Πανεπιστήμιο Ιωαννίνων – Τμήμα Μηχανικών Η/Υ και Πληροφορικής
Δομές Δεδομένων [ΠΛΥ302] – Χειμερινό Εξάμηνο 2013

Λυμένες Ασκήσεις – Σετ Α: Ανάλυση Αλγορίθμων

Άσκηση 1

Πραγματοποιήσαμε μια σειρά μετρήσεων του χρόνου εκτέλεσης τριών αλγόριθμων και λάβαμε τα παρακάτω αποτελέσματα:

αλγόριθμος	μέγεθος εισόδου N		
	512	1024	2048
Α	226	320	452
Β	54	208	836
Γ	356	621	1214

Οι ασυμπτωτικοί χρόνοι εκτέλεσης αυτών των αλγόριθμων είναι κατά φθίνουσα σειρά $O(N^2)$, $O(N)$ και $O(\sqrt{N})$. Αντιστοιχήστε τον κάθε αλγόριθμο με την καταλληλότερη από τις παραπάνω πολυπλοκότητες. Δικαιολογείστε τις απαντήσεις σας.

Απάντηση

Έστω $A(N)$, $B(N)$ και $\Gamma(N)$ οι χρόνοι εκτέλεσης των αλγορίθμων Α, Β και Γ αντίστοιχα. Υπολογίζουμε τον ρυθμό αύξησης του κάθε χρόνου εκτέλεσης καθώς διπλασιάζεται το N και τον συγκρίνουμε με την αντίστοιχη αύξηση των συναρτήσεων N^2, N και $\sqrt{N}$.

Ισχύει

$$\frac{(2N)^2}{N^2} = 4, \qquad \frac{2N}{N} = 2, \qquad \frac{\sqrt{2N}}{\sqrt{N}} = \sqrt{2} = 1.41$$

Για $N = 512$ έχουμε

$$\frac{A(2N)}{A(N)} = 1.42, \qquad \frac{B(2N)}{B(N)} = 3.85, \qquad \frac{\Gamma(2N)}{\Gamma(N)} = 1.74$$

ενώ για $N = 1024$ έχουμε

$$\frac{A(2N)}{A(N)} = 1.41, \qquad \frac{B(2N)}{B(N)} = 4.02, \qquad \frac{\Gamma(2N)}{\Gamma(N)} = 1.95$$

Αντιστοιχώντας τους χρόνους $A(N)$, $B(N)$ και $\Gamma(N)$ στις συναρτήσεις που παρουσιάζουν παρόμοια αύξηση λαμβάνουμε $A(N) = O(\sqrt{N})$, $B(N) = O(N^2)$ και $\Gamma(N) = O(N)$.

Άσκηση 2

Διατάξτε τις παρακάτω συναρτήσεις ως προς το ρυθμό αύξησης, από τη μικρότερη ως τη μεγαλύτερη.

$$f_1(n) = n!\,,\ f_2(n) = 2^{2^n},\ f_3(n) = n,\ f_4(n) = \frac{1000}{\lg n} \text{ και } f_5(n) = 10^n$$

Απάντηση

Έχουμε $f_1(n) = n! = 1 \cdot 2 \cdot 3 \cdots (n-1) \cdot n \leq n^n$, για $n \geq 1$.

Επιπλέον

$$n^n = 2^{\lg n^n} = 2^{\,n \lg n} = o(2^{2^n})$$

επειδή

$$\lim_{n\to\infty} \frac{n \lg n}{2^n} = 0$$

Άρα $f_1(n) = o(f_2(n))$.

Για $n \geq 1$ έχουμε

$$f_1(n) = n! = 1 \cdot 2 \cdot 3 \cdots (n-1) \cdot n \geq \left(\frac{n}{2}+1\right) \cdot \left(\frac{n}{2}+2\right) \cdots (n-1) \cdot n \geq \left(\frac{n}{2}\right)^{\frac{n}{2}} = 10^{\log\left(\frac{n}{2}\right)^{\frac{n}{2}}} = 10^{\frac{n}{2}\log\frac{n}{2}}$$

άρα $f_5(n) = o(f_1(n))$, επειδή

$$\lim_{n\to\infty} \frac{n}{\frac{n}{2}\log\frac{n}{2}} = 0$$

Για τις f_4 και f_3 έχουμε

$$\lim_{n\to\infty} \frac{f_4(n)}{f_3(n)} = \lim_{n\to\infty} \frac{1000}{n \lg n} = 0$$

Δηλαδή $f_4(n) = o(f_3(n))$.

