# Quicksort

**Δημήτρης Φωτάκης**

Τμήμα Μηχανικών Πληροφοριακών και Επικοινωνιακών Συστημάτων

Πανεπιστήμιο Αιγαίου

## Διαίρει-και-Βασίλευε

- Γενική μέθοδος σχεδιασμού αλγορίθμων:
  - **Διαίρεση** σε ($\geq 2$) υπο-προβλήματα (σημαντικά) μικρότερου μεγέθους.
  - **Ανεξάρτητη** (αναδρομική) επίλυση υπο-προβλημάτων (για μικρά υπο-προβλήματα εφαρμόζουμε στοιχειώδεις αλγορίθμους).
  - **Σύνθεση** λύσης αρχικού προβλήματος από λύσεις υπο-προβλημάτων.
- Ισχυρή μέθοδος, πολλές σημαντικές εφαρμογές !
- (Εύκολη) ανάλυση με **αναδρομικές εξισώσεις.**
- **Ταξινόμηση :** merge-sort, **quicksort**.



## Quicksort [Hoare, 62]

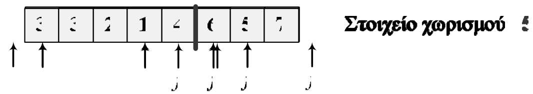
- Στοιχείο χωρισμού (pivot), π.χ. πρώτο, τυχαίο, …
- Αναδιάταξη και διαίρεση εισόδου σε δύο υπο-ακολουθίες:
  - Στοιχεία αριστερής υπο-ακολ. < στοιχείο χωρισμού.
  - Στοιχεία δεξιάς υπο-ακολ. > στοιχείο χωρισμού.
- Ταξινόμηση υπο-ακολουθιών αναδρομικά.
- Ακολουθία ταξινομημένη – όχι σύνθεση!

```
quickSort(int A[], int left, int right) {
  if (left >= right) return; // At most 1 element
  q = partition(A, left, right);
  quickSort(A, left, q);
  quickSort(A, q+1, right);   }
```



## Διαίρεση

- Στοιχείο χωρισμού (pivot), π.χ. **πρώτο**, τυχαίο, …
- Διαίρεση σε ένα πέρασμα :
  - Σάρωση από αριστερά (με δείκτη *i*) μέχρι A[*i*] $\geq$ pivot.
  - Σάρωση από δεξιά (με δείκτη *j*) μέχρι A[*j*] $\leq$ pivot.
  - Δεν έχουν εξεταστεί όλα τα στοιχεία ($i < j$): αντιμετάθεση(A[*i*] , A[*j*]) και συνέχεια.
  - Έχουν εξεταστεί όλα: επιστροφή ορίου χωρισμού (δείκτη *j*).



# Ανάλυση Διαίρεσης

- Ορθότητα **partition** :
  - Διατηρεί και επεκτείνει αριστερή περιοχή με στοιχεία ≤ pivot και δεξιά περιοχή με στοιχεία ≥ pivot.
  - A[*i*] ≥ pivot : επέκταση αριστερής περιοχής σταματά.
  - A[*j*] ≤ pivot : επέκταση δεξιάς περιοχής σταματά.
  - Ξένες περιοχές : αντιμετάθεση στοιχείων και συνέχεια.
  - Επικάλυψη : ολοκλήρωση διαίρεσης.
  - Τελικά τα στοιχεία αριστερά ≤ pivot και τα στοιχεία δεξιά ≥ pivot, **όπως απαιτείται.**
- Κάθε περιοχή > 1 στοιχείο. **Quicksort τερματίζει.** (1 ≤ σημείο χωρισμού ≤ *n* – 1)
  - Απαραίτητα: *i* και *j* σταματούν στο pivot.

