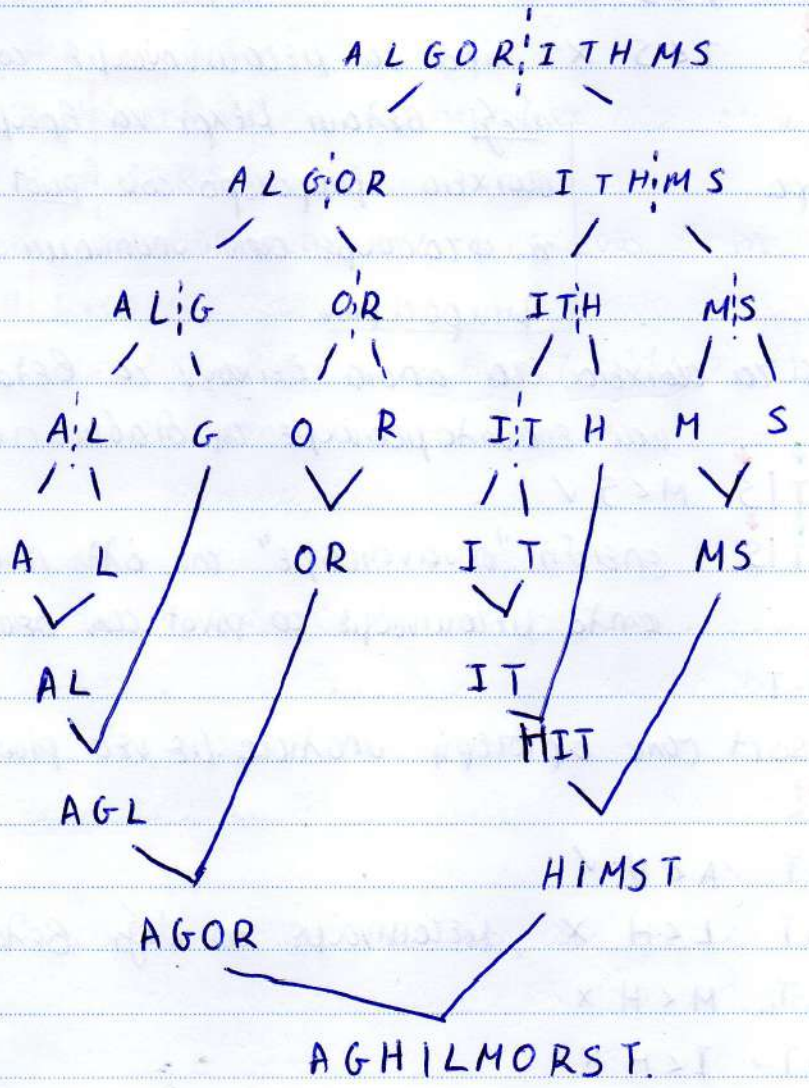
ΑΛΓΟΡΙΘΜΟΙ

• ΦΡΟΝΤΙΣΤΗΡΙΟ 3 •

(Διαίρει & Βασίλευε).

Άσκηση 1

Να ταξινομήσετε τα γράμματα της λέξης 'Algorithms' χρησιμοποιώντας τον αλγόριθμο Mergesort.



Άσκηση 2

Χρησιμοποιήσατε τον αλγόριθμο Quick Sort για να ταξινομήσετε την αλφαβητοσειρά 'Algorithms'.

ALGORITHMS → Παίρνουμε για pivot το μεσαίο στοιχείο και το μεταφέρουμε στο τέλος, αντιλλαίσσοντας το με το τελευταίο

ALGOSITHM|R → Έχουμε 2 "δείκτες" έναν στην αρχή και ένα στο τέλος του subarray για να κάνουμε partition.

A<R → δεξιά

ALGOSITHM|R → Μετακινούμε τον αριστερό δείκτη δεξιά μέχρι να συναντήσουμε στοιχείο ίσο ή μεγαλύτερο του pivot

L<R → δεξιά

ALGOSITHM|R → Όταν βρούμε ίσο ή μεγαλύτερο σταματάμε την κίνηση σε αυτό το βελάκι και συνεχίζουμε με το δεξί προχωρώντας προς τα αριστερά μέχρι να βρούμε τιμή μικρότερη του pivot, ή προσπεράσουμε το άλλο βελάκι.

