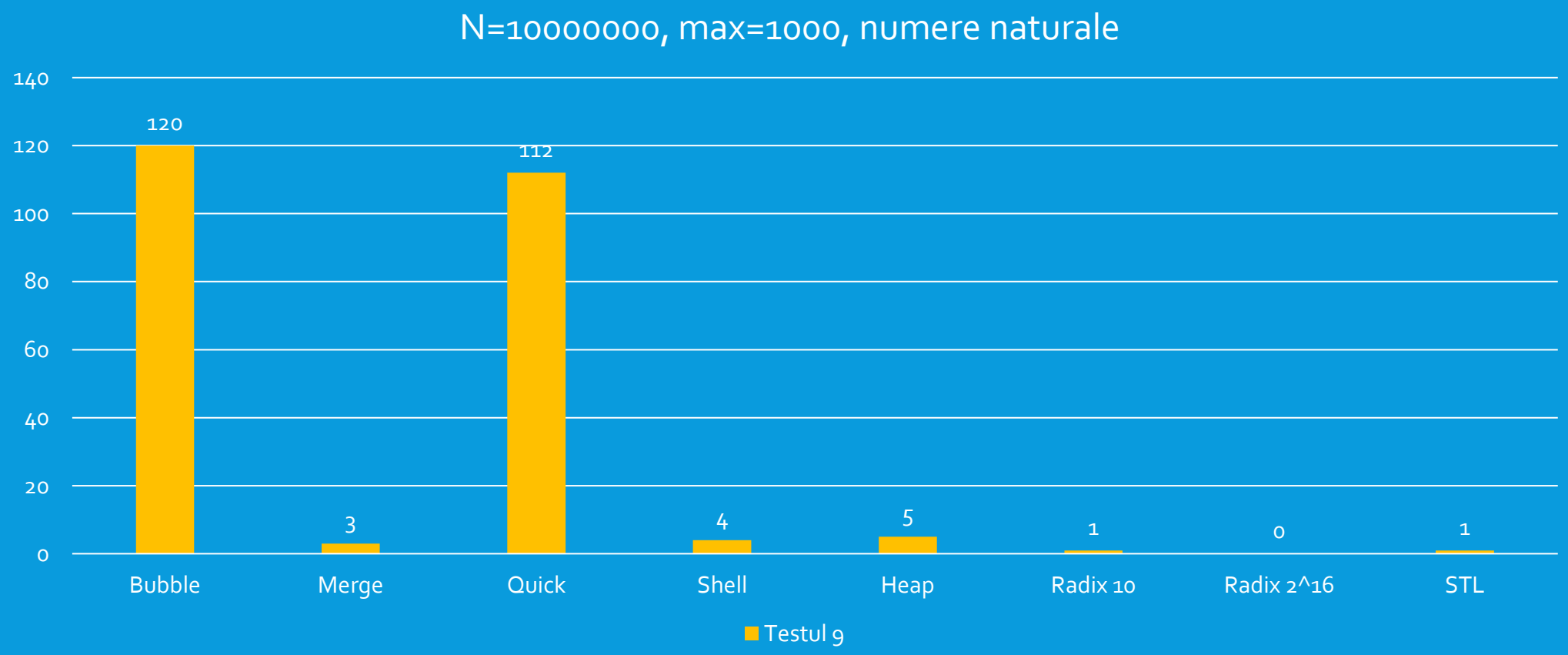
TESTUL 7


N=6000000, max=1000, numere naturale
140
120
100
80
60
40
20
0
120
2
34
2
3
0
0
1
Bubble
Merge
Quick
Shell
Heap
Radix 10
Radix 2^16
STL
Testul 7

