

Διαίρει-και-Βασίλευε

Διδάσκοντες: **Σ. Ζάχος, Δ. Φωτάκης**
Επιμέλεια διαφανειών: **Δ. Φωτάκης**

Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών

Εθνικό Μετσόβιο Πολυτεχνείο



Διαίρει-και-Βασίλευε

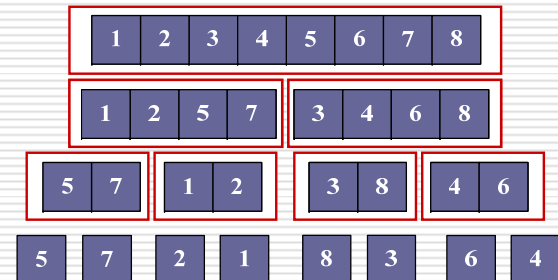
- Γενική μέθοδος σχεδιασμού αλγορίθμων:
 - **Διαίρεση** σε (≥ 2) υποπροβλήματα (σημαντικά) μικρότερου μεγέθους.
 - **Ανεξάρτητη επίλυση** υπο-προβλημάτων (αναδρομικά) (για μικρά υποπροβλήματα εφαρμόζονται στοιχειώδεις αλγόριθμοι).
 - **Σύνθεση** λύσης αρχικού προβλήματος από λύσεις υποπροβλημάτων.
- **Ισχυρή μέθοδος, με πολλές σημαντικές εφαρμογές!**
 - Ταξινόμηση – Επιλογή: MergeSort, QuickSort, QuickSelect.
 - Πολλαπλασιασμός αριθμών, πινάκων, FFT.
 - «Εκτέλεση»: Δυσδική αναζήτηση, ύψωση σε δύναμη.
- (Εύκολη) ανάλυση με **αναδρομικές σχέσεις**.
 - Μη γραμμικές, συγκριμένης μορφής.

MergeSort

- Πρόβλημα Ταξινόμησης:
 - **Είσοδος** : ακολουθία n αριθμών (a_1, a_2, \dots, a_n).
 - **Έξοδος** : μετάθεση (a'_1, a'_2, \dots, a'_n) με αριθμούς σε **αύξουσα σειρά** ($\forall i, a'_i \leq a'_{i+1}$).
- MergeSort (ταξινόμηση με **συγχώνευση**):
 - **Διαίρεση** ακολουθίας εισόδου (n στοιχεία) σε δύο υπο-ακολουθίες ίδιου μήκους ($n/2$ στοιχεία).
 - **Ταξινόμηση** υπο-ακολουθιών **αναδρομικά**.
 - **Συγχώνευση** (merge) δύο ταξινομημένων υπο-ακολουθιών σε **μία ταξινομημένη ακολουθία**.

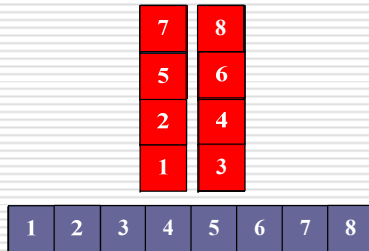
```
MergeSort(int A[], int left, int right) {  
    if (left >= right) return;  
    mid = (left + right) / 2;  
    mergeSort(A, left, mid);  
    mergeSort(A, mid+1, right);  
    merge(A, left, mid, right); }
```

MergeSort



Συγχώνευση

- **Συγχώνευση** ταξινομημένων $A[\text{left} \dots \text{mid}]$ και $A[\text{mid}+1 \dots \text{right}]$ σε **ταξινομημένο** $A[\text{left} \dots \text{right}]$.



