Algoritmi şi structuri de date (I). Seminar 9: Aplicaţii ale tehnicii reducerii. Analiza complexităţii algoritmilor recursivi.

Problema 1 *Căutare eficientă a poziţiei de inserare într-un tablou ordonat.* Se consideră un tablou $a[1..n]$ ordonat crescător şi v o valoare. Să se determine, folosind un algoritm din $\mathcal{O}(\lg n)$, poziţia unde poate fi inserată valoarea v în tabloul a astfel încât acesta să rămână ordonat crescător. Să se analizeze ce efect are utilizarea acestui algoritm de căutare a poziţiei de inserţie în cazul în care este utilizat în cadrul algoritmului de sortare prin inserţie.

Indicaţie. Se poate utiliza ideea de la căutarea binară.

Problema 2 *Metoda bisecţiei.* Fie $f : [a, b] \to R$ o funcţie continuă având proprietăţile: (i) $f(a)f(b) < 0$; (ii) există un unic x^* cu proprietatea că $f(x^*) = 0$. Să se aproximeze x^* cu precizia $\epsilon > 0$. Stabiliţi ordinul de complexitate al algoritmului propus.

Indicaţie. A determina pe x^* cu precizia ϵ înseamnă a identifica un interval de lungime ϵ care conţine pe x^* sau chiar un interval de lungime 2ϵ dacă se consideră ca aproximare a lui x^* mijlocul intervalului. Se poate aplica exact aceeaşi strategie ca la căutarea binară ţinându-se cont că x^* se află în intervalul pentru care funcţia f are valori de semne opuse în extremităţi.

Complexitatea este determinată de dimensiunea intervalului $[a, b]$ şi de precizia dorită a aproximării, ϵ, prin urmare se poate considera că dimensiunea problemei este $n = (b - a)/\epsilon$

Problema 3 *Căutare ternară.* Tehnica căutării binare poate fi extinsă prin divizarea unui subşir $a[li..ls]$ în trei subşiruri $a[li..m1]$, $a[m1 + 1..m2]$, $a[m2 + 1..ls]$, unde $m1 = li + \lfloor (ls - li)/3 \rfloor$ iar $m2 = li + 2\lfloor (ls - li)/3 \rfloor$. Descrieţi în pseudocod algoritmul de căutare ternară şi stabiliţi ordinul de complexitate.

Indicaţie. Se poate utiliza Teorema Master pentru a stabili ordinul de complexitate.

Probleme suplimentare

1. Se consideră un caroiaj de dimensiuni $2^k \times 2^k$ (pentru $k = 3$ se obţine o tablă de şah clasică) în care unul dintre pătrăţele este marcat. Propuneţi o strategie bazată pe tehnica divizării care acoperă întreaga tablă (cu excepţia pătrăţelului marcat) cu piese constând din 3 pătrăţele aranjate în forma de L (indiferent de orientare).

2. Propuneţi un algoritm de complexitate medie $\mathcal{O}(n \log n)$ pentru a verifica dacă elementele unui tablou sunt distincte sau nu.

3. Se consideră un tablou ordonat crescător $a[1..n]$. Presupunem că tabloul a a fost transformat printr-o deplasare circulară la dreapta cu k poziţii. De exemplu, dacă $a = [2, 3, 6, 8, 9, 11, 14]$ iar $k = 3$ tabloul transformat este $a = [9, 11, 14, 2, 3, 6, 8]$.
   a) In ipoteza că valoarea k este cunoscută propuneţi un algoritm de complexitate $\mathcal{O}(1)$ care determină cea mai mare valoare din a.
   b) In ipoteza că valoarea k este necunoscută propuneţi un algoritm de complexitate $\mathcal{O}(\log n)$ care determină cea mai mare valoare din a.

4. Se consideră un tablou cu numere naturale distincte $a[1..n]$ ordonat crescător. Propuneţi un algoritm de complexitate $\mathcal{O}(\log n)$ care verifică dacă există $i \in \{1, \ldots, n\}$ cu proprietatea că $a[i] = i$.