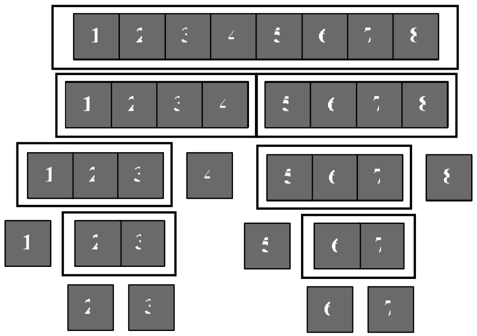


# Ανάλυση Διαίρεσης

- Χρόνος εκτέλεσης **partition** :
  - Κάθε στοιχείο συγκρίνεται με pivot μία φορά (εκτός από στοιχεία εκατέρωθεν σημείου χωρισμού).
  - Τελικά *i* και *j* «δείχνουν» είτε γειτονικές είτε ίδια θέση γιατί όπου πέρασε το *i* δεν συνεχίζει *j*.
  - Χρόνος εκτέλεσης **partition για n στοιχεία = Θ(n).**
- Μετά τη διαίρεση, κανένα στοιχείο δεν αλλάζει «πλευρά» (δηλ. αριστερά μένουν αριστερά, δεξιά μένουν δεξιά).
- Υπάρχουν πολλές άλλες μορφές διαίρεσης, π.χ. pivot παίρνει τελική του θέση στον πίνακα, διαίρεση στα τρία, …



# Παράδειγμα Quicksort





# Ορθότητα Quicksort

- Συνέπεια ορθότητας **partition** :
  - Τερματισμός : μέγεθος υπο-ακολουθιών ≤ *n* – 1.
  - Ταξινόμηση :
    - Αριστερά στοιχεία ≤ pivot ≤ δεξιά στοιχεία.
    - Επαγωγικά, αριστερή περιοχή και δεξιά περιοχή ταξινομημένες.
    - Συνολικά, πίνακας ταξινομημένος.

## Χρόνος Εκτέλεσης (χ.π.)

- Χρόνος εκτελ. αναδρομικών αλγ. με διατύπωση και λύση αναδρομικής εξίσωσης λειτουργίας.
- Χρόνος εκτέλεσης **partition**(*n* στοιχεία) : Θ(*n*)
- ***T*(*n*)** : χρόνος (χ.π.) για ταξινόμηση *n* στοιχείων.
  - **Θ(*n*)** : αναδιάταξη και διαίρεση εισόδου.
  - ***T*(*k*)** : ταξινόμηση αριστερού τμήματος (*k* στοιχεία).
  - ***T*(*n* – *k*)** : ταξινόμηση δεξιού τμήματος (*n* – *k* στοιχεία).

$$T(n) = \Theta(n) + \max_{1 \leq k \leq n-1} \{T(k) + T(n-k)\}\,, \quad T(1) = \Theta(1)$$



## Χρόνος Εκτέλεσης (χ.π.)

$$T(n) = \Theta(n) + \max_{1 \leq k \leq n-1} \{T(k) + T(n-k)\}\,, \quad T(1) = \Theta(1)$$

- Χειρότερη περίπτωση : *k* = 1 ή *k* = *n* – 1 (σε κάθε κλήση).
  - Ουσιαστικά δεν γίνεται διαίρεση (μόνο αναδιάταξη) !
  - Partition «βοηθάει ελάχιστα» τον αλγόριθμο.

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

- Στιγμιότυπα που quicksort χρειάζεται χρόνο Ω($n^2$);



## Χρόνος Εκτέλεσης

- **Καλύτερη περίπτωση** : *k* = *n* / 2 (σε κάθε κλήση).
  - Ουσιαστικά τέλεια διαίρεση !
  - Partition «βοηθάει τα μέγιστα» !

  $T(n) = 2\,T(n/2) + \Theta(n) \Rightarrow T(n) = \Theta(n \log n)$
- Αν $\min\{k, n-k\} \geq 3n/4$ (περίπου ίδιο μεγέθος)

  $T(n) = \Theta(n) + T(n/4) + T(3n/4) \;\Rightarrow T(n) = \Theta(n \log n)$
- Χειρότερη και καλύτερη περίπτωση σπάνιες !
- Αν τυχαίο στοιχείο pivot, πιθανότητα διαίρεσης (*n* / 4, 3*n* / 4) ή καλύτερης ≥ **1/2** !

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



## Πιθανοτική Quicksort

- Τυχαίο στοιχείο σαν στοιχείο χωρισμού (pivot).
- Για κάθε $k \in [n-1]$, πιθανότητα διαίρεσης (*k*, *n* – *k*) = $\dfrac{1}{n-1}$

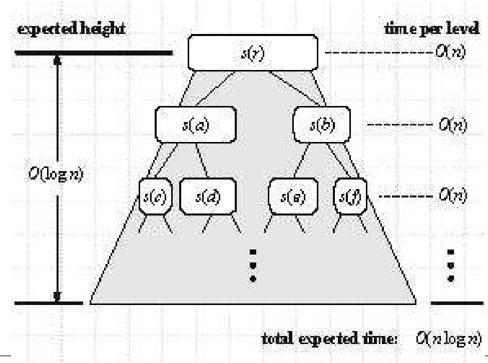
# Χρόνος Εκτέλεσης (μ.π.)

