**3. Laboratorijska vježba**

# Algoritmi sortiranja

U datoteci *sortiranja.c* se nalaze različite implementacije algoritma za sortiranje (insertion, selection i quick sortiranje). Uz to ima i nekoliko pomoćnih funkcija (za generiranje niza, mjerenje izvođenja funkcije sortiranja i neke pomoćne funkcije). Potrebno je napraviti projekt sa *main.c* i *sortiranja.c* datotekama. U projekt dodati i implementirati *sortiranja.h* koja će sadržati sve potpise funkcija koje će se koristiti u *main.c* .

U funkciju *partition* je potrebno dodati odabir pivota tako da se provjere vrijednosti na indeksima *0, n/2 i n-1* i uzme se srednja vrijednost među njima (medijan). Tu vrijednost se zamijeni sa vrijednošću na indeksu 0. U *sortiranja.c* i *sortiranja.h* dodati globalnu boolean varijablu *choose\_pivot* tako da se biranje pivota može isključiti/uključiti iz main funkcije po potrebi. Izmjeriti vremena izvođenja quicksorta sa i bez odabira pivota za slučajni i za sortirani niz dužine *n = [1000 .. 10000]* sa korakom 1000 (po potrebi uvećati sve vrijednosti 5-10 puta da bi dobili izmjerljiva vremena). Ideja je dokazati da, za sortirane nizove, vrijeme sortiranja sa odabirom pivota je zaista u skladu sa srednjom složenosti quicksorta, a bez biranja pivota se približava kvadratnoj složenosti (najgori slučaj osnovnog algoritma).

Dodati novu, izmijenjenu, quicksort funkciju tako da za neki n\_min (dužina primljenog niza) zove selectionsort ili insertionsort funkciju umjesto quicksort funkcije. Pronaći optimalni n\_min mjerenjem vremena izvođenja za različite vrijednosti u rasponu *[0 .. 200]* sa korakom 5 (prilagoditi po potrebi). Da bi mijenjali n\_min, potrebno je u *sortiranja.c* i *sortiranja.h* dodati globalnu deklarirati globalnu varijablu n\_min koja će se mijenjati u main funkciji prije poziva prilagođenog quicksorta. Ideja je da se za vrlo kratke nizove poziva „slabiji“ algoritam sortiranja i da ukupno izvođenje sortiranja bude što brže. Ako se slabiji algoritam poziva za prekratke nizove, neće biti velike koristi jer će quicksort uzeti više vremena, a za preduge nizove će sporost slabijeg algoritma prevladati.

*Dodatno*:

Implementirati merge i mergesort funkcije. Pomoćne nizove možete alocirati/oslobađati po potrebi. Izmjeriti vrijeme izvođenja implementirane funkcije za slučajni i sortirani niz dužine *n = [1000 .. 10000]* (prilagoditi po potrebi kao i prije).

***Prilikom testiranja se može dogoditi prevelik broj rekurzija (quicksort na već sortiranom nizu bez biranja pivota) pa je potrebno povećati stack size u postavkama projekta: Properites / Linker / System / Stack Reserve Size postaviti na 16000000.***