Teorija izračunljivosti

Ivan Prokić
kabinet 117, F blok
prokic@uns.ac.rs
http://imft.ftn.uns.ac.rs/∼iprokic/

Novi Sad

Sadržaj predmeta

1. Pojam i definicija algoritma, analiza algoritama;
2. Rast funkcija, asimptotske notacije;
3. Analiza vremena rada algoritma;
4. Rekurzivne funkcije;
5. Tjuringove mašine;
6. Klase složenosti, P i NP.

Način polaganja

- Pismeni:
  1. Pojam i definicija algoritma, analiza algoritama, rast funkcija, asimptotske notacije, analiza vremena rada algoritma: 30 bodova (min. 15 bodova)
  2. Rekurzivne funkcije, Tjuringove mašine, klase složenosti, P i NP: 30 bodova (min. 15 bodova)
- Seminarski i prezentacija: 30 bodova
- Usmeni: 10 bodova

Literatura

- T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, *Introduction to Algorithms*, MIT Press, Cambridge, 2009.
- I. Dolinka, *Kratak uvod u Analizu algoritama*, Prirodno-matematički fakultet, Novi Sad, 2008.

Tema 1

Pojam i neformalna definicija algoritma

Algoritimi (neformalno)

Algoritam:

- je ”dobro definisana” procedura koja tansformiše ulazne podatke u izlazne podatke;
- je oruđe za rešavanje ”dobro definisanog” problema izračunavanja

Primer (Sortiranje niza)

Ulaz: *Niz od* n *brojeva* $<a_1, a_2, \ldots, a_n>$
Izlaz: *Niz* $<a'_1, a'_2, \ldots, a'_n>$ *dobijen permutacijom elemenata ulaznog niza, tako da važi* $a'_1 \leq a'_2 \leq \ldots \leq a'_n$.

Na primer, za ulazni niz $<12, 45, 31, 33, 17>$, algoritam za sortiranje treba da vrati izlaz $<12, 17, 31, 33, 45>$. Ovo bi bio primer **instance problema**, koja sadrži jedan ulaz za algoritam.

Svojstva algoritima

Ključna svojstva algoritma:

- ulaz
- izlaz
- konačnost (broja koraka izračunvanja)
- određenost (koraka izračunavanja)
- efektivnost (”lako” razumljiv, ali i efikasan)

Algoritam je **tačan** ako se za svaku instancu problema zaustavlja (eng. halts) i daje tačan izlaz.

Specifikacija algoritma

U prvom delu kursa:

- algoritme pišemo u **pseudo-kodu**;
- računarski model je **RAM mašina** (*random-access machine*), tj. algoritmi će biti implementirani slično kao kompjuterski programi.

Tema 2

Analiza algoritama

Primer: Bubble sort

Primer (Sortiranje niza)

Ulaz: *Niz* A *od* n *brojeva* $<a_1, a_2, \ldots, a_n>$
Izlaz: *Niz* $<a'_1, a'_2, \ldots, a'_n>$ *dobijen od* A*, tako da* $a'_1 \leq a'_2 \leq \ldots \leq a'_n$.

Algorithm 1: Bubble sort (A)

```
for i = 1 to A.lenght − 1 do
    for j = A.lenght downto i + 1 do
        if A[j] < A[j − 1] then
            key = A[j]
            A[j] = A[j − 1]
            A[j − 1] = key
return A
```

Bubble sort: invarijante petlji i tačnost alogritma

Invarijanta prve for petlje:
U svakom prolasku kroz petlju podniz $A[1 \ldots i-1]$ sadrži elemente početnog niza u sortiranom redosledu i element $A[i-1]$ je manji ili jednak od svih elemenata podniza $A[i \ldots n]$.

Treba pokazati da važe:

- **Inicijalizacija petlje**: invarijanta je tačna pre prve iteracije petlje.
- **Očuvanje**: ako je invarijanta tačna pre iteracije petlje onda će biti tačna i nakon sledeće iteracije.
- **Terminacija**: Kada petlja terminira invarijanta nam pomaže da zaključimo da je agoritam tačan.

U našem primeru: **Inicijalizacija petlje**: za $i = 1$ imamo prazan podniz (invarijanta zadovoljena trivijalno). **Očuvanje**: Pre i-tog prolaska invarijanta je tačna. U i-tom prolasku kroz petlju najmanji element podniza $A[i \ldots n]$ dodaje se na kraj sortiranog niza $A[1 \ldots i-1]$. **Terminacija**: Nakon posledenje iteracije niz je sortiran.

Tema 3

Analiza vremena rada algoritma

Složenost algoritama

Efikasnost možemo meriti kroz:

- **vremensku složenost** (utrošeno vreme);
- prostornu složenost (utrošena memorija);
- druge.

Mi ćemo **ocenjivati vremensku složenost** algoritama, tj. vreme rada u **najgorem mogućem slučaju**, u zavisnosti od veličine ulaza. Vreme rada se izražava kroz broj primitivnih operacija, odnosno izvršenih **koraka**.

Primer (Bubble sort)

Veličina ulaza je broj elemenata niza n. Vreme rada algortima ćemo izraziti kroz funkciju $T(n)$ koja meri broj koraka potrebnih da se algoritam izvrši.

U ovakvom poluformalnom pristupu treba obratiti pažnju na: šta je prava veličina ulaza i šta su elementarni koraci mašine koja izvršava algoritam.

Ocena složenosti Bubble sort algoritma

Posmatramo niz A on n elemenata. Računamo broj koraka potrebnog za izvršenje algoritma u najgorem mogućem slučaju (za Bubble sort - kada je ulazni niz sortiran u obrnutom redosledu).

Algorithm 2: Bubble sort (A)

```
for i = 1 to A.lenght − 1 do
    for j = A.lenght downto i + 1 do
        if A[j] < A[j − 1] then
            key = A[j]
            A[j] = A[j − 1]
            A[j − 1] = key
return A
```

korak	cena	koliko puta
for (i)	c_1	$n-1$
for (j)	c_2	$\sum_{i=1}^{n-1} n-i$
if	c_3	$\sum_{i=1}^{n-1} n-i$
$key = A[j]$	c_4	$\sum_{i=1}^{n-1} n-i$
$A[j] = A[j-1]$	c_5	$\sum_{i=1}^{n-1} n-i$
$A[j-1] = key$	c_6	$\sum_{i=1}^{n-1} n-i$
return	c_7	1

Za $C = c_2 + \ldots + c_6$ imamo:

$$\begin{aligned} T(n) &= c_1(n-1) + C\sum_{i=1}^{n-1}(n-i) + c_7 = c_1(n-1) + C\frac{n(n-1)}{2} + c_7 \\ &= \frac{C}{2}n^2 + \left(c_1 - \frac{C}{2}\right)n + (c_7 - c_1) \end{aligned}$$

Kažemo da je Bubble sort algoritam **kvadratne složenosti**.

Složenost algoritama

- Tačna složenost algoritma nama nije bitna - nas zanima (što bolja) **ocena složenosti**.
- Mi ćemo određivati **red veličine** složenosti algoritma.
- Preciznije, određivaćemo **asimptotsku efikasnost algoritma**.