Divide et Impera

Metoda generala de programare

Legaturi stranse cu apelurile recursive

Recursivitatea: probleme cu evaluarea complexitatii

Astazi:

- Descriere: divide et impera
- cum se foloseste
- cum se evalueaza complexitatea
- o teorema pentru o clasa larga de asemenea probleme
- exemple

- apaun@fmi.unibuc.ro

Divide et impera

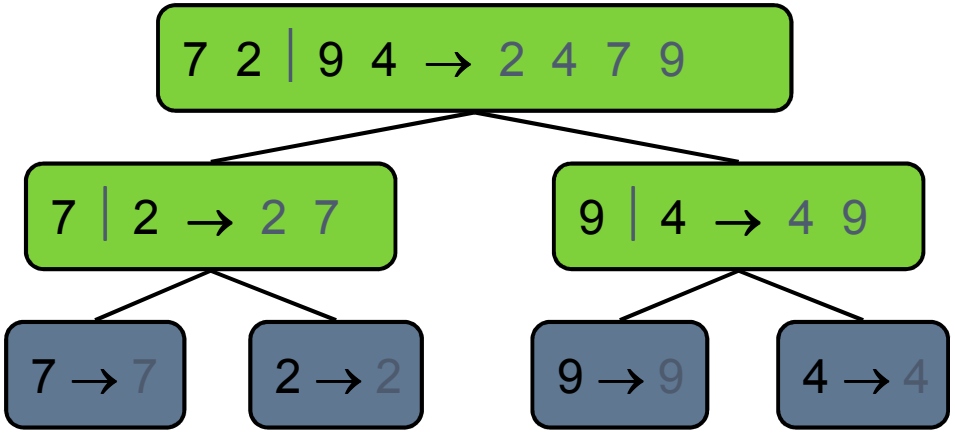
- Maxima latina
- Divide si conduce (divide si cucereste)
- Atribuita lui Cezar
  - folosita inainte de Cezar de Gabinius

- Idee: un inamic trebuie divizat in mai multe componente mici care pot fi cucerite succesiv

- Inamic: problema de programare

mergesort

Algorithm *mergeSort*(*S*, *C*)
  Input sequence *S* with *n* elements, comparator *C*
  Output sequence *S* sorted according to *C*
  if *S.size*() > 1
    (*S*$_1$, *S*$_2$) ← *partition*(*S*, *n*/2)
    mergeSort(*S*$_1$, *C*)
    mergeSort(*S*$_2$, *C*)
    S ← *merge*(*S*$_1$, *S*$_2$)



$$T(n) = O(g(n)) \text{ iff}$$

$$\exists\, c > 0, \exists\, n_0 > 0 \text{ such that } \forall\, n > n_0,$$

$$T(n) \leq c \cdot g(n).$$

Complexitatea algoritmului

- Relatie de recurenta
- Din secventa de n numere ajungem la doua secvente de cate n/2 numere pe care trebuie sa le rezolvam, plus timpul pentru "merge"

- $T(n)=2T(n/2)+cn$

- Vom vedea cum se rezolva acestea in general: $T(n)$ este $O(n \log n)$

Divide et impera

- Se da o instanta x pentru o problema
- Metoda divide et impera functioneaza in felul urmator

```
Del (x)
begin
   Daca x e suficient de mic se rezolva direct, solutia este y
   Altfel
         se sparge problema x in subproblemele x1, x2, …,xk;
         for i=1 to k do   yi=Del(xi);
         combinam y1, y2, …, yk intr-o solutie y pentru x;
   return (y);
end
```

Analiza metodei prezentate

- Deobicei $x_1, x_2, \ldots, x_k$ sunt de aceeasi dimensiune, sa zicem [n/b]
- Complexitatea metodei exprimata de o recurenta

$$T(n) = \begin{cases} c & \text{daca } n \le n_0 \\ aT(\lfloor n/b \rfloor) + f(n) & \text{daca } n > n_0 \end{cases}$$

unde $f(n)$ este timpul necesar pentru spargerea lui x si combinarea y_i-urilor

- cine e *c*, cine e n_0

Demonstratie

- Se inlocuiesc T-urile cu variante de dimensiune mai mica pana la T[1]
- Se obtine o suma de doua progresii
- …

$$T(n) = \sum_{i=0}^{\log_b n} a^i f(n/b^i) + O(n^{\log_b a})$$

Teorema Master

1. if $f(n)$ is $O(n^{\log_b a-\varepsilon})$, then $T(n)$ is $\Theta(n^{\log_b a})$
2. if $f(n)$ is $\Theta(n^{\log_b a}\log^k n)$, then $T(n)$ is $\Theta(n^{\log_b a}\log^{k+1} n)$
3. if $f(n)$ is $\Omega(n^{\log_b a+\varepsilon})$, then $T(n)$ is $\Theta(f(n))$,
   provided $af(n/b) \le \delta f(n)$ for some $\delta < 1$.