$$S(n) = \Theta(n) + \frac{1}{n-1}\sum_{k=1}^{n-1}[S(k) + S(n-k)]$$
$$= \Theta(n) + \frac{2}{n-1}\sum_{k=1}^{n-1} S(k)$$

- Λύση αναδρομής : $S(n) = \Theta(n \log n)$
  Αυτός ο χρόνος εκτέλεσης με μεγάλη πιθανότητα !
- Πιθανότητα διαίρεσης ($n$ / 4, 3$n$ / 4) ή καλύτερης ≥ **1/2** !
  - Κατά «μέσο όρο», κάθε 2 επίπεδα στο δέντρο της αναδρομής, έχουμε «επιτυχημένη» διαίρεση.
  - Σε κάθε επίπεδο, συνολικός χρόνος διαίρεσης Θ($n$).
  - Θ($n$ log $n$) από «επιτυχημένες» διαιρέσεις +
    Θ($n$ log $n$) από «αποτυχημένες» διαιρέσεις.



# Δέντρο Αναδρομής






# Χρόνος Εκτέλεσης (μ.π.)

- Πιθανότητα «αποτυχημένες» διαιρέσεις > $c$ log $n$
  είναι εξαιρετικά μικρή !
  - Χρόνος εκτέλεσης Θ($n$ log $n$) με **μεγάλη πιθανότητα** !
- Μέση περίπτωση δεν εξαρτάται από είσοδο !
  Αφορά στη συμπεριφορά του αλγόριθμου.
- Εξαιρετικά μικρή πιθανότητα χειρότερης περίπτωσης.
  - Ανάλυση χειρότερης περίπτωσης δεν έχει νόημα !



# Σύνοψη

- Γρήγορη ταξινόμηση (quicksort):
  - Πιθανοτικός αλγόριθμος.
  - Χρόνος χειρότερης περ. : Θ( $n^2$ )
  - Χρόνος μέσης περίπτωσης : Θ($n$ log $n$)
  - Χώρος : σχεδόν in-place.
  - Αναδρομή καθυστερεί και απαιτεί μνήμη.
    - Επαναληπτική υλοποίηση.
  - Εύκολη και γρήγορη υλοποίηση.
  - Γρηγορότερος αλγόριθμος στην πράξη (για n > 30).

## Σύνοψη

| **Αλγόριθμος** | Καλύτερη | Μέση | Χειρότερη | Χώρος |
|---|---|---|---|---|
| BubbleS | $\Omega(n)$ | $O(n^2)$ | $O(n^2)$ | $O(1)$ |
| InsertionS | $\Omega(n)$ | $O(n^2)$ | $O(n^2)$ | $O(1)$ |
| SelectionS | $\Omega(n)$ | $O(n^2)$ | $O(n^2)$ | $O(1)$ |
| HeapS | $\Omega(n \log n)$ | $O(n \log n)$ | $O(n \log n)$ | $O(1)$ |
| MergeS | $\Omega(n \log n)$ | $O(n \log n)$ | $O(n \log n)$ | $O(n)$ |
| QuickS | $\Omega(n \log n)$ | $O(n \log n)$ | $O(n^2)$ | ? |



## Πιθανοτικοί Αλγόριθμοι

- Ντετερμινιστικοί αλγόριθμοι:
  - Προκαθορισμένη συμπεριφορά για κάθε είσοδο.
  - Υπάρχει χειρότερη περίπτωση και μπορεί να συμβεί.
- Πιθανοτικοί αλγόριθμοι:
  - Συμπεριφορά από είσοδο και τυχαίες επιλογές.
  - Χρήση τυχαιότητας ώστε χειρότερη περίπτωση να συμβαίνει με πολύ μικρή πιθανότητα.
  - Ποια είναι η χειρότερη περ. για πιθανοτική quicksort;
  - Χρόνος (απόδοση) κατά μέση τιμή. Ορθότητα με μεγάλη πιθανότητα.
  - Las-Vegas: αποτέλεσμα σωστό, χρόνος τυχαία μετ/τη.
  - Monte-Carlo: χρόνος προκαθορισμένος, μπορεί λάθος αποτέλεσμα (αλλά με πολύ μικρή πιθανότητα).