Minciunescu Tudor-Andrei
Grupa 134

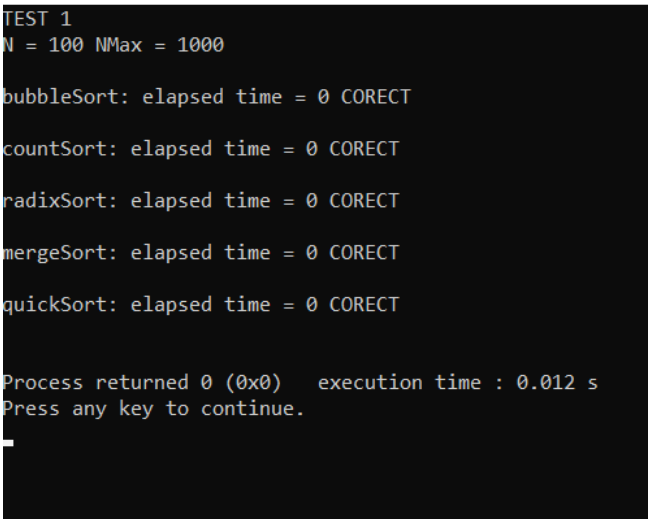
Prezentare algoritmi de sortare

Am implementat in c++ urmatorii algoritmi:

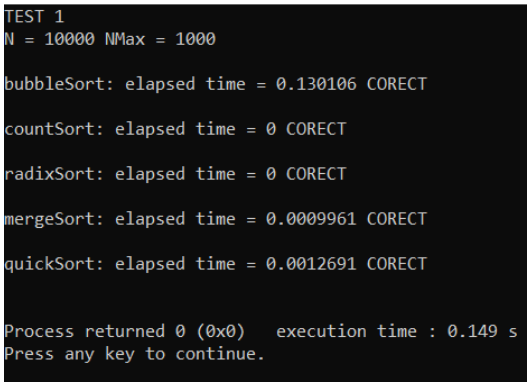
- BubbleSort
- CountSort
- RadixSort
- MergeSort
- QuickSort

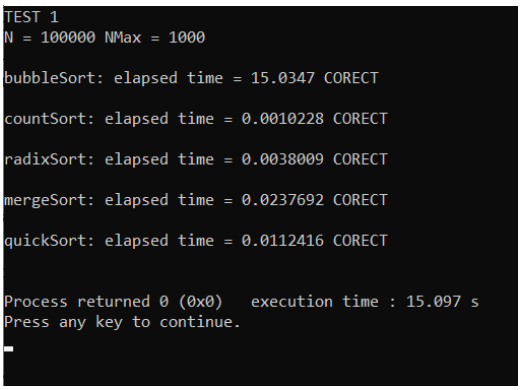
Dupa ce am masurat timpii de rulare pentru fiecare algoritm pe mai multe teste am putut observa diferenta dintre ei in functie de datele de intrare.

Pentru putine numere toti algoritmii sunt foarte rapizi.



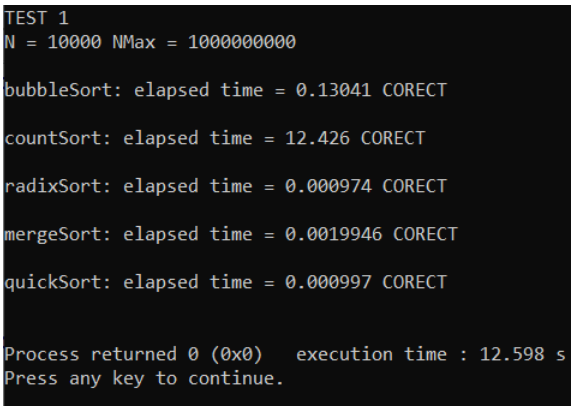
De la 10^4 deja apar diferente intre ei, iar pentru 10^5 bubbleSort avand complexitatea $O(n^2)$ este foarte lent, de aceea nu o sa il las sa sorteze mai mult de atat





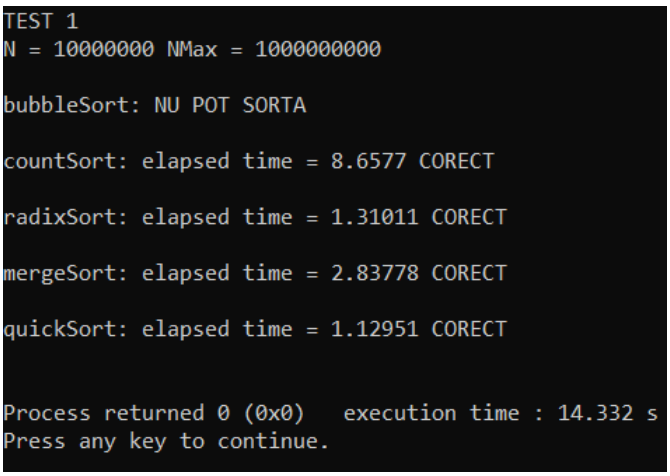
CountSort este foarte rapid pentru numere mici, dar daca marim Nmax creste si timpul de rulare. Acum este mult mai lent chiar si decat bubbleSort.