Τέλος

$$\lim_{n\to\infty} \frac{f_3(n)}{f_5(n)} = \lim_{n\to\infty} \frac{n}{10^n} = 0$$

Δηλαδή $f_3(n) = o(f_5(n))$.

Άρα η διάταξη από το μικρότερο προς το μεγαλύτερο ρυθμό αύξησης είναι f_4, f_3, f_5, f_1, f_2.

Άσκηση 3

Λύστε την παρακάτω αναδρομική σχέση με την μέθοδο της αντικατάστασης.

$$T(n) = 9\,T\left(n/3\right) + n^2, n \geq 2$$

με αρχική συνθήκη $T(1) = 1$.

Απάντηση

Υποθέτουμε ότι το n είναι δύναμη του 3, δηλαδή $n = 3^k$ για κάποιο θετικό ακέραιο k. Έχουμε

$$\begin{aligned}
T(n) = 9\,T\left(n/3\right) + n^2 = \\
&= 9\left[9\,T\left(n/3^2\right) + \left(\frac{n}{3}\right)^2\right] + n^2 = 9^2 T\left(n/3^2\right) + 9\frac{n^2}{9} + n^2 = 9^2 T\left(n/3^2\right) + 2n^2 \\
&= 9^2\left[9\,T\left(n/3^3\right) + \left(\frac{n}{3^2}\right)^2\right] + 2n^2 = 9^3 T\left(n/3^3\right) + 9^2\frac{n^2}{9^2} + 2n^2 = 9^3 T\left(n/3^3\right) + 3n^2 \\
&= 9^3\left[9\,T\left(n/3^4\right) + \left(\frac{n}{3^3}\right)^2\right] + 3n^2 = 9^4 T\left(n/3^4\right) + 9^3\frac{n^2}{9^3} + 3n^2 = 9^4 T\left(n/3^4\right) + 4n^2
\end{aligned}$$

Μετά από k αντικαταστάσεις λαμβάνουμε

$$T(n) = 9^k T\left(n/3^k\right) + kn^2 = 9^k T(1) + kn^2 = 9^k + kn^2$$

Επιπλέον, $n = 3^k \Rightarrow k = \log_3 n$, άρα

$$T(n) = 9^{\log_3 n} + n^2 \log_3 n = n^2 + n^2 \log_3 n = \Theta(n^2 \lg n)$$

Άσκηση 4

Έστω ότι ο χρόνος εκτέλεσης $T(n)$ ενός αλγορίθμου ικανοποιεί την παρακάτω αναδρομική σχέση

$$T(n) = 2\,T\left(n/2\right) + n\lg n, n > 2$$

με αρχική συνθήκη $T(2) = 2$. Δείξτε με χρήση επαγωγής ότι $T(n) = \Theta(n(\log n)^2)$.

Απάντηση

Θα δείξουμε πρώτα με επαγωγή, ότι υπάρχουν σταθερές $c > 0$ και $n_0 \geq 0$ τέτοιες ώστε $T(n) \leq cn(\lg n)^2$ για κάθε $n \geq n_0$. Για τη βάση της επαγωγής, από την αρχική συνθήκη έχουμε

$$T(2) = 2 \leq c2(\lg 2)^2 = 2c$$

το οποίο ισχύει για $c \geq 1$.

Η επαγωγική υπόθεση είναι ότι για κάθε $k < n$, $T(k) \leq ck(\lg k)^2$. Για το επαγωγικό βήμα πρέπει να δείξουμε ότι $T(n) \leq cn(\lg n)^2$. Από την αναδρομική σχέση και την επαγωγική υπόθεση,

$$T(n) = 2T(n/2) + n\lg n \le 2c\frac{n}{2}\left(\lg\frac{n}{2}\right)^2 + n\lg n = cn(\lg n - \lg 2)^2 + n\lg n$$
$$= cn((\lg n)^2 - 2\lg n + 1) + n\lg n = cn(\lg n)^2 - 2cn\lg n + cn + n\lg n$$
$$\le cn(\lg n)^2 - 2cn\lg n + cn\lg n + cn\lg n = cn(\lg n)^2$$

αφού για $n, c \ge 2$, $cn \le cn\lg n$ και $n\lg n \le cn\lg n$.