Exemplul 1

$$T(n) = 4T(n/2) + n$$

- Cine e a, b?

$$T(n) = \begin{cases} c & \text{daca } n \le n_0 \\ aT(\lfloor n/b \rfloor) + f(n) & \text{daca } n > n_0 \end{cases}$$

unde $f(n)$ este timpul necesar pentru spargerea lui x si combinarea y_i-urilor

- a=4, b=2, f(n)=n,
- verificam f(n) cu $n^{\log_b a}$

1. if $f(n)$ is $O(n^{\log_b a-\varepsilon})$, then $T(n)$ is $\Theta(n^{\log_b a})$
2. if $f(n)$ is $\Theta(n^{\log_b a}\log^k n)$, then $T(n)$ is $\Theta(n^{\log_b a}\log^{k+1} n)$
3. if $f(n)$ is $\Omega(n^{\log_b a+\varepsilon})$, then $T(n)$ is $\Theta(f(n))$,
   provided $af(n/b) \le \delta f(n)$ for some $\delta < 1$.

- Deci n este $O(n^2)$ cazul 1

if $f(n)$ is $O(n^{\log_b a-\varepsilon})$, then $T(n)$ is $\Theta(n^{\log_b a})$

- Adica $T(n)\ este\ \Theta(n^2)$

Exemplul 2 $T(n) = 2T(n/2) + n \log n$

- a=2, b=2, f=n log n

$$T(n) = \begin{cases} c & \text{if } n < d \\ aT(n/b) + f(n) & \text{if } n \geq d \end{cases}$$

1. if $f(n)$ is $O(n^{\log_b a - \varepsilon})$, then $T(n)$ is $\Theta(n^{\log_b a})$
2. if $f(n)$ is $\Theta(n^{\log_b a} \log^k n)$, then $T(n)$ is $\Theta(n^{\log_b a} \log^{k+1} n)$
3. if $f(n)$ is $\Omega(n^{\log_b a + \varepsilon})$, then $T(n)$ is $\Theta(f(n))$, provided $af(n/b) \leq \delta f(n)$ for some $\delta < 1$.

- suntem in cazul 2 deci

T(n) este $O(n \log^2 n)$

Exemplul 3

$$T(n) = T(n/3) + n \log n$$

- a=1, b=3, f=n log n

$$T(n) = \begin{cases} c & \text{if } n < d \\ aT(n/b) + f(n) & \text{if } n \geq d \end{cases}$$

1. if $f(n)$ is $O(n^{\log_b a - \varepsilon})$, then $T(n)$ is $\Theta(n^{\log_b a})$
2. if $f(n)$ is $\Theta(n^{\log_b a} \log^k n)$, then $T(n)$ is $\Theta(n^{\log_b a} \log^{k+1} n)$
3. if $f(n)$ is $\Omega(n^{\log_b a + \varepsilon})$, then $T(n)$ is $\Theta(f(n))$, provided $af(n/b) \leq \delta f(n)$ for some $\delta < 1$.

- suntem in cazul 3 deci

T(n) este O(n log n)

Exemplul 4

$$T(n) = 8T(n/2) + n^2$$

- a=8, b=2, f=n^2

$$T(n) = \begin{cases} c & \text{if } n < d \\ aT(n/b) + f(n) & \text{if } n \geq d \end{cases}$$

1. if $f(n)$ is $O(n^{\log_b a - \varepsilon})$, then $T(n)$ is $\Theta(n^{\log_b a})$
2. if $f(n)$ is $\Theta(n^{\log_b a} \log^k n)$, then $T(n)$ is $\Theta(n^{\log_b a} \log^{k+1} n)$
3. if $f(n)$ is $\Omega(n^{\log_b a + \varepsilon})$, then $T(n)$ is $\Theta(f(n))$, provided $af(n/b) \leq \delta f(n)$ for some $\delta < 1$.

