**4. Laboratorijska vježba**

# Mjerenje vremena izvođenja algoritma (a posteriori)

Preuzeti iduće funkcije sa primjera na predavanjima:

* *generiraj(n)* – generira niz dužine *n* sa slučajnim elementima (svaki element se pojavljuje samo jednom u nizu)
* *shuffle(skup)* – miješa elemente u nizu *skup*

Napisati iduće funkcije:

* *presjek(skupA, skupB)* – funkcija računa dužinu presjeka dva skupa
* *presjek\_jedan\_sortiran(skupA, skupB)* – funkcija računa dužinu presjeka dva skupa, gdje je skupB sortirani niz tako da se funkcija oslanja na upotrebu upotrebom *bsearch()* funkcije (iz stdlib.h)
* *presjek\_oba\_sortirana(skupA, skupB)* – funkcija računa dužinu presjeka dva skupa, gdje je su oba skupa sortirana i moguće je samo ići naprijed sa indeksima po oba skupa
* *presjek\_po\_indeksima(skupA, skupB)* – funkcija kreira dva nova niza (indexA i indexB) i ispunjava ih sa 0. Dužine nizova su određene najvećim brojem u skupA i skupB. Zatim se za svaki broj iz skupA upiše 1 na njegov indeks u njegovom „indeks“ nizu indexA. Isto se napravi i za skupB. Zatim se u novi niz (presjek) broje samo brojevi koji se pojavljuju u oba niza na istom indeksu odnosno imaju 1 u indexA i indexB nizovima.

Skupovi su predstavljeni dinamički alociranim nizovima.

Izmjeriti vrijeme izvođenja svake pojedine funkcije za veličine skupova od 100K do 3M (s korakom 300K) brojeva i nacrtati graf *dužina/vrijeme* za svaku funkciju.

Oba skupa će imati istu veličinu, ali možda će biti potrebno malo prilagoditi veličine skupova da se dobiju izmjerljiva vremena.

Kod funkcija gdje se skup sortira, u vrijeme izvođenja ulazi i sortiranje nizova sa *qsort*() funkcijom (iz stdlib.h).

Nizove koji predstavljaju skupove alocirati i oslobađati za svako mjerenje (ne ulazi u vrijeme mjerenja).

Za mjerenje izvođenja pojedinog algoritma koristiti funkciju *clock()*.

Parametre funkcije proširiti sa dužinom niza.

Za svaku funkciju će se moći ispisati vremena, a graf možete složiti u Excelu.

Provjerite da rast vremena izvođenja ugrubo odgovaraju očekivanoj složenosti svakog algoritma. Objasniti eventualne razlike.