

Camp Προετοιμασίας 2012: Διαίρει-και-Βασίλευε

Δημήτρης Φωτάκης

Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών

Εθνικό Μετσόβιο Πολυτεχνείο



Διαίρει-και-Βασίλευε

- Γενική μέθοδος σχεδιασμού αλγορίθμων:
 - **Διαίρεση** σε ($\square 2$) 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: MergeSort, QuickSort.
 - Πολλαπλασιασμός αριθμών, πινάκων, FFT.
 - «**Εκλέπτυνση**»: Δυναμική αναζήτηση, QuickSelect, ύψωση σε δύναμη.
- (Εύκολη) ανάλυση με **αναδρομικές σχέσεις**.
 - Μη γραμμικές, συγκεκριμένης μορφής.

Πρόβλημα Ταξινόμησης

- Πρόβλημα Ταξινόμησης:
 - **Είσοδος** : ακολουθία n αριθμών (a_1, a_2, \dots, a_n) .
 - **Έξοδος** : αναδιάταξη $(a'_1, a'_2, \dots, a'_n)$ με αριθμούς σε **αύξουσα** σειρά $(\forall a'_i \leq a'_{i+1})$.
- **Στατική** συλλογή δεδομένων (όχι εισαγωγή και διαγραφή).
- **Θεμελιώδες** αλγοριθμικό πρόβλημα.
 - Πολλές **εφαρμογές** (περ. 25% υπολογιστικού χρόνου).
 - Ταχύτατη **αναζήτηση** σε ταξινομημένα δεδομένα.
 - Σημαντικές αλγοριθμικές **ιδέες και τεχνικές**.

Μέθοδοι Ταξινόμησης

- **Αντιμετάθεση** κάθε ζεύγους **στοιχείων εκτός διάταξης** (bubble sort).
- **Εισαγωγή** στοιχείου σε **κατάλληλη θέση** ταξινομημένου πίνακα (insertion sort).
- **Επιλογή** **μεγαλύτερου** στοιχείου και τοποθέτηση **στο τέλος** (selection sort, heapsort).
- **Συγχώνευση** ταξινομημένων **πινάκων** : **Διαίρεση στη μέση**, ταξινόμηση, συγχώνευση (mergesort).
- **Διαίρεση** σε μικρότερα και μεγαλύτερα **από** **στοιχείο-διαχωρισμού** και ταξινόμηση (quicksort).

Συγκριτικοί Αλγόριθμοι

- Ταξινόμηση **αποκλειστικά** με συγκρίσεις και αντιμεταθέσεις στοιχείων.
 - Δεν εκμεταλλεύονται τύπο δεδομένων (π.χ. αριθμούς, συμβολοσειρές) : γενική εφαρμογή.
 - **Ανάλυση** : πλήθος συγκρίσεων.
πλήθος αντιμεταθέσεων.
 - Κάτω φράγμα #συγκρίσεων : $\Omega(n \log n)$.
- Ταξινόμηση (με αλγόριθμους που δεν είναι συγκριτικοί) σε χρόνο $\Theta(n+k)$ για n φυσικούς αριθμούς στο $\{1, \dots, k\}$ (π.χ. bin sort, radix sort).

