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 ≠ 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 aceeași ieșire pentru aceleași date de intrare).

Elaborarea algoritmilor

Elaborarea algoritmilor

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

Corectitudinea algoritmilor

Corectitudinea algoritmilor

- **Terminarea programului**

- **Corectitudinea parțială:** presupunând că algoritmul se termină, rezultatul este corect;
  - **Invarianţi** =relaţii ce trebuie îndeplinite la orice trecere a programului prin acel loc

Corectitudinea algoritmilor

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

Corectitudinea algoritmilor

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

Algoritm:

```
x ← a; y ← b;

while x≠y

    if x>y then x ← x-y;
           else y ← y-x;

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

Corectitudinea algoritmilor

- Exemplu 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

- Exemplu **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

- `{(x,y)=(a,b); xv+yu = 2ab} (*)` **– este invariant**
- **<u>Demonstrație</u>** – **Inducție** după numărul de pași (exercițiu)

Corectitudinea algoritmilor

- **{(x,y)=(a,b); xv+yu = 2ab} (*) - este invariant**

  ⇒ **Dacă se termină avem**

  (a,b) = (x,x) =x

  (u+v)/2 = ab/x = ab / (a,b) = [a,b]

⇒ **corectitudine parțială**

- **Algoritmul se termină deoarece**

- şirul **$\{x_n+y_n\}$** este un şir *strict* descrescător de **numere naturale pozitive**, unde $\{x_n\}$, $\{y_n\}$ este şirul de valori succesive ale variabilelor x și y.

