Ταξινόμηση

Παύλος Εφραιμίδης

pefraimi <at> ee.duth.gr

Το πρόβλημα της ταξινόμησης

Ταξινόμηση

- Δίνεται πολυ-σύνολο Σ με στοιχεία από κάποιο σύμπαν U (πχ. U = το σύνολο των ακεραίων αριθμών).
- **Ταξινόμηση** του Σ είναι η επιβολή μιας διάταξης στα στοιχεία του συνόλου Σ.
- Ένας **αλγόριθμος ταξινόμησης**
  - δέχεται ως είσοδο ένα πολυ-σύνολο Σ και με βάση μια δεδομένη σχέση διάταξης δίνει στην έξοδο τα στοιχεία του Σ διατεταγμένα.

Αλγόριθμοι ταξινόμησης

- Φυσαλιδωτή ταξινόμηση (Bubble sort)
- Ενθετική ταξινόμηση (Insertion sort)
- Ταχυταξινόμηση (Quicksort)
- Συγχωνευτική ταξινόμηση (Merge sort)

Φυσαλιδωτή Ταξινόμηση (Bubble sort)

- Ένας απλός και ιδιαίτερα διαδεδομένος αλγόριθμος ταξινόμησης
- Βασική αρχή του αλγορίθμου:
  - Σαρώνει τη λίστα των αριθμών εναλλάσσοντας παρακείμενα στοιχεία που είναι εκτός διάταξης
  - Μετά από ένα αριθμό σαρώσεων η λίστα θα ταξινομηθεί

Ψευδοκώδικας

```
1.  for (int i = 0; i < A.length; i++ ) {
2.    for (int j = A.length - 1; j > i; j-- ) {
3.      if (A[j] < A[j-1]) {
4.        "αντιαλλαγή (swap)" A[j-1], A[j]
5.      }
6.    }
7.  }
```

Πολυπλοκότητα

- Πολυπλοκότητα χειρότερης περίπτωσης για πίνακα με n στοιχεία
- Στο 1ο πέρασμα θα χρειαστούν n-1 συγκρίσεις
- Στο 2ο πέρασμα n-2 συγκρίσεις
- …
- Στο n-1 πέρασμα, 1 σύγκριση
- **Σύνολο:** $(n-1) + (n-2) + \ldots + 1 = \frac{1}{2}n(n-1) = O(n^2)$

Ταξινόμηση Σέικερ (Shaker sort)

- Παραλλαγή της ταξινόμησης φυσαλίδων
- Βασική αρχή του αλγορίθμου:
  - Η **shaker sort** είναι μια απλά παραλλαγή της ταξινόμησης φυσαλίδων (bubble sort) στην οποία η σάρωση του πίνακα γίνεται εναλλάξ από την αρχή προς το τέλος και από το τέλος προς την αρχή
- Ποια η πολυπλοκότητα χειρότερης περίπτωσης;

Ενθετική ταξινόμηση (Insertion sort)

- Βασική αρχή:
  - Επιλέγει ένα-ένα τα στοιχεία της μη-ταξινομημένης ακολουθίας και τα εισαγάγει σε μια ταξινομημένη λίστα

Ψευδοκώδικας

```
1. for (int j = 2; j < A.length; j++ ) {
2.   κλειδί = A[j];
3.   // Ενθέτουμε το A[j] στην ταξινομημένη
     // ακολουθία A[1..j-1]
4.   i = j - 1;
5.   while (i > 0 && A[i] > κλειδί) {
6.     A[i+1]=A[i];
7.     i--;
8.   }
9.   A[i+1] = κλειδί
10.}
```

