Αλγόριθμοι και Πολυπλοκότητα

Γαβαλάς Νίκος, ΑΜ 03113121

Οκτώβριος 2018

1η Γραπτή Σειρά Ασκήσεων

Ασκηση 1

α

Τελική διάταξη:

1. $\sum_{k=1}^{n} k2^{-k}$
2. $(\log n)^2 / \log\log n$
3. $\log(\binom{n}{\log n})$
4. $\log^4 n$
5. $\log(n!)/(\log n)^3$
6. $\log(\binom{2n}{n}), n2^{2^{2^{100}}} = \Theta(n)$
7. n^2
8. $n^3/(\log n)^8$
9. $\binom{n}{6} = \Theta(n^6)$
10. $(\log_2 n)^{\log_2 n}$
11. $(\sqrt{n})^{\log_2 \log_2 (n!)}$
12. $2^{(\log_2 n)^4} = \Theta(n^{\log^3 n})$
13. $\sum_{k=1}^{n} k2^k, n\sum_{k=0}^{n}\binom{n}{k} = \Theta(n2^n)$
14. $(\sqrt{n})!$

β

1. $T(n) = 2T(n/3) + n\log n : n^{\log_3 2 \approx 0.63}, nlogn = f(n) = \Omega(n^{\log_3 2})$, ικανοποιεί την regularity condition, άρα *Περίπτωση 3* και $T(n) = \Theta(n\log n)$

2. $T(n) = 3T(n/3) + n\log n : n^{\log_3 3} + nlogn, nlogn = f(n) = \Omega(n)$, δεν ικανοποιεί όμως την regularity condition, αφού διαφέρουν κατά λογαριθμικό όρο, όχι πολυωνυμικό. Με δέντρο αναδρομής εύκολα φαίνεται ότι $\Theta(n\log^2 n)$

3. $T(n) = 4T(n/3) + n\log n : n^{\log_3 4 \approx 1.26}, n\log n = f(n) = O(n^{\log_3 4})$, άρα *Περίπτωση 1* και $T(n) = \Theta(n^{\log_3 4})$

4. $T(n) = T(n/2) + T(n/3) + n$: βλέπουμε ότι $\frac{n}{2} + \frac{n}{3} < n$, οπότε λογικά $T(n) = \Theta(n)$ (απόδειξη με δέντρο αναδρομής)

5. $T(n) = T(n/2) + T(n/3) + T(n/6) + n$: εδώ ισχύει $\frac{n}{2} + \frac{n}{3} + \frac{n}{6} = n$, οπότε κάνοντας δέντρο αναδρομής βλέπουμε πως $\Theta(n\log n)$

6. $T(n) = T(n^{5/6}) + \Theta(\log n)$: ...

7. $T(n) = T(n/4) + \sqrt{n}$: θέτουμε $n = k^2$ και η αναδρομική σχέση γίνεται $U(k) = U(k/2) + \Theta(k)$ με $U(k) = T(k^2)$. Από MT, $U(k) = \Theta(k) \Rightarrow T(k^2) = \Theta(k) \Rightarrow T(n) = \Theta(\sqrt{n})$

Ασκηση 2

α

1.

Πραγματοποιούμε quickselect στον πίνακα και βρίσκουμε τον median των στοιχείων σε χρόνο $O(n)$. Ύστερα κάνουμε partition τον πίνακα χρησιμοποιώντας τον median ως pivot, που παίρνει χρόνο επίσης $O(n)$, και τέλος κάνουμε αναδρομικά την ίδια διαδικασία στα δύο partitioned κομμάτια. Με δέντρο αναδρομής φαίνεται άμεσα ότι η πολυπλοκότητα του αλγορίθμου αυτού είναι $O(n\log k)$.

2.