Merge

- **Συγχώνευση** ταξινομημένων $A[\text{left} \dots \text{mid}]$ και $A[\text{mid}+1 \dots \text{right}]$ σε **ταξινομημένο** $A[\text{left} \dots \text{right}]$.
- $X[\text{right}-\text{left}+1] \leftarrow A[\text{left} \dots \text{right}]$; // προσωρινή αποθήκευση
 $i : 0 \leq i \leq \text{xmid}$ // δείκτης αριστερό τμήμα
 $j : \text{xmid}+1 \leq j \leq \text{xright}$ // δείκτης δεξιό τμήμα
 $k : \text{left} \leq k \leq \text{right}$ // δείκτης στον συγχωνευμένο πίνακα
- $X[i]$: μικρότερο διαθέσιμο στοιχείο στο **αριστερό** τμήμα.
 $X[j]$: μικρότερο διαθέσιμο στοιχείο στο **δεξιό** τμήμα.
- while $((i \leq \text{xmid}) \ \&\& \ (j \leq \text{xright}))$
if $(X[i] \leq X[j])$ $A[k++] = X[i++]$;
else $A[k++] = X[j++]$;
- Όταν ένα τμήμα **εξαντληθεί**, αντιγράφουμε όλα τα στοιχεία του άλλου στο $A[]$.

Ορθότητα

- Ορθότητα **merge** επειδή τα τμήματα είναι **ταξινομημένα**.
 - Όταν ένα στοιχείο μεταφέρεται στον $A[]$, δεν υπάρχει μικρότερο διαθέσιμο στοιχείο στα δύο τμήματα.
- Ορθότητα **mergeSort** αποδεικνύεται **επαγωγικά** :
 - Βάση (ένα στοιχείο) τετριμμένη.
 - Δύο τμήματα σωστά ταξινομημένα (**επαγωγική υποθ.**) και συγχωνεύονται σωστά (**ορθότητα merge**) \Rightarrow **Σωστά ταξινομημένος** πίνακας $A[]$.

Χρόνος Εκτέλεσης

- Χρόνος εκτέλεσης **merge** (για n στοιχεία) : $\Theta(n)$ (γραμμικός)
 - $\Theta(1)$ λειτουργίες για κάθε στοιχείο.
- Χρόνος εκτέλεσης αλγόριθμων «διαιρεί-και-βασίλευε» με διατύπωση και λύση αναδρομικής εξίσωσης λειτουργίας.
- $T(n)$: χρόνος (χ.π.) για ταξινόμηση n στοιχείων.
 - $T(n/2)$: ταξινόμηση **αριστερού** τμήματος ($n/2$ στοιχεία).
 - $T(n/2)$: ταξινόμηση **δεξιού** τμήματος ($n/2$ στοιχεία).
 - $\Theta(n)$: **συγχώνευση** ταξινομημένων τμημάτων.
$$T(n) = 2T(n/2) + \Theta(n), T(1) = \Theta(1)$$
- Χρόνος εκτέλεσης **MergeSort**: $T(n) = ???$

Δέντρο Αναδρομής

$$T(n) = 2T(n/2) + \Theta(n),$$

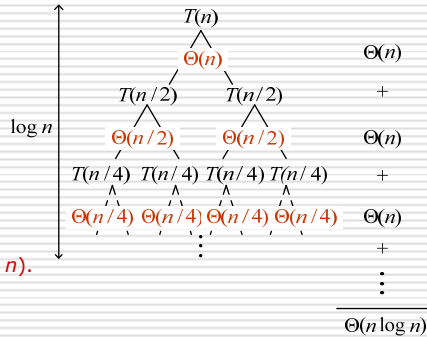
$$T(1) = \Theta(1)$$

Δέντρο αναδρομής :

Ύψος : $\Theta(\log n)$
 # κορυφών : $\Theta(n)$

Χρόνος / επίπεδο : $\Theta(n)$

Συνολικός χρόνος : $\Theta(n \log n)$.



Master Theorem

- Ανάλυση χρόνου εκτέλεσης αλγορίθμων «διαιρεί-και-βασίλευε» με αναδρομικές σχέσεις της μορφής

$$T(n) = aT(n/b) + f(n), T(1) = \Theta(1)$$

όπου a, b σταθερές και $f(n)$ θετική συνάρτηση.

- Επίλυση με Θεώρημα Κυρίαρχου Όρου (Master Theorem)

1. Αν $f(n) = O(n^{\log_b a - \epsilon})$, $\epsilon > 0$, τότε $T(n) = \Theta(n^{\log_b a})$

2. Αν $f(n) = \Theta(n^{\log_b a})$, τότε $T(n) = \Theta(n^{\log_b a} \log n)$

3. Αν $f(n) = \Omega(n^{\log_b a + \epsilon})$, $\epsilon > 0$, και $af(n/b) < f(n)$, τότε $T(n) = \Theta(f(n))$

- Ασυμπτωτικά μεγαλύτερος από $f(n)$ και $n^{\log_b a}$ καθορίζει λύση.

