Tehnici avansate de programare

Marinescu-Ghemeci Ruxandra

verman@fmi.unibuc.ro

Programa

Programa

- Tehnici de programare:
  - Greedy
  - Divide et Impera
  - Programare dinamica
  - Backtracking
  - Branch and Bound
- NP-completitudine
- Algoritmi euristici. Algoritmi probabiliști. Algoritmi genetici
- Principiul lui Dirichlet

Obiectiv general

- Însușirea principalelor tehnici de elaborare a algoritmilor și a tipurilor de probleme la care se pretează acestea

Obiective specifice

- cunoașterea principalelor tehnici de programare
- abilități de utilizare a structurilor de date și tehnicilor potrivite în rezolvarea unei probleme
- abilități de justificare a **corectitudinii** algoritmilor propuși si de determinare a **complexității** acestora

Obiective. Motivații

- **Tehnici de programare**
  - algoritmi eficienți

    "*Perhaps the most important principle for the good algorithm designer is to refuse to be content*" –

    Aho, Hopcroft, and Ullman, The Design and Analysis of Computer Algorithms

Obiective. Motivații

- **Tehnici de programare**
  - algoritmi eficienți

    Exemple de probleme

    - Aflarea minimului și maximului dintr-un vector
    - Cele mai apropiate două puncte dintr-o mulțime de puncte din plan dată
    - Numărul de inversiuni dintr-un vector
    - Înmulțirea a două numere / matrice

Obiective. Motivații

- **Tehnici de programare**
  - algoritmi corecți

    Exemple de probleme

    - Dată o mulțime de intervale, să se determine o submulțime de cardinal maxim de intervale care nu se suprapun
    - Dată o mulțime de intervale, fiecare interval având asociată o pondere, să se determine o submulțime de intervale care nu se suprapun având ponderea totală maximă

Obiective. Motivații

- **Tehnici de programare**
  - algoritmi eficienți (chiar dacă există soluții evidente polinomiale – se poate mai bine?)
  - corectitudinea algoritmilor – demonstrații
  - probleme dificile -> NP-completitudine
  - pentru ce tipuri de probleme se aplica metodele

  - Complexitate – structuri de date

Obiective. Motivații

- Numeroase aplicații
  - Bioinformatică, procesare texte, imagini
  - Geometrie computațională
  - Căutare web, similitudini, aliniere
  - Probleme de planificare
  - Proiectare, jocuri, strategii
  - Baze de date – arbori de căutare optimi
- Probleme interviuri

Resurse

- http://moodle.fmi.unibuc.ro/course/

BIBLIOGRAFIE

- Jon Kleinberg, Éva Tardos, **Algorithm Design**, Addison-Wesley 2005 http://www.cs.princeton.edu/~wayne/kleinberg-tardos/
- Horia Georgescu. **Tehnici de programare**. Editura Universităţii din Bucureşti 2005
- S. Dasgupta, C.H. Papadimitriou, U.V. Vazirani, **Algorithms**, McGraw-Hill, 2008

BIBLIOGRAFIE

- T.H. Cormen, C.E. Leiserson, R.R. Rivest – **Introducere in algoritmi**, Mit Press, trad. Computer Libris Agora
- Leon Livovschi, Horia Georgescu. **Sinteza şi analiza algoritmilor**. 1986
- http://www.cplusplus.com/
- Dana Lica, Mircea Paşoi, **Fundamentele programării,** L&S Infomat

BIBLIOGRAFIE

- **coursera.org**

  Algorithms, Part II – Princeton University

  Algorithms: Design and Analysis – Stanford University

- **MIT** https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-006-introduction-to-algorithms-fall-2011/

- infoarena.ro

Evaluare

Evaluare

- Test de laborator ⇒ 10 puncte

  Nota test laborator ≥ 5 puncte

- Teme de laborator ⇒ 3 puncte

  – 3 teme

  – nu se pot recupera la reexaminari

- Examen scris ⇒ 7 puncte
  - !!! în ultima săptamână din semestru, nu în sesiune

