

# ISPITIVANJE KVALITETA SOFTVERSKI GENERISANIH SEGMENTATA U OBLASTI VREMENSKE SLOŽENOSTI ALGORITAMA ZA AUTOMATIZOVANO SASTAVLJANJE ISPITA

**Dorđe Pešić<sup>1</sup>, Jelica Protić<sup>2</sup>, Milena Vujošević Janičić<sup>3</sup>, Marko Mišić<sup>4</sup>**

<sup>1,2,4</sup>Univerzitet u Beogradu, Elektrotehnički fakultet, Beograd, Srbija

<sup>3</sup>Univerzitet u Beogradu, Matematički fakultet, Beograd, Srbija

<sup>1</sup>djpjesic@yahoo.com, <sup>2</sup>jeca@etf.bg.ac.rs, <sup>3</sup>milena@matf.bg.ac.rs, <sup>4</sup>marko.misic@etf.bg.ac.rs

**Kratak sadržaj:** U ovom radu prikazali smo ispitivanje kvaliteta rezultata softvera koji generiše programske segmente i računa njihovu vremensku složenost. Segmenti se koriste kao ispitna pitanja, za potrebe predmeta Programiranje 1, na prvoj godini Elektrotehničkog fakulteta, Univerziteta u Beogradu. Studenti određuju vremensku složenost tih segmenata. Za parametre kvaliteta uzeta je raspodela broja segmenata po grupama segmenata iste složenosti i međusobna sličnost segmenata unutar jedne grupe. Dobijena je relativno neravnomerna raspodela, dok je međusobna sličnost unutar jedne grupe zadovoljavajućeg kvaliteta, sa prostorom za dalja unapređenja.

**Ključne reči:** automatsko sastavljanje testova, algoritmi, analiza složenosti

## TESTING THE QUALITY OF SOURCE CODE SEGMENTS IN THE FIELD OF TIME COMPLEXITY ASSEMBLED BY SOFTWARE FOR AUTOMATED EXAM PREPARATION

**Abstract:** In this paper, we measure the quality of source code segments, assembled by software. The software also calculates their time complexity. Segments are used as exam questions for the Programming 1 course, at the first year of the University of Belgrade, School of Electrical Engineering. During the exam, students determine the segments time complexity. Quality measurement parameters are the distribution of the number of segments by groups of segments with the same complexity, and the similarity of segments within one group. We got relatively non-uniform distribution, while the similarity measurements are satisfactory, with the possibility for improvements.

### 1. UVOD

Sa porastom broja studenata i povećanim brojem ispita i kolokvijuma na Elektrotehničkom fakultetu Univerziteta u Beogradu, javila se potreba da se ispitna pitanja generišu softverskim putem, kako bi se unapredila efikasnost sastavljanja i pregledanja testova. Postoje razna rešenja za automatsko sastavljanje i pregledanje ispitnih pitanja. Ona mogu biti bazirana na odabiru iz gotove baze pitanja, uz vođenje računa o raznim parametrima (oblast, težina, ponavljanja na prethodnim ispitima). Ovakvi softverski sistemi često koriste veštačku inteligenciju, logičko programiranje i genetičke algoritme [1]. Takođe se mogu koristiti parametrizovana pitanja [2], kako bi se povećao skup pitanja koji se može dobiti generisanjem.

Na predmetu Programiranje 1, na prvog godini Elektrotehničkog fakulteta, izučavaju se teme iz osnova programiranja, među koje spada i vremenska složenost algoritma. Ona se izučava na rudimentarnom nivou, sa ciljem da studenti steknu svest o postojanju ograničenja u resursima i potrebi vođenja računa o efikasnosti programa. Ispitni zadaci obično sadrže segment programskog koda, izrađen od kontrolnih struktura i naredbi dodele vrednosti, za koji studenti treba da odrede vremensku složenost. Ove zadatke trenutno sastavlja nastavno osoblje, dok je pregledanje delimično automatizovano. Ceo ispit, koji sadrži i pitanja iz ostalih oblasti kursa, radi se na papiru, ali je štampan u posebnom formatu, koji omogućava da se popunjen ispit skenira, da bi se konačno bodovanje izvršilo softverskim putem [3].