Με βάση τον άνωθι αλγόριθμο, ταξινομούμε το καθένα από τα k κομμάτια σε χρόνο $O(\frac{n}{k}\log\frac{n}{k})$ το καθένα. Συνολικά έχουμε k κομμάτια, οπότε ο χρόνος για όλον τον πίνακα θα είναι $k \cdot O(\frac{n}{k}\log\frac{n}{k}) = O(n\log\frac{n}{k})$. Ο χρόνος αυτός είναι βέλτιστος για συγκριτικό αλγόριθμο, και αυτό επειδή οποιοσδήποτε συγκριτικός αλγόριθμος ταξινόμησης ξέρουμε ότι δεν γίνεται να έχει χρόνο καλύτερο από $O(n\log n)$, αφού για n στοιχεία υπάρχουν $n!$ συνδυασμοί αυτών, και πραγματοποιώντας σε κάθε βήμα μία σύγκριση μειώνουμε το $n!$ σε $\frac{n!}{2}$. Άρα συνολικά δηλαδή θα θέλουμε χρόνο $\log_2(n!) = \Theta(n\log_2 n)$ (από Stirling's approximation).

β

Έστω m το πλήθος των διαφορετικών στοιχείων του A. Αρχικά κάνουμε κάτι σαν mergesort, όπου χωρίζουμε τον πίνακα σε δύο μέρη και δουλεύουμε αναδρομικά, με τη διαφορά ότι στο βήμα της συγχώνευσης όταν βρούμε διπλότυπο στοιχείο, το πετάμε. Έτσι προκύπτει ένας νεός πίνακας A', που έχει ακριβώς m στοιχεία ταξινομημένα. Η διαδικασία αυτή γίνεται σε χρόνο $m\log n$ αφού $T(n) = 2T(n/2) + \Theta(m)$. Ύστερα έχοντας αυτόν τον A' πίνακα, διασχίζουμε τον A και για κάθε στοιχείο αυτού κάνουμε δυαδική αναζήτηση στον A' και βρίσκουμε σε ποια θέση είναι εκεί, οπότε αυξάνουμε έναν counter σε έναν πίνακα με counters στην αντίστοιχη θέση. Τέλος, από

τον πίνακα A' και αυτόν με τους counters, σχηματίζουμε τον αρχικό, ταξινομημένο. Η τελευταία διαδικασία χρειάζεται χρόνο $n \log m$. Τώρα αν $m = O(\log^d n)$, η συνολική πολυπλοκότητα είναι $\Theta(\log^d n \log n + n \log \log^d n) = \Theta(dn \log \log n) = \Theta(n \log \log n)$.

Ασκηση 3

α

Ξεκινάμε θέτοντας σε μια μεταβλητή "min" τη τιμή $|A_1[0] - A_2[0]|$ και αρχικοποιώντας τους i και j στις αρχές των δύο πινάκων. Ύστερα σε loop αυξάνουμε το index που αντιστοιχεί στο στοιχείο με τη μικρότερη τιμή, και κάνουμε update την min όταν μειώνεται η διαφορά, μέχρι να δείξει σε τιμή που ξεπερνάει αυτή του στοιχείου που δείχνει το άλλο index, οπότε ύστερα αυξάνουμε το άλλο (που τώρα έχει δείχνει στο μικρότερο από τα δύο), έως ότου βρούμε διαφορά μηδέν ή φτάσει ένας από τους δείκτες στα όρια του πίνακα του.
Ο αλγόριθμος είναι ορθός γιατί για να μειωθεί η διαφορά, αρκεί να αυξηθεί η τιμή του μικρότερου όρου ή να μειωθεί η τιμή του μεγαλύτερου. Όμως εδώ επειδή είναι ταξινομημένοι οι πίνακες, μπορούμε μόνο να μειώσουμε τη διαφορά αυξάνοντας τον μικρότερο, οπότε κάνουμε αυτό έως ότου βρούμε στοιχείο που υπερβαίνει την τιμή που δείχνει ο άλλος δείκτης όπου και αρχίζουμε να μετακινούμε τον άλλον.
Σε ό,τι αφορά χρόνική πολυπλοκότητα, ο αλγόριθμος είναι γραμμικού χρόνου αφού για κάθε στοιχείο καθενός πίνακα πραγματοποιούμε μια σύγκριση που μειώνει το μέγεθος του προβλήματος κατά ένα στοιχείο. Άρα $\Theta(n_1 n_2)$.