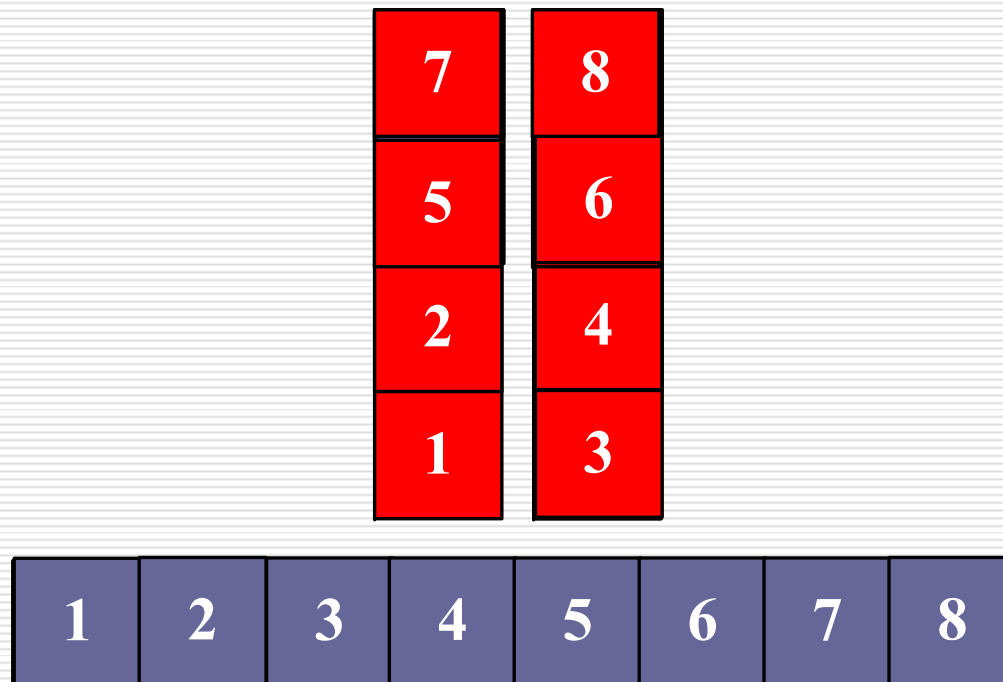
MergeSort

- 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); }
```

Συγχώνευση

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



Συγχώνευση

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

MergeSort: 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**) □
Σωστά ταξινομημένος πίνακας $A[]$.