G<R → δεξιά

ALGOSITHM|R

O<R → δεξιά

ALGOSITHM|R

S>R ! STOP

ALGOSITHM|R

M<R ! STOP

ΑΝΤΑΛΑΣΣΟΥΜΕ ΤΑ ΣΤΟΙΧΕΙΑ :

ALGOMITHS|R → Εφόσον βρούμε μικρότερη τιμή σταματάμε και ανταλλάσσουμε τα στοιχεία στα οποία δείχνουν τα βελάκια. Μετακινούμε τους δείκτες με τον ίδιο τρόπο από τις θέσεις που ήδη βρίσκονται.

M<R → δεξιά

ALGOMITHS|R

I<R → δεξιά

ALGOMITHS|R

T>R ! STOP

δεξί βελάκι S>R → αριστερά

ALGOMITHS|R

H<R ! STOP

ΑΝΤΑΛΛΑΓΗ → Τώρα που φτάσαμε τα βελάκια να "συναντώνται" τα προχωράμε με τον ίδιο τρόπο μέχρι να περάσει το ένα στη μέση του άλλου

ALGOMIHTS|R

H<R → δεξιά

ALGOMIHTS|R

T>R ! STOP αριστερό βελάκι

A L G O M I H T S | R

T > R → αριστερά το δεξί θέλει

A L G O M I H T S | R

τα θέλαμα "άλλαξαν μερίδες"

→ Όταν συμβαίνει αυτό τότε έχουμε σιγουρέψει πως όλα τα στοιχεία αριστερά του αριστερού δείκτη (A) είναι μικρότερα του pivot, και αντίστοιχα δεξιά του δεξιού δείκτη (Δ) βρίσκονται στοιχεία μεγαλύτερα ή ίσα του pivot.

→ Αντιλαλάσσουμε το pivot με το T μετακινώντας το στη σωστή του θέση

A L G O M I H, R S T

κάνουμε quicksort στην υπόλοιπα αριστερά του παλιού μας pivot:

new pivot

A L G O M I H R S T

A L G H M I | O R | S T

A < O → δεξιά το αριστερό

A L G H M I | O R S T

L < O → δεξιά

A L G H M I | O R S T

G < O → δεξιά

A L G H M I | O R S T

H < O → δεξιά

A L G H M I | O R S T

M < O → δεξιά

A L G H M I | O R S T

I < O → δεξιά

(συνέχεια δίπλα σελίδα)

A L G H M I | O R S T

O = O ! STOP

Τα θέλαμα άλλαξαν μερίδες

Quicksort στην υπόλοιπα αριστερά του O

A L G H M I O R S T

A L I H M | G O R S T

A < G → δεξιά

A L I H M | G O R S T

L > G ! STOP M ~~>~~ G → αριστερά

A L I H M | G O R S T

H > G → αριστερά

A L I H M | G O R S T

I > G → αριστερά

A Δ
A L I H M | G O R S T

Δ A L > G → αριστερά
A L I H M | G O R S T

Βελάκια άλλαξαν θέσεις
Βάζουμε το G στη θέση του
(μετά τις μικρότερες αξίες)

A, G I H M L O R S T

η αριστερή υπολίστα περιέχει μόνο ένα στοιχείο άρα είναι ταξινομημένη

Quicksort στην υπολίστα αριστερά του G και δεξιά του Q (μεταξύ των pivots)

new pivot
A G I H M L O R S T

A G I L M | H O R S T

I > H ! STOP

M > H → αριστερά
A G I L M | H O R S T

A Δ L > H → αριστερά
A G I L M | H O R S T

Δ A I > G → αριστερά
A G I L M | H O R S T

Βελάκια άλλαξαν θέσεις
αντάλλαξη I με H
A G H L M I O R S T

Quicksort στην υπολίστα
L M I

new pivot
A G H L M I O R S T

A Δ
A G H L I | M O R S T

L < M → δεξιά
A Δ Δ
A G H L I | M O R S T

I < M → δεξιά
Δ Δ A
A G H L I | M O R S T

τα βελάκια άλλαξαν θέσεις
οπότε όλα αριστερά του M είναι μικρότερα του αρά βρίσκεται στη σωστή θέση

Quicksort στην LI

new pivot
A G H L I M O R S T

A Δ
A G H I | L M O R S T

I < L → δεξιά το A
Δ Δ A
A G H I | L M O R S T

τα βελάκια άλλαξαν μεριές, το L στη σωστή του θέση.

