Διαίρει και Βασίλευε

Δ. Μάγος

12 Φεβρουαρίου 2020

Διαίρει και Βασίλευε

Η μέθοδος αυτή επιλύει ένα πρόβλημα διασπώντας το σε μικρότερα υποπροβλήματα, τις λύσεις των οποίων συνδυάζει για την επίτευξη της τελικής λύσης. Αναλυτικά, τα βήματα είναι:

- Διαίρει: το αρχικό πρόβλημα διασπάται σε απλούστερα υποπροβλήματα όσο το δυνατόν ίδιου μεγέθους.
- Βασίλευε: κάθε υποπρόβλημα επιλύεται με *αναδρομικό* τρόπο.
- Συνδύασε:οι επιμέρους λύσεις συνδυάζονται προκειμένου να δοθεί λύση στο αρχικό πρόβλημα.

Αλγόριθμοι αυτού του τύπου είναι η δυαδική αναζήτηση (quicksort) ή συγχωνευτική ταξινόμηση (mergesort), κλπ.

Περιγραφή

Μία σειρά από ακεραίους $a_1, \ldots, a_n$ ονομάζεται *μονότροπη (unimodal)* αν υπάρχει ένα στοιχείο a_k, $1 \leq k \leq n$, για το οποίο ισχύει

$$a_1 < a_2 < \cdots < a_{k-1} < a_k > a_{k+1} > \cdots > a_{n-1} > a_n.$$

Από τον παραπάνω ορισμό προκύπτει ότι το σημείο καμπής μπορεί να είναι το πρώτο ή το τελευταίο στοιχείο της σειράς. Το στοιχείο a_k ονομάζεται *σημείο καμπής*.

Πρόβλημα Δίνεται μία μονότροπη σειρά ακεραίων αποθηκευμένη στις θέσεις $1, \ldots, n$ του πίνακα A. Να βρεθεί το σημείο καμπής.

Email:dmagos@uniwa.gr

Ιδέα

Έστω $A[r]$ ένα στοιχείο της σειράς. Αν $A[r] < A[r+1]$ τότε το σημείο καμπής βρίσκεται σε κάποια από τις θέσεις $r+1, \ldots, n$. Διαφορετικά, (αν $A[r] > A[r+1]$) τότε το σημείο καμπής βρίσκεται σε κα'ποια από τις θέσεις $1, \ldots, r$.

Είναι προφανές ότι μπορούμε να επιλύσουμε το πρόβλημα εκτελώντας δυαδική αναζήτηση πάνω στα στοιχεία του πίνακα A. Η συνάρτηση unimodal υλοποιεί την ιδέα.

Αλγόριθμος

int unimodal($int\ A[], int\ lb, int\ ub$)
if lb = ub **then**
  return $A[lb]$;
end if
$mid \leftarrow \lfloor \frac{lb+ub}{2} \rfloor$;
if $A[mid] < A[mid+1]$ **then**
  return unimodal($A, mid+1, ub$);
else
  return unimodal(A, lb, mid);
end if

Η αρχική κλήση της συνάρτησης είναι

$$\text{unimodal}(A, 1, n)$$

Ανάλυση

Ακριβώς ίδια με τη δυαδική αναζήτηση. Θεωρώντας ότι το n είναι δύναμη του 2, έχουμε:

$$T(n) = T(n/2) + c \Rightarrow T(n) = \Theta(\lg n).$$

Το πρόβλημα ξανά

- Πρόβλημα: Δεδομένου ενός πίνακα A, ζητείται η αναδιάταξη των στοιχείων (ακεραίων) $A[1], \ldots, A[n]$ σε αύξουσα σειρά.

Το πρόβλημα ξανά

- Πρόβλημα: Δεδομένου ενός πίνακα A, ζητείται η αναδιάταξη των στοιχείων (ακεραίων) $A[1], \ldots, A[n]$ σε αύξουσα σειρά.
- Ιδέα: Τοποθέτηση ενός στοιχείου στη *σωστή θέση*: στη θέση που θα βρίσκεται στην ταξινομημένη σειρά. Αναδρομική κλήση με τα στοιχεία του πίνακα που βρίσκονται αριστερότερα και δεξιότερα της σωστής θέσης.

