Analiza algoritmilor

Tema 1

Deadline: 15.11.2019
Ora 23:55

Responsabili temă: Mihai Nan, David Iancu

Profesor titular: Andrei-Horia Mogoș

Facultatea de Automatică şi Calculatoare
Universitatea Politehnica din Bucureşti
Anul universitar 2019 - 2020
Seria CC

1 Problema 1

1.1 Clase de complexitate

1. Verificați valoarea de adevăr a relațiilor, argumentând pentru fiecare răspunsul.

- $0.01 \cdot n \cdot \log n - 2000 \cdot n + 6 = O(n \log n)$
- $\Theta(n \log n) \cup o(n \log n) \neq O(n \log n)$
- $\sqrt{n} \log n! = \Theta(n^{\frac{3}{2}} \log n)$
- $n! = \Omega(5^{\log n})$
- $\log^{2019} n = o(n)$

2. Calculați estimativ, oferind explicații, complexitatea temporală pentru următorii algoritmi, pentru cele trei cazuri (favorabil, defavorabil și mediu):

```
Algoritm1(A[0..n], n)
    for i = 1..n
        if (A[i] > 0)
            for j = 1..i
                if (A[j] mod 2 == 0)
                    for k = 1..j
                        s = s + i + j + k
    return s
```

```
Algoritm2(A[0..n], n)
    s = 0
    for i = 1..n
        for j = i..n
            if (A[i] > A[j])
                for k = 1..i
                    s = s + A[i]*A[k]
    return s
```

```
Algoritm3(n)
    s = 0
    for i = 1..n
        for j = i..i*i
            s = s + j
    return s
```

```
Algoritm4(n)
        s = 0
        i = n/2
        while (i < n)
                j = i
                while (j <= n)
                        k = 1
                        while (k <= n)
                                s += 1
                                k = k * 2
                    j = j * 2
                i = i + 1
        return s
```

Pentru algoritmul `Algoritm2` calculați complexitatea exactă, folosind metrica neomogenă.

3. Dându-se un vector cu N elemente numere întregi și un număr întreg K, scrieți un algoritm, în pseudocod, care să determine subsecvența maximă, din punct de vedere al numărului de elemente, de sumă K. Fie un vector v cu N elemente întregi, o subsecvență de numere din vector este de forma $v[i], v[i+1], \ldots, v[j]$, unde $i \leq j$. Calculați estimativ, oferind explicații sumare, complexitatea temporală a algoritmului pentru cazul cel mai defavorabil. Pentru punctajul maxim, algoritmul propus trebuie să fie unul optim din punct de vedere al complexității temporale.

1.2 Rezolvarea unei recurențe

4. Găsiți clasa de complexitate, folosind una dintre cele trei metode predate la curs (iterativă, arbore de recurență sau Master), pentru

- $T(n) = \begin{cases} 3T(\frac{n-2}{2}) + k_2 n^2 & \text{dacă } n > 1, k_2 \in R_+^* \\ k_1 & \text{dacă } n = 1, k_1 \in R_+^* \end{cases}$

Hint: $S(n) = T(4n-2)$

- $T(n) = \begin{cases} 3T(\frac{n}{4}) + k_2 n^2 & \text{dacă } n > 1, k_2 \in R_+^* \\ k_1 & \text{dacă } n = 1, k_1 \in R_+^* \end{cases}$
- $T(n) = \begin{cases} 3T(\frac{n}{4}) + n \log n & \text{dacă } n > 1, k_2 \in R_+^* \\ k_1 & \text{dacă } n = 1, k_1 \in R_+^* \end{cases}$
- $T(n) = \begin{cases} 3T(\frac{n}{3}) + \frac{n}{\log n} & \text{dacă } n > 1, k_2 \in R_+^* \\ k_1 & \text{dacă } n = 1, k_1 \in R_+^* \end{cases}$

Atenție!

Trebuie să alegeți metodele de rezolvare indicate (**iterativă**, **arbore de recurență** sau **Master**) astfel încât să folosiți fiecare metodă cel puțin pentru una dintre recurențe.

