Σχεδίαση και Ανάλυση Αλγορίθμων

Μιχάλης Πλεξουσάκης

Τμήμα Μαθηματικών και Εφαρμοσμένων Μαθηματικών

Χειμερινό Εξάμηνο 2024-25

Η σημερινή διάλεξη

1. Ο αλγόριθμος της ταχυταξινόμησης (quicksort)

2. Επίδοση της ταχυταξινόμησης

3. Κάτω φράγματα αλγόριθμων ταξινόμησης

Η ταχυταξινόμηση

Η διαδικασία της ταχυταξινόμησης

- βασίζεται στην τεχνική σχεδίασης διαίρει-και-κυρίευε
- έχει αναμενόμενο χρόνο εκτέλεσης $\Theta(n \lg n)$, όπως η ταξινόμηση σωρού
- είναι *επιτόπια*, όπως η ενθετική ταξινόμηση
- έχει χρόνο εκτέλεσης χειρότερης περίπτωσης $O(n^2)$

Για την ταξινόμηση της ακολουθίας στοιχείων $A[p \,..\, r]$ ακολουθούμε τα εξης βήματα:

- Διαμερίζουμε τη συστοιχία $A[p \,..\, r]$ σε δύο μικρότερες υποσυστοιχίες $A[p \,..\, q-1]$ και $A[q+1 \,..\, r]$ έτσι ώστε κάθε στοιχείο της $A[p \,..\, q-1]$ να είναι μικρότερο ή ίσο του $A[q]$ το οποίο με τη σειρά του να είναι μικρότερο ή ίσο κάθε στοιχείου της $A[q+1 \,..\, r]$.
- Ταξινομούμε τις υποσυστοιχίες $A[p \,..\, q-1]$ και $A[q+1 \,..\, r]$ με αναδρομικές κλήσεις της διαδικασίας της ταξινόμησης

Η ταχυταξινόμηση

Αφού οι υποσυστοιχίες ταξινομούνται επιτόπου, δεν χρειάζεται καμία επιπλέον εργασία κατά τον συνδυασμό τους. Η συστοιχία $A[p\,..\,r]$ είναι ήδη ταξινομημένη.

$\textsc{Quicksort}(A, p, r)$

  if $p < r$

    $q = \textsc{Partition}(A, p, r)$

    $\textsc{Quicksort}(A, p, q - 1)$

    $\textsc{Quicksort}(A, q + 1, r)$

Η αρχική κλήση για την ταξινόμηση της συστοιχίας $A[1\,..\,n]$ είναι $\textsc{Quicksort}(A, 1, n)$.

Το βασικό συστατικό της αλγόριθμου της ταχυταξινόμησης είναι η διαδικασία της διαμέρισης της συστοιχίας $A[p\,..\,r]$

Η ταχυταξινόμηση

Η διαδικασία της διαμέρισης (Nico Lomuto, 1993):

```
PARTITION(A, p, r)
x = A[r]
i = p - 1
for j = p to r - 1
        if A[j] ≤ x
                i = i + 1
                exchange A[i] ↔ A[j]
exchange A[i + 1] ↔ A[r]
return i + 1
```

Η διαδικασία PARTITION επιλέγει το στοιχείο $x = A[r]$ ως *οδηγό* (*pivot*), δηλαδή, ως το στοιχείο γύρω από το οποίο θα διαμεριστεί η υποσυστοιχία $A[p \mathinner{..} r]$.

Η ταχυταξινόμηση

Καθώς εκτελείται η διαδικασία Partition η συστοιχία διαμερίζεται σε τέσσερις (πιθανόν κενές) περιοχές οι οποίες ικανοποιούν τις παρακάτω συνθήκες (οι απαιτήσεις 1, 2, και 4 μπορούν να διατυπωθούν ως μια αναλλοίωτη συνθήκη βρόχου):

1. Αν $p \leq k \leq i$, τότε $A[k] \leq x$.
2. Αν $i + 1 \leq k \leq j - 1$, τότε $A[k] > x$.
3. Τα στοιχεία στις θέσεις $j \leq k \leq r - 1$ δεν έχουν εξεταστεί ακόμα
4. Αν $k = r$ τότε $A[k] = x$.