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**

  - dat de numărul de operații elementare în funcție de n

Timpul de executare a algoritmilor

- Se numără operații elementare (de atribuire, aritmetice, de decizie, de citire/scriere)

- Numărare aproximativă => ordinul de mărime al numărului de operații elementare

- Pentru simplitate – se fixează operație de bază

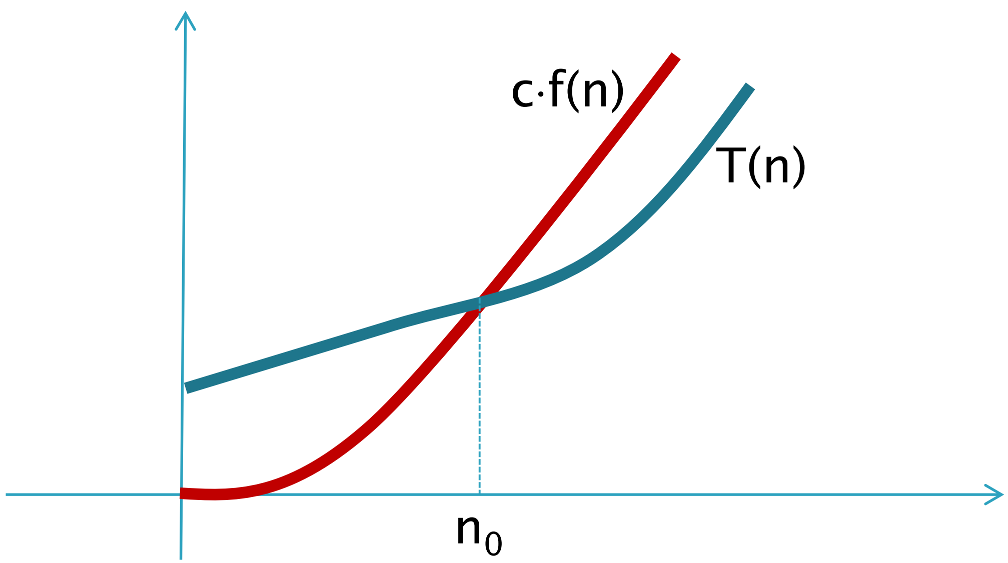
Timpul de executare a algoritmilor

- În majoritatea cazurilor ne mărginim la a evalua **ordinul de mărime** al timpului de executare = **ordin de complexitate al algoritmului**

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

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

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

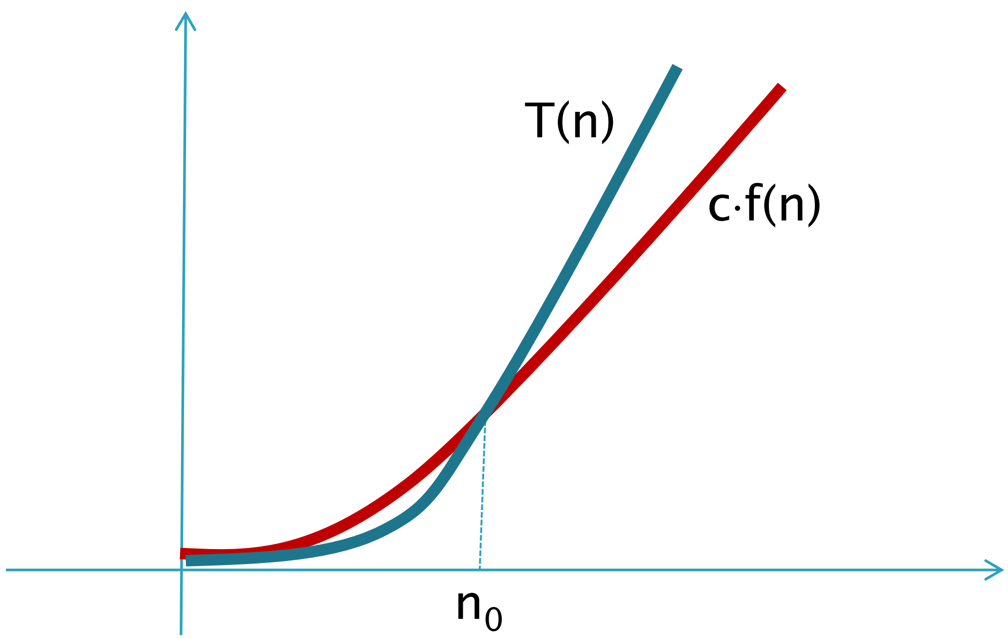


Timpul de executare a algoritmilor

- **T(n)= Ω(f(n)) (Suplimentar)**

  $\exists c, n_0$ – **constante a.î** $\forall n \geq n_0$

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

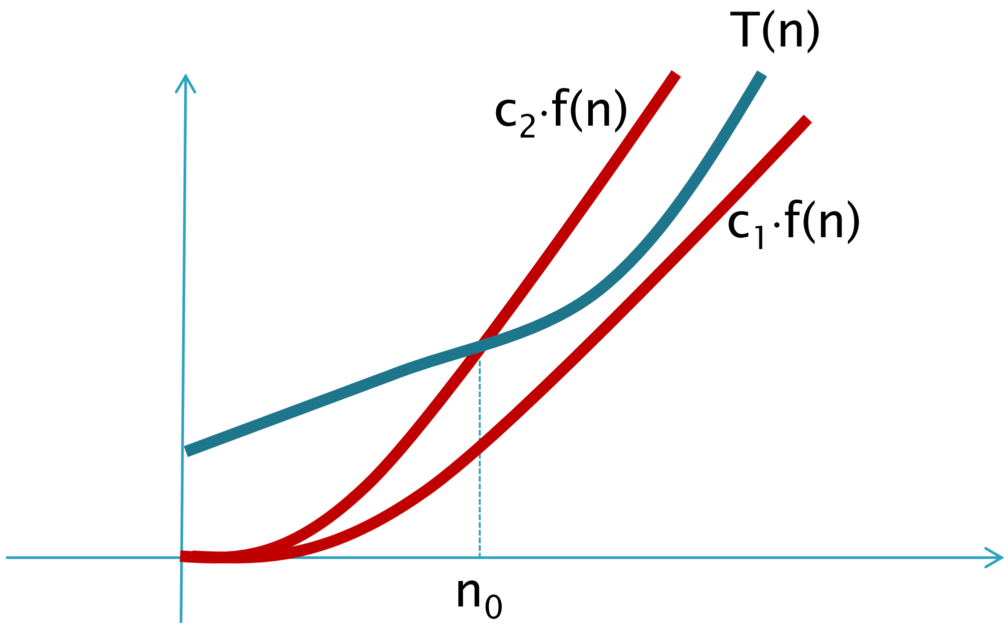


Timpul de executare a algoritmilor

- **T(n)= Θ(f(n)) (Suplimentar)**

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

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



Timpul de executare a algoritmilor

- Notație: `T(n) = O(f(n))`
- comportare **asimptotică**
- caz defavorabil
- O(expresie) = O(termen dominant)

  $O(2n) \Rightarrow O(n)$

  $O(2n^2+4n+1) \Rightarrow O(n^2)$

  $O(n^2 - n) \Rightarrow O(n^2)$

Timpul de executare a algoritmilor

- **Exemplul 1**