Da bi se omogućila automatizacija sastavljanja ispitnih pitanja iz oblasti vremenske složenosti, razvija se softver koji generiše programske segmente, poznate vremenske složenosti [4]. Softver je izgrađen od nekoliko različitih komponenti [5], [6]: grafičkog interfejsa, baze segmenata od kojih započinje generisanje, jezgra koje je zaduženo za generisanje jednog segmenta i računanje njegove složenosti, komponente za završnu obradu generisanog segmenta, i komponente za generisanje skupa segmenata, koja koristi sve ostale komponente i predstavlja najviši sloj softvera. U nastavku sledi opis algoritma za generisanje skupa segmenata.

### 2. ALGORITAM GENERISANJA SKUPA SEGMENTATA

Ulazni podaci za algoritam su: maksimalan broj petlji koje smeju da sadrže generisani segmenti (maxM), željeni broj generisanih segmenata i skup segmenata koji predstavljaju osnovu za generisanje (različite vrste

kontrolnih struktura i dodela vrednosti). Prvi korak algoritma je da se na osnovu početnog skupa segmenata generišu svi mogući segmenti koji imaju jednu petlju. Najzastupljeniji su segmenti linearne složenosti, za koje je utvrđeno da se u zadacima koje su sastavljali ispitivači pojavljuju češće od segmenata sa drugačijom složenosti.

Sledeći korak algoritma generiše segmente sa više petlji, korišćenjem različitih metoda. Jedna od tih metoda generiše segmente bez zavisnosti po podacima između petlji, dok ostale metode generišu segmente sa zavisnostima po podacima između petlji. U ovom slučaju, segmenti koji imaju zavisnosti po podacima mogu imati maksimalno dve petlje i opis načina njihovog generisanja prevazilazi okvire ovog rada. Segmenti bez zavisnosti po podacima generišu se kombinovanjem. Neka je skup segmenata sa  $M$  petlji definisan kao tekući skup ( $M=2, \dots, \max M$ ). Slučajno se izabere jedan segment sa jednom petljom i ukrsti se sa svim segmentima iz tekućeg skupa. Ovako se dobija skup segmenata sa  $M+1$  petljom, koji postaje tekući skup. Ova procedura se ponavlja dok se ne dostigne  $\max M$  petlji u segmentima. Ukrštanje je slučajno odabrana operacija između ugnežđavanja, sekvenciranja ili selekcije.