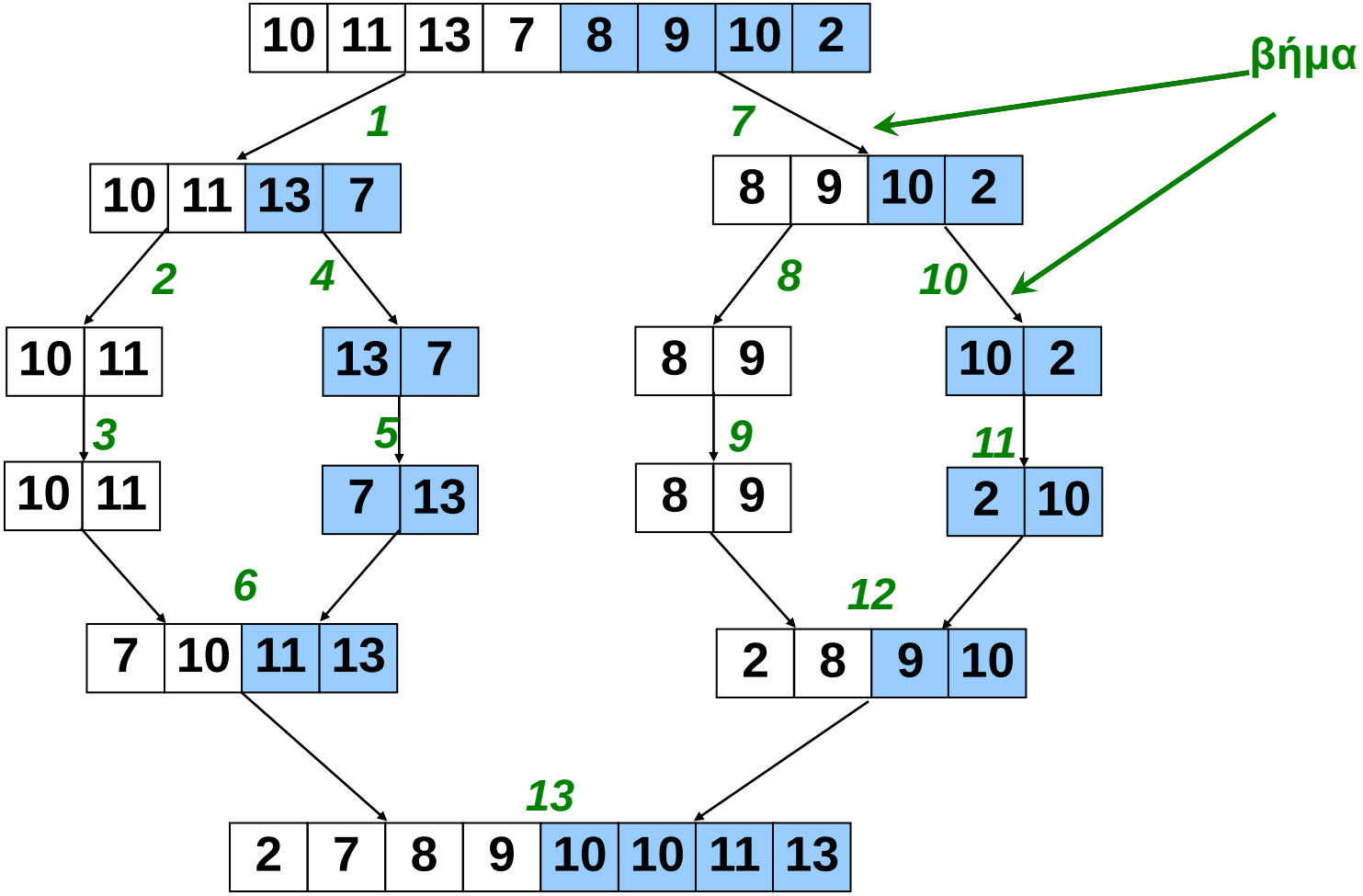
Πολυπλοκότητα

- Η πολυπλοκότητα χειρότερης περίπτωσης είναι $O(n^2)$

Συγχωνευτική ταξινόμηση (Merge sort)

- **Βασική αρχή:** Για να ταξινομήσουμε τα στοιχεία ενός μιας ακολουθίας
  - **Διαίρει:** *Χωρίζουμε την ακολουθία σε δύο ίσου (ή σχεδόν ίσου) μεγέθους τμήματα*
  - **Κυρίευε:** *Εφαρμόζουμε αναδρομικά τη συγχωνευτική ταξινόμηση σε κάθε ένα από τα τμήματα*
  - **Συνδύασε:** *Όταν τα δύο τμήματα έχουν ταξινομηθεί, τα συγχωνεύουμε (merge) σε μία ενιαία ταξινομημένη ακολουθία*

Παράδειγμα MergeSort



Ψευδοκώδικας

```
1. mergeSort (A, p, r) {
2.  if (p < r) {
3.   q = (p+r)/2
4.   mergeSort(A, p, q);
5.   mergeSort(A, q+1, r);
6.   merge(A, p, q, r);
7.  }
8. }
```