Quicksort στο I, που είναι L στοιχείο αρα ταξινομημένο:
A G H I L M O R S T

Μένει Quicksort στην ST

Pivot
A G H I L M O R S T
A G H I L M O R T S

T > S ! STOP A.

T > S → αριστερά Δ
Δ A
A G H I L M O R T S

βελάκια αντάλλαξαν θέσεις, βάζουμε το S στη σωστή του θέση:
A G H I M O R S T : SORTED

Άσκηση 3

Ποιος είναι ο χρόνος εκτέλεσης του Quick Sort:

α) Όταν όλα τα στοιχεία του πίνακα Α είναι ίσα;

β) Όταν ο πίνακας Α είναι αναποδα ταξινομημένος και όλα τα στοιχεία του είναι άνισα;

Λύση

α) Σε κάθε βήμα γίνεται 1 κλήση της Quick Sort για πίνακα κατά 1 θέση μικρότερο από πριν. Το "partition" σε κάθε βήμα απαιτεί χρόνο $O(n)$. Έχουμε συνολικά n αναδρομικές κλήσεις όπου $O(n)$ η πολυπλοκότητα της κάθε κλήσης, άρα η συνολική πολυπλοκότητα είναι $n \cdot O(n) = O(n^2)$.

β) Το pivot, αν ο πίνακας είναι σε φθίνουσα ταξινόμηση και εμείς θέλαμε να κάνουμε αύξουσα ταξινόμηση, είναι πάντα το μικρότερο στοιχείο, οπότε στην επόμενη κλήση θα κάνουμε partition σέναν πίνακα $n-1$ στοιχείων με pivot πάλι το μικρότερο στοιχείο όπως είπαμε. Ομοίως, στην μεσεπόμενη περίπτωση θα έχουμε partition σε πίνακα $n-2$ κλπ. Συνεπώς θα γίνουν n συνολικά αναδρομικές κλήσεις που σε αυτές το partition επίσης απαιτεί χρόνο $O(n)$, άρα η πολυπλοκότητα θα είναι $n \cdot O(n) = O(n^2)$.

Ή απλώς λύνοντας την αναδρομική εξίσωση του αλγορίθμου για πίνακα n που προκύπτει από ταξινόμηση $n-1$ έχουμε:

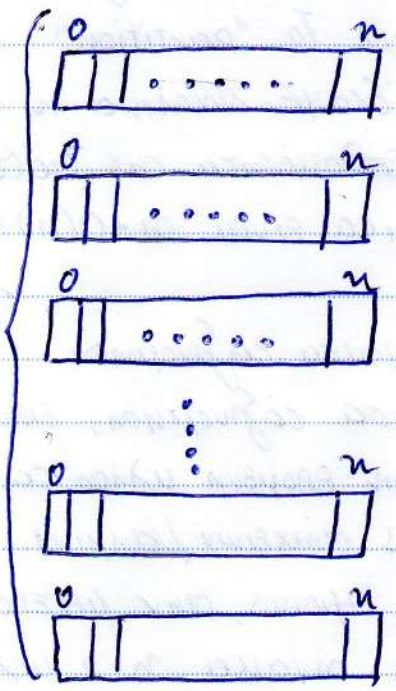
$$T(n) = T(n-1) + n = \sum_{k=1}^{n} k = \frac{(n+1)n}{2} = O(n^2).$$

→ Περισσότερα για την πολυπλοκότητα εν γένει της QuickSort στις διαφάνειες "Διαίρει και βασίλευε" εδώ σε σύγχρονα. ←