Τώρα θα δείξουμε με επαγωγή ότι υπάρχουν σταθερές $c' > 0$ και $n'_0 \ge 0$ τέτοιες ώστε $T(n) \ge c'n(\lg n)^2$ για κάθε $n \ge n'_0$. Για τη βάση της επαγωγής, από την αρχική συνθήκη έχουμε

$$T(2) = 2 \ge c'2(\lg 2)^2 = 2c'$$

το οποίο ισχύει για $c' \le 1$.

Η επαγωγική υπόθεση είναι ότι για κάθε $k < n$, $T(k) \ge c'k(\lg k)^2$. Για το επαγωγικό βήμα πρέπει να δείξουμε ότι $T(n) \ge c'n(\lg n)^2$. Από την αναδρομική σχέση και την επαγωγική υπόθεση,

$$T(n) = 2T(n/2) + n\lg n \ge 2c'\frac{n}{2}\left(\lg\frac{n}{2}\right)^2 + n\lg n = c'n(\lg n - \lg 2)^2 + n\lg n$$
$$= c'n((\lg n)^2 - 2\lg n + 1) + n\lg n = c'n(\lg n)^2 - 2c'n\lg n + c'n + n\lg n$$
$$\ge c'n(\lg n)^2 - 2c'n\lg n + n\lg n = c'n(\lg n)^2 + (1 - 2c')\lg n \ge c'n(\lg n)^2$$

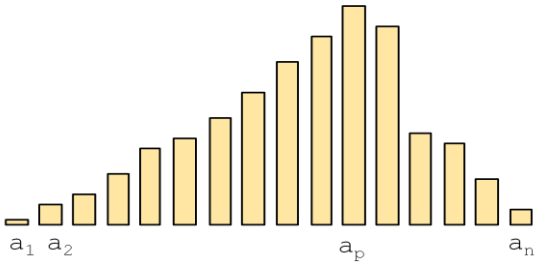
το οποίο ισχύει για $1 - 2c' \ge 0 \Rightarrow c' \le 1/2$.

Άσκηση 5

Μία ακολουθία αριθμών $A = (a_1\ a_2\ \ldots\ a_n)$ ονομάζεται *μονοκόρυφη* εάν για κάποιο p με $1 \le p \le n$, έχουμε $a_1 < a_2 < \ldots < a_p$ και $a_p > a_{p+1} > \ldots > a_n$. Θέλουμε να βρούμε το μέγιστο στοιχείο a_p μιας μονοκόρυφης ακολουθίας, διαβάζοντας όσο το δυνατό λιγότερα στοιχεία της. Σχεδιάστε έναν αποδοτικό αλγόριθμο για αυτό το πρόβλημα και αναλύστε τον ασυμπτωτικό χρόνο εκτέλεσης του. Υποθέστε ότι η ακολουθία είναι αποθηκευμένη σε έναν πίνακα, οπότε η προσπέλαση ενός στοιχείου της γίνεται σε $O(1)$ χρόνο.

Απάντηση

Μια τέτοια ακολουθία έχει τη μορφή του παρακάτω σχήματος:



Μπορούμε να βρούμε το μέγιστο στοιχείο a_p σε $p = O(n)$ βήματα με ακολουθιακή αναζήτηση, δηλαδή ξεκινάμε από τη θέση $k = 1$ και αυξάνουμε το k κατά 1 εάν $a_k < a_{k+1}$, διαφορετικά έχουμε $a_k = a_p$. Για να πετύχουμε καλύτερο χρόνο χρησιμοποιούμε δυαδική αναζήτηση ως εξής. Θέτουμε $m = \lfloor (n+1)/2 \rfloor$ και ελέγχουμε τη σχέση των a_{m-1}, a_m και a_{m+1}. Υπάρχουν τρεις περιπτώσεις:

α) $a_{m-1} < a_m$ και $a_m > a_{m+1}$. Τότε το a_m είναι το μέγιστο στοιχείο.