Πολυπλοκότητα

- Διαίσθηση:
  - Για n στοιχεία θα έχουμε logn+1 επίπεδα διαιρέσεων
  - Σε κάθε επίπεδο το κόστος είναι της τάξης του n
- Σύνολο;

συγχωνευτική ταξινόμηση

- αναδρομική σχέση για το χρόνο εκτέλεσης:

$$T(n) = \begin{cases} \Theta(1) & ,\text{εάν } n = 1 \\ 2T(n/2) + \Theta(n) & ,\text{εάν } n > 1 \end{cases}$$

- λύση:

$$T(n) = \Theta(n \cdot \log n)$$

Ταχυταξινόμηση (Quicksort)

- Βασική αρχή:
  - Σε κάθε βήμα του αλγορίθμου
    - επιλέγεται ένα τυχαίο στοιχείο A[q] του πίνακα A,
    - Διαίρει: τα στοιχεία του A χωρίζονται σε αυτά που είναι μικρότερα από το A[q] και σε αυτά που είναι μεγαλύτερα από το A[q]
    - Κυρίευε: Εφαρμόζεται αναδρομικά η ίδια διαδικασία σε κάθε ένα από τα τμήματα του A που προέκυψαν
    - Συνδύασε: Δεν απαιτεί καμία ενέργεια

Ψευδοκώδικας Ταχυταξινόμησης

```
1.quickSort(A, p, r) {
2.  if (p < r) {
3.    q = Διαμέριση(A, p, r);
4.    quickSort(A, p, q-1);
5.    quickSort(A, q+1, r);
6.  }
7.}
```

Πολυπλοκότητα

- **Χειρότερη περίπτωση: $O(n^2)$**
- **Ιδανική Περίπτωση: O(nlogn)**
- **Μέση/Αναμενόμενη περίπτωση: O(nlogn)**, *όπου ο συμβολισμός O(nlogn) κρύβει μια σταθερά λίγο μεγαλύτερη από ότι στην ιδανική περίπτωση*

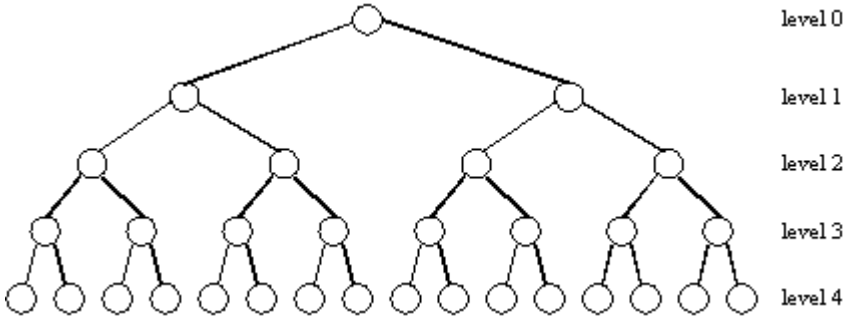
Κάτω όριο (lower bound)

- Ταξινόμηση με συγκρίσεις
  - επιτρέπεται μόνο η σύγκριση στοιχείων

- Υπάρχει κάποιο κάτω όριο για την πολυπλοκότητα χρόνου της ταξινόμησης με συγκρίσεις

Κάτω όριο (lower bound)

- Έστω n διαφορετικά στοιχεία
- Πιθανές διατάξεις: n! $\rightarrow$ Μία από αυτές είναι η σωστή
- Δυαδικό δέντρο με τουλάχιστον n! φύλλα (ένα φύλλο για κάθε πιθανή διάταξη)
  - τι βάθος έχει;



- Χρησιμοποιώντας την προσέγγιση του Stirling για το παραγοντικό (συγκεκριμένα την παρακάτω ανισότητα),

$$n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n \qquad \sqrt{2\pi}\, n^{n+1/2}e^{-n} \leq n! \leq e\, n^{n+1/2}e^{-n},$$

προκύπτει ότι: log (n!) = Ω(n logn)

Πηγές/Αναφορές

- Εισαγωγή στους αλγορίθμους, Cormen, Leiserson, Rivest and Stein, Κεφάλαια 2, 3 και 7
- Δομές Δεδομένων, Π. Μποζάνης, Κεφάλαια 3, 4
- Αλγόριθμοι και Δομές Δεδομένων, N. Wirth, Κεφάλαιο 2