Το πρόβλημα ξανά

- Πρόβλημα: Δεδομένου ενός πίνακα A, ζητείται η αναδιάταξη των στοιχείων (ακεραίων) $A[1], \ldots, A[n]$ σε αύξουσα σειρά.
- Ιδέα: Τοποθέτηση ενός στοιχείου στη *σωστή θέση*: στη θέση που θα βρίσκεται στην ταξινομημένη σειρά. Αναδρομική κλήση με τα στοιχεία του πίνακα που βρίσκονται αριστερότερα και δεξιότερα της σωστής θέσης.
- Ποιο στοιχείο θα τοποθετηθεί στη σωστή θέση;

Το πρόβλημα ξανά

- Πρόβλημα: Δεδομένου ενός πίνακα A, ζητείται η αναδιάταξη των στοιχείων (ακεραίων) $A[1], \ldots, A[n]$ σε αύξουσα σειρά.
- Ιδέα: Τοποθέτηση ενός στοιχείου στη *σωστή θέση*: στη θέση που θα βρίσκεται στην ταξινομημένη σειρά. Αναδρομική κλήση με τα στοιχεία του πίνακα που βρίσκονται αριστερότερα και δεξιότερα της σωστής θέσης.
- Ποιο στοιχείο θα τοποθετηθεί στη σωστή θέση;
  το δεξιότερο στοιχείο του πίνακα (ανάλογα με την υλοποίηση).

Το πρόβλημα ξανά

- Πρόβλημα : Δεδομένου ενός πίνακα A, ζητείται η αναδιάταξη των στοιχείων (ακεραίων) $A[1], \ldots, A[n]$ σε αύξουσα σειρά.
- Ιδέα : Τοποθέτηση ενός στοιχείου στη *σωστή θέση* : στη θέση που θα βρίσκεται στην ταξινομημένη σειρά. Αναδρομική κλήση με τα στοιχεία του πίνακα που βρίσκονται αριστερότερα και δεξιότερα της σωστής θέσης.
- Ποιο στοιχείο θα τοποθετηθεί στη σωστή θέση ; το δεξιότερο στοιχείο του πίνακα (ανάλογα με την υλοποίηση).
- Ποια είναι η σωστή θέση ;

Το πρόβλημα ξανά

- Πρόβλημα: Δεδομένου ενός πίνακα A, ζητείται η αναδιάταξη των στοιχείων (ακεραίων) $A[1], \ldots, A[n]$ σε αύξουσα σειρά.
- Ιδέα: Τοποθέτηση ενός στοιχείου στη *σωστή θέση*: στη θέση που θα βρίσκεται στην ταξινομημένη σειρά. Αναδρομική κλήση με τα στοιχεία του πίνακα που βρίσκονται αριστερότερα και δεξιότερα της σωστής θέσης.
- Ποιο στοιχείο θα τοποθετηθεί στη σωστή θέση;
  το δεξιότερο στοιχείο του πίνακα (ανάλογα με την υλοποίηση).
- Ποια είναι η σωστή θέση;
  αν η σωστή θέση είναι η θέση pos τότε

$$A[i] \leq A[pos], \forall 1 \leq i < pos,$$
$$A[pos] \leq A[j], \forall n \geq j > pos.$$

Σωστή Θέση

```
int pos(int A[], int lb, int ub)
i ← lb; j ← ub;
while i < j do
    while A[i] ≤ A[ub] do
        i++;
        if i = j then
            break;
        end if
    end while
    if i < j then
        while A[j] ≥ A[ub] do
            j--;
            if i = j then
                break;
            end if
        end while
    end if
    swap(A[i], A[j]);
end while
swap(A[i], A[ub]);
return i
```

Αλγόριθμος

void quicksort($int\ A[], int\ lb, int\ ub$)
if $lb < ub$ **then**
  $i \leftarrow \mathrm{pos}(A, lb, ub)$;
  quicksort($A, lb, i-1$);
  quicksort($A, i+1, ub$);
end if

