1η Σειρά Γραπτών Ασκήσεων

Αλγόριθμοι και Πολυπλοκότητα

ΣΗΜΜΥ, Εθνικό Μετσόβιο Πολυτεχνείο

Άσκηση 1α): Ασυμπτωτικός Συμβολισμός, Αναδρομικές Σχέσεις

1. $\sum_{k=1}^{n} k2^{-k}$
2. $\log(\binom{n}{\log n})$
3. $\log(10n!)/(\log n)^9$
4. $\log\left(\binom{2n}{n}\right) = \Theta(n) = n3^{4^{5^6}}$
5. $\sum_{k=1}^{n} k^3 = \Theta(n^4)$
6. $n^5/\log^{10} n$
7. $(\log_2(3n))^{\log_2(7n)}$
8. $2^{(\log_2 n)^3} = \Theta(n^{\log^2 n})$
9. $\sum_{k=1}^{n} k2^k = \Theta(n2^n) = n\sum_{k=1}^{n}\binom{n}{k}$
10. $\binom{2n}{n/4}$
11. $\sqrt{n!}$

Άσκηση 1β): Ασυμπτωτικός Συμβολισμός, Αναδρομικές Σχέσεις

1. $T(n) = 3T(n/4) + n\log\log^7 n = \Theta(n\log\log^7 n)$
   Μ.Τ. περίπτωση 3
2. $T(n) = 4T(n/4) + n\log^5 n = \Theta(n\log^6 n)$
   Όχι Μ.Τ. διότι δεν είναι πολυωνυμικά διαχωρίσιμες. Επειδή $4n/4 = n$ και $f(n) = n\log^5 n$ υποπτευόμαστε ότι $T(n) = \Theta(n\log^6 n)$ και το δείχνουμε με επαγωγή ή με δένδρο αναδρομής.
3. $T(n) = 9T(n/5) + n\log^{14} n = \Theta(n^{\log_5 9})$
   Μ.Τ. περίπτωση 1
4. $T(n) = T(n-1) + \log n = \Theta(n\log n)$
   $T(n) = T(1) + \log 2 + \ldots + \log n = \Theta(1) + \log n!$
5. $T(n) = 2T(\sqrt{n}) + \Theta(\log n) = \Theta(\log\log n\log n)$
   Θέτουμε $m = \log n$ και εργαζόμαστε με τη σχέση

Άσκηση 2: Σωροί

- α) Ενώνουμε τους 2 σωρούς σε ενα σε χρόνο $\Theta(n)$.
- β) Προσθέτουμε απο το σωρό με $o(n)$ στοιχεία, στο σωρο με $\Theta(n)$ στοιχεία. Συνολικά, $o(n\log n)$.
  Αν και τα δύο έχουν $\Theta(n)$ στοιχεία τότε, $\Theta(n\log n)$.

Άσκηση 3α): Ταξινόμηση

Θέλουμε να ταξινομήσουμε τον πίνακα $A[1, \ldots, n]$ με μέγιστο στοιχείο M=$O(n)$ σε γραμμικό χρόνο.

- Θεωρούμε πίνακα $B[0, \ldots, M]$ και αρχικοποιούμε τα στοιχεία του στο 0.
- Διατρέχουμε τον πίνακα A μια φορά και για κάθε στοιχείο $A[i] = j \leq M$, προσθέτουμε στο B[j] συν 1.
- Ξεκινάμε με τον κενό πίνακα C.
- Διατρέχουμε τον πίνακα και για κάθε $B[j] = i$, βάζουμε στο τέλος του πίνακα C, i φορες το στοιχειο $B[j]$.
- Επιστρέφουμε τον πίνακα C.

Άσκηση 3α): Ταξινόμηση

- Συνολικά, $\Theta(n+M) = \Theta(n)$.
- Το κάτω φράγμα $\Omega(n\log n)$ ισχύει μόνο στην περίπτωση των συγκριτικών αλγορίθμων, δηλαδή αλγορίθμων που κάνουν μόνο συγκρίσεις μεταξύ των στοιχείων του πίνακα.
  Ο προηγούμενος αλγόριθμος δεν εμπίπτει σε αυτή την κατηγορία, καθως κάνει χρήση της αριθμητικής τιμής του κάθε στοιχείου.