Άσκηση 4

Δίνονται k ταξινομημένοι πίνακες n στοιχείων ο καθένας. Να σχεδιάσετε αλγόριθμο δυαίρει και βασίλευε που να καταλαμβάνει χώρο για έναν ταξινομημένο πίνακα με όλα τα στοιχεία και να υπολογίσετε την πολυπλοκότητά του

Λύση



Αλγόριθμος (αναδρομικός)

- Χωρίζω τους πίνακες σε 2 σύνολα πινάκων ($k/2$ και $k/2$)
- Καλώ τον αλγόριθμο - ταξινομώ - τον υποπίνακα που προκύπτει από το κάθε σύνολο (κάθε σύνολο μπορεί να θεωρηθεί ως ενιαίος πίνακας $\frac{k \cdot n}{2}$ στοιχείων)
- Συγχωνεύω (merge) τους 2 υποπίνακες

Η βάση της αναδρομής μας θα είναι να έχουμε έναν πίνακα σε κάθε σύνολο, οπότε τότε απλώς συγχωνεύουμε τους 2 πίνακες.

Αναδρομική εξίσωση: $T(k) = 2\,T\left(\frac{k}{2}\right) + O(kn)$

$(a=2,\ b=2,\ f(n) = kn)$.

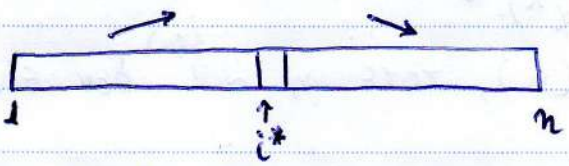
Άρα $T(k) = kn \log k$.

Άσκηση 5 / 3 (Διαίρει και Βασίλευε).

Δίνεται πίνακας n ακεραίων Α με την εξής ιδιότητα: Υπάρχει δείκτης i^* τέτοιος ώστε $1 \le i^* \le n$, τα A[1], A[2], ... A[i^*] είναι σε αύξουσα διάταξη και τα A[i^*], A[i^*+1], ..., A[n] είναι

σε φθίνουσα σειρά. Να δώσετε αλγόριθμο που υπολογίζει το i^* σε χρόνο $O(\log n)$.

Λύση



Το πρόβλημα ανήκει Binary Search καθώς:

1) Δίνεται πίνακα που έχει ταξινόμηση και αναζητείται ένα στοιχείο του.
2) Ζητείται αλγόριθμος πολυπλοκότητας $O(\log n)$ που είναι ίδια με τον χρόνο της δυαδικής αναζήτησης.

Αλγόριθμος (αρχική θέση, τελευταία θέση)

```
Unimodal(A, p, r)
if p=r then
    return p;
else if A[⌊(p+r)/2⌋] < A[⌊(p+r)/2⌋ + 1]
    return Unimodal(A, ⌊(p+r)/2 + 1⌋, r)
else
    return Unimodal(A, p, ⌊(p+r)/2⌋).
```

Αν το μεσαίο και το επόμενο του στοιχείο στον πίνακα ακολουθούν την ίδια διάταξη (εδώ φθίνουσα) "πετάμε" το άχρηστο κομμάτι και προχωράμε στο άνω μισό, που αποτελεί εκείνο όπου βρίσκεται η εναλλαγή σειράς.

Πολυπλοκότητα: (κάθε φορά καλούμε για το μισό μέρος του αρχικού)

$$T(n) = T(n/2) + \Theta(1)$$ ← σταθερό πλήθος συγκρίσεων σε κάθε κλήση οπότε $\Theta(1)$.

Από δεύτερη περίπτωση Master Theorem ($a=1$, $b=2$, $f(n) = n^0 * n^{\log_2 1} = n^{0+1}$).