Η ταχυταξινόμηση

Η λειτουργία Partition για μια συστοιχία με στοιχεία 13, 19, 9, 5, 12, 8, 7, 4, 2, 21, 2, 6, 11. Τα ελαφρά σκιασμένα στοιχεία είναι αυτά που ανήκουν στην πρώτη περιοχή, δηλαδή, έχουν τιμές μικρότερες από αυτή του οδηγού, ενώ τα έντονα σκιασμένα στοιχεία είναι αυτά που ακήνουν στη δεύτερη περιοχή.

i	p, j											r
	13	19	9	5	12	8	7	4	2	21	2	6	11

i	p	j											r
	13	19	9	5	12	8	7	4	2	21	2	6	11

i	p		j										r
	13	19	9	5	12	8	7	4	2	21	2	6	11

Η ταχυταξινόμηση

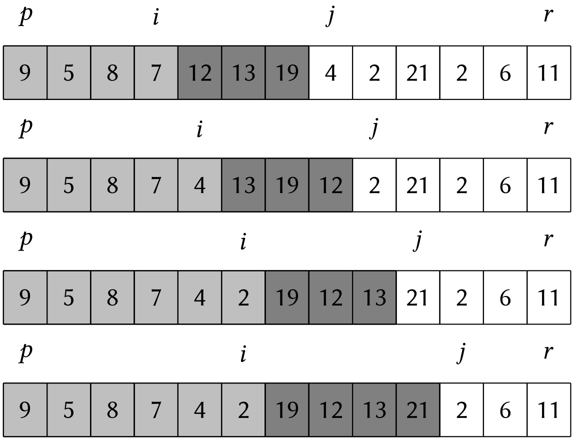
p, i			j									r
9	19	13	5	12	8	7	4	2	21	2	6	11

p	i			j								r
9	5	13	19	12	8	7	4	2	21	2	6	11

p	i				j							r
9	5	13	19	12	8	7	4	2	21	2	6	11

p		i				j						r
9	5	8	19	12	13	7	4	2	21	2	6	11

Η ταχυταξινόμηση



Η ταχυταξινόμηση

p						i					j	r
9	5	8	7	4	2	2	12	13	21	19	6	11

p							i					r
9	5	8	7	4	2	2	6	13	21	19	12	11

Η τελευταία πράξη της Partition είναι εναλλαγή των στοιχείων $A[i+1]$ και $A[r]$ και η επιστροφή του δείκτη $i+1$ ως τον δείκτη του οδηγού:

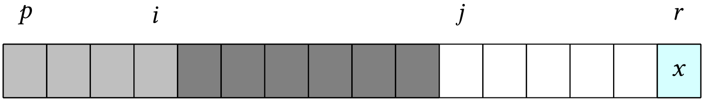
p							i					r
9	5	8	7	4	2	2	6	11	21	19	12	13

Η ταχυταξινόμηση

Ορθότητα του αλγόριθμου της διαμέρισης: Η διαδικασία Partition τηρεί τέσσερεις περιοχές στην υποσυστοιχία $A[p \mathinner{..} r]$.

- Τα στοιχεία στην περιοχή $A[p \mathinner{..} i]$ είναι όλα μικρότερα ή ίσα του x
- Τα στοιχεία της περιοχής $A[i + 1 \mathinner{..} j - 1]$ είναι μεγαλύτερα του x,
- $A[r] = x$.

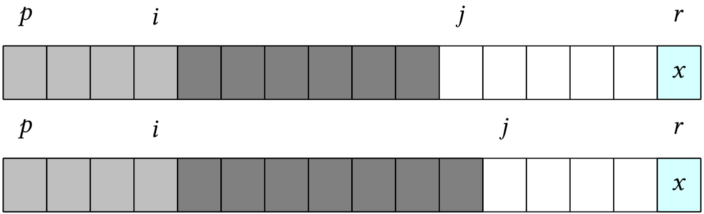
Τα στοιχεία της περιοχής $A[j \mathinner{..} r - 1]$ μπορεί να έχουν οποιαδήποτε τιμή.