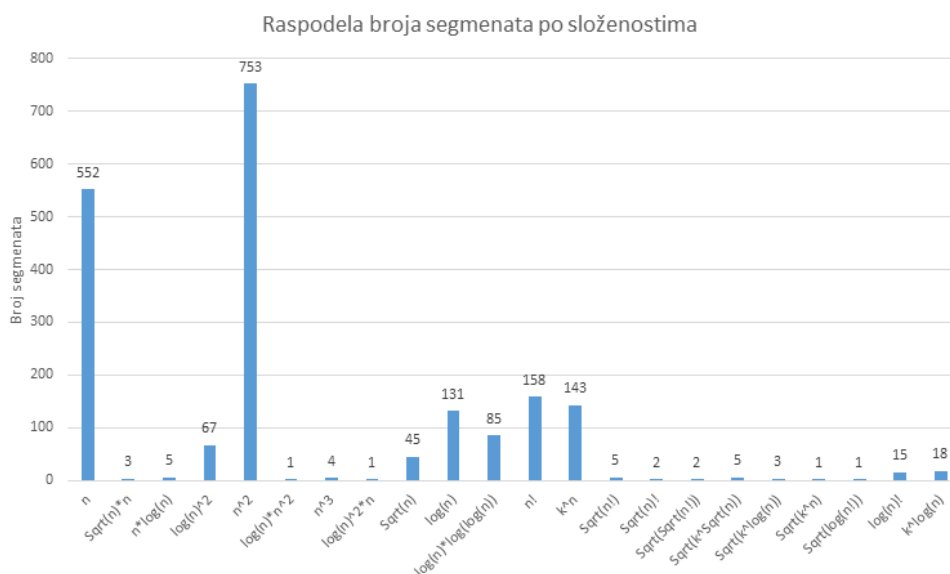
### 3. REZULTATI GENERISANJA

Svako pokretanje softvera za generisanje daje drugačiji skup segmenata. Broj segmenata koji se može dobiti u jednom pokretanju kreće se između 2000 i 3000, što su ograničenja uvedena samo za potrebe testiranja. Generisani segmenti se grupišu po vremenskoj složenosti. Korisnik sistema na kraju odabira neke od generisanih segmenata. Broj izlaznih segmenata je ograničen, zato što se prilikom jednog pokretanja izvršava jedan od mogućih scenarija, koji je dobijen kombinacijom slučajnih parametara u samom programu.

Za meru kvaliteta generisanih segmenata, odabrali smo raspodelu broja segmenata po grupama i stepen međusobne sličnosti segmenata unutar jedne grupe. Program treba da ima što ravnomerniju raspodelu broja segmenata po grupama segmenata iste složenosti, da bi imali ravnoopravan izbor segmenata različite složenosti. Unutar jedne grupe, potrebno je proceniti međusobnu sličnost, da bi ispitivač imao uvid u realan izbor više segmenata iste složenosti.

Za merenje sličnosti dva segmenta koriste se alati za detekciju sličnosti u programskom kodu, čiji je primarni domen otkrivanje plagijarizma. Najpoznatiji alati su Moss i JPlag i za naše potrebe odabran je JPlag. JPlag je kompletna open-source aplikacija koja se može pokretati na lokalnom računaru. On poredi segmente tako što koristi algoritme za poređenje stringova. U naš program je integrisan kao biblioteka.

JPlag koristimo tako što ga pokrećemo nad grupama segmenata iste složenosti. Segmenti su zapisani u pseudokodu koji prati C sintaksu, pa se koriste semantičko (procesiranje C parserom) i čisto tekstualno poređenje. Za svaku grupu segmenata, dobijaju se dve matrice sličnosti, pošto se segmenti porede svaki sa svakim. Sledeći korak je obrada matrice sličnosti. Želimo da izvršimo klasterizaciju unutar grupe segmenata po stepenima sličnosti. Prvo se vrši klasterizacija po semantičkoj sličnosti. Kada se oforme klasteri, svaki od njih se još jednom klasterizuje po tekstualnoj sličnosti. Na ovaj način se dobija uvid u to koliko se generiše semantički različitih grupa segmenata iste složenosti, a takođe se izdvajaju i podgrupe segmenata koje sadrže segmente visoke međusobne semantičke sličnosti koji su tekstualno različiti.



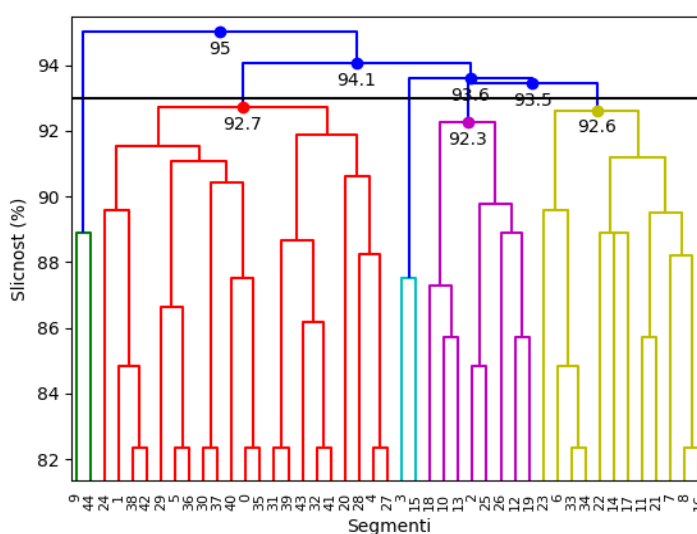
Slika 1. Raspodela broja segmenata po složenostima

Kada se rezultati ovako klasterizuju, sastavljač ispita može da uspostavi različite scenarije odabira segmenata, a moguće je i iskoristiti rezultate za automatsko, računarski podržano sastavljanje testova, gde svaki student dobija drugačije pitanje (Computer Aided Testing, CAT) Na taj način mogu se fokusirati pitanja vezana za istu složenost na odabranu grupu studenata.