```
pentru i = 1, n executa
    pentru j = 1,i executa
        scrie j
    scrie linie noua
```

Afișare
1
1 2
1 2 3
1 2 3 4
1 2 ....n

Timpul de executare a algoritmilor

- **Exemplul 1**

```
pentru i = 1, n executa
     pentru j = 1,i executa
          scrie j
     scrie linie noua
```

Afișare
1
1 2
1 2 3
1 2 3 4
1 2 …….n

$1 + 2 + 3 + \ldots + n = n(n+1)/2$ afișări ale lui j

$O(n^2)$

Timpul de executare a algoritmilor

- **Exemplul 2**

```
p = 1
pentru i = 1, n+1 executa
     pentru j = 1,p executa
           scrie j
     scrie linie noua
     p = p * 2
```

Afișare

1

1 2

1 2 3 4

1 2 3 4 5 6 7 8

1 2 .... 2^n

Timpul de executare a algoritmilor

- **Exemplul 2**

```
i = 0
p = 1
pentru i = 1, n+1 executa
     pentru j = 1,p executa
           scrie j
     scrie linie noua
     p = p * 2
```

Afișare

1

1 2

1 2 3 4

1 2 3 4 5 6 7 8

1 2 .... 2^n

$1 + 2 + 2^2 + \ldots + 2^n = 2^{n+1} - 1$ afișări ale lui j

$O(2^n)$

Timpul de executare a algoritmilor

- Exemplul 3 – Cea mai mică putere a lui 2 mai mare ca n

```
p = 1
cat timp p<=n executa:
  p = p * 2
scrie p
```

Timpul de executare a algoritmilor

- Exemplul 3 – Cea mai mică putere a lui 2 mai mare ca n

```
p = 1
cat timp p<=n executa:
  p = p * 2
scrie p
```

$O(\log_2(n))$

Timpul de executare a algoritmilor

- **Exemplul 4 –** 2-SUM pentru șir crescător

Se dă un vector ordonat cu n elemente întregi distincte. Să se afișeze toate perechile de elemente din vector cu suma 0

Timpul de executare a algoritmilor

- **Exemplul 4** – 2–SUM pentru șir crescător

```
pentru i = 1, n-1 executa
    pentru j = i+1,n executa
        daca v[i]+v[j]==0 atunci
            scrie i,j
```

Timpul de executare a algoritmilor

- **Exemplul 4** – 2-SUM pentru șir crescător

```
pentru i = 1, n-1 executa
    pentru j = i+1,n executa
        daca v[i]+v[j]==0 atunci
            scrie i,j
```

$O(n^2)$

Timpul de executare a algoritmilor

- **Exemplul 4** – 2–SUM pentru șir crescător

```
i = 0
j = n-1
cat timp i<j executa
    daca v[i]+v[j]==0 atunci
        scrie i,j
        i = i + 1
        j = j - 1
    altfel
        daca v[i] + v[j] < 0 atunci
            i = i + 1
        altfel
            j = j - 1
```

Timpul de executare a algoritmilor

- **Exemplul 4** – 2-SUM pentru șir crescător