Η ταχυταξινόμηση

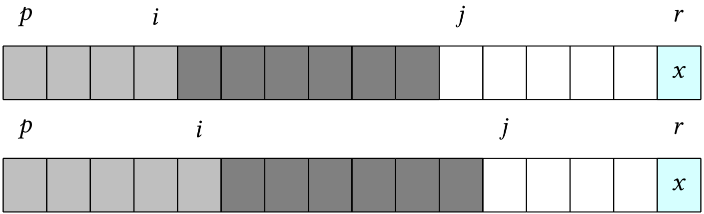
Ορθότητα του αλγόριθμου της διαμέρισης: Πριν από την πρώτη επανάληψη του βρόχου, έχουμε $i = p - 1$ και $j = p$, επομένως δεν υπάρχει αριθμός μεταξύ των p και i, αλλά ούτε και μεταξύ των $i + 1$ και $j - 1$. Επομένως οι δύο πρώτες απαιτήσεις της αναλλοίωτης συνθήκης ικανοποιούνται κατά τετριμμένο τρόπο. Η τρίτη απαίτηση εκπληρώνεται από την ανάθεση στη γραμμή 1 της διαδικασίας Quicksort.

Η αναλλοίωτη συνθήκη διατηρείται σε κάθε επανάληψη: Όταν $A[j] > x$ η μόνη ενέργεια που πραγματοποιείται είναι η αύξηση του j κατά ένα.



Η ταχυταξινόμηση

Η αναλλοίωτη συνθήκη διατηρείται σε κάθε επανάληψη: Αν $A[j] \leq x$ τότε αυξάνεται το i κατά ένα, εναλλάσσουν θέσεις τα στοιχεία $A[i]$ και $A[j]$, και στη συνέχεια αυξάνεται το j κατά ένα.



Λόγω αυτής της εναλλαγής έχουμε $A[i] \leq x$, και επομένως η απαίτηση 1 ικανοποιείται. Επίσης έχουμε $A[j-1] > x$, αφού το στοιχείο που μετακινήθηκε στη θέση $A[j-1]$ είναι μεγαλύτερο του x. Αυτό δείχνει ότι η απαίτηση 2 ικανοποιείται επίσης.

Η ταχυταξινόμηση

Η αναλλοίωτη συνθήκη διατηρείται μετά τον τερματισμό των επανάληψεων: έχουμε $j = r$ και κάθε στοιχεία της συστοιχίας ανήκει σε μια από τις τρεις περιοχές που περιγράφονται στην αναλλοίωτη συνθήκη. Οι δύο τελευταίες γραμμές στη διαδικασίας Partition μετακινούν το στοιχείο–οδηγό εναλλάσοντάς το με το πρώτο στοιχείο, το οποίο είναι μεγαλύτερο του x.

Παρατήρηση. Ο αλγόριθμος της ταχυταξινόμησης, όπως διατυπώθηκε, εμπεριέχει δύο αναδρομικές κλήσεις στον εαυτό του. Η δεύτερη αναδρομική κλήση μπορεί να αποφευχθεί μέσω μιας επαναληπτικής δομής ελέγχου:

```
Quicksort2(A, p, r)
    while p < r
        q = Partition(A, p, r)
        Quicksort2(A, p, q − 1)
        p = q + 1
```

Η ταχυταξινόμηση

Quicksort2(A, p, r)

while $p < r$

$q = $ Partition(A, p, r)

Quicksort2$(A, p, q - 1)$

$p = q + 1$

- τόσο η Quicksort όσο και η Quicksort2 βασίζονται στην ίδια διαδικασία διαμέρισης και στη συνέχεια καλούν αναδρομικά τον εαυτό τους με ορίσματα $A, p, q - 1$
- μετά η Quicksort καλεί αναδρομικά τον εαυτό με ορίσματα $A, q + 1, r$ ενώ η Quicksort2 θέτει $p = q + 1$ και εκτελεί ξανά μια επανάληψη του βρόχου **while** .
- Η επανάληψη αυτή εκτελεί ακριβώς τις ίδιες πράξεις όπως η αναδρομική κλήση με ορίσματα $A, q + 1, r$.

