0.1 Uvod u algoritme 2024/2025, ispitni rok - primer

Ime, prezime i broj indeksa: _____
Ime profesora: Filip Marić ili Predrag Janičić _____

1. (a) Šta je to verifikovanje programa?
   (b) Šta je to testiranje?
   (c) Da li se testiranjem može dokazati prisustvo grešaka u programu da/ne? Da li se testiranjem može dokazati odsustvo grešaka u programu da/ne?

2. Dokazati da naredna funkcija vraca proizvod prvih n elemenata nepraznog niza (pretpostaviti da je dužina niza `a` bar `n` i da prilikom množenja ne dolazi do prekoračenja):

```
int proizvodPodniza(int a[], int n) {
  if (n == 1)
    return a[0];
  else {
    int r = proizvodPodniza(a, n-1);
    return r * a[n-1];
  }
}
```

3. (a) Ukratko opisati kako se koristi debager tj. koje su osnove funkcionalnosti koje programerima pruža.
   (b) Koje se informacije o programu mogu dobiti njegovim profajliranjem?

4. Precizno definisati kada važi $f(n) = O(g(n))$, a kada da važi $f(n) = \Omega(g(n))$?

5. (a) Ako program složenosti $O(n^2)$ ulaz dimenzije $n = 10^4$ obradi za 0,01s, koliko će mu vremena biti potrebno da bi obradio ulaz dimenzije $n = 10^5$?
   (b) Ako je za izvršavanje jedne instrukcije potrebno 10^{-9}s, a algoritam za ulaz dimenzije n izvršava oko $5 \cdot n \cdot \log_2 n$ instrukcija, koliko će mu otprilike biti potrebno vremena da obradi ulaz dimenzije $n = 10^6$?

6. (a) Koji su osnovni resursi o kojima se vodi računa prilikom procene složenosti programa?
   (b) Kada je preporučljivo unapred izvršiti potrebna izačunavanja i rezultate čuvati u samom programu, a kada je preporučljivo ne čuvati nikakve rezultate izračunavanja, već izračunavanja vršiti kad je potreban njihov rezultat?

7. Koja je složenost narednog koda?

```
int s = 0;
for (int a = 1; a <= n; a++)
   s += a*a*a;
cout << "Zbir kubova je " << s << endl;
```

   Kako ga treba modifikovati da bi radio brže? Koja se složenost tada može postići?

8. (a) Da bi se binarnom pretragom proverilo da li niz brojeva sadrži dati broj, potrebno je da je taj niz _____.

(b) Šta vraćaju funkcije `lower_bound` i `upper_bound`?

9. (a) Koje su dobre a koje loše strane rekurzije?

   (b) Navesti bar jedan algoritam koji se po pravilu implementira rekurzivno i objasniti zašto je tako.

10. (a) Ukratko opisati osnovnu ideju algoritma *merge sort*.

    (b) Napisati rekurentu jednačinu koja opisuje njegovu složenost i objasniti kako se do te jednačine dolazi.

11. (a) Opisati strategije izbora pivota u algoritmu *quick sort*.

    (b) Ako se pivot nalazi na poziciji l u nizu a, opisati implementaciju bar jednog algoritma particionisanja dela niza a na pozicijama iz intervala $[l, d]$.

12. Definisati funkciju koja za datu varijaciju sa ponavljanjima brojeva od 1 do n određuje sledeću varijaciju u leksikografskom redosledu. Funkcija vraća `true` ako takva varijacija postoji ili `false` ako je polazna varijacija leksikografski najveća.

```
bool sledecaVarijacija(int n, vector<int>& varijacija) {









}
```

13. Navesti primer problema koji se može rešiti primenom algoritma pretrage sa povratkom i ukratko opisati njegovo rešenje.

14. Data je funkcija

```
int f(int n) {
  if (n == 0) return 1;
  if (n == 1) return 2;
  return 2*f(n-1) + 3*f(n-2);
}
```

    (a) Koja je složenost ove funkcije? (a) logaritamska (b) linearna (c) kvazilinearna (d) kvadratna (e) ekpsonencijalna

    (b) Primenom nekog oblika tehnike dinamičkog programiranja poboljšati složenost prethodne funkcije.

15. (a) Opisati princip rada gramzivih (pohlepnih) algoritama.

    (b) Da li gramzivi algoritmi uvek garantuju pronalaženje optimalnog rešenja?

    (c) Da li se gramzivi algoritam može primeniti na problem vraćanja kusura sa minimalnim brojem novčića? Diskutovati razne mogućnosti.