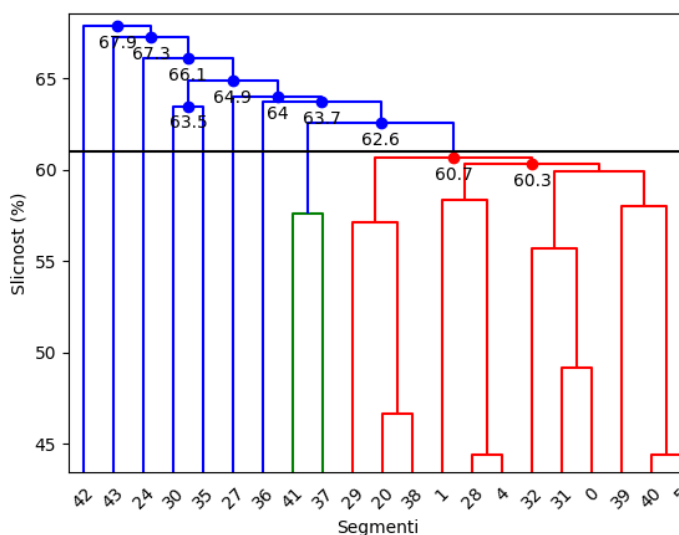
U nastavku ce biti prikazana analiza jednog pokretanja program. Željena veličina izlaznog skupa je postavljena na 2000, dok je maksimalni broj petlji u jednom segmentu postavljen na 3. Slika 1 prikazuje raspodelu broja segmenata po različitim vremenskim složenostima (konstanta  $k$  se može odabrati u softveru).

Može se primetiti da prikazana raspodela dosta neravnomerna. Određene složenosti su daleko zastupljenije, dok se neke druge složenosti nalaze u zanemarljivom broju. Ovo je posledica činjenice da za složenosti koje su zastupljenije (kao što je linearna), postoji više mogućih načina za njihovo dobijanje. Samim tim, kada se odabiraju slučajni parametri koji učestvuju u radu programa, veća je verovatnoća da se generišu zastupljenije složenosti. Složenosti koje nastaju od zastupljenijih, su takođe brojnije (kao što je  $n^2$ ).

Proces klasterizacije po semantičkoj sličnosti, prikazan je nad grupom od 45 segmenata, složenosti  $\text{Sqrt}(n)$ . Koristi se hijerarhijska UPGMA klasterizacija [7], čiji je rezultat dendrogram, prikazan na slici 2. Sa ovog dendograma se može primetiti da se skup može podeliti u 5 velikih klastera. Prag sličnosti je 93 procenata.. Dobijeni klasteri su dalje klasterizovani po tekstualnoj sličnosti. Na slikama 3 i 4, prikazani su novi dendogrami za prva dva najveća klastera. Za oba dendograma, pravogi sličnosti su 61%. Sa slike 3 se vidi da unutar prvog najvećeg klastera postoji čak 9 potklastera, što govori da postoji приметnija tekstualna različitost između segmenata, koji su grupisani kao semantički slični. Sa slike 4 se vidi da su unutar drugog najvećeg klastera, izolovana su 3 potklastera, i da je ovde međusobna tekstualna sličnost nešto veća.