Ο παραπάνω αλγόριθμος καλείται αρχικώς σαν

$$\text{quicksort}\ (A, 1, n)$$

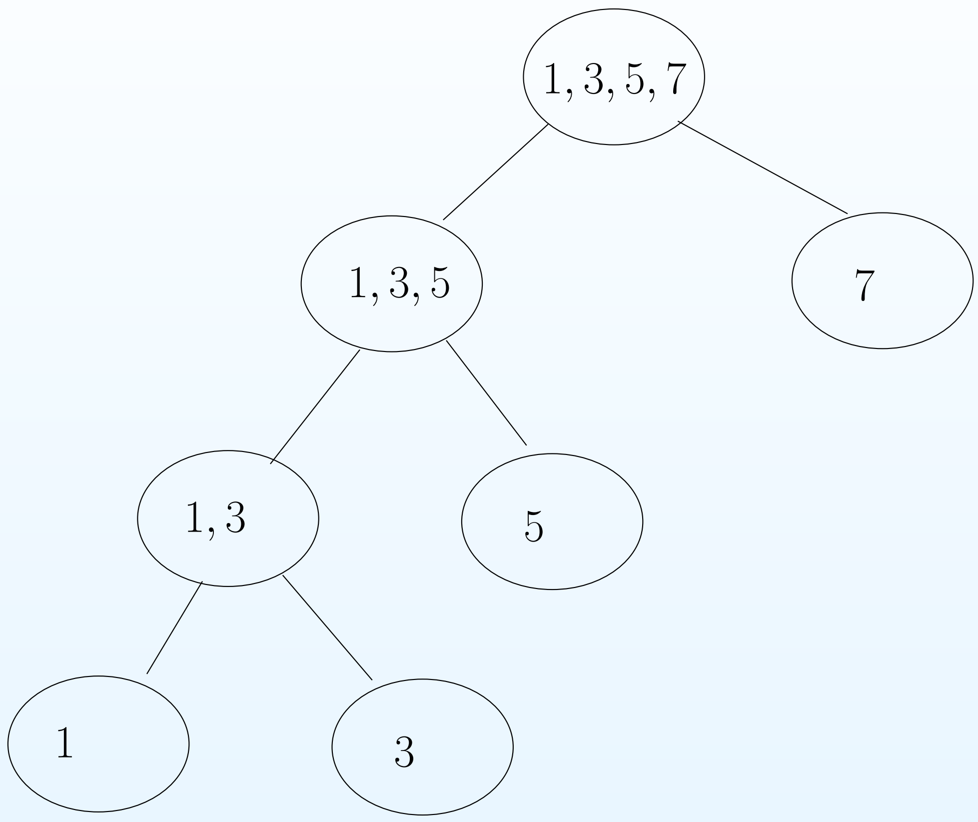
Παράδειγμα

Να ταξινομηθεί η σειρά

$$1 \quad 3 \quad 5 \quad 7$$

Το δένδρο της αναδρομής απεικονίζεται στη συνέχεια.

Παράδειγμα



Πολυπλοκότητα

- Η διαδικασία pos εκτελεί γραμμικό αριθμό στοιχειωδών πράξεων σε σχέση με το πλήθος των αριθμών, ήτοι $\Theta(n)$.
- Σε κάθε επίπεδο, στη χειρότερη περίπτωση καταλήγουμε με ένα πρόβλημα ταξινόμησης το οποίο περιέχει ένα στοιχείο λιγότερο από το πρόβλημα του πρόγονου κόμβου.

- Το δένδρο αναδρομής είναι δυαδικό και επομένως έχει ύψος το πολύ $n-1$.

Επομένως ο αριθμός των στοιχειωδών πράξεων δίνεται από τη σχέση

$$T(n) \begin{cases} \leq T(n-1) + \Theta(n), & n > 1, \\ = 1, & n = 1. \end{cases}$$

Πολυπλοκότητα (συνεχ.)

Με διαδοχική (επαναληπτική) αντικατάσταση έχουμε

$$\begin{aligned} T(n) &\le T(n-1) + cn \le T(n-2) + c(n-1) + cn \\ &\le \dots \\ &\le T(1) + c\sum_{j=2}^{n} j = c\sum_{j=1}^{n} j \Rightarrow \end{aligned}$$

$$T(n) \le c\frac{n(n+1)}{2} \Rightarrow$$

$$T(n) = O(n^2).$$

Εισαγωγή

Οι δύο βασικές αριθμητικές πράξεις είναι η *πρόσθεση* και ο *πολλαπλασιασμός*. Μέχρι τώρα θεωρούσαμε ότι οι πράξεις αυτές είναι από άποψη χρόνου ισοδύναμες: μετράγαμε σαν μία στοιχειώδη πράξη τόσο την πρόσθεση δύο ακεραίων όσο και τον υπολογισμό του γινομένου τους.