```
i = 0
j = n-1
cat timp i<j executa
    daca v[i]+v[j]==0 atunci
        scrie i,j
        i = i + 1
        j = j - 1
    altfel
        daca v[i] + v[j] < 0 atunci
            i = i + 1
        altfel
            j = j - 1
```

O(n)

Timpul de executare a algoritmilor

Alte exemple

```
pentru i = 1, n executa
    pentru j = 1, m executa
        operatii O(1)
    pentru k = 1, p executa
        operatii O(1)
```

Timpul de executare a algoritmilor

Alte exemple

```
pentru i = 1, n executa
    pentru j = 1, m executa
          operatii O(1)
    pentru k = 1, p executa
          operatii O(1)
```

$O(n(m+p))$

Timpul de executare a algoritmilor

Alte exemple

- Înmulțirea a două matrice A(n,m) B(m,p)
- Intersecția a două mulțimi cu n respectiv m elemente
- Reuniunea a două mulțimi ordonate cu n respectiv m elemente

Timpul de executare a algoritmilor

Alte exemple

- Înmulțirea a două matrice A(n,m) B(m,p) **O(nmp)**
- Intersecția a două mulțimi cu n respectiv m elemente **O(nm)**
- Reuniunea a două mulțimi ordonate cu n respectiv m elemente- similar **Interclasare O(n+m)**

Timpul de executare a algoritmilor

- Interclasare

```
i = 1; j = 1
cat timp (i<=m) and (j<=n) executa
      daca a[i]<=b[j] atunci
            scrie a[i]; i = i+1
      altfel
            scrie b[j]; j = j+1
cat timp i<=m executa
       scrie a[i]; i=i+1
cat timp j<=n executa
       scrie b[j]; j=j+1
```

Timpul de executare a algoritmilor

Notație: $T(n) = O(f(n))$

Clase de complexitate uzuale:

- **Complexitate logaritmică** $O(\log_2(n))$, $O(\log(n))$

  Exemplu: căutarea binară

-
-

Timpul de executare a algoritmilor

Notație: $T(n) = O(f(n))$

Clase de complexitate uzuale:

- **Complexitate logaritmică** $O(\log_2(n))$, $O(\log(n))$

  Exemplu: căutarea binară

- **Complexitate liniară** $O(n)$

  Exemplu: minimul dintr-un vector

-

Timpul de executare a algoritmilor

Notație: `T(n) = O(f(n))`

Clase de complexitate uzuale:

- **Complexitate logaritmică** $O(\log_2(n))$, $O(\log(n))$

  Exemplu: căutarea binară

- **Complexitate liniară** $O(n)$

  Exemplu: minimul dintr-un vector

- **Complexitate** $O(n \log_2(n))$ **liniară logaritmică**

  Exemplu: sortarea prin interclasare MergeSort

Timpul de executare a algoritmilor

- **Complexitate pătratică** $O(n^2)$
  Exemplu: suma elementelor unei matrice nxn, sortarea prin metoda bulelor, prin selecție

- **Complexitate polinomială** $O(n^k)$, $k \geq 3$
  Exemplu: înmulțirea a două matrice pătratice de dimensiune n

Timpul de executare a algoritmilor

- **Complexitate pătratică** $O(n^2)$
  Exemplu: suma elementelor unei matrice nxn, sortarea prin metoda bulelor, prin selecție

- **Complexitate polinomială** $O(n^k)$, $k \geq 3$
  Exemplu: înmulțirea a două matrice pătratice de dimensiune n

- **Complexitate exponențială** $O(k^n)$, $k \geq 2$
  Exemplu: generarea submulțimilor unei mulțimi cu n elemente

- **Complexitate factorială** $O(n!)$
  Exemplu: generarea permutărilor unui vector cu n elemente