```
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); }
```

Χρόνος Εκτέλεσης

- Χρόνος εκτέλεσης **merge** (για n στοιχεία) : $\Theta(n)$ (γραμμικός)
 - $\Theta(1)$ λειτουργίες για **κάθε στοιχείο**.
- Χρόνος εκτέλεσης αλγόριθμων «**διαίρει-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.

$$T(n) = 2 T(n / 2) + \Theta(n), T(1) = \Theta(1)$$

- Χρόνος εκτέλεσης **MergeSort**: **$T(n) = ???$**

Δέντρο Αναδρομής

$$\begin{aligned} T(n) &= 2 T(n / 2) + \Theta(n), \\ T(1) &= \Theta(1) \end{aligned}$$

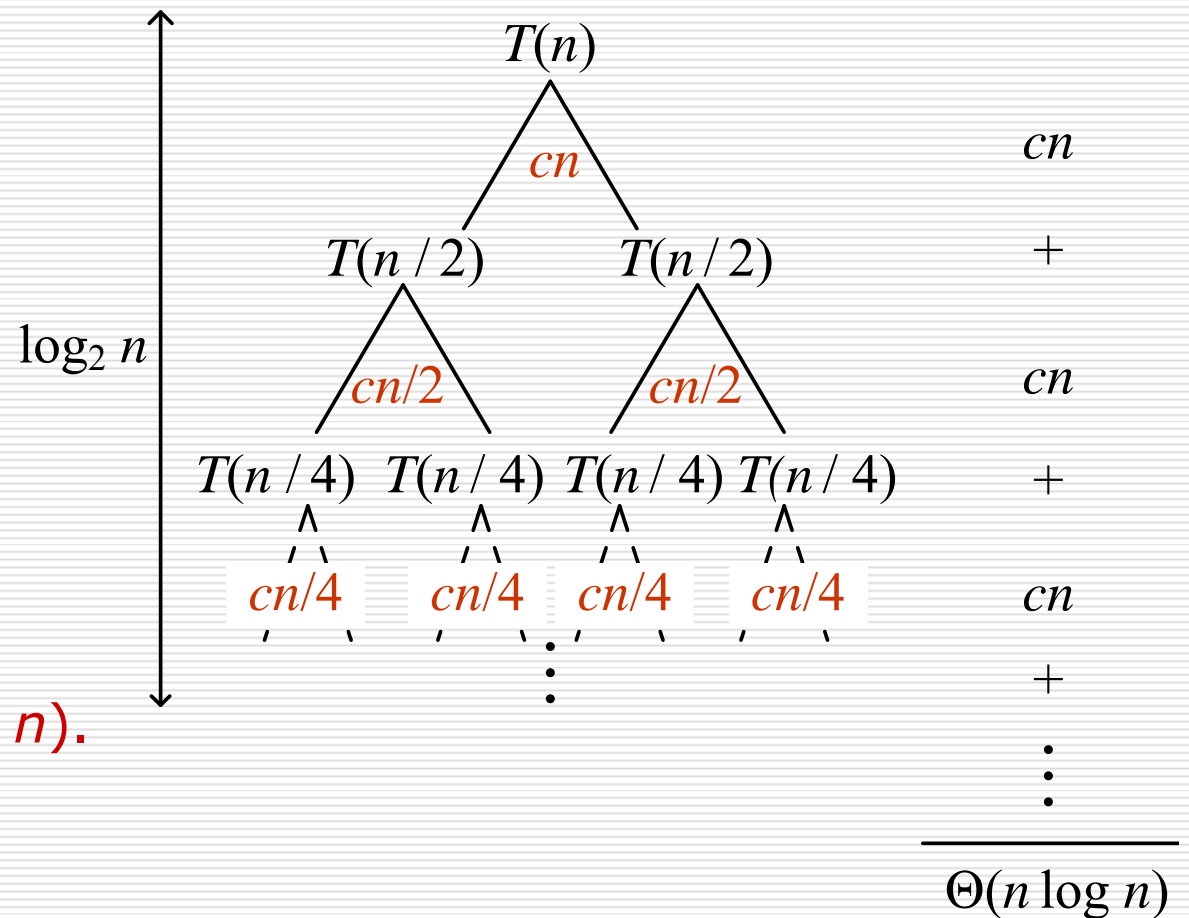
Δέντρο αναδρομής :

Ύψος : $\Theta(\log n)$

#κορυφών : $\Theta(n)$

Χρόνος / επίπεδο : $\Theta(n)$

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



Master Theorem

- Ανάλυση χρόνου εκτέλεσης αλγορίθμων «διαίρει-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 $\gamma_1 + \gamma_2 < 1 - \varepsilon$, 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

- $T(n) = 9 T(n/3) + n.$ $T(n) = \Theta(n^2)$ (περ. 1)
- $T(n) = T(2n/3) + 1.$ $T(n) = \Theta(\log n)$ (περ. 2)
- $T(n) = 3 T(n/4) + n \log n.$ $T(n) = \Theta(n \log n)$ (περ. 3)
- $T(n) = 2 T(n/2) + n.$ $T(n) = \Theta(n \log n)$ (περ. 2)
- $T(n) = 2 T(n/2) + n \log n.$
 - Δεν εμπίπτει! 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;
- Διαίρει-και-Βασίλευε:
 - Διαίρεση: $x = 2^{n/2}x_h + x_l$, $y = 2^{n/2}y_h + y_l$
$$x \times y = 2^n \overbrace{x_h y_h}^{z_h} + 2^{n/2} \overbrace{(x_h y_l + x_l y_h)}^{z_m} + \overbrace{x_l y_l}^{z_l} = 2^n z_h + 2^{n/2} z_m + z_l$$
 - 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: $2576 \square 7935 = 20440560$

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

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

$$\begin{aligned} z_m &= (25 + 76)(79 + 35) - 1975 - 2660 = \\ &= 101 \times 114 - 1975 - 2660 = 11514 - 1975 - 2660 = 6879 \end{aligned}$$

$$x \times y = 1975 \cdot 10^4 + 6879 \cdot 10^2 + 2660 = 20404560$$

Πολλαπλασιασμός Πινάκων

- Υπολογισμός **γινομένου** $C = A \square B$.
 A, B τετραγωνικοί πίνακες $n \square n$.
- 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).

- Διαίρει-και-Βασίλευε:

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

- 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 **Αλίκης** και **Βασίλη** σε κρυπτογραφικό **κλειδί**.
Εύα παρακολουθεί για να «κλέψει» το κλειδί.
- Α, Β συμφωνούν **δημόσια** σε **πρώτο** p και **ακέραιο** $q < p$.
Ε γνωρίζει p, q .
 - **Εμπλεκόμενοι αριθμοί** είναι **πολυψήφιοι** (π.χ. 512 ψηφία).
- Α διαλέγει τυχαία $a < p$ και υπολογίζει $q_a = q^a \bmod p$
Β διαλέγει τυχαία $b < p$ και υπολογίζει $q_b = q^b \bmod p$
Α, Β **ανταλλάσσουν** q_a, q_b και τα μαθαίνει Ε.
- Α, Β υπολογίζουν **K (μόνοι τους)**. **Ε δεν ξέρει K** .
$$K = q_b^a \bmod p = (q^b \bmod p)^a \bmod p = q^{ab} \bmod p$$
- Για K , **Ε 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!!!
 - Διαίρει-και-Βασίλευε (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( $x \bmod p$ );  
 $t \leftarrow \text{ExponRec}(x, \lfloor n/2 \rfloor, p)$ ;  
 $t \leftarrow t^2 \bmod p$ ;  
if  $n$  is odd then return( $t \times x \bmod p$ );  
else return( $t$ );
```