- Nota finala

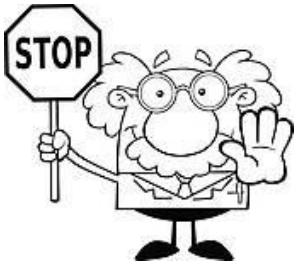
  = (test laborator+teme+examen scris)/2

Structura

- Curs
  - 2 ore pe săptămâna
  - finalizat cu examen scris
- Laborator
  - 2 ore pe săptămână
  - teme, limbaj C/C++
  - finalizat cu test de laborator
- Seminar
  - 2 ore la două săptămâni
  - discuții idei probleme curs/laborator, complexități
  - nu este notat

Consultații

- verman@fmi.unibuc.ro
- sala 318 (catedra de informatică)



- învățat pe de rost
- copy – paste

Despre algoritmi

De ce despre algoritmi?

- numeroase aplicații
- în practică este importantă eficiența algoritmilor
- ar fi util să știm dacă algoritmii pe care îi propunem sunt corecți
  - ☺ corectitudine $\neq$ nu a găsit cineva încă un contraexemplu

Aspecte generale care apar la rezolvarea unei probleme

- *Teoretic*, **paşii elaborării un algoritm sunt următorii:**

1. demonstrarea faptului că este **posibilă** elaborarea unui algoritm pentru determinarea unei soluţii
2.
3.
4.
5.

Aspecte generale care apar la rezolvarea unei probleme

- *Teoretic*, **paşii elaborării un algoritm sunt următorii:**

1. demonstrarea faptului că este **posibilă** elaborarea unui algoritm pentru determinarea unei soluţii
2. **elaborarea** algoritmului
3. demonstrarea **corectitudinii** algoritmului
4.
5.

Aspecte generale care apar la rezolvarea unei probleme

- *Teoretic*, **paşii elaborării un algoritm sunt următorii:**

1. demonstrarea faptului că este **posibilă** elaborarea unui algoritm pentru determinarea unei soluţii
2. **elaborarea** algoritmului
3. demonstrarea **corectitudinii** algoritmului
4. determinarea **timpului de executare** a algoritmului
5. demonstrarea **optimalităţii** algoritmului

Existența algoritmilor

Existența algoritmilor

- **Problemă nedecidabilă** =pentru care nu poate fi elaborat un algoritm.

1. **Problema opririi programelor**: pentru orice program şi orice valori de intrare să se decidă dacă programul se termină.
2. **Problema echivalenţei programelor:** să se decidă pentru orice două programe dacă sunt echivalente (produc aceeaşi ieşire pentru aceleaşi date de intrare).

Elaborarea algoritmilor

Elaborarea algoritmilor

- **Cursurile viitoare**: metode de elaborare a algoritmilor

Corectitudinea algoritmilor

Corectitudinea algoritmilor

- **Terminarea programului**
- **Corectitudinea parţială**
  - **Invarianţi** =relaţii ce trebuie îndeplinite la orice trecere a programului prin acel loc

Corectitudinea algoritmilor

- <u>Exemplul 1</u> **Determinarea concomitentă a cmmdc şi cmmmc a două numere naturale** $a,b \in \mathbf{N}^*$.

Corectitudinea algoritmilor

- Exemplul 1 **Determinarea concomitentă a cmmdc şi cmmmc a două numere naturale** $a,b \in \mathbf{N}^*$**.**

Algoritm:

```
x ← a; y ← b; u ← a; v ← b;

while x≠y

   if x>y then x ← x-y; u ← u+v
          else y ← y-x; v ← u+v

write(x,(u+v)/2)
```

Corectitudinea algoritmilor

- Exemplul 1 **Determinarea concomitentă a cmmdc şi cmmmc a două numere naturale** $a,b \in \mathbf{N}^*$.

