**Arhitektura računara projektni grupa 2 zadatak 5.**

***Miloš Jaćimović* *1160/17***

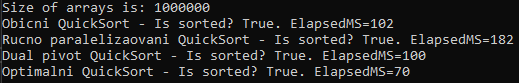
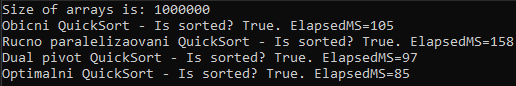
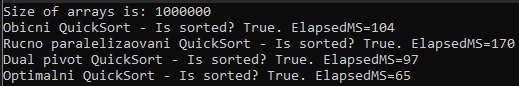
Projekat se sastoji iz 5 klasa.

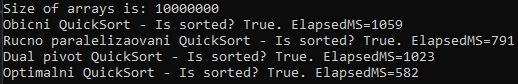
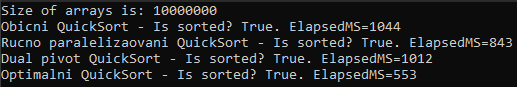
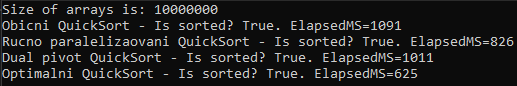
* *StandardQuickSort* sadrži samo obični QuickSort bez optimizacije.
* *ParallelQuickSort* slično prethodnoj klasi sadrži QuickSort, ali je urađena optimizacija tako što su rekurzivni pozivi paralelizovani metodom System.Threading.Tasks.Parallel.Invoke. Ovdje sam uveo i parametar maxDepth da bi ograničio dubinu paralelizacije. U slučaju da ne postoji nikakvo ograničenje program će kreirati 2 nova taska za svako dijeljenje niza, te nas dovodi do lošijih perfomansi u odnosu da obični Quick Sort.
* *DualPivotQuickSort* je optimizacija korištenjem dva pivota.
* *OptimalQuickSort* prestavlja kombinaciju prethodne dvije optimizacije (tri rekurzivna poziva DualPivotQuickSort-a su paralelno pozvana metodom System.Threading.Tasks.Parallel.Invoke i uveden je parametar maxDepth iz gore navedenih razloga)
* *Utility* sadrži generator random niza intova i provjeru da li je niz sortiran.

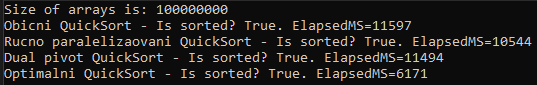
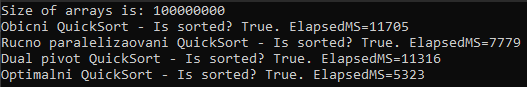
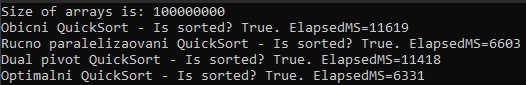
**Mjerenja:**

Mjerenje je vršeno za 1 000 000, 10 000 000 i 100 000 000 elemenata tipa int koji su nasumično generisani. Svako mjerenje izvršeno je triput te je ispis svakog od njih prikazan ispod, dok je u grafičkom prikazu prikazano prosječno vrijeme za sve veličine nizova. Mjerenja su vršena sa kompajlerskom optimizacijom i bez nje. maxDepth parametar u ovim mjerenjima postavljen je na 4.

**Rezultati mjerenja sa kompajlerskom optimizacijom**:

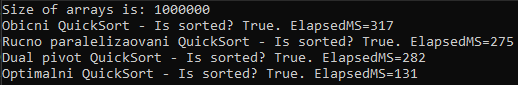
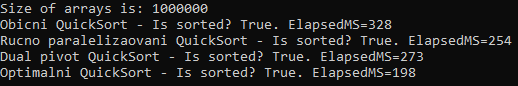
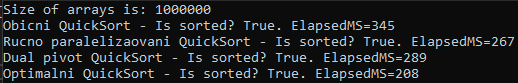


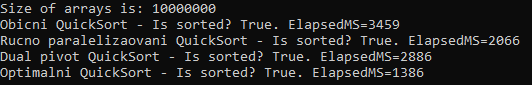
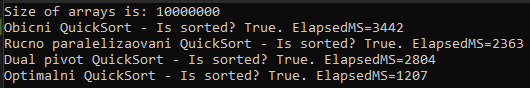
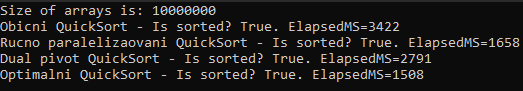


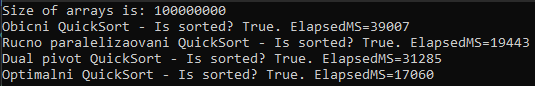
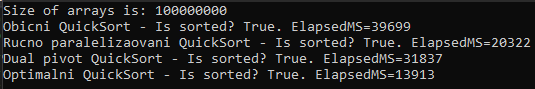
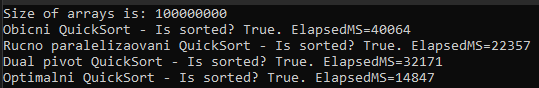


**Grafički prikaz mjerenja sa kompajlerskom optimizacijom:**

**Rezultati mjerenja bez kompajlerske optimizacije**:

****

****

****

**Grafički prikaz mjerenja bez optimizacije kompajlera:**

**Komentar na mjerenja:** Paralelizovani QuickSort tek nakon otprilike 1 000 000 elemenata donosi bolje perfomanse u odnosu na obični QuickSort zato što je sam poziv metode Parralel.Inovke realtivno skup, a zbog malog broja elemenata i njihovog tipa brzo se sortiraju i sa običnim algoritmom. Kompajlerska optimizacija je očekivano donijela bolje perfomnase sve četiri algoritma. Takođe očekivano Optimalni QuickSort (kombinacija paralelnog algoritma i dual pivota) je postigla najbolje perfomanse u svakom mjerenju bez izuzetka.

**Perfomanse računara na kojem je vršeno mjerenje:**