Προϋποθέσεις Εφαρμογής

- Διαίρεση **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.

$$f_n = f_{n-1} + f_{n-2}, n \geq 2 \quad \text{long fibRec(long n) \{}$$
$$f_0 = 0, f_1 = 1 \quad \text{if (n <= 1) return(n);}$$
$$\text{return(fibRec(n-1) + fibRec(n-2)); \}}$$

- Χρόνος εκτέλεσης:

$$T(n) = \Theta(1) + T(n-1) + T(n-2), \quad T(1) = \Theta(1)$$

- Λύση: $T(n) = \Theta(\varphi^n)$, $\varphi = \frac{1+\sqrt{5}}{2} \approx 1.618$

- Επικαλυπτόμενα στιγμ.:

Εκθετικός χρόνος!

- Αλγόριθμος γραμμικού χρόνου;

- Καλύτερος αλγόριθμος;

```
long fib(long n) {  
    long cur = 1, prev = 0;  
    for (i = 2; i <= n; i++) {  
        cur = cur + prev;  
        prev = cur - prev;  
    }  
    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 $F_n = A^{n-1} \times F_1, F_1 = [1, 0]$
- Διαιρεί-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: $T(n) = T(n/2) + \Theta(1) \Rightarrow T(n) = \Theta(\log n)$

Πρόβλημα Επιλογής

- Πίνακας $A[]$ με n στοιχεία (όχι ταξινομημένος).
Αριθμός k , $1 \leq k \leq n$.
- Υπολογισμός του k -οστού μικρότερου στοιχείου
(στοιχείο θέσης $A[k]$ αν A ταξινομημένος).
 - $k = 1$: ελάχιστο. $k = n$: μέγιστο.
 $k = n / 2$: ενδιάμεσο (median).

5	3	2	6	4	1	3	7
---	---	---	---	---	---	---	---

Ελάχιστο : **1**

Μέγιστο : **7**

Ενδιάμεσο : **3**

Εφαρμογές

- Υπολογισμός **στατιστικού ενδιάμεσου** (median).
 - Χρήσιμες πληροφορίες για κατανομή.
 - Ανήκει η Ελλάδα στο φτωχότερο 25% των χωρών ΕΕ;
 - Ανήκει κάποιος φοιτητής στο καλύτερο 10% του έτους του;
- **Ισομερής διαίρεση** (partition) πίνακα σε **ομάδες «ταξινομημένες»** μεταξύ τους.
- Ενδιαφέρον **αλγοριθμικό πρόβλημα!**

Μέγιστο / Ελάχιστο

- Μέγιστο (ελάχιστο) εύκολα σε χρόνο $\Theta(n)$, με $n - 1$ συγκρίσεις μεταξύ στοιχείων.