Algoritm:

```
x ← a; y ← b; u ← a; v ← b;

while x≠y
  {(x,y)=(a,b); xv+yu = 2ab}          (*)
    if x>y then x ← x-y; u ← u+v
           else y ← y-x; v ← u+v

write(x,(u+v)/2)
```

Corectitudinea algoritmilor

- Exemplul 2 **Metoda de înmulţire a ţăranului rus**

Fie $a, b \in \mathbf{N}^*$. Să se calculeze produsul ab

Ţăranul rus ştie doar:

- să adune două numere;
- să verifice dacă un număr este par sau impar;
- să afle câtul împărţirii unui număr la 2.

Corectitudinea algoritmilor

- **Exemplul 2 Metoda de înmulţire a ţăranului rus**

Algoritm:

```
x ← a; y ← b; p ← 0
while x>0


    if x impar then p ← p+y
    x ← x div 2; y ← y+y
write(p)
```

Corectitudinea algoritmilor

- Exemplul 2 **Metoda de înmulţire a ţăranului rus**

Algoritm:

```
x ← a; y ← b; p ← 0
while x>0
    { xy+p=ab }                 (*)
    if x impar then p ← p+y
    x ← x div 2; y ← y+y
write(p)
```

Timpul de executare

Timpul de executare a algoritmilor

- se măsoară în funcţie de lungimea n a datelor de intrare
- **`T(n)`** = timpul de executare pentru orice set de date de intrare de lungime **`n`**.

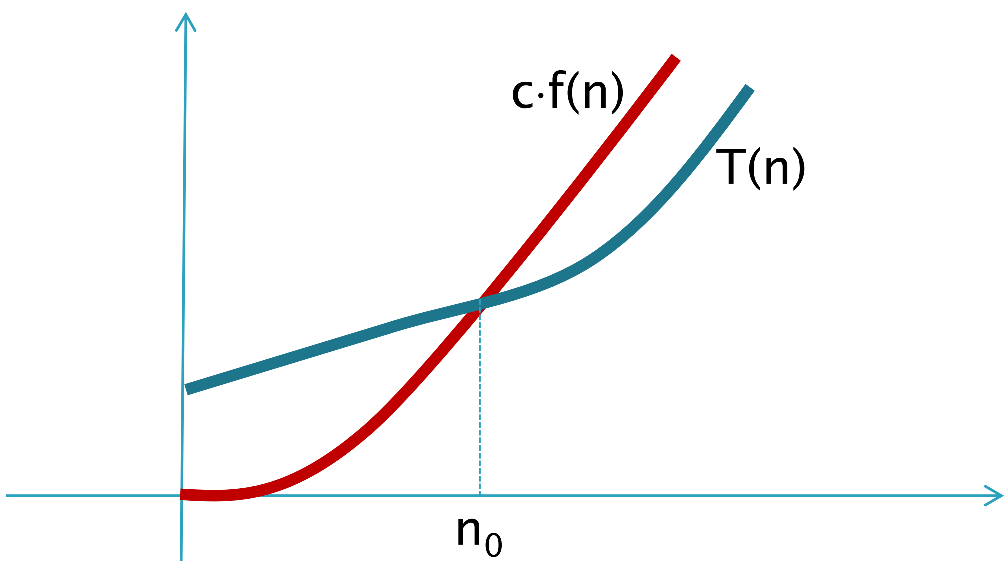
Timpul de executare a algoritmilor

- în majoritatea cazurilor ne mărginim la a evalua **ordinul de mărime** al timpului de executare