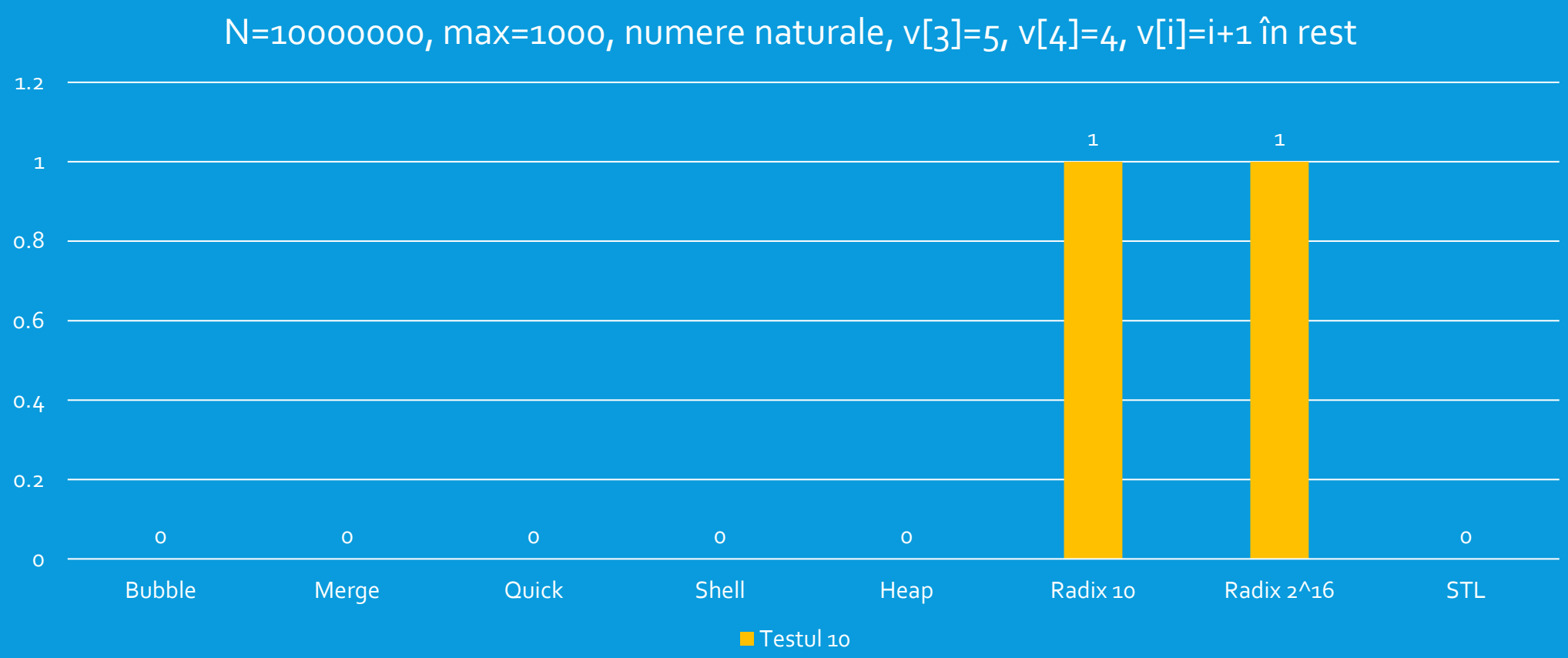
TESTUL 8


N=6000000, max=1000, numere reale
140
120
100
80
60
40
20
0
120
2
1
4
3
120
120
1
Bubble
Merge
Quick
Shell
Heap
Radix 10
Radix 2^16
STL
Testul 8

TESTUL 9


N=10000000, max=1000, numere naturale
140
120
100
80
60
40
20
0
120
3
112
4
5
1
0
1
Bubble
Merge
Quick
Shell
Heap
Radix 10
Radix 2^16
STL
Testul 9

TESTUL 10


N=10000000, max=1000, numere naturale, v[3]=5, v[4]=4, v[i]=i+1 în rest
1.2
1
0.8
0.6
0.4
0.2
0
0
0
0
0
0
1
1
0
Bubble
Merge
Quick
Shell
Heap
Radix 10
Radix 2^16
STL
Testul 10

CONCLUZII

- Bubble Sort este de regulă unul din cei mai slabi algoritmi de sortare, dar în cazul în care vectorul este aproape sortat el devine printre cei mai eficienți(best case: O(n))
- Merge, Quick, Shell și Heap Sort au timpi similari, însă Quick Sort pierde mult timp pe vectorii cu numere naturale din cauza dublurilor
- Radix Sort este mai eficient decât toți algoritmii menționați anterior, iar varianta 2^16 este defapt un counting sort deoarece avem elemente mai mici de 1000(în afară de ultimul test).
- De regulă, algoritmul STL din C++ este cel mai rapid, dar din cauza numerelor mici, Radix Sort 2^16 a reușit să sorteze vectorii mai rapid.

MULȚUMIM PENTRU ATENȚIE!