- suntem in cazul 1 deci

T(n) este $O(n^3)$

Exemplul 5

$$T(n) = 9T(n/3) + n^3$$

- a=9, b=3, f=n^3

$$T(n) = \begin{cases} c & \text{if } n < d \\ aT(n/b) + f(n) & \text{if } n \geq d \end{cases}$$

1. if $f(n)$ is $O(n^{\log_b a - \varepsilon})$, then $T(n)$ is $\Theta(n^{\log_b a})$
2. if $f(n)$ is $\Theta(n^{\log_b a} \log^k n)$, then $T(n)$ is $\Theta(n^{\log_b a} \log^{k+1} n)$
3. if $f(n)$ is $\Omega(n^{\log_b a + \varepsilon})$, then $T(n)$ is $\Theta(f(n))$, provided $af(n/b) \leq \delta f(n)$ for some $\delta < 1$.

- suntem in cazul 3 deci

T(n) este $O(n^3)$

Exemplul 6

$$T(n) = T(n/2) + 1$$

- a=1, b=2, f=1

$$T(n) = \begin{cases} c & \text{if } n < d \\ aT(n/b) + f(n) & \text{if } n \geq d \end{cases}$$

1. if $f(n)$ is $O(n^{\log_b a - \varepsilon})$, then $T(n)$ is $\Theta(n^{\log_b a})$
2. if $f(n)$ is $\Theta(n^{\log_b a} \log^k n)$, then $T(n)$ is $\Theta(n^{\log_b a} \log^{k+1} n)$
3. if $f(n)$ is $\Omega(n^{\log_b a + \varepsilon})$, then $T(n)$ is $\Theta(f(n))$, provided $af(n/b) \leq \delta f(n)$ for some $\delta < 1$.

- suntem in cazul 2 cu k=0 deci

T(n) este O(log n)

Exemplul 7 $T(n) = 2T(n/2) + \log n$

- a=2, b=2, f=log n

$$T(n) = \begin{cases} c & \text{if } n < d \\ aT(n/b) + f(n) & \text{if } n \geq d \end{cases}$$

1. if $f(n)$ is $O(n^{\log_b a - \varepsilon})$, then $T(n)$ is $\Theta(n^{\log_b a})$
2. if $f(n)$ is $\Theta(n^{\log_b a} \log^k n)$, then $T(n)$ is $\Theta(n^{\log_b a} \log^{k+1} n)$
3. if $f(n)$ is $\Omega(n^{\log_b a + \varepsilon})$, then $T(n)$ is $\Theta(f(n))$, provided $af(n/b) \leq \delta f(n)$ for some $\delta < 1$.

- suntem in cazul 1 deci

T(n) este O(n)

O schita de demonstratie a teoremei

- Se substituie iterativ termenii

$$
\begin{aligned}
T(n) &= aT(n/b) + f(n) \\
&= a(aT(n/b^2)) + f(n/b)) + bn \\
&= a^2 T(n/b^2) + af(n/b) + f(n) \\
&= a^3 T(n/b^3) + a^2 f(n/b^2) + af(n/b) + f(n) \\
&= \dots \\
&= a^{\log_b n} T(1) + \sum_{i=0}^{(\log_b n)-1} a^i f(n/b^i) \\
&= n^{\log_b a} T(1) + \sum_{i=0}^{(\log_b n)-1} a^i f(n/b^i)
\end{aligned}
$$

- Si avem trei cazuri acum:
  - Primul termen este dominant, sau
  - Amandoi termenii sunt "la fel" de mari, sau
  - Suma este o serie geometrica

exemplu

- Traversarile de arbori
  - Preordine
  - Inordine
  - Postordine

$$T(n) = \begin{cases} c & \text{if } n < d \\ aT(n/b) + f(n) & \text{if } n \geq d \end{cases}$$

1. if $f(n)$ is $O(n^{\log_b a - \varepsilon})$, then $T(n)$ is $\Theta(n^{\log_b a})$
2. if $f(n)$ is $\Theta(n^{\log_b a} \log^k n)$, then $T(n)$ is $\Theta(n^{\log_b a} \log^{k+1} n)$
3. if $f(n)$ is $\Omega(n^{\log_b a + \varepsilon})$, then $T(n)$ is $\Theta(f(n))$,
   provided $af(n/b) \leq \delta f(n)$ for some $\delta < 1$.

T(n)=2T(n/2)+1

Complexitate O(n)

Exemplu: inmultirea a doi intregi

- Se dau I si J doi intregi scrisi in baza 2
- Vrem sa aflam rezultatul inmultirii
- Cum rezolvam?