Άσκηση 3β): Ταξινόμηση

- Εφόσον $M = n^d - 1$, κάθε στοιχείο του A περιγράφεται με d bits, $A[i] = b_1, \ldots, b_d$ όπου $b_k \in \{0, n-1\}$.
- Θα εφαρμόσουμε τον αλγόριθμο ταξινόμησης *RadixSort*.
- Συνολικά διατρέχουμε τον A, d φορές και σε κάθε επανάληψη πληρώνουμε $O(n)$.
- Συνολική Πολυπλοκότητα, $O(dn) = O(n)$, γιατί d είναι σταθερά.

Άσκηση 3γ): Ταξινόμηση

Θέλουμε να υπολογίσουμε το πλήθος των διαφορετικών ταξινομημένων ακολουθιών μήκους n, με στοιχεία που παίρνουν τιμές στο $\{1, \ldots, M\}$

- Για να προσδιορίσουμε μια τέτοια ακολουθία, αρκεί να προσδιορίσουμε το πλήθος στοιχείων της με τιμές 1,2,...,M.
- Έστω A_k το το πλήθος στοιχείων με τιμή k.
- Αρκεί να υπολογίσουμε με πόσους τρόπους μπορουμε να αναθέσουμε τιμές από το 0 στο n, στους αριθμούς $A_1, \ldots, A_M$ έτσι ώστε $A_1 + \ldots + A_M = n$
- Το παραπάνω μπορεί να γίνει με $\binom{n+M-1}{n}$ τρόπους.
- Άρα κάθε συγκριτικός αλγόριθμος χρειάζεται τουλάχιστον $\Omega(\log\binom{n+M-1}{n}) = \Omega(n \cdot \log M)$ βήματα.

Άσκηση 3γ): Ταξινόμηση

- Για να υπάρχει συγκριτικός αλγόριθμός με χρόνο εκτέλεσης $O(n\log\log n)$, θα πρέπει το $M = O(\log^d n)$ για σταθερά d.
- Αν γνωρίζουμε το μέγιστο στοιχείο του A είναι , μπορούμε να παράξουμε ενα συγκριτικό αλγόριθμο με πολυπλοκότητα $O(n\log M)$?
- Θα λυσουμε ένα λίγο πιο γενικό πρόβλημα: Δεδομένου ότι ο A περιέχει M διαφορετικά στοιχεία, θα φτιάξουμε ένα ενα συγκριτικό αλγόριθμο με πολυπλοκότητα $O(n\log M + M\log n)$.

Άσκηση 3γ): Ταξινόμηση

- Ιδέα: Έστω ότι έχουμε ενα υποπίνακα A' του A, ο οποίος περιέχει M διαφορετικά στοιχεία του A σε αύξουσα σειρά.
- Διασχίζουμε τον A και για κάθε στοιχείο που συναντάμε, κάνουμε δυαδική αναζήτηση στον A' για να βρούμε με ποιο στοιχείο του A' είναι ίσο.
- Αυξάνουμε ενα Counter σε εκείνο το σημείο και βάσει αυτού μπορούμε να παράξουμε μια ταξινόμηση για τον A.
- Πολυπλοκότητα: $O(n \log M)$.

Άσκηση 3γ): Ταξινόμηση

Μας μένει να υπολογίσουμε τον A'. Θα χρησιμοποιήσουμε μια παραλλαγή της Mergesort.

- Χωρίζουμε τον A στην μέση και παίρνουμε τους πίνακες A_1 και A_2.
- Αναδρομικά υπολογίζουμε τους A'_1 και A'_2, καθένας τους περιέχει το πολύ M στοιχεία.
- Συγχωνεύουμε τους A'_1 και A'_2 με την διαφορά οτι αν συναντησουμε 2 ίσα στοιχεία, πετάμε το ένα.
- Ο πίνακας που προκύπτει είναι ο A'.
- Πολυπλοκότητα:
  $T(n) = 2T(n/2) + \Theta(M) \Longrightarrow T(n) = O(M\log n)$.