```
int maximum(int A[], int n) {  
    int max = A[0];  
    for (i = 1; i < n; i++)  
        if (A[i] > max) max = A[i];  
    return(max);  
}
```

- Μέγιστο και ελάχιστο με $3 \lfloor n/2 \rfloor$ συγκρίσεις ! Πώς;

Κάτω Φράγμα για Μέγιστο

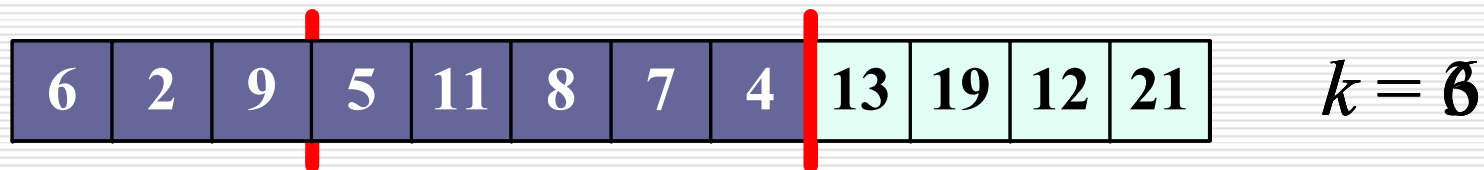
- Κάθε ντετερμινιστικός συγκριτικός αλγόριθμος χρειάζεται □
 $n - 1$ συγκρίσεις για μέγιστο (ελάχιστο).
 - «Πρωτάθλημα» μεταξύ στοιχείων.
 - Σύγκριση στοιχείων : αγώνας όπου κερδίζει μεγαλύτερο.
 - Κάθε «αήττητο» στοιχείο είναι υποψήφιο μέγιστο.
 - Για μοναδικό μέγιστο, πρέπει τα υπόλοιπα να «ηττηθούν».
 - Κάθε αγώνας δίνει ένα «ηττημένο» στοιχείο
 - □ $n - 1$ αγώνες / συγκρίσεις για μοναδικό μέγιστο.

Επιλογή

- Σε χρόνο $O(n \log n)$ με ταξινόμηση.
- Μέγιστο ($k = 1$), ελάχιστο ($k = n$) : χρόνος $\Theta(n)$.
- Άλλες τιμές k : χρόνος $O(n \log n)$ ή $O(n)$;
- Επιλογή σε γραμμικό χρόνο με διαίρει-και-βασίλευε βασισμένη σε διαχωρισμό της quicksort !

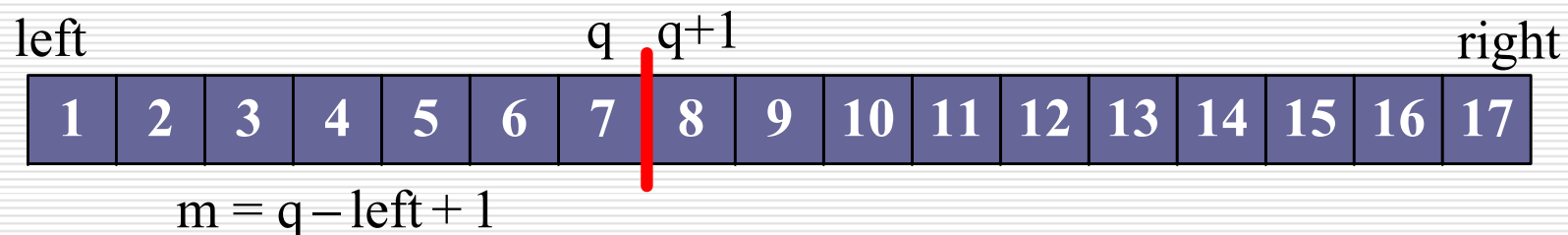
Πιθανοτική Quickselect

- Έστω υπο-πίνακας $A[l...r]$ και αναζητούμε k -οστό στοιχείο.
- **Τυχαίο** στοιχείο διαχωρισμού (pivot).
- **Αναδιάταξη** και **διαίρεση** εισόδου σε δύο υπο-ακολουθίες:
 - Στοιχεία αριστερής $[l...q]$ υπο-ακολ. □ στοιχείο διαχωρισμού.
 - Στοιχεία δεξιάς $[q+1...r]$ υπο-ακολ. □ στοιχείο διαχωρισμού.
- Αν $k \leq q-l+1$, αναδρομική λύση $(A[l...q], k)$
Αν $k > q-l+1$, αναδρομική λύση $(A[q+1...r], k-(q-l+1))$



Ορθότητα Quickselect

- Τερματισμός : μέγεθος υπο-ακολουθιών □ $n - 1$.
- Επαγωγικά υποθέτω ότι $1 \leq k \leq \text{right} - \text{left} + 1$.
 - Πλήθος στοιχείων στα αριστερά: $m = q - \text{left} + 1$.
 - Αν $k \leq m$, δεξιά στοιχεία «αποκλείονται».
 - Αν $m < k$, αριστερά στοιχεία «αποκλείονται» και k μειώνεται αντίστοιχα ($k' = k - m$).



Πιθανοτική Quickselect