5. Găsiți clasa de complexitate, folosind metoda substituției și notația Θ, pentru:

- $T(n) = \begin{cases} 10T(\frac{n}{3}) + k_2 n^2 & \text{dacă } n > 1, k_2 \in R_+^* \\ k_1 & \text{dacă } n = 1, k_1 \in R_+^* \end{cases}$
- $T(n) = \begin{cases} 2T(\frac{n}{3}) + k_2 n^4 & \text{dacă } n > 1, k_2 \in R_+^* \\ k_1 & \text{dacă } n = 1, k_1 \in R_+^* \end{cases}$

1.3 Găsirea și rezolvarea unei recurențe

6. Fie următorii doi algoritmi care calculează produsul a două numere naturale, având n cifre.

Algorithm 1 Inmultire 1

```
procedure ALGORITHM1(x, y, n)
    if n = 1 then
        return x * y
    else
        m ← ⌈n/2⌉
        p ← 10^m
        a ← x div p
        b ← x mod p
        c ← y div p
        d ← y mod p
        e ← algorithm1(a, c, m)
        f ← algorithm1(b, d, m)
        g ← algorithm1(b, c, m)
        h ← algorithm1(a, d, m)
        return p^2 * e + p * (g + h) + f
```

Algorithm 2 Inmultire 2

```
procedure ALGORITHM2(x, y, n)
    if n = 1 then
        return x * y
    else
        m ← ⌈n/2⌉
        p ← 10^m
        a ← x div p
        b ← x mod p
        c ← y div p
        d ← y mod p
        e ← algorithm2(a, c, m)
        f ← algorithm2(b, d, m)
        g ← algorithm2(a − b, c − d, m)
        return p^2 * e + p * (e + f − g) + f
```

Găsiți recurența de complexitate pentru fiecare algoritm și aplicați una dintre cele patru metode predate la curs (iterativă, arbore de recurență, substituție sau Master) pentru a determina clasa de complexitate.

2 Problema 2

În viața reală sunt foarte multe date care sunt reținute sub forma unor secvențe de elemente (de exemplu frame-urile unui video, temperaturile înregistrate în zilele unei luni etc.). De aceea, există multe tipuri de structuri de date care pot fi folosite pentru modelarea unor astfel de date secvențiale. Spre exemplu, vectorii alocați dinamic ne permit să realizăm operația de tip `lookup` în $O(1)$, operația de tip adăugare la final în $O(1)$, dar operația de inserare la începutul structurii va avea complexitatea $O(n)$. Listele dublu înlănțuite ne permite să realizăm inserarea în $O(1)$, indiferent de poziția la care dorim să inserăm elementul, dar accesul la un element de pe o poziție aleatoare se realizează în $O(n)$.

În cadrul acestei probleme vă propunem să implementați o structură de date care permită realizarea unui număr mare de operații cu o complexitate cât mai bună (puteți considera drept referința $O(\log n)$) pentru fiecare tip în parte.

Structura de date pe care o veți implementa folosind limbajul C trebuie să permită următoarele operații:

1. `insert(struct, item, index)` – care va insera în structura `struct` elementul `item` la poziția `index`;
2. `delete(struct, index)` – va șterge elementul de pe poziția `index` din structura `struct`;
3. `lookup(struct, index)` – care va returna valoarea elementului de pe poziția `index` din structura `struct`;
4. `set(struct, item, index)` – care va înlocui valoarea elementului de pe poziția `index` cu `item`;
5. `size(struct)` – care va returna numărul de elemente din structura `struct`;
6. `split(struct, index)` – care va împărți secvența reținută în structura `struct` în două subsecvențe S_1 și S_2, unde S_1 va conține elementele de la poziția 0 până la poziția *index*, iar S_2 va conține elementele de la poziția $index + 1$ până la poziția $N - 1$, unde $N =$ `size(struct)`;
7. `concat(struct1, struct2)` – va adăuga la finalul secvenței reținută în structura `struct1` elementele din secvența reținută în structura `struct2`.

Implementați în limbajul **C** o structură de date care să permită operațiile indicate și analizați în **README** complexitatea pentru fiecare operație în parte.