Άσκηση 3γ): Ταξινόμηση

- Συνολική Πολυπλοκότητα: $O(n\log M + M\log n)$.
- Στην περίπτωση μας,
  $O(n\log\log^d n) = O(d \cdot n\log\log n) = O(n\log\log n)$.

Άσκηση 4α): Αναζήτηση

Χαρακτηρίζουμε κάθε θέση του πίνακα ως εξής

$$j \in H \Leftrightarrow A[j] - j > \frac{k+1}{2}, \quad j \in L \Leftrightarrow A[j] - j < -\frac{k+1}{2},$$

$$j \in X \Leftrightarrow |A[j] - j| \leq \frac{k+1}{2}$$

Υποθέτουμε ότι $C = \emptyset$. Τότε έχουμε ότι $0 \in A$ και $n \in B$ επειδή $1 \leq A[j] \leq n$.

Άσκηση 4α): Αναζήτηση

Αφού δεν ανήκουν όλα τα στοιχεία **μόνο** στο H ή **μόνο** στο L υπάρχουν διαδοχικές θέσεις ώστε $j \in H$ και $j+1 \in L$. Έχουμε λοιπόν

$$A[j] - j > \frac{k+1}{2} \Leftrightarrow A[j] - j - \frac{k+1}{2} > 0 \quad (1)$$

$$A[j+1] - j - 1 < \frac{k+1}{2} \Leftrightarrow \frac{k+1}{2} - A[j+1] + j + 1 > 0 \quad (2)$$

Προσθέτοντας κατά μέλη παίρνουμε

$$A[j] - A[j+1] > k$$

Άτοπο.

Άσκηση 4α): Αναζήτηση

Κάθε φορά που έχουμε θέσεις $i < j$ έτσι ώστε $A[i] \in H$ και $A[j] \in L$, γνωρίζουμε ότι υπάρχει $i < k < j$ ώστε $k \in X$. Άρα μπορούμε να κάνουμε binary search.

- Ξεκινάμε με $i = 0$ και $j = n$
- Ελέγχουμε αν $i \in X$ ή $j \in X$.
- Αν δεν είναι τότε βρίσκω σε ποια κατηγορία από τις L, H, X ανήκει ο (ακέραιος) διάμεσος m των i, j.
  - Αν ανήκει στο X τον επιστρέφω.
  - Αν ανήκει στο H τότε συνεχίζω την αναζήτηση στο (i, m).
  - Αν ανήκει στο L τότε συνεχίζω την αναζήτηση στο (m, j).

Άσκηση 4β): Αναζήτηση

Αφού $m \leq n$ μπορούμε να κάνουμε binary search σε κάθε στήλη κάνοντας $O(m \log n)$ συγκρίσεις, αν ο πίνακας είναι τετράγωνος κάνουμε $O(n \log n)$.
Έτσι δεν εκμεταλλευόμαστε το γεγονός ότι ο πίνακας είναι ταξινομημένος και ως προς γραμμές και ως προς στήλες.

- Ξεκινάμε από το πάνω δεξιά στοιχείο $A[m, 1]$ του πίνακα.
- Αν $A[m, 1] = k$ τότε επιστρέφουμε τη θέση $[m, 1]$ και συνεχίζουμε την αναζήτηση στον υποπίνακα $A[2 \ldots n-1, 2 \ldots m-1]$.
- Αν $A[m, 1] < k$ τότε συνεχίζουμε στον υποπίνακα $A[2 \ldots n-1, 1 \ldots m-1]$.
- Αν $A[m, 1] > k$ τότε συνεχίζουμε στον υποπίνακα $A[1 \ldots n-1, 2 \ldots m-1]$.

Συνολικός χρόνος $O(n+m) = O(n)$.