```
int RQuickSelect(int A[], int left, int right, int k)
{
    if (left == right) return(A[left]); // 1 στοιχείο
    pivot = random(left, right); // τυχαίο pivot
    swap(A[left], A[pivot]);
    q = partition(A, left, right); // διαίρεση
    nel = q - left + 1; // #στοιχείων στο αριστερό μέρος
    if (k <= nel) return(RQuickSelect(A, left, q, k));
    else return(RQuickSelect(A, q+1, right, k - nel));
}
```

Χρόνος Εκτέλεσης (χ.π.)

- Χρόνος εκτελ. **αναδρομικών αλγ.** με διατύπωση και λύση **αναδρομικής εξίσωσης** λειτουργίας.
- **$T(n)$** : χρόνος (χ.π.) για **επιλογή από n στοιχεία**.
- Χρόνος εκτέλεσης **partition**(n στοιχεία) : $\Theta(n)$
- Χειρότερη περίπτωση : **ένα στοιχείο «αποκλείεται»** σε κάθε διαίρεση!

$$T(n) = \Theta(n) + T(n - 1), \quad T(1) = \Theta(1)$$

- **Πιθανοτικός αλγ.:** χειρότερη περίπτωση έχει **εξαιρετικά μικρή πιθανότητα** να συμβεί (για κάθε είσοδο) !
$$T(n) = \Theta(n) + \Theta(n - 1) + \Theta(n - 2) + \dots + \Theta(1) = \Theta(n^2)$$

Χρόνος Εκτέλεσης (μ.π.)

- **Καλή περίπτωση** : διαίρεση $(n/4, 3n/4)$ ή καλύτερη.
 - Τουλάχιστον $n/4$ στοιχεία «αποκλείονται».
- Πιθανότητα «καλής περίπτωσης» □ **1/2** !
 - Κατά «μέσο όρο», μία «κακή διαίρεση» πριν από «καλή διαίρεση» που μειώνει στοιχεία από n σε □ $3n/4$.

- Λύση αναδρομής: $S(n) = \Theta(n) + S(3n/4)$
 - Γεωμετρική σειρά $S(n) = \Theta(n)$

$$S(n) \leq cn + \frac{3}{4}cn + \left(\frac{3}{4}\right)^2 cn + \left(\frac{3}{4}\right)^3 cn + \dots + c = \Theta(n)$$

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----

Χρόνος Εκτέλεσης (μ.π.)

□ Τυχαίο στοιχείο σαν στοιχείο χωρισμού (pivot).

□ Για κάθε $i \in [n-1]$

πιθανότητα διαίρεσης $(i, n-i) = \frac{1}{n-1}$

$$S(n) = \Theta(n) + \frac{1}{n-1} \sum_{i=1}^{n-1} S(\max\{i, n-i\})$$