Veți porni implementarea structurii de date de la următorul fișier header (`sequence.h`).

```
typedef int Type;

typedef struct sequence {
        // TODO
} *Sequence;

/*
*        Returnează structura vidă
*/
Sequence init();

/*
*        Inserează un element în structură pe poziția index
*/
Sequence insert(Sequence struct, Type item, int index);

/*
*         Șterge elementul de pe poziția index din structură
*/
Sequence delete(Sequence struct, int index);

/*
*        Returnează valoarea elementului de pe poziția index
*/
Type lookup(Sequence struct, int index);
```

```
/*
*        Înlocuiește valoarea elementului de pe poziția index
*/
Sequence set(Sequence struct, Type item, int index);

/*
*        Returnează dimensiunea structurii
*/
int size(Sequence struct);

/*
*        Returnează cele două structuri rezultate în urma divizării
*         (rezultatul este reținut într-un Sequence* cu 2 elemente)
*/
Sequence* split(Sequence struct, int index);

/*
*        Returnează structura rezultată după concatenare
*/
Sequence concat(Sequence struct1, Sequence struct2);
```

3 Punctaj

⚠ IMPORTANT !

⚠ Punctajul complet pe fiecare task se acordă doar dacă task-ul a fost **complet** și **corect** rezolvat.
Tema valorează **1 punct** din nota finală de la AA.

Problema	Punctaj
Problema 1.1.1	10 puncte
Problema 1.1.2	10 puncte
Problema 1.1.3	10 puncte
Problema 1.2.4	10 puncte
Problema 1.2.5	10 puncte
Problema 1.3.6	10 puncte
Problema 2	40 de puncte (30p implementare ***eficientă*** + 10p explicații și calcul complexitate)

Atenție!

Tema este individuală! Toate soluțiile trimise vor fi verificate, folosind o unealtă pentru detectarea plagiatului. Orice parte a temei copiată, de pe Internet sau de la colegi, duce la anularea punctajului pentru temă. În cazul copierii de la un alt coleg se anulează și punctajul colegului respectiv pentru această temă.

Dacă o temă este uploadată dupa termenul limită, tema se anulează.

Observație

Se va acorda și punctaj parțial pe exercițiile propuse la **Problema 1** (1), în funcție de cât de mult s-a rezolvat corect.

În cazul **Problemei 2** (2), dacă fișierele sursă nu compilează sau nu rulează corect (conform specificațiilor din enunț), punctajul maxim pe task va deveni 50% din punctajul maxim specificat anterior și se va acorda punctaj parțial, în funcție de cât de mult din task a fost rezolvat. Dacă nu se va realiza o variantă de implementare propusă, nu se va acorda nici punctajul aferent explicațiilor din comparație pentru aceasta.

4 Precizări generale pentru trimiterea temei

4.1 Cum se trimite tema?

Creați o arhivă cu denumirea **nume_prenume_grupa.zip**. De exemplu, studentul *Ionel Popescu* de la grupa *326CC* va crea arhiva *popescu_ionel_326CC.zip*.

Încărcați arhiva pe cs.curs.pub.ro până la data indicată (**15.11.2019, ora 23:55**). **NU** se acceptă teme trimise pe e-mail sau altfel decât prin intermediul platformei de curs.

Atenție! Formatul arhivei trebuie să fie **zip**.

4.2 Ce trebuie să conțină arhiva?

Un fișier **README**, având formatul **PDF**, care să fie semnat cu **nume**, **prenume** și **grupă**. Acest fișier trebuie să conțina răspunsul la intrebările și cerințele din enunț. Pentru redactarea fișierului **PDF**, puteți utiliza, de exemplu, LaTeXsau orice procesor de documente pentru desktop sau online în care să puteți edita ecuații (exemple ar fi: Google Docs, LibreOffice Writer, Microsoft Word etc.) sau puteți rezolva exercițiile pe foi și să le scanați și să le adăugați într-un fișier **PDF**.

Un folder **aa-tema1-2** care să conțină sursele și Makefile-ul cerute la **Problema 2**(2).

⚠ IMPORTANT !

⚠ Acestea trebuie să se afle în rădăcina arhivei și **NU** în alte foldere!

Atenție!

Orice întrebare legată de temă se va adresa pe forumul dedicat temei.