$$T(n) = O(f(n))$$

$$\exists c,\ n_0 \text{ – constante a.î } \forall n \geq n_0$$

$$T(n) \leq c \cdot f(n)$$



Timpul de executare a algoritmilor

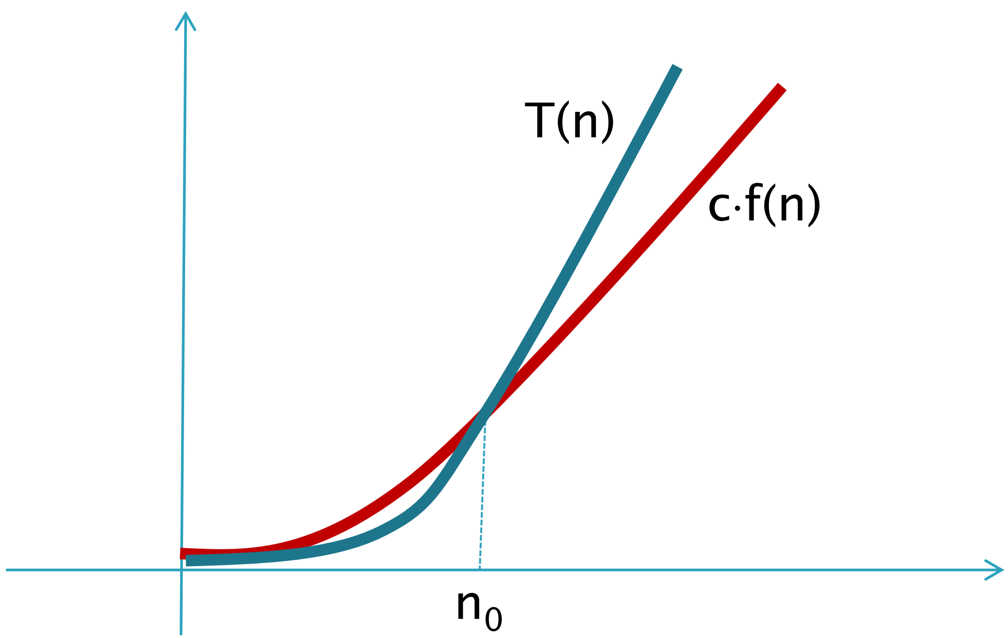
- Notație: `T(n) = O(f(n))`
- comportare asimptotică
- caz defavorabil