Master Theorem: Ειδικές Μορφές

- Όταν $f(n) = \Theta(n)$, δηλ. $T(n) = aT(n/b) + \Theta(n), T(1) = \Theta(1)$

1. Αν $a > b$, τότε $T(n) = \Theta(n^{\log_b a})$

2. Αν $a = b$, τότε $T(n) = \Theta(n \log n)$

3. Αν $a < b$, τότε $T(n) = \Theta(n)$

- Αν $T(n) = T(\gamma_1 n) + T(\gamma_2 n) + \Theta(n), T(1) = \Theta(1)$

με $\gamma_1 + \gamma_2 < 1$, τότε $T(n) = \Theta(n)$

- Όταν $f(n) = \Theta(n^d)$, δηλ. $T(n) = aT(n/b) + \Theta(n^d), T(1) = \Theta(1)$

1. Αν $d < \log_b a$, τότε $T(n) = \Theta(n^{\log_b a})$

2. Αν $d = \log_b a$, τότε $T(n) = \Theta(n^d \log n)$

3. Αν $d > \log_b a$, τότε $T(n) = \Theta(n^d)$

Παραδείγματα

□ $T(n) = 9T(n/3) + n$. $T(n) = \Theta(n^2)$ (περ. 1)

□ $T(n) = T(2n/3) + 1$. $T(n) = \Theta(\log n)$ (περ. 2)

□ $T(n) = 3T(n/4) + n \log n$. $T(n) = \Theta(n \log n)$ (περ. 3)

□ $T(n) = 2T(n/2) + n$. $T(n) = \Theta(n \log n)$ (περ. 2)

□ $T(n) = 2T(n/2) + n \log n$.

- Δεν επιπτεί! Με δέντρο αναδρομής βρίσκουμε ότι $T(n) = \Theta(n \log^2 n)$.

Πολλαπλασιασμός Αριθμών

- Υπολογισμός αθροίσματος $x + y$, x και y αριθμοί n -bits.

- Κλασσικός αλγόριθμος πρόσθεσης, χρόνος $\Theta(n)$.

- Υπολογισμός γινομένου $x \times y$, x και y αριθμοί με n -bits.

- Κλασσικός αλγόριθμος πολ/μού, χρόνος $\Theta(n^2)$.

- Καλύτερος αλγόριθμος;

- Διαιρεί-και-Βασίλευε:

- Διάρθρωση: $x = 2^{n/2}x_h + x_l$, $y = 2^{n/2}y_h + y_l$

$$x \times y = 2^n \overbrace{x_h y_h}^{z_h} + 2^{n/2} \overbrace{(x_h y_l + x_l y_h)}^{z_m} + \overbrace{x_l y_l}^{z_l} = 2^n z_h + 2^{n/2} z_m + z_l$$

- 4 πολλαπλασιασμοί $(n/2)$ -bits, 2 ολισθήσεις, 3 προσθέσεις.

- Χρόνος: $T_1(n) = 4T_1(n/2) + \Theta(n) \Rightarrow T_1(n) = \Theta(n^2)$

Πολλαπλασιασμός Αριθμών

$$x \times y = 2^n \overbrace{x_h y_h}^{z_h} + 2^{n/2} \overbrace{(x_h y_l + x_l y_h)}^{z_m} + \overbrace{x_l y_l}^{z_l} = 2^n z_h + 2^{n/2} z_m + z_l$$

- Όμως z_m υπολογίζεται με 1 μόνο πολ/μο $(n/2+1)$ -bits.

$$z_m = (x_h + x_l)(y_h + y_l) - x_h y_h - x_l y_l$$

- 3 πολλαπλασιασμοί $(n/2)$ -bits, 2 ολισθήσεις, 6 προσθέσεις.