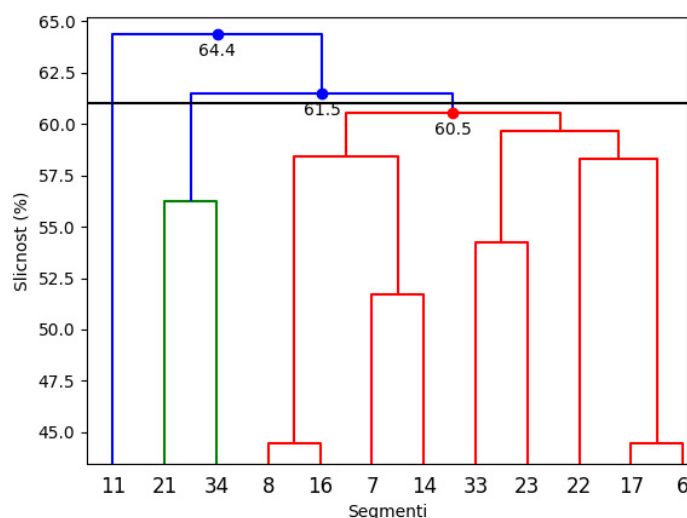


Slika 2: Hijerarhijski dendrogram skupa segmenata složenosti  $\text{Sqrt}(n)$



Slika 3. Hijerarhijski dendrogram najvećeg klastera segmenata semantičke složenosti

Na osnovu ovih rezultata, može se primetiti da postojeći načini generisanja segmenata postižu visoku semantičku sličnost i nešto manju tekstualnu sličnost.



Slika 4. Hijerarhijski dendrogram drugog najvećeg klastera segmenata semantičke

#### 4. ZAKLJUČAK

U ovom radu prikazali smo ispitivanje kvaliteta softvera za generisanje programskih segmenata, koji se koriste kao ispitna pitanja na ispitu iz predmeta Programiranje 1, na prvoj godini Elektrotehničkog fakulteta, Univerziteta u Beogradu. Rezultati generatora su zadovoljavajućeg kvaliteta, jer postoji primetna razlika između semantičke i tekstualne sličnosti generisanih segmenata, kao i većeg broja različitih vremenskih složenosti koje je moguće generisati.

Algoritam za generisanje može se unaprediti na više načina. Da bi se dodatno povećao kvalitet generisanih segmenata, potrebno je postići ravnomerniju raspodelu broja segmenata po složenostima. Moguće je dodavanje novih metoda za generisanje segmenata, uz korišćenje različitih složenosti koje u ovom eksperimentu nisu zastupljene. Takođe, kao kriterijum prekidanja generisanja novih segmenata može se koristiti vreme rada generatora. Ocena kvaliteta generisanih segmenata dobijena opisanim postupkom može se uporediti sa rezultatima koje generiše kontrolna grupa studenata u dužem vremenskom periodu.

#### 5. LITERATURA

- [1]. Andrija Bošnjaković, Jelica Protić, Dragan Bojić, Igor. Tartalja, *Automating the Knowledge Assessment Workflow for Large Student Groups: A Development Experience*, International Journal of Engineering Education, Vol. 31, No. 4, 2015, pp. 1058 – 1070.
- [2]. Hanneke Geerlings, Wim J. van der Linden, Cees A. W. Glas, *Optimal test design with rule-based item generation*, Applied psychological measurement 37, no. 2 (2013): 140-161.
- [3]. Marko Mišić, Marko Lazić, Jelica Protić, *A software tool that helps teachers in handling, processing and understanding the results of massive exams*, 5th Balkan Conference in Informatics, pp. 259-262, ACM New York, NY, USA, Novi Sad, Serbia, Sep, 2012.
- [4]. Đorđe Pešić, Sofija Purić, Marko Mišić, Jelica Protić, *Softversko generisanje pitanja i odgovora iz oblasti analize složenosti algoritama za testove na kursovima programiranja*, XXII Skup Trendovi razvoja: Nove tehnologije u nastavi, Zlatibor, 16.-19.02.2016, pp. 53-56.
- [5]. Đorđe Pešić, Marko Mišić, Jelica Protić, Milena Vujošević Janičić, *The Program Segment Assembling System for Examination in the Field of Algorithmic Time Complexity*, ETRAN 2017, Kladovo, June 2017, pp. RT3.1.1 – 6. (In Serbian).
- [6]. Đorđe Pešić, Marko Mišić, Jelica Protić, Milena Vujošević-Jančić, *Prototype implementation of segment assembling software*, Serbian Journal of Electrical Engineering 15, no. 1 (2018): 71-83.
- [7]. Anil K Jain, Richard C. Dubes. "Algorithms for clustering data." (1988).