Inmultirea prin divide et impera

- Divide: consideram jumatatea mai semnificativa si jumatatea nesemnificativa pentru cei doi intregi

$$I = I_h 2^{n/2} + I_l$$

$$J = J_h 2^{n/2} + J_l$$

- Asadar inmultirea devine:

$$I * J = (I_h 2^{n/2} + I_l) * (J_h 2^{n/2} + J_l)$$

$$= I_h J_h 2^n + I_h J_l 2^{n/2} + I_l J_h 2^{n/2} + I_l J_l$$

- Deci ajungem la relatia:
  - T(n)=4T(n/2)+n

Pentru T(n)=4T(n/2)+n

- a=4, b=2, f=n

$$T(n) = \begin{cases} c & \text{if } n < d \\ aT(n/b) + f(n) & \text{if } n \geq d \end{cases}$$

1. if $f(n)$ is $O(n^{\log_b a - \varepsilon})$, then $T(n)$ is $\Theta(n^{\log_b a})$
2. if $f(n)$ is $\Theta(n^{\log_b a} \log^k n)$, then $T(n)$ is $\Theta(n^{\log_b a} \log^{k+1} n)$
3. if $f(n)$ is $\Omega(n^{\log_b a + \varepsilon})$, then $T(n)$ is $\Theta(f(n))$, provided $af(n/b) \leq \delta f(n)$ for some $\delta < 1$.

- Care e complexitatea?

- E mai rapid decat inmultirea “de mana”?

Imbunatatire pentru inmultire

- Spargem tot in jumatatea mai semnificativa si jumatatea mai nesemnificativa

$$I = I_h 2^{n/2} + I_l$$

$$J = J_h 2^{n/2} + J_l$$

$$\begin{aligned} I * J &= I_h J_h 2^n + [(I_h - I_l)(J_l - J_h) + I_h J_h + I_l J_l] 2^{n/2} + I_l J_l \\ &= I_h J_h 2^n + [(I_h J_l - I_l J_l - I_h J_h + I_l J_h) + I_h J_h + I_l J_l] 2^{n/2} + I_l J_l \\ &= I_h J_h 2^n + (I_h J_l + I_l J_h) 2^{n/2} + I_l J_l \end{aligned}$$

- Deci T(n)=3T(n/2)+n

Pentru T(n)=3T(n/2)+n

- a=3, b=2, f=n

$$T(n) = \begin{cases} c & \text{if } n < d \\ aT(n/b) + f(n) & \text{if } n \geq d \end{cases}$$

1. if $f(n)$ is $O(n^{\log_b a - \varepsilon})$, then $T(n)$ is $\Theta(n^{\log_b a})$
2. if $f(n)$ is $\Theta(n^{\log_b a} \log^k n)$, then $T(n)$ is $\Theta(n^{\log_b a} \log^{k+1} n)$
3. if $f(n)$ is $\Omega(n^{\log_b a + \varepsilon})$, then $T(n)$ is $\Theta(f(n))$, provided $af(n/b) \leq \delta f(n)$ for some $\delta < 1$.

- Care e complexitatea?
- T(n) este $O(n^{1.585})$ (Karatsuba)
- Cel mai rapid O(n log n log log n)

Alte probleme

- De acum incolo in aceasta lectie consideram multiplicarea ca operatia de baza in complexitate

  (de complexitate O(1))

Inmultire de polinoame

- $A(x)=a_0+a_1x+\ldots+a_{n-1}x^{n-1}$
- $B(x)=b_0+b_1x+\ldots+b_{n-1}x^{n-1}$
- Vrem sa calculam $C(x)=A(x)B(x)$

- cu $c_i=a_0*b_i+a_1*b_{i-1}+\ldots+a_i*b_0$
- si $C(x)=c_0+c_1x+c_2x^2\ldots+c_{2n-2}x^{2n-2}$

Inmultirea polinoamelor

- Implementarea naiva: $O(n^2)$ pentru fiecare coeficient avem n inmultiri, si avem 2n-1 coeficienti

- Divide et impera!
- Spargem polinoamele in polinoame mai mici

- $A(x)=a_0+a_1x+\ldots+a_{n-1}x^{n-1}$
- $A(x)=a_0+a_1x+\ldots+a_{n/2-1}x^{n/2-1}$
  $+x^{n/2}(a_{n/2}+a_{n/2+1}x+\ldots\ a_{n-1}x^{n/2-1/2})$
- $A(x)=P(x)+x^{n/2}Q(x)$
- La fel: $B(x)= S(x)+x^{n/2}R(x)$
- $C(x)=PS+x^{n/2}\ (PR+QS)+x^{n}QR$

- La fel ca la inmultirea intregilor!!