- Χρόνος: $T(n) = 3T(n/2) + \Theta(n) \Rightarrow T(n) = \Theta(n^{\log_2 3})$

- Παράδειγμα: $2576 \times 7935 = 20440560$

$$x_h = 25, x_l = 76, y_h = 79, y_l = 35$$

$$z_h = 25 \times 79 = 1975, z_l = 76 \times 35 = 2660$$

$$z_m = (25 + 76)(79 + 35) - 1975 - 2660 =$$

$$= 101 \times 114 - 1975 - 2660 = 11514 - 1975 - 2660 = 6879$$

$$x \times y = 1975 \cdot 10^4 + 6879 \cdot 10^2 + 2660 = 20440560$$

Πολλαπλασιασμός Πινάκων

- Υπολογισμός γινομένου $C = A \times B$.

A, B τετραγωνικοί πίνακες $n \times n$.

- Εφαρμογή ορισμού: $C[i, j] = \sum_{k=1}^n A[i, k]B[k, j]$

- Χρόνος $\Theta(n^3)$ (n^2 στοιχεία, χρόνος $\Theta(n)$ για καθένα).

- Διαιρεί-και-Βασίλευε:

$$C_{11} = A_{11}B_{11} + A_{12}B_{21}$$

$$C_{12} = A_{11}B_{12} + A_{12}B_{22}$$

$$C_{21} = A_{21}B_{11} + A_{22}B_{21}$$

$$C_{22} = A_{21}B_{12} + A_{22}B_{22}$$

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}, C = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}$$

- 8 πολ/μοι και 4 προσθέσεις πινάκων $\frac{n}{2} \times \frac{n}{2}$

- Χρόνος: $T_1(n) = 8T_1(n/2) + \Theta(n^2) \Rightarrow T_1(n) = \Theta(n^3)$

Αλγόριθμος Strassen (1969)

$$D_1 = (A_{21} + A_{22} - A_{11})(B_{22} - B_{12} + B_{11})$$

$$D_2 = A_{11}B_{11}$$

$$D_3 = A_{12}B_{21}$$

$$D_4 = (A_{11} - A_{21})(B_{22} - B_{12})$$

$$D_5 = (A_{21} + A_{22})(B_{12} - B_{11})$$

$$D_6 = (A_{12} - A_{21} + A_{11} - A_{22})B_{22}$$

$$D_7 = A_{22}(B_{11} + B_{22} - B_{12} - B_{21})$$

$$C_{11} = D_2 + D_3$$

$$C_{12} = D_1 + D_2 + D_5 + D_6$$

$$C_{21} = D_1 + D_2 + D_4 - D_7$$

$$C_{22} = D_1 + D_2 + D_4 + D_5$$

- 7 πολ/μοι και 24 προσθέσεις πινάκων $\frac{n}{2} \times \frac{n}{2}$

- Χρόνος: $T(n) = 7T(n/2) + \Theta(n^2) \Rightarrow T(n) = \Theta(n^{\log_2 7})$

Υπολογισμός Δύναμης (Diffie-Hellman)

- Συμφωνία **Αλίκη** και **Βασίλη** σε κρυπτογραφικό **κλειδί**.
Εύα παρακολουθεί για να «κλέψει» το κλειδί.
- A, B συμφωνούν **δημόσια** σε **πρώτο** p και **ακέραιο** $q < p$.
E γνωρίζει p, q .
 - Εμπλεκόμενοι αριθμοί είναι **πολυψήφιοι** (π.χ. 512 ψηφία).
- A διαλέγει τυχαία $a < p$ και υπολογίζει $q_a = q^a \bmod p$
B διαλέγει τυχαία $b < p$ και υπολογίζει $q_b = q^b \bmod p$
A, B **ανταλλάσσουν** q_a, q_b και τα μαθαίνει E.
- A, B υπολογίζουν K (μόνοι τους). E **δεν ξέρει** K .
$$K = q_a^b \bmod p = (q^b \bmod p)^a \bmod p = q^{ab} \bmod p$$
- Για K, E **χρειάζεται** a, b (δεν μεταδόθηκαν).
Επίλυση **διακριτού λογαρίθμου** (πολύ δύσκολο).

