Scrieti un program, care citeste dintr-un fisier un vector, apelează o functie de sortare si apoi verifica daca vectorul citit a fost sortat corect...

sort (vector<int> &v) {

return;

}

sort(v);

if (verif\_sort ( ????????)){

cout<<”functia de sortare a sortat bine”;

} else cout <<”nasol”;

fisier intrare

3

1 2 3 -> DA

3

1 3 2 -> NU

Trebuie sa fim atenti ca outputul sa aibe același element cu inputul (1 3 2 -> 1 2 3 nu 1 1 1 de exemplu)

qsort cum aleg pivotul ?

* random
* mediana din 3 sau 5
* mediana medianelor
* media aritmetica
* primul / ultimul
* cel din mijloc
* alegem pivotul oricum si daca avem maxim 5/10/20/40/64 elemente sa facem insertion sort..

Bubble sort/insertion sort/ min sort o sortare in O(n^2)... sa dati multe teste cu 100 elemente cu 1000, 10000, 50000, numere random sortate nesortate in forma de munte…

start =time(0)

sort(...)

end = time(0)

cout << (end - start) / 1000

Teste trebuie sa evidentieze cazurile favorabile/defavorabile pentru fiecare algoritm….

Solutia lui [Radu](https://pastebin.com/diJqME0p) a mers pentru un test random cu 1.000.000 elementein

* 5.3 cu pivotul la capat
* 5.8 cu pivot random
* 6.3 cu mediana din 3

Sortat crescator 20.000

* **500 milisecunde** cu pivotul la capat
* 2 cu pivot random
* 3 cu mediana din 3

1 1 1 1 …. 1 de 1.000.000 ori -> N^2 pentru majoritatea implementarilor….