Analiza timpului de rulare a algoritmilor recursivi

Exemplu: metode de sortare

- Quicksort
- Merge sort
- Heap sort

$O(n \log n)$

- Radix Sort
- Bubble sort
- Count sort
- Insertion sort

Nu sunt bazați pe comparații între chei

$O(n^2)$

Insertion Sort

39 5 7 33 200 96 70

I. 39 rămâne pe loc

II. căutăm poziția pe care să îl inserăm pe 5

5 39 7 33 200 96 70

III. căutăm poziția pe care să îl inserăm pe 7

5 7 39 33 200 96 70

IV. 5 7 33 39 200 96 70

V. 200 este deja unde trebuie

VI. 5 7 33 39 96 200 70

VII. 5 7 33 39 70 96 200

Complexitate: La pasul i facem (în cel mai defavorabil caz) i operații

În total vom avea: $1 + 2 + \ldots + n = O(n^2)$

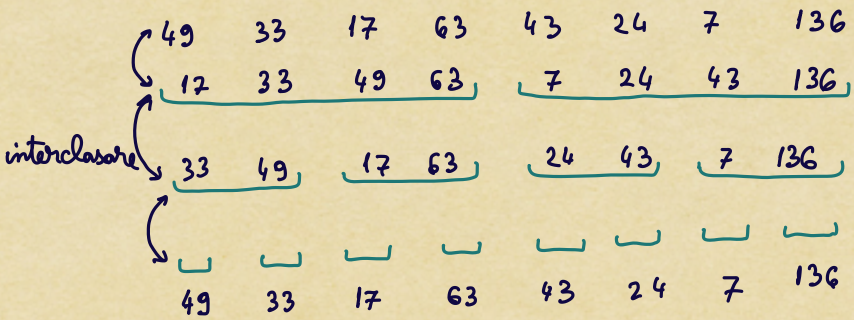
$T(n)$ (= timpul de rulare pt a rezolva o problemă de dimensiune n)

$T(n) = T(n-1) + O(n) = O(n^2)$

Ex: $T(n) = T(n/3) + T(2n/3) + O(n) = ?$

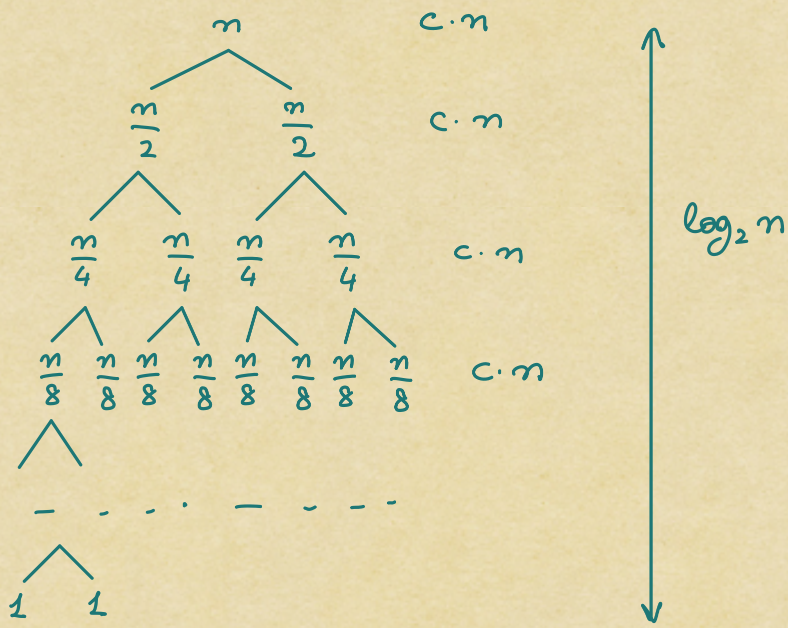
Merge Sort

Idee: Împărțim problema în subprobleme mai mici pe care le rezolvăm și combinăm rezultatele



$T(n) = 2T\left(\frac{n}{2}\right) + c \cdot n = O(n \log n)$

Nu scriem baza pt că putem oricum să transformăm



Căutare binară

$$T(n) = T\left(\frac{n}{2}\right) + 1 = O(\log n)$$

$$\begin{array}{ccc}
n & 1 & \uparrow \\
| & & \\
\frac{n}{2} & 1 & \\
| & & \log_2 n \\
\frac{n}{4} & 1 & \\
\vdots & \vdots & \\
1 & 1 & \downarrow
\end{array}$$

