User: teodor_mihai.cotet
Nume: Cotet Teodor Mihai 333CA
Problema 1(1.cpp)

Ideea: generez și testez.

Împart procesoarele în bucăți de câte N, acum fiecare bucata de N procesoare va fi responsabila de o sortare. Fiecare procesor va fi responsabil de un element din vector. Fiecare procesor genereaza o poziție random (intre 0 și N) pe care va scrie elementul de care este responsabil (id % N in cod) într-un vector (vectorul fiind initializat cu -1, am mai mulți vectori pentru fiecare încercare de sortare, fiecare bucata de N procesoare). S-ar putea ca doua procesoare sa scrie în același timp într-o casuța din vector, dar nu contează pentru ca dacă se întâmpla asta, conform principiului cutiei o alta casuța din vector va rămâne fără element scris (avem N procesoare și N elemente, e suficient ca 2 procesoare sa scrie în același loc ca să se strice bijectia) va avea valoarea default, adică -1, și sortarea va fi reperata ca invalida ulterior. (nici nu ma interesează ce se scrie în casuța unde este concurenta).

Apoi fiecare din aceste N procesoare va verifica dacă elementul de care era răspunzător initial este în regulă (adică dacă e mai mare sau egal ca anteriorul și dacă nu este scrisa valoarea default (-1) în vectorul generat). Dacă găsește una dintre aceste condiții falsa va scrie într-un vector (sorted) pe indexul bucatii caruia e asignat 1, adică respectiva bcuata (de N procesoare) a eșuat (sorted e initializat cu 0). Apoi fiecare procesor verifica dacă bucata de N procesoare din care face parte e valida (adică valoarea e 0), și dacă da scrie în varaibila indexSorted valoare indexului bucatii de N procesoare respective. (unde se reține și vectorul sortat).

Numărul maxim de threaduri pentru care am putut sa testez pe stația mea este 30k, este un numar mic, așadar metoda va găsi un vector sortat doar pentru numere N mici (N == 2 sau N == 4, deși la N == 4 nu merge intotdeauna)

Este o metoda probabilista, dacă vrem sa fim siguri ca timpul este O(1) avem nevoie de fix o infinitate de procesoare, nu exista un numar minimal.
Timpii nu am putut să-i masor, intrucat trebuie simulat pe mașini cu multe core-uri.

Problema 2(2.cpp)

Am folosit scheletul de la laborator. Ca sa pot sorta și vectori cu elemente egale am asignat pe lângă valoarea elementului indexul din vectorul initial, ca sa am un criteriu de departajare în caz de egalitate la căutarea binara pentru merge.

Paralelizez mergesortul clasic, pe niveluri, iar la fiecare merge am grija sa intru în functia respectiva cu un numar de threaduri egal cu numărul de elemente. Apoi fiecare thread cauta binar poziția unui element în celălalt vector, ca sa știe unde sa-l pună în vectorul final.

Complexitate:

Dacă merge-ul nu ar fi paralelizat am avea:

$n + n/2 + n/4 + n/8 + ... + 1 = O(n)$

acum ca mergesorul e paralelizat avem:

$\log(n) + \log n/2 + \log n/4 + ... + \log 1 <= \log n + \log n + \log n + ... + \log n$ – ceva $<= \log n * \log n = O((\log n) ^ 2)$ cu n procesoare.

Pentru testare numărul de procesoare trebuie să fie egal cu N. Timpii nu am putut să-i masor, intru-cat trebuie simulat pe mașini cu multe core-uri ca aceștia să fie relevanti.