β) $a_{m-1} < a_m < a_{m+1}$. Τότε το μέγιστο στοιχείο βρίσκεται στην ακολουθία $A_{>m} = (a_{m+1}\ a_{m+2}\ \dots\ a_n)$, οπότε επαναλαμβάνουμε την ίδια διαδικασία για την $A_{>m}$.

γ) $a_{m-1} > a_m > a_{m+1}$. Τότε το μέγιστο στοιχείο βρίσκεται στην ακολουθία $A_{<m} = (a_1\ a_2\ \dots\ a_{m-1})$, οπότε επαναλαμβάνουμε την ίδια διαδικασία για την $A_{<m}$.

Ο χρόνος εκτέλεσης του παραπάνω αλγορίθμου είναι ασυμπτωτικά ίσος με το χρόνο δυαδικής αναζήτησης, δηλαδή $O(\log n)$.

Άσκηση 6

Η παρακάτω μέθοδος υπολογίζει τα αθροίσματα $A[i] + A[i+1] + \cdots + A[j]$ για όλα τα i και j τέτοια ώστε $0 \le i \le j \le N-1$, όπου A ένας μονοδιάστατος πίνακας N ακέραιων. Τα αθροίσματα αποθηκεύονται σε ένα διδιάστατο $N \times N$ πίνακα ακεραίων B. (Υποθέτουμε ότι κάθε $B[i][j]$ έχει ήδη αρχικοποιηθεί με την τιμή 0.)

```
void partialSums(int[] A, int[][] B)
{
   int N = A.length;
   for (int i=0; i<=N-1; i++)
      for (int j=i; j<=N-1; j++)
         for (int k=i; k<=j; k++)
            B[i][j] += A[k];
}
```

α) Ποίος είναι ο ασυμπτωτικός χρόνος εκτέλεσης $T(N)$ της `partialSums`;

β) Δώστε ένα βελτιωμένο τρόπο υπολογισμού του πίνακα B, με ασυμπτωτικά καλύτερο χρόνο εκτέλεσης $T'(N)$. Δηλαδή θέλουμε $T'(N) = o(T(N))$.

Απάντηση

α) Ο χρόνος εκτέλεσης της `partialSums` είναι ανάλογος του αριθμού των προσθέσεων που εκτελεί. Για τον υπολογισμό του $B[i][j]$ εκτελεί $(j-i+1)$ προσθέσεις, επομένως κάνει N προσθέσεις για το $B[0][N-1]$, $(N-1)$ προσθέσεις για τα $B[1][N-1]$ και $B[2][N-2]$, $(N-3)$ προσθέσεις για τα $B[0][N-3]$, $B[1][N-2]$ και $B[2][N-1]$ κ.ο.κ. Δηλαδή συνολικά

$$N \cdot 1 + (N-1) \cdot 2 + (N-2) \cdot 3 + \cdots + 1 \cdot N = \sum_{j=1}^{N} (N-j+1) \cdot j =$$

$$= \sum_{j=1}^{N} Nj - \sum_{j=1}^{N} j^2 + \sum_{j=1}^{N} j = \frac{N^2(N+1)}{2} - \frac{N(N+1)(2N+1)}{6} + \frac{N(N+1)}{2} =$$

$$= \frac{N^3 + 3N^2 + 2N}{6} = \Theta(N^3)$$

β) Παρατηρούμε ότι $B[i][i] = A[i]$ και για $i < j$, $B[i][j] = B[i][j-1] + A[j]$. Έτσι λαμβάνουμε την παρακάτω βελτιωμένη μέθοδο

```
void partialSums(int[] A, int[][] B)
{
   int N = A.length;
   for (int i=0; i<=N-1; i++) B[i][i] = A[i];

   for (int i=0; i<=N-1; i++)
           for (j=i+1; j<=N-1; j++)
                       B[i][j] = B[i][j-1]+A[j];
}
```

Ο χρόνος εκτέλεσης είναι ανάλογος του αριθμού των προσθέσεων, που είναι ίσος με τον αριθμό των $B[i][j]$ με $i < j$, δηλαδή

$$(N-1) + (N-2) + \cdots + 2 + 1 = \frac{N(N-1)}{2} = \Theta(N^2)$$