Προκύπτει ότι $T(n) = \Theta(\log n)$ άρα προφανώς και $T(n) = O(\log n)$

ΑΛΓΟΡΙΘΜΟΙ

• ΦΡΟΝΤΙΣΤΗΡΙΟ 4 •

Άσκηση 6/3 (Διαίρει & Βασίλευε)

Χρησιμοποιώντας τη μέθοδο Master να υπολογίσετε ασυμπτωτικά όρια στις παρακάτω αναδρομές:

α. $T(n) = 4T\left(\frac{n}{2}\right) + n$

β. $T(n) = 4T\left(\frac{n}{2}\right) + n^2$

γ. $T(n) = 4T\left(\frac{n}{2}\right) + n^3$

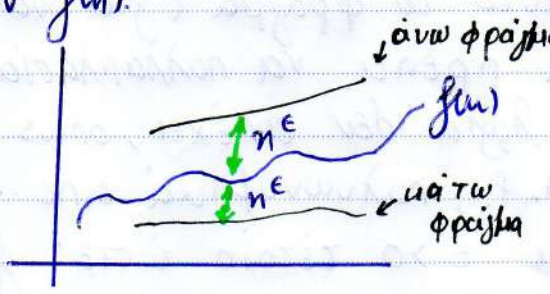
Λύση

1) Υπολογίζω $n^{\log_b a}$, εδώ σε όλα τα ερωτήματα είναι:

$$n^{\log_2 4} = n^2 \quad (\text{αφού } a=4,\ b=2 - a\,T\left(\tfrac{n}{b}\right)-).$$

2) Τώρα συγκρίνω τις $f(n)$ με το n^2 για να κρίνουμε αν μπορεί να εφαρμοστεί το Master.

→ Το Master Theorem μπορεί να εφαρμοστεί όταν το φράγμα που έχουμε διαφέρει πολυωνυμικά από την $f(n)$.



α) $f(n) = O(n^{2-\epsilon})$ για $\epsilon = 1$, $f(n) = n$

Από πρώτη περίπτωση Μ. Τ. έχουμε: $T(n) = \Theta(n^2)$.

β) $f(n) = \Theta(n^2)$, $f(n) = n^2$.

Από δεύτερη περίπτωση Μ. Τ. έχουμε: $T(n) = \Theta(n^2 \log n)$.

g) $f(n) = \Omega(n^{2+\epsilon})$, για $\epsilon = 1$, $f(n) = n^3$.
Πρέπει επίσης:

$$4\left(\frac{n}{2}\right)^3 = 4\,\frac{n^3}{8} = \frac{n^3}{2} \le c \cdot f(n) = c \cdot n^3$$

για $c \in [1/2, 1)$.

Από τρίτη περίπτωση Μ. Τ. έχουμε: $T(n) = \Theta(n^3)$.

Άσκηση 7/3 (Διαίρει και βασίλευε)
Μπορούμε να εφαρμόσουμε τη μέθοδο Master στην ακόλουθη αναδρομή; $T(n) = 2T(n/2) + n\log n$.

Λύση

$T(n) = 2T(n/2) + n\log n$

$a = 2 \quad b = 2 \quad f(n) = n\log n$.

1) $n^{\log_b a} = n$

2) $f(n) = n\log n = \Omega(n)$, η $f(n)$ φαίνεται μεγαλύτερη από την n, ωστόσο έχει λογαριθμική διαφορά από αυτό το φράγμα (για να πάω από n σε $n\log n$ πρέπει να πολλαπλασιάσω με $\log n$). Με άλλα λόγια δεν απέχει, όπως θα ήταν ιδανικό για Μ. Τ., πολυωνυμικά από τη n, άρα δεν υπάρχει $\epsilon > 0$ τέτοιο ώστε $f(n) = \Omega(n^{1+\epsilon})$. Συνεπώς δεν μπορούμε να εφαρμόσουμε Master.

Εναλλακτικά: Έστω οποιοδήποτε $\epsilon > 0$ έχουμε:

$$\lim_{n\to\infty} \frac{f(n)}{n^{1+\epsilon}} = \lim_{n\to\infty} \frac{n\log n}{n \cdot n^{\epsilon}} = \lim_{n\to\infty} \frac{\log n}{n^{\epsilon}} = 0.$$

Επομένως $f(n) \neq \Omega(n^{1+\epsilon})$, άρα η μέθοδος Master δεν μπορεί να εφαρμοστεί.