Έστω x, y δύο ακέραιοι με n ψηφία ο καθένας ($n \in \mathbb{N}$). Με μία προσεκτικότερη ματιά προκύπτει εύκολα ότι το άθροισμα τους απαιτεί τάξης $O(n)$ προσθέσεις μονοψήφιων ακεραίων. Όμως ο υπολογισμός του γινομένου τους απαιτεί τάξης $O(n^2)$ πολλαπλασιασμούς μονοψήφιων ακεραίων.

Συμπέρασμα: Ο πολλαπλασιασμός ακεραίων είναι πιο "ακριβή" πράξη από την πρόσθεση.

Παράδειγμα

Το γινόμενο $x \cdot y$ απαιτεί n^2 πολλαπλασιασμούς προκειμένου να υπολογιστεί.

Παράδειγμα 1 *Θεωρήστε ότι* $x = 81, y = 26$ *και άρα* $n = 2$. *Τότε*

$$
\begin{array}{ccccc}
 & & & 8 & 1 \\
\times & & & 2 & 6 \\
\hline
 & & 4 & 8 & 6 \\
 & 1 & 6 & 2 & \\
\hline
 & 2 & 1 & 0 & 6
\end{array}
$$

Για να υπολογιστεί ο αριθμός 486 *εκτελέστηκαν* 2 *πολλαπλασιασμοί* $(6 \cdot 1, 6 \cdot 8)$. *Αντίστοιχα,* 2 *πολλαπλασιασμοί εκτελέστηκαν για τον υπολογισμό του* 162. *Άρα σύνολο* $n^2 = 2^2 = 4$ *πολλαπλασιασμοί.*

Το πρόβλημα

Μπορούμε να υπολογίσουμε το γινόμενο δύο $n-$ψήφιων ακεραίων με $o(n^2)$ πολλαπλασιασμούς;

Το πρόβλημα

Μπορούμε να υπολογίσουμε το γινόμενο δύο n−ψήφιων ακεραίων με $o(n^2)$ πολλαπλασιασμούς;

Ναι - με τη χρήση της τεχνικής διαίρει και βασίλευε.

Θεμελίωση

Χωρίς μεγάλη βλάβη της γενικότητας θεωρούμε ότι $n = 2^r, r \in \mathbb{N}$.

Χωρίζουμε τον $n-$ψήφιο ακέραιο x σε δύο τμήματα έστω X^L, x^R όπου το πρώτο περιλαμβάνει τα πρώτα $n/2$ ψηφία του ενώ το δεύτερο τα υπόλοιπα $n/2$. Μπορούμε να γράψουμε τον x σαν

$$x = x^L \cdot 10^{\frac{n}{2}} + x^R.$$

Ομοίως

$$y = y^L \cdot 10^{\frac{n}{2}} + y^R.$$

Επομένως,

$$\begin{aligned} x \cdot y &= (x^L \cdot 10^{\frac{n}{2}} + x^R) \cdot (y^L \cdot 10^{\frac{n}{2}} + y^R) \\ &= 10^n (x^L \cdot y^L) + (x^R \cdot y^R) + 10^{\frac{n}{2}} (x^L \cdot y^R + x^R \cdot y^L). \quad (1) \end{aligned}$$

Θεμελίωση (συνέχ.)

Επεξεργαζόμαστε την τελευταία παρένθεση στην δεξία πλευρά της εξίσωσης (1) ως εξής:

$$
\begin{aligned}
x^L \cdot y^R + x^R \cdot y^L &= x^L \cdot y^R + x^R \cdot y^L + x^L \cdot y^L + x^R \cdot y^R \\
&\quad - (x^L \cdot y^L + x^R \cdot y^R) \\
&= (x^L + x^R) \cdot (y^L + y^R) - (x^L \cdot y^L + x^R \cdot y^R).
\end{aligned}
$$

Αντικαθιστώντας στην (1) έχουμε

$$
\begin{aligned}
x \cdot y = {} & 10^n (x^L \cdot y^L) + (x^R \cdot y^R) \\
& + 10^{\frac{n}{2}} [(x^L + x^R) \cdot (y^L + y^R) - (x^L \cdot y^L + x^R \cdot y^R)]. \qquad (2)
\end{aligned}
$$

Από την (2) είναι φανερό ότι ο υπολογισμός του γινομένου $x \cdot y$ απαιτεί τρεις πολλαπλασιασμούς (γιατί;). Ο κάθε πολλαπλασιασμός μπορεί να γίνει πάλι με τον ίδιο τρόπο μέχρι να έχουμε μονοψήφιους αριθμούς.