Inmultirea polinoamelor

- Se poate mai bine cu divide et impera ajungem la O(n log n)

Ridicarea la o putere

- Consideram ca adunarile si inmultirile sunt de cost O(1)
- Input: numar a si exponent pe n biti b
- Vrem a^b
- Implementarea naiva: O(b) deci $O(2^n)$
- Divide et impera!

- Ce sa dividem? exponentul

- Algoritm:

  power (a,b)

  Daca b=1 return a

  Daca b este par return power(a,b/2)^2

  Daca b este impar

  return a*power (a,(b-1)/2) ^{2}

  Deci T(n)=T(n-1)+O(1)

  Rezolvam prin Teorema Master: O(n)

Calcularea numerelor Fibonacci

- Implementare naiva:

fib(n)
Daca n<=1 return 1
Altfel return fib(n-1)+fib(n-2)

- Complexitate: T(n)=T(n-1)+T(n-2)+1
- Aproape de valoarea numarului Fib(n) care este exponentiala in n

Fibonacci

- Daca mergem de jos in sus si tinem minte ultimele doua valori putem obtine F(n) in timp polinomial
- La fel si cu divide et impera:
- Vrem sa reducem problema Fibonacci la o problema de inmultire de matrici
- A=$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$ $\mathbf{F}(n) = \begin{pmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{pmatrix}$

- Teorema: $F(n)=A^n$

- Folosim algoritmul divide et impera de mai sus pentru inmultirea numerelor pe matrici patratice
- Unde pentru a avem A si pentru b avem n
- Asadar obtinem F(n) in timp O(log n)

- Daca nu consideram ca inmultirea de numere este O(1) algoritmul cu matrici ramane mai rapid:
- Pentru “bottom-up” avem:
  - n adunari de numere de n digiti: $O(n^2)$
- Pentru matrici avem:
  - O(log n) inmultiri de numere de n digiti
  - Daca inmultirea e implementata “prost” avem $O(n^2 \log n)$ care e evident mai prost decat $O(n^2)$
  - Karatsuba: $O(n^{1.53} \log n)$
  - Cel mai bine $O(n \log^2 n \log \log n)$

Inmultire de matrici patratice

- 2 matrici de nXn de inmultit
- Implementare naiva: $O(n^3)$
- Divide et impera: spargem matricile de dimensiune n in 4 matrici de dimensiune n/2

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix} B = \begin{pmatrix} e & f \\ g & h \end{pmatrix} \text{ and } C = \begin{pmatrix} r & s \\ t & u \end{pmatrix}$$

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix} B = \begin{pmatrix} e & f \\ g & h \end{pmatrix} \text{ and } C = \begin{pmatrix} r & s \\ t & u \end{pmatrix}$$

- r=ae+bg, s=af+bh, t=ce+dg, u=cf+dh

- Deci T(n)=8T(n/2)+4 adunari de matrici de dimensiune n/2
- $T(n)=8T(n/2)+4cn^2$
- Cazul 1 in teorema Master, $O(n^3)$
- La fel ca implementarea naiva!!!

- Cum imbunatatim? STRASSEN

Inmultire de matrice 2x2 rapida

- 7 inmultiri si 18 adunari

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix} B = \begin{pmatrix} e & f \\ g & h \end{pmatrix} \text{ and } C = \begin{pmatrix} r & s \\ t & u \end{pmatrix}$$

$$P1 = a(f - h)$$
$$P2 = (a + b)h$$
$$P3 = (c + d)e$$
$$P4 = d(g - e)$$
$$P5 = (a + d)(e + h)$$
$$P6 = (b - d)(g + h)$$
$$P7 = (a - c)(e + f)$$

$$\text{Then, } r = P5 + P4 - P2 + P6$$
$$s = P1 + P2$$
$$t = P3 + P4$$
$$u = P5 + P1 - P3 - P7$$

- Deci relatia de recurenta devine
- $T(n)=7T(n/2)+O(n^2)$

- Tot in cazul 1 in teorema Master
- $T(n)=O(n^{\log 7})=O(n^{2.81})$

- Algoritm complicat: Coppersmith and Winograd $T(n)=O(n^{2.376})$