OBLIKOVANJE I ANALIZA ALGORITAMA — 1. kolokvij

18. 11. 2009.

1. (10) Između ponuđenih odgovora

$$\Theta(1), \quad \Theta(\lg n), \quad \Theta(n), \quad \Theta(n \lg n), \quad \Theta(n^2), \quad \Theta(n^2 \lg n), \quad \Theta(n^3), \quad \Theta(2^n),$$

nađite točan red veličine za broj koliko puta se izvršava naredba `x = x + 1` u svakom od sljedećih dijelova programa (`/` je operator cjelobrojnog dijeljenja, kao u C-u):

(a)
```
for i = 1 to n
    for j = 1 to i / 2
      x = x + 1;
```

(b)
```
for i = 1 to n {
    j = i;
    while (j >= 1) {
      for k = 1 to i
        x = x + 1;
      j = j / 2;
    }
}
```

Ukratko **argumentirajte** odgovore!

2. (10) Zadana je rekurzivna relacija

$$T(n) = 3\,T(n/4) + f(n), \quad f(n) = n,$$

uz početni uvjet $T(1) = d > 0$. Nađite uvjetno asimptotsko ponašanje relacijom Θ za rješenje $T(n)$, ako je n potencija od 4. Može li se dobiveno rješenje proširiti tako da asimptotsko ponašanje vrijedi bezuvjetno, za svaki dovoljno veliki $n \in \mathbf{N}$, ako u rekurziji piše $\lceil n/4 \rceil$, umjesto $n/4$?

3. (20) Neka je a uzlazno sortirano polje objekata nekog jednostavnog tipa. Treba provjeriti nalazi li se zadana vrijednost *key* u komadu polja $a[i], \ldots, a[j]$. Ulazni argumenti algoritma su: polje a, indeksi i, j, i vrijednost *key*. Pretpostavljamo da zadani indeksi i, j korektno indeksiraju neke elemente u polju a, pa granice ne treba provjeravati, ali dozvoljavamo da je $i > j$ na ulazu. Algoritam treba vratiti 1 (istina) ako postoji indeks k takav da je $i \le k \le j$ i $a[k] = key$. U protivnom, treba vratiti 0 (laž).

(a) Sastavite iterativni (**ne** rekurzivni) algoritam za **binarno** traženje odgovora, koji provjerava element na mjestu $(i + j)/2$ i tu, po potrebi, "reže" polje. Kolika je njegova vremenska složenost u najgorem slučaju?

Zamislite da, u tom algoritmu, izraz $(i + j)/2$ zamijenimo izrazom

(b) $i + (j - i)/3$, (c) $j - 2$, (d) $\max\{i, j - 2\}$,

bez ikakvih drugih promjena. Analizirajte pojedinačno svaki od tri dobivena algoritma. Radi li odgovarajući algoritam korektno, tj. nalazi li korektan odgovor na pitanje provjere? Ako da, kolika je njegova vremenska složenost u najgorem slučaju? Ako ne radi korektno, pokažite to primjerom i objasnite što se tada događa.

OKRENITE!

4. (10) Neka je A polje od n elemenata nekog tipa. Na tom tipu **nije** definirana relacija uređaja (poput $\leq$), tako da elemente u polju A smijemo "testirati" samo operatorima jednakosti ili različitosti. Elementarna operacija je test jednakosti ili različitosti jednog para elemenata, a mjera složenosti je **točan broj** takvih testova u ovisnosti o n.

Kažemo da je element x **većinski** ili **dominirajući** element u polju A, ako je strogo preko polovine elemenata u A jednako x, tj. ako je

$$\text{card}\{i \mid A[i] = x\} > \frac{n}{2}.$$

Koliko različitih većinskih elemenata može biti u polju A?

Sastavite algoritam koji provjerava postoji li dominirajući element u polju A. Ako takav element postoji, algoritam treba vratiti taj element (ili pokazivač na njega). U protivnom, treba vratiti objekt s imenom `nema` (ili `NULL` pokazivač). Zabranjeno je koristiti dodatna polja!

Red veličine složenosti algoritma mora biti $O(n^2)$. Analizirajte složenost vašeg algoritma i pokažite da ona zadovoljava ovaj uvjet.

Napomena: Broj bodova ovisi o složenosti algoritma. Složenost $O(n^2)$ vrijedi najviše 10 bodova. **Bonus**: složenost $O(n \log n)$ vrijedi 10 bodova više!

REZULTATI: ponedjeljak, 23. 11. 2009. u 11 sati. Saša Singer