Υπολογισμός Δύναμης

- Εφαρμογή υποθέτει **αποδοτικό** αλγόριθμο υπολογισμού $\text{exp}(x, n, p) = x^n \bmod p$, x, n, p **πολυψήφιοι** ακέραιοι.
 - Υπολογισμός δυνάμεων με τη σειρά (1, 2, 3, ...):
αν μήκος 512 bits, χρειάζεται περίπου 2^{512} **πολ/μους!!!**
 - Διαιρε-και-Βασίλειε (έστω n **δύναμη του 2**):
 - Υπολογίζουμε αναδρομικά $\text{exp}(x, n/2, p) = x^{n/2} \bmod p$
 - ... και $\text{exp}(x, n, p) = \text{exp}(x, n/2, p) \times \text{exp}(x, n/2, p)$
 - #πολλαπλασιασμών: $T(n) = T(n/2) + \Theta(1)$
 $\Rightarrow T(n) = O(\log n)$
 - p με μήκος 512 bits:
περίπου 2^{10} **πολ/μους**.
- ```
ExponRec(x, n, p)
if n = 1 then return(x mod p);
t ← ExponRec(x, ⌊n/2⌋, p);
t ← t2 mod p;
if n is odd then return(t × x mod p);
else return(t);
```

## Προϋποθέσεις Εφαρμογής

- Διάρθρωση **σημαντικά ευκολότερη** από επίλυση αρχικού.
- Σύνθεση **σημαντικά ευκολότερη** από επίλυση αρχικού.
- Υπο-στιγμιότυπα **σημαντικά μικρότερα** από αρχικό  
(π.χ. αρχικό μέγεθος  $n$ , υπο-στιγμ. μεγέθους  $n/c, c > 1$ ).
- **Ανεξάρτητα υπο-στιγμιότυπα** που λύνονται από ανεξάρτητες αναδρομικές κλήσεις.
  - Ίδια ή **επικαλυπτόμενα** υπο-στιγμιότυπα :  
σημαντική και **αδικοιολόγητη** αύξηση χρόνου εκτέλεσης.
  - **Επικαλυπτόμενα** υπο-στιγμιότυπα :  
**Δυναμικός Προγραμματισμός**

## (Αντι)παράδειγμα

- Υπολογισμός  $n$ -οστού όρου **ακολουθίας Fibonacci**.  
 $f_n = f_{n-1} + f_{n-2}, n \geq 2$   
 $f_0 = 0, f_1 = 1$ 

```
long fibRec(long n) {
 if (n <= 1) return(n);
 return(fibRec(n-1) + fibRec(n-2)); }
```
- Χρόνος εκτέλεσης:  
 $T(n) = \Theta(1) + T(n-1) + T(n-2), T(1) = \Theta(1)$
- Λύση:  $T(n) = \Theta(\varphi^n), \varphi = \frac{1+\sqrt{5}}{2} \approx 1.618$
- **Επικαλυπτόμενα** στιγμ.: **fib(n)** {  
**Εκθετικός** χρόνος!

```
int cur = 1, prev = 0;
for (i = 2; i <= n; i++) {
 cur = cur + prev;
 prev = cur - prev; }
```
- Αλγόριθμος **γραμμικού** χρόνου;
- **Καλύτερος** αλγόριθμος; 

```
return(cur); }
```

# Ακολουθία Fibonacci

- Ακολουθία Fibonacci:  $f_n = f_{n-1} + f_{n-2}, n \geq 2$   
 $f_0 = 0, f_1 = 1$
- Θεωρούμε πίνακα  $A = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$  και  $F_n = [f_n, f_{n-1}]$ 
  - Παρατηρούμε ότι  $A \times F_n = [f_n + f_{n-1}, f_n] = F_{n+1}$
  - Με επαγωγή αποδεικνύουμε ότι  $F_n = A^{n-1} \times F_1, F_1 = [1, 0]$
- Διάρει-και-Βασίλευε:
  - Υπολογισμός  $A^n$  σε χρόνο  $O(\log n)$  (όπως με αριθμούς).
  - Υπολογίζω αναδρομικά το  $A^{n/2}$  και  $A^n = A^{n/2} \times A^{n/2}$
  - Χρόνος:  $T(n) = T(n/2) + \Theta(1) \Rightarrow T(n) = \Theta(\log n)$