Metoda substituției pentru rezolvarea recurențelor

Exemplu: $T(n) = 2 \cdot T\left(\frac{n}{2}\right) + n = O(n \log n)$

1. „Ghicim" soluția
2. Arătăm că $T(n) \leq c \cdot n \log_2 n$

Presupunem că $T(n/2) \leq c \cdot \frac{n}{2} \log_2 \frac{n}{2}$

Știm că $T(n) = 2T(n/2) + n$

$$T(n) \leq 2 \cdot c \cdot \frac{n}{2} \cdot \log_2 \frac{n}{2} + n$$

$$T(n) \leq cn \cdot \log \frac{n}{2} + n$$

$$T(n) \leq cn \cdot \log_2 n - cn \log_2 2 + n$$

$$T(n) \leq cn \log_2 n + \underbrace{n(1-c)}_{\text{dacă}} \qquad 1 - c \leq 0$$

Adevărat pt $c \geq 1$

Avem $T(n) \leq c \cdot n \log_2 n$

- $T(n) = T\left(\frac{n}{2}\right) + 1 = O(\log_2 n)$

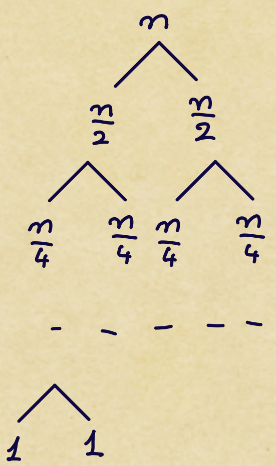
Vrem să arătăm că $T(n) \leq c \cdot \log_2 n$

Presupunem că $T\left(\frac{n}{2}\right) \leq c \cdot \log_2 \frac{n}{2}$

$$T(n) \leq c \cdot \log_2 \frac{n}{2} + 1 = c \cdot \log_2 n \underbrace{- c + 1}_{\substack{\leq 0 \\ \forall c \geq 1}}$$

$$\leq c \log_2 n \quad \forall c \geq 1$$

- $T(n) = 2T\left(\frac{n}{2}\right) + 1 = O(n)$



1

2

4

$2^{\log_2 n} = n$

$1 + 2 + 4 + 8 + \ldots = O(n)$

$\frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \ldots = O(\log n)$

Formule Kinda?

Vrem să arătăm că $T(n) \leq c \cdot n$

Presupunem că $T\left(\frac{n}{2}\right) \leq c \cdot \frac{n}{2}$

$$T(n) \leq 2 \cdot c \cdot \frac{n}{2} + 1 = c \cdot n + 1$$

Vrem să arătăm că $T(n) \leq c \cdot n - d$

constantă

Pp. că $T\left(\frac{n}{2}\right) \leq c \cdot \frac{n}{2} - d$

$$T(n) \leq 2 \cdot \left(c \cdot \frac{n}{2} - d\right) + 1 = c \cdot n - 2d + 1 = c \cdot n - d \underbrace{- d + 1}_{\leq 0 \ \forall d \geq 1}$$

Teorema Master: Dacă avem recurențe de forma $T(n) = aT(\frac{n}{b}) + f(n)$

Trei cazuri:

1) $f(n) \in O(n^{\log_b a - \varepsilon})$ pt un $\varepsilon > 0$

atunci $T(n) \in \Theta(n^{\log_b a})$

2) $f(n) \in \Theta(n^{\log_b a})$

$T(n) = \Theta(n^{\log_b a} \log_2 n)$

3) $f(n) \in \Omega(n^{\log_b a + \varepsilon})$ pt $\varepsilon > 0$ și

$a f(\frac{n}{b}) \leq c \cdot f(n)$ pt $c \geq 0$

$T(n) \in \Theta(f(n))$

- $T(n) = T(\frac{n}{2}) + 1$

$f(n) = 1$

$n^{\log_b a} = n^{\log_2 1} = n^0 = 1$

$f(n) \in \Theta(1)$

- $T(n) = 9T(\frac{n}{3}) + n$

$f(n) = n$

$a = 9$

$b = 3$

$n^{\log_b a} = n^2$

$f(n) \in O(n^{2-\varepsilon})$

$T(n) = \Theta(n^2)$

- $T(n) = T(\frac{n}{3}) + T(\frac{2n}{3}) + 1$

nu pot aplica Th. Master