Complexitatea lui CountSort este O(n+max) pentru timp si O(max) pentru spatiu, ceea ce il face ineficient pentru numere mari.



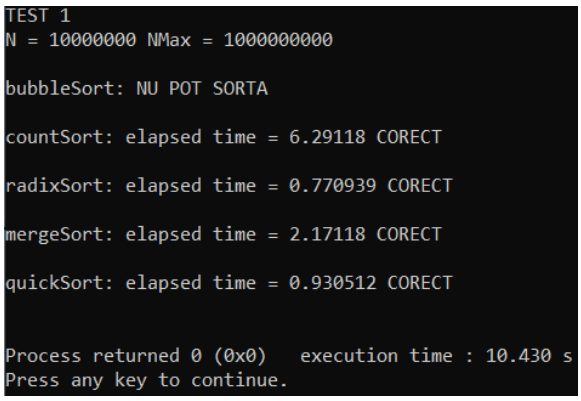
RadixSort (implementat lsd in baza 10) are timpii de executare mai rapizi decat quicksort pentru numere mici, iar pentru numere mari timpii de rulare sunt comparabili, dar radixSort foloseste memorie in plus.

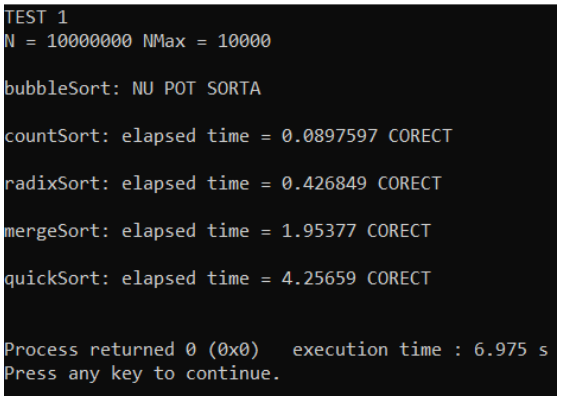
RadixSort are complexitatea O(n log max) pentru timp si O(n+b) pentru spatiu.



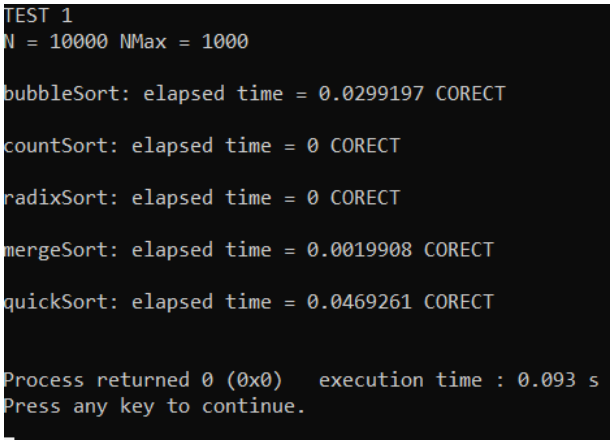
In toate testele pe care le-am facut mergeSort ruleaza mai lent decat radixSort si quicksort, avantajul sau insa este ca are complexitatea O(n log n) indiferent de marimea vectorului sau de intervalul din care apartin numerele.

Astfel, in testele de mai jos vedem ca mergeSort are aproximativ acelasi timp de rulare pentru NMax $= 10^4$ cat si pentru NMax = 10^9 spre deosebire de ceilalti algorimti unde vedem variatii mai mari.





QuickSort are complexitatea medie O(n log n) insa poate varia foarte mult in functie de cum alegem pivotul dupa care este impartit vectorul. In implementarea mea, am ales ca pivot ultimul element iar algoritmul este foarte rapid pe majoritatea testelor de mai sus. Problema apare cand vectorul este constant sau deja sortat deoarece complexitatea ajunge la $O(n^2)$. Mai jos am testat cu doar 10^4 elemente iar quicksort este mai lent chiar si decat bubbleSort, deoarece vectorul este constant.



Avantajul la quicksort este sortarea efectuata in-place, fara a folosi memorie aditionala.

Am comparat timpii de rulare si cu functia sort() din c++ care foloseste introSort. IntroSort este un algoritm hibrid, acesta combinand mai multi algoritmi (quickSort, heapSort, insertionSort).