$$= \Theta(n) + \frac{2}{n-1} \sum_{i=n/2}^{n-1} S(i)$$

□ Λύση αναδρομής :

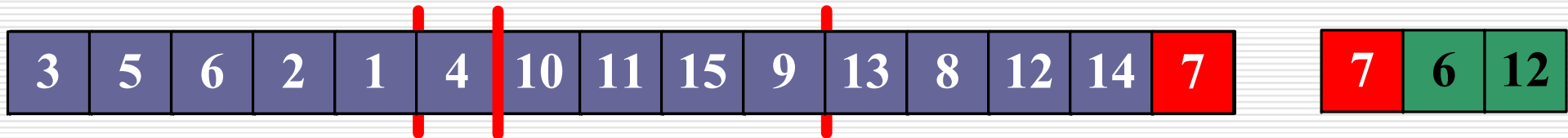
$$S(n) = \Theta(n)$$

Ντετερμινιστική Επιλογή

- «Καλή διαίρεση» ντετερμινιστικά:
 - Χρήση ρινότ **κοντά** στο ενδιάμεσο: πρόβλημα επιλογής!
 - Φαύλος κύκλος : γρήγορη επιλογή → καλή διαίρεση → γρήγορη επιλογή.
- Προσεγγιστική επιλογή : όχι «ενδιάμεσο» αλλά «κοντά στο ενδιάμεσο» για ρινότ.
 - Επιλογή **κατάλληλου** δείγματος (π.χ. $n / 5$ στοιχεία).
 - Ενδιάμεσο δείγματος είναι «κοντά στο ενδιάμεσο» για σύνολο στοιχείων.
 - Αναδρομικά ενδιάμεσο στοιχείο του δείγματος.
 - Ενδιάμεσο δείγματος για ρινότ εγγυάται «καλή διαίρεση».

Ντετερμινιστική Επιλογή

- **Δείγμα:** Χωρίζουμε στοιχεία σε 5άδες.
Βρίσκουμε **ενδιάμεσο** κάθε 5άδας: $n / 5$ στοιχεία.
 - Χρόνος : $\Theta(n)$.
- Αναδρομικά, **ενδιάμεσο** στοιχείο δείγματος.
 - Χρόνος : $T(n / 5)$
- **Διαίρεση** με **ενδιάμεσο** δείγματος σαν pivot.
 - Χρόνος : $\Theta(n)$.
 - Μεγαλύτερος υποπίνακας έχει
- Αναδρομική επιλογή: **χρόνος** $T(7n/10) \leq 7n/10$ στοιχεία.



Ντετερμινιστική Επιλογή

- Χρόνος χειρότερης περίπτωσης:

$$T(n) \leq \Theta(n) + T(n/5) + T(7n/10), \quad T(1) = \Theta(1)$$

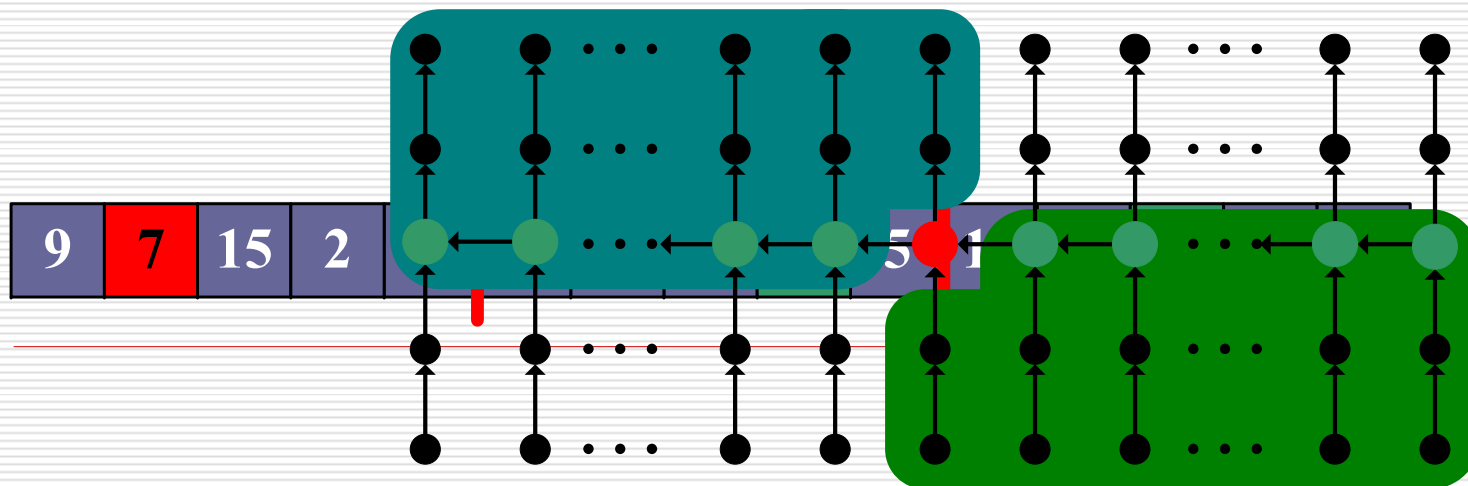
- Λύση αναδρομής : $T(n) = \Theta(n)$
- Ντετερμινιστική επιλογή σε γραμμικό χρόνο!

Ενδιάμεσο Δείγματος

- Διάρθρωση με ενδιάμεσο δείγματος σαν pivot.
 - Μεγαλύτερος υποπίνακας □ $7n / 10$ στοιχεία.
 - Μικρότερος υποπίνακας □ $3n / 10$ στοιχεία.
- Ταξινομούμε 5αδες και βάζουμε σε αύξουσα σειρά των ενδιάμεσων στοιχείων τους (δείγματος).
- Ενδιάμεσος δείγματος στη $(n / 10)$ -οστή στήλη.
- Ενδιάμεσος δείγματος

$$\geq 3 \times n/10 \text{ στοιχεία.}$$

$$\leq 3 \times n/10 \text{ στοιχεία.}$$



4	1	3
5	2	8
6	7	12
10	9	13
11	15	14