Επίδοση της ταχυταξινόμησης

Ο χρόνος εκτέλεσης της ταχυταξινόμησης εξαρτάται από το κατά πόσο η διαμέριση της συστοιχίας είναι ισομερής, και αυτό με τη σειρά του εξαρτάται από την επιλογή του οδηγού. Θα δείξουμε ότι η χειρότερη περίπτωση προκύπτει όταν η διαδικασία Partition οδηγεί σε δύο υποπροβλήματα μεγέθους $n-1$ και 0, αντίστοιχα.

Ας υποθέσουμε ότι αυτή η διαμέριση προκύπτει σε κάθε αναδρομική κλήση. Δεδομένου του ότι

- το κόστος της διαμέρισης είναι $\Theta(n)$
- το κόστος μιας κλήσης της διαδικασίας ταχυταξινόμησης για ένα πρόβλημα μηδενικού μεγέθους είναι $\Theta(1)$

έχουμε για την αναδρομική σχέση για τον χρόνο εκτέλεσης

$$T(n) = T(n-1) + T(0) + \Theta(n) = T(n-1) + \Theta(n).$$

Επίδοση της ταχυταξινόμησης

Η λύση της αναδρομικής σχέσης

$$T(n) = T(n-1) + \Theta(n).$$

είναι $T(n) = \Theta(n^2)$. Αυτό σημαίνει ότι για τη δεδομένη ανισομερή διαμέριση, η ταχυταξινομήση δεν υπερτερεί ούτε καν της ενθετικής ταξινόμησης!

Ακόμα, ο χρόνος $\Theta(n^2)$ αφορά και στην περίπτωση που η συστοιχία εισόδου είναι ήδη ταξινομημένη, περίπτωση για την οποία η ενθετική ταξινόμηση εκτελείται σε χρόνο $O(n)$.

Επίδοση της ταχυταξινόμησης

Στον πλέον ισόρροπο διαχωρισμό, η διαδικασία Partition παράγει δύο υποπροβλήματα μεγέθους $\lfloor n/2 \rfloor$ και $\lceil n/2 \rceil - 1$, αντίστοιχα, μεγέθη μικρότερα από $n/2$. Σε αυτή την περίπτωση η αναδρομική σχέση για τον χρόνο εκτέλεσης είναι

$$T(n) \leq 2T(n/2) + \Theta(n).$$

Από την περίπτωση 2 του κεντρικού θεωρήματος παίρνουμε τη λύση $T(n) = O(n \lg n)$, δηλαδή έναν ασυμπτωτικά ταχύτερο αλγόριθμο.

Μπορεί να αποδειχθεί ότι ο χρόνος εκτέλεσης της μέσης περίπτωσης προσεγγίζει πολύ περισσότερο την καλύτερη περίπτωση παρά τη χειρότερη. Ειδικότερα, οποιοσδήποτε διαχωρισμός με *σταθερή* αναλογία δίνει χρόνο εκτέλεσης $O(n \lg n)$.

Επίδοση της ταχυταξινόμησης

Αν, για παράδειγμα, υποθέσουμε ότι ο αλγόριθμος διαμέρισης δίνει πάντοτε υποπροβλήματα με αναλογία μεγεθών 9 προς 1, τότε ο χρόνος εκτέλεσης της ταχυταξινόμησης περιγράφεται από την αναδρομική σχέση

$$T(n) \leq T(9n/10) + T(n/10) + cn,$$

με τη σταθερά c αυτή που εμπεριέχεται στον όρο $\Theta(n)$. Στο δένδρο αναδρομής για αυτή τη σχέση βλέπουμε ότι κάθε επίπεδο του δένδρου έχει κόστος cn και η αναδρομή τερματίζεται σε βάθος $\log_{10/9} n = \Theta(\lg n)$. Επομένως το συνολικό κόστος είναι $O(n \lg n)$.