Παράδειγμα (συνέχ.)

Στο Παράδειγμα 15, έχουμε

$$x^L = 8, x^R = 1, y^L = 2, y^R = 6,$$
$$x^L + x^R = 9, y^L + y^R = 8.$$

Επομένως, η (2) γίνεται

$$\begin{aligned} 10^2(8 \cdot 2) + (1 \cdot 6) + 10[(8+1) \cdot (2+6) - (8 \cdot 2) - (1 \cdot 6)] \\ = 1600 + 6 + 10[72 - 16 - 6] \\ = 1600 + 6 + 500 \\ = 2106. \end{aligned}$$

Ο αλγόριθμος

Στην περιγραφή του αλγόριθμου κάνουμε τις παρακάτω συμβάσεις:

- $[a]^k$: τα πρώτα k ψηφία του ακέραιου a,
- $[a]_k$: τα τελευταία k ψηφία του ακέραιου
- ο αριθμός των ψηφίων ενός ακεραίου x επιστρέφεται από τη συνάρτηση psifia(x)

int fastmult($int\ x,\ int\ y$)
if psifia(x) $= 1$ **then**
  return $x \cdot y$;
end if
$x^L \leftarrow [x]^{n/2}$; $x^R \leftarrow [x]_{n/2}$;
$y^L \leftarrow [y]^{n/2}$; $y^R \leftarrow [y]_{n/2}$;
$p_1 \leftarrow \text{fastmult}(x^L, y^L)$;
$p_2 \leftarrow \text{fastmult}(x^R, y^R)$;
$p_3 \leftarrow \text{fastmult}(x^L + x^R, y^L + y^R)$;
return $10^n p_1 + p_2 + 10^{n/2}(p_3 - p_1 - p_2)$;

Ανάλυση

Σε κάθε κλήση ο αριθμός των στοιχειωδών πράξεων είναι τάξης $O(n)$. Ο αλγόριθμος σε κάθε κλίση συνδυάζει τρία υποπροβλήματα μεγέθους $n/2$. Ο αριθμός των στοιχειωδών πράξεων περιγράφεται από τη σχέση

$$T(n) = 3T(\frac{n}{2}) + O(n).$$

Ισοδύναμα

$$T(n) \leq T'(n), \qquad (3)$$

όπου

$$T'(n) = \begin{cases} 3T'(n/2) + cn, & n \geq 2, \\ 2 & n \leq 1. \end{cases}$$

Ανάλυση(συνεχ.)

Η ασυμπτωτική συμπεριφορά της $T'(n)$ μπορεί να υπολογιστεί από το κεντρικό θεώρημα. Έχουμε

$$a = 3, b = 2, f(n) = cn.$$

Παρατηρούμε ότι

$$f(n) = cn \leq cn^{\lg 3 - \epsilon}$$

για $\lg \frac{3}{2} \geq \epsilon \geq 0$. Από την πρώτη περίπτωση του κεντρικού θεωρήματος, έχουμε ότι

$$T'(n) = \Theta(n^{\lg 3}). \tag{4}$$

Από (3),(4) συνεπάγεται ότι

$$T(n) = O(n^{\lg 3}).$$

Παρατηρήσεις

Η υλοποίηση του αλγόριθμου με βάση την παραπάνω περιγραφή χρήζει προσοχής στον υπολογισμό του p_3. Τα αθροίσματα $(x^L + x^R)$, $(y^L + y^R)$ μπορεί να μην έχουν τον ίδιο αριθμό ψηφίων. Προκειμένου να ξεπεραστεί το πρόβλημα θα πρέπει να γίνει ‘‘ευθυγράμμιση’’ προσθέτοντας στο άθροισμα με το μικρότερο αριθμό ψηφίων ένα ‘‘προπορευόμενο’’ μηδενικό (γιατί ;).