Άσκηση 4γ): Αναζήτηση

Στην περίπτωση $m = n$, αν υπάρχουν r εμφανίσεις του στοιχείου k στον πίνακα κάθε αλγόριθμος για αυτό το πρόβλημα πρέπει να τυπώσει μια ακολουθία από θέσεις $(i_l, j_l)_{l=1}^{r}$. Το πλήθος των δυνατών ακολουθιών για κάθε τιμή του r είναι τουλάχιστον $\binom{n}{r}$ αφού μπορούμε να διαλέξουμε σε ποιες από τις n γραμμές θα τοποθετήσουμε κάθε εμφάνιση του k. Το συνολικό πλήθος των δυνατών ακολουθιών εξόδου είναι

$$\sum_{r=1}^{n} \binom{n}{r} = 2^n$$

Άρα (όμοια με το lower bound της ταξινόμησης) κάθε αλγόριθμος πρέπει να κάνει τουλάχιστον $\log_2(2^n) = n$ συγκρίσεις μέχρι να σταματήσει. Συνεπώς χρειάζονται $\Omega(n)$ συγκρίσεις.

Άσκηση 5: Πλειοψηφία - Λύση 1

Ξεκινάμε με έναν βουλευτή και ξεκινάμε να τον βάζουμε να χαιρετηθεί με τους υπόλοιπους. Αρχικοποιούμε έναν μετρητή στο 1.

- Κάθε φορά που βρίσκουμε πολιτικό του αντίπαλο μειώνουμε τον μετρητή κατά 1, αλλιώς τον αυξάνουμε.
- Αν κάποια στιγμή ο μετρητής γίνει 0 τότε αλλάζουμε τον βουλευτή με τον επόμενό του και θέτουμε τον μετρητή στο 1.

Αν υπάρχει κόμμα με απόλυτη πλειοψηφία, ο βουλευτής που έχουμε στο τέλος ανήκει σε αυτό, οπότε τον βάζουμε να χαιρετηθεί ξανά με όλους και βρίσκουμε και όλους τους υπόλοιπους στο κόμμα του.
Αν τελικά οι βουλευτές προκύψουν λιγότεροι τότε δεν υπάρχει απόλυτη πλειοψηφία και απαντάμε ”ΟΧΙ”, αλλιώς απαντάμε ”ΝΑΙ” και δίνουμε την λίστα των βουλευτών.
Συνολικοί χαιρετισμοί $O(n)$.

Άσκηση 5: Πλειοψηφία - Λύση 2

Διαλέγω έναν βουλευτή στην τύχη, τον βάζω να χαιρετηθεί με όλους τους άλλους και μετράω πόσοι και ποιοι ανήκουν στο κόμμα του. Αν το πλήθος των βουλευτών στο κόμμα του πρώτου είναι τουλάχιστον $n/2 + 1$ τότε απαντάω ”ΝΑΙ” και δίνω την λίστα των βουλευτών, αλλιώς απαντάω ”ΟΧΙ”.

- Περιοριζόμαστε στην περίπτωση που υπάρχει πλειοψηφικό κόμμα, αφού αλλιώς ο αλγόριθμος απαντάει πάντα σωστά.
- Αν υπάρχει κόμμα με απόλυτη πλειοψηφία η πιθανότητα να ανήκει σε αυτό είναι τουλάχιστον $1/2 + 1/n$.

Άσκηση 5: Πλειοψηφία - Λύση 2

Ο αλγόριθμος απαντάει σωστά με πιθανότητα τουλάχιστον $1/2$ σε χρόνο $O(n)$. Αν τον τρέξουμε k φορές τότε η πιθανότητα να μην πετύχουμε βουλευτή του πλειοψηφικού κόμματος είναι $(1/2)^k$. Διαλέγουμε $k = \log(1/\delta)$ και έχουμε ότι η πιθανότητα αποτυχίας του αλγορίθμου είναι το πολύ δ.
Συνολικοί χαιρετισμοί $O(n \log(1/\delta))$.