Επίδοση της ταχυταξινόμησης

Δείχνουμε ότι ο χρόνος εκτέλεσης της χειρότερης περίπτωσης του αλγόριθμου της ταχυταξινόμησης είναι $O(n^2)$. Αν $T(n)$ είναι ο χρόνος εκτέλεσης

$$T(n) \le \max_{0 \le q \le n-1} (T(q) + T(n-q-1)) + \Theta(n), \tag{1}$$

με q μεταξύ 0 και $n-1$ γιατί η διαδικασία Partition παράγει δύο υποπροβλήματα συνολικού μεγέθους $n-1$. Εικάζουμε ότι $T(n) \le cn^2$. Αντικαθιστώντας στην (1)

$$\begin{aligned} T(n) &\le \max_{0 \le q \le n-1} \left(cq^2 + c(n-q-1)^2\right) + \Theta(n) \\ &\le c \max_{0 \le q \le n-1} \left(q^2 + (n-q-1)^2\right) + \Theta(n). \end{aligned}$$

Η ποσότητα $q^2 + (n-q-1)^2$ λαμβάνει τη μέγιστη τιμή της όταν $q = n-1$ και αυτή είναι ίση με $(n-1)^2$. Τότε,

$$T(n) \le cn^2 - c(2n-1) + \Theta(n) \le cn^2,$$

επιλέγοντας τη σταθερά c ώστε να επικρατεί του όρου $\Theta(n)$. Συνεπώς, $T(n) = O(n^2)$.

Επίδοση της ταχυταξινόμησης

$$T(n) \leq \max_{0 \leq q \leq n-1} (T(q) + T(n-q-1)) + \Theta(n),$$

Έχουμε ήδη δει ότι στην περίπτωση μιας ειδικής ανισομερούς διαμέρισης η ταχυταξινόμηση απαιτεί χρόνο $\Omega(n^2)$. Επομένως ο χρόνος εκτέλεσης χειρότερης περίπτωσης της ταχυταξινόμησης είναι $\Theta(n^2)$.

Κάτω φράγματα αλγόριθμων ταξινόμησης

Όλοι οι αλγόριθμοι ταξινόμησης που μελετήσαμε έχουν ως κοινό χαρακτηριστικό ότι η ταξινομημένη διάταξη που παράγουν βασίζεται μόνο σε συγκρίσεις των στοιχείων εισόδου. Αυτού του είδους οι αλγόριθμοι ταξινόμησης ονομάζονται αλγόριθμοι *συγκριτικής ταξινόμησης* (*comparison sort algorithms*).

Οι αλγόριθμοι συγκριτικής ταξινόμησης μπορούν να αναπαρασταθούν σε αφηρημένο επίπεδο μέσω *δένδρων αποφάσεων*.

Ένα δένδρο αποφάσεων είναι ένα πλήρες δυαδικό δένδρο το οποίο αντιπροσωπεύει τις συγκρίσεις που πραγματοποιούνται μεταξύ στοιχείων από κάποιο αλγόριθμο ταξινόμησης.

Κάτω φράγματα αλγόριθμων ταξινόμησης

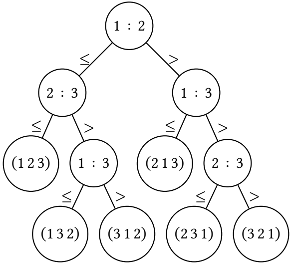
Αν υποθέσουμε ότι όλα τα στοιχεία εισόδου είναι διαφορετικά μεταξύ τους, συγκρίσεις της μορφής $a_i = a_j$ είναι περιττές και μπορούμε να υποθέσουμε ότι δεν εκτελούνται.

Οι συγκρίσεις όμως $a_i \leq a_j$, $a_i \geq a_j$, $a_i < a_j$ και $a_j < a_i$ είναι όλες ισοδύναμες μεταξύ τους αφού παρέχουν την ίδια ακριβώς πληροφορία σχετικά με τη διάταξη των στοιχείων a_i και a_j.

Μπορούμε επομένως να υποθέσουμε ότι όλες οι συγκρίσεις είναι της μορφής $a_i \leq a_j$.

