Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή ΗΜ&ΜΥ
Αλγόριθμοι και Πολυπλοκότητα
$7^{ο}$ εξάμηνο, Ροή Λ
Ακαδημαϊκή περίοδος: 2010-2011

$1^{η}$ Σειρά Γραπτών Ασκήσεων

Γερακάρης Βασίλης
Α.Μ.: 03108092

8 Δεκεμβρίου 2011

1 Ασυμπτωτικός συμβολισμός, Αναδρομικές Σχέσεις

i) Ταξινόμηση

(1) $\log n^3 = 3\log n = \Theta(logn)$

(2) $\sqrt{n} * \log^{50} n$

(3) $\frac{n}{\log\log n}$

(4) $\log n! = \Theta(nlogn)$

(5) $n * \log^{10} n$

(6) $n^{1.01}$

(7) $5^{\log_2 n} = n^{log_2 5} = \Theta(n^{2.322})$

(8) $\sum_{k=1}^{n} k^5 = k^6$

(9) $\log^{\log n} n = n^{\log\log n}$

(10) $2^{\log_2^4 n}$

(11) $\log^{\sqrt{n}} n$

(12) $e^{\frac{n}{\ln n}}$

(13) $n3^n$

(14) 2^{2n}

(15) $\sqrt{n!}$

ii) Τάξη Μεγέθους

(1) $T(n) = 5T(n/7) + n\log n$, $(a = 5, b = 7)$
$\Rightarrow n^{\log_7 5} = n^{0.827} \Rightarrow n^{0.827} < n < n\log n$
$\Rightarrow T(n) \in \Theta(n\log n)$ M.Th (case #3)

(2) $T(n) = 4T(n/5) + n/\log^2 n$, $(a = 4, b = 5)$
$\Rightarrow n^{\log_5 4} = n^{0.861} \Rightarrow n^{0.861} < n/\log^2 n$
$\Rightarrow T(n) \in \Theta(n/\log^2 n)$ M.Th (case #3)

(3) $T(n) = T(n/3) + 3T(n/7) + n$ $(\frac{1}{3} + \frac{1}{7} + \frac{1}{7} + \frac{1}{7} < 1)$
$\Rightarrow T(n) \in \Theta(n)$ M.Th (special case)

(4) $T(n) = 6T(n/6) + n$
$\Rightarrow n^{\log_6 6} = n$
$\Rightarrow T(n) \in \Theta(n\log n)$ M.Th (sp. case #2)

(5) $T(n) = T(n/3) + T(2n/3) + n$
$\Rightarrow T(n) \in \Theta(n\log n)$

(6) $T(n) = 16T(n/4) + n^3\log^2 n$, $(a = 16, b = 4)$
$\Rightarrow n^{\log_4 16} = n^2 \Rightarrow n^2 < n^3\log^2 n$
$\Rightarrow T(n) \in \Theta(n^3\log^2 n)$ M.Th (case #3)

(7) $T(n) = T(\sqrt{n}) + \Theta(\log\log n)$ (Θέτω $k = \log n, f(k) = T(n)$)
$\Rightarrow T(\sqrt{n}) = f(\sqrt{k}) = f(\log\sqrt{n}) = f(\frac{\log n}{2}) = f(\frac{k}{2})$
$\Rightarrow f(k) = f(\frac{k}{2}) + \Theta(\log k)$
$\Rightarrow \Theta(n)$ (M.Th (case #1)

(8) $T(n) = T(n-3) + \log n$ $(\frac{n}{3} * \log n)$
$\Rightarrow T(n) \in \Theta(nlogn)$

2 Ταξινόμηση σε Πίνακα με Πολλά Ίδια Στοιχεία

Για να κάνουμε ταξινόμηση σε ένα τέτοιο πίνακα σε πρώτο στάδιο μετράμε τα στοιχεία που είναι ίδια σε κάθε πέρασμα και τα βγάζουμε από τον πίνακα. Για να ολοκληρώσουμε αυτή τη διαδικασία και με δεδομένο πως το πλήθος των διαφορετικών στοιχείων είναι μόλις $\log^d n$ προκύπτει ο αναδρομικός τύπος $T(n) = T(n * \frac{m-1}{m}) + f(n), f(n) \in O(n), m = \log^d n$. Επιλύοντας με χρήση του *Master Theorem* καταλήγουμε σε πολυπλοκότητα $O(n)$ για το πρώτο βήμα. Έπειτα με mergesort ταξινομούμε τα στοιχεία που έχουμε και τα επεκτείνουμε ανάλογα με το πλήθος του καθενός έτσι ώστε το τελικό αποτέλεσμα να είναι και πάλι μήκους n. Η διαδικασία ολοκληρώνεται σε άλλο $\log^d n * \log\log^d n = d * \log^d n * \log\log n$. Προφανώς το κύριο κομμάτι του χρόνου καταναλώνεται στο δεύτερο στάδιο. Συνεπώς η πολυπλοκότητα του αλγορίθμου είναι $\Theta(\log^d n * \log\log n)$ άρα και $O(n * \log\log n)$.