Timpul de executare a algoritmilor

- $T(n) = \Omega(f(n))$

$$\exists c,\ n_0 \text{ – constante a.î } \forall n \geq n_0$$

$$T(n) \geq c \cdot f(n)$$

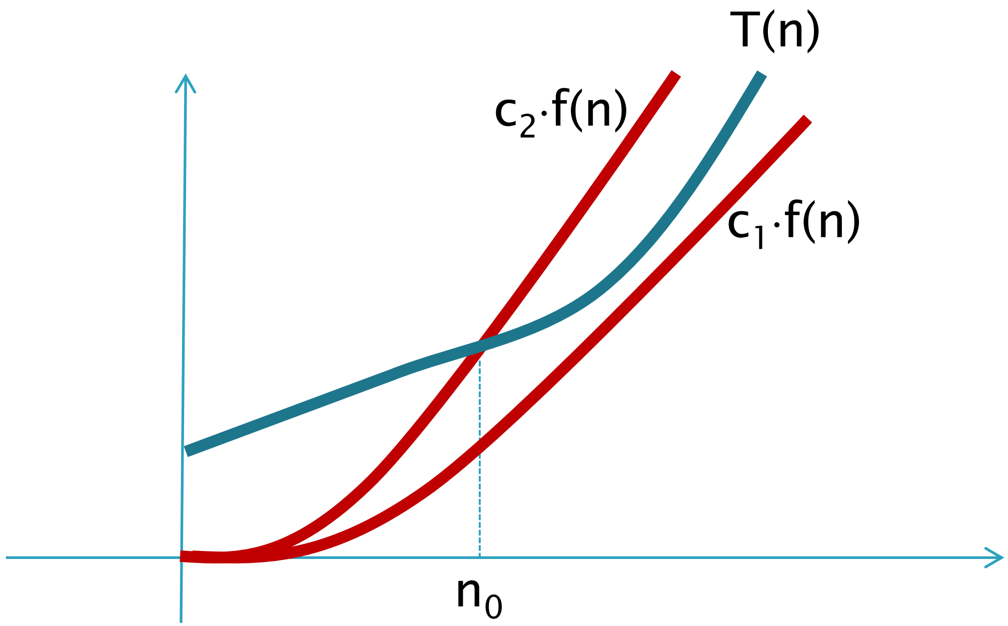


Timpul de executare a algoritmilor

- $T(n) = \Theta(f(n))$

$\exists c_1, c_2, n_0$ constante a.î $\forall n \geq n_0$

$$c_1 \cdot f(n) \leq T(n) \leq c_2 \cdot f(n)$$



Timpul de executare a algoritmilor

- **T(n)= O(f(n))**
- **f(n)** = n^k – algoritm **polinomial**

  – algoritm "**acceptabil**"

	n	$n \log_2 n$	n^2	n^3	1.5^n	2^n	$n!$
$n = 10$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	4 sec
$n = 30$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	18 min	10^{25} years
$n = 50$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	11 min	36 years	very long
$n = 100$	< 1 sec	< 1 sec	< 1 sec	1 sec	12,892 years	10^{17} years	very long
$n = 1{,}000$	< 1 sec	< 1 sec	1 sec	18 min	very long	very long	very long
$n = 10{,}000$	< 1 sec	< 1 sec	2 min	12 days	very long	very long	very long
$n = 100{,}000$	< 1 sec	2 sec	3 hours	32 years	very long	very long	very long
$n = 1{,}000{,}000$	1 sec	20 sec	12 days	31,710 years	very long	very long	very long

Jon Kleinberg, Éva Tardos, **Algorithm Design**, Addison-Wesley 2005,

procesor – 1 milion de operații pe secundă

Temă

Problema 3-SUM: Dat un vector de n numere întregi distincte, să se afișeze tripletele de elemente din vector care au suma 0

Optimalitatea algoritmilor

Optimalitatea algoritmilor

- Să presupunem că pentru o anumită problemă am elaborat un algoritm şi am putut calcula şi timpul său de executare T(n).

Algoritmul nostru este "cel mai bun" sau există un alt algoritm cu timp de executare mai mic.

Optimalitatea algoritmilor

- Exemplul 1. Să se determine minimul elementelor unui vector.

- Exemplul 2. Să se determine minimul şi maximul elementelor unui vector.

- Arătați că algoritmii propuși sunt optimali

Optimalitatea algoritmilor

- Exemplul 1. **Să se determine minimul elementelor unui vector.**

Optimalitatea algoritmilor

- Exemplul 1. **Să se determine minimul elementelor unui vector.**

$m \leftarrow a_1$

for $i=2,n$

    if $a_i<m$ then $m \leftarrow a_i$

- $T(n)=?$

Optimalitatea algoritmilor

- Exemplul 2. **Să se determine minimul şi maximul elementelor unui vector.**

Optimalitatea algoritmilor

- Exemplul 2. **Să se determine minimul şi maximul elementelor unui vector.**

```
if n impar then m ← a_1; M ← a_1; k ← 1
else if a_1<a_2 then m ← a_1; M ← a_2
     else m ← a_2; M ← a_1;
     k ← 2
while k ≤ n-2
   if a_{k+1}<a_{k+2} then if a_{k+1}<m then m ← a_{k+1}
                           if a_{k+2}>M then M ← a_{k+2}
                      else if a_{k+2}<m then m ← a_{k+2}
                           if a_{k+1}>M then M ← a_{k+1}
   k ← k+2
```

- $T(n)=?$

Despre algoritmi

- **Nu interesează în general demonstrarea teoretică a existenţei algoritmilor, ci accentul este pus pe elaborarea algoritmilor**