Κάτω φράγματα αλγόριθμων ταξινόμησης

Το δένδρο αποφάσεων για τον αλγόριθμο της ενθετικής ταξινόμησης για την ακολουθία στοιχείων $a_1 = 6$, $a_2 = 8$ και $a_3 = 5$. Κόμβοι με ετικέτες της μορφής $i : j$ υποδεικνύουν μια σύγκριση μεταξύ των a_i και a_j. Στους καταληκτικούς κόμβους αναφέρεται η διάταξη των στοιχείων εισόδου.



Είναι προφανές ότι αφού υπάρχουν $3! = 6$ δυνατές μεταθέσεις των στοιχείων εισόδου, το δένδρο αποφάσεων θα πρέπει να έχει τουλάχιστον 6 καταληκτικούς κόμβους.

Κάτω φράγματα αλγόριθμων ταξινόμησης

Κάθε ορθός αλγόριθμος συγκριτικής ταξινόμησης θα πρέπει να έχει τη δυνατότητα να παράγει οποιαδήποτε μετάθεση από τις $n!$ μεταθέσεις των n στοιχείων εισόδου. Κάθε μια από αυτές τις μεταθέσεις θα πρέπει να εμφανίζεται σε έναν από τους καταληκτικούς κόμβους του δένδρου αποφάσεων και κάθε τέτοιος κόμβος θα πρέπει να είναι προσβάσιμος από τη ρίζα μέσω κάποιας διαδρομής.

Το ύψος του δένδρου αποφάσεων είναι το πλήθος των συγκρίσεων στη χειρότερη περίπτωση για έναν αλγόριθμο συγκριτικής ταξινόμησης. Επομένως ένα κάτω φράγμα για τα ύψη των δένδρων αποφάσεων στα οποία οποιαδήποτε μετάθεση εμφανίζεται σε κάποιον καταληκτικό κόμβο αντιπροσωπεύει ένα κάτω φράγμα για τον χρόνο εκτέλεσης οποιουδήποτε αλγόριθμου συγκριτικής ταξινόμησης.

Κάτω φράγματα αλγόριθμων ταξινόμησης

Θεώρημα. Οποιοσδήποτε αλγόριθμος συγκριτικής ταξινόμησης απαιτεί $\Omega(n \lg n)$ συγκρίσεις στη χειρότερη περίπτωση.

Απόδειξη. Έστω ένα δένδρο αποφάσεων με ύψος h και με l καταληκτικούς κόμβους. Πρέπει $n! \leq l$, αφού κάθε μια από τις $n!$ μεταθέσεις της εισόδου εμφανίζεται σε κάποιον καταληκτικό κόμβο.

Γνωρίζουμε ότι ένα δυαδικό δένδρο με ύψος h έχει το πολύ 2^h καταληκτικούς κόμβους, συνεπώς

$$n! \leq l \leq 2^h.$$

Παίρνοντας λογάριθμους έχουμε ότι $h \geq \lg(n!)$. Δείχνουμε τώρα ότι $\lg(n!) = \Omega(n \lg n)$. Πράγματι, από τον τύπο του Stirling έχουμε ότι

$$n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n e^{\alpha_n},$$

όπου

$$\frac{1}{12n+1} < \alpha_n < \frac{1}{12n}.$$

Κάτω φράγματα αλγόριθμων ταξινόμησης

Θεώρημα. Οποιοσδήποτε αλγόριθμος συγκριτικής ταξινόμησης απαιτεί $\Omega(n \lg n)$ συγκρίσεις στη χειρότερη περίπτωση.

Απόδειξη (συνέχεια). Παίρνοντας λογάριθμους έχουμε

$$\begin{aligned} \lg(n!) &= n \lg\left(\frac{n}{e}\right) + \frac{1}{2}\lg n + \frac{1}{2}\lg(2\pi) + (\lg e)\alpha_n \\ &\geq n \lg\left(\frac{n}{e}\right) \\ &\geq cn \lg n, \end{aligned}$$

για κάποια θετική σταθερά c. □

Πόρισμα. Η ταξινόμηση σωρού και η συγχωνευτική ταξινόμηση είναι ασυμπτωτικά βέλτιστοι αλγόριθμοι συγκριτικής ταξινόμησης.