3 Δυαδική Αναζήτηση

i) Ξεκινάμε (για $i = 1$) συγκρίνοντας το $A[i]$ με το x. Όσο το x είναι μεγαλύτερος από αυτό, κρατάμε την τιμή του $(i+1)$ σε μια άλλη μεταβλητή k και διπλασιάζουμε το i. Μόλις φτάσουμε σε μεγαλύτερο αριθμό από τον x, εφαρμόζω ένα έλεγχο. Αν το στοιχείο $A[i]$ είναι διάφορο του ∞, τότε εφαρμόζω δυαδική αναζήτηση για το στοιχείο x στο τμήμα $[k..i]$ του πίνακα και έχω το αποτέλεσμα για πιθανή ύπαρξη & θέση του στοιχείου.
Αν το στοιχείο $A[i]$ είναι ίσο με ∞, τότε ελέγχω αν $k = i$ (τερματική συνθήκη) και αν όχι ελέγχω το $A[(k+i)div2]$.

- Αν είναι ∞ τότε θέτω $i = (k+i)div2$ και επαναλαβάνω.
- Αν είναι αριθμός m τον συγκρίνω με το x.
  - Αν $x > m$ τότε θέτω $k = (k+i)div2$ και επαναλαμβάνω.
  - Αν $x < m$ τότε εφαρμόζω δυαδική αναζήτηση στο τμήμα $A[k..i]$ του πίνακα.
  - Αν $x = m$ τότε είναι το ζητούμενο στοιχείο.

Η τελική πολυπλοκότητα είναι $O(\log n)$.

ii) Εργαζόμαστε με τα k πρώτα στοιχεία από κάθε πίνακα (συμπληρώνοντας τις κενές θέσεις με + ∞ , αν ο πίνακας δεν έχει τόσα στοιχεία). Συγκρίνουμε τα $A[k/2]$ με $B[k/2+1]$ και $A[k/2+1]$ με $B[k/2]$.

- Αν $A[k/2] < B[k/2+1]$ και $B[k/2] < A[k/2+1]$ τοτε το k-οστό στοιχείο είναι το $max\{A[k/2], B[k/2]\}$.
- Αν $A[k/2] < B[k/2+1]$ και $B[k/2] > A[k/2+1]$ τότε επαναλαμβάνουμε αναδρομικά τη διαδικασία στους υποπίνακες $A[k/2+1]..A[k]$ και $B[1]..B[k/2]$.

4 Συλλογή Comics

Ελέγχουμε αρχικά το 1^o bit για όλα τα τεύχη, χωρίζοντάς τα σε 2 σύνολα με πλήθος στοιχείων $\frac{n}{2}$ και $\frac{n}{2}-1$. Το τεύχος που λείπει είναι στο σύνολο με τα λιγότερα στοιχεία, οπότε αποθηκεύουμε την τιμή αυτού του bit.
Εφαρμόζουμε αναδρομικά τη μέθοδο για το επόμενο bit, κάθε φορά στο σύνολο με τα λιγότερα στοιχεία, κάνοντας τις μισές ερωτήσεις από το προηγούμενο βήμα σε κάθε αναδρομή.
Τελικά, θα καταλήξουμε στον κωδικό του τεύχους που λείπει από τη συλλογή, έχοντας ρωτήσει $n + \frac{n}{2} + \frac{n}{4} + .. + 1 \approx 2n$ φορές.

Πρακτικά, είναι: $(n-1) + (\frac{n}{2}-1) + (\frac{n}{4}-1) + .. = \sum_{i=0}^{\log n}(\frac{n}{2^i}-1) = 2n - logn - 2$

5 Πολυκατοικίες χωρίς Θέα

Αρχίζουμε αρχικοποιώντας $B[1] = 0$, μιάς και ο $1^{ος}$ δεν έχει κτήριο δυτικά του.
Για κάθε κτήριο i ανατολικά του, θέτουμε $k = i - 1$ και συγκρίνουμε το ύψος του $A[i]$ με του $A[k]$.

- Αν $A[i] < A[k]$, τότε $B[i] = k$ και συνεχίζουμε στο επόμενο δεξιά κτήριο ($i = i + 1$) .
- Αν $A[i] \geq A[k]$,
  - Αν $B[k] = 0$ τότε $B[i] = 0$ και συνεχίζουμε στο επόμενο δεξιά κτήριο.
  - Αλλιώς θέτουμε $k = B[k]$ και επαναλαμβάνουμε τον έλεγχο.