β

Εντελώς ανάλογα με το 1ο ερώτημα, ξεκινάμε με 3 δείκτες στην αρχή του κάθε πίνακα και πραγματοποιούμε ακριβώς την ίδια διαδικασία, αυξάνοντας τον δείκτη που δείχνει στο μικρότερο στοιχείο σε σύγκριση με τα στοιχεία που δείχνουν οι υπόλοιποι, και σε κάθε αύξηση επανυοπολογίζουμε min και max και κάνουμε update τη διαφορά. Η διαδικασία αυτή χρόνο $\Theta(m \sum_{k=1}^{m} n_k)$ αφού σε κάθε βήμα χρειαζόμαστε χρόνο m για να βρούμε τα min-max. Είναι ορθός για τον ίδιο λόγο που είναι και ο αλγόριθμος του a ερωτήματος.

γ

Η διαδικασία εύρεσης min-max σε κάθε βήμα μπορεί να επιταχυνθεί με χρήση ενός max-heap και ενός min-heap. Θα χρειάζεται χρόνος $\log m$ για την αναδιάταξη των στοιχείων σε κάθε βήμα αλλά θα έχουμε το min και το max σε χρόνο $O(1)$. Έτσι η τελική πολυπλοκότητα είναι $\Theta(\log m \sum_{k=1}^{m} n_k)$

Ασκηση 4

α

Αριθμούμε τις φιάλες στο δυαδικό, και χρησιμοποιούμε $log_2 1000000 \approx 20$ δοκιμαστές, στους οποίους λέμε να δοκιμάσουν ο 1ος τις φιάλες με $LSB = bit_0 = 1$, ο 2ος εκείνες με $bit_1 = 1$ και ούτω καθεξής. Μετά από 24 ώρες τους διατάσσουμε βάσει του bit που δοκίμαζε ο καθένας και όποιος έχει επηρεαστεί από το φίλτρο τον μετράμε ως 1, αντιθέτως 0. Ο δυαδικός που σχηματίζεται αντιστοιχεί στη φιάλη που αναζητούμε. Η πολυπλοκότητα της λύσης αυτής είναι $\Theta(\log n)$.

β

Κατ΄ αρχάς, η μέγιστη ημερήσια αποστάση είναι προφανώς τουλάχιστον ίση με τη μέγιστη από τις d_i, και το πολύ ίση με το $\sum_{i=k}^{n} d_i$ (αφού μπορούμε να βάλουμε σε κάθε μέρα και από μία πόλη και στο τέλος όσες περισσεύουν να τις βάλουμε μαζεμένες στην τελευταία μέρα). Αυτές οι δύο τιμές ορίζουν ένα εύρος. Αυτό που μπορούμε λοιπόν να κάνουμε είναι για κάθε τιμή σε αυτό το εύρος, να ακολουθήσουμε την εξής τακτική: Για κάθε πόλη, αν η d_i χωράει στην τρέχουσα τιμή, τη προσθέτουμε, αν όχι, μηδενίζουμε το τρέχον άθροισμα και αλλάζουμε μέρα. Αφού τρέξουμε όλες τις πόλεις και έχουμε $\{\text{πλήθος ημερών}\}<= k$, κρατάμε το max. Τέλος για κάθε τιμή του εύρους κρατάμε το μικρότερο max, και επιστρέφουμε το μικρότερο όλων. Η πολυπλοκότητα της λύσης αυτής είναι $O(n \sum_{i=1}^{n} d_i)$. Νομίζω ότι μπορούμε να κάνουμε δυαδική αναζήτηση στις τιμές του εύρους αλλά δεν είμαι σίγουρος (αυτό θα μείωνε την πολυπλοκότητα σε $O(n \log \sum_{i=1}^{n} d_i)$.)

Ασκηση 5

α

Για την εύρεση του k-οστού μικρότερου ξεκινάμε με κλήση της $F_S(\frac{M}{2})$. Αν το αποτέλεσμα είναι $< k$, καλούμε την $F_S(\frac{M}{2} + \frac{M}{4})$, διαφορετικά την $F_S(\frac{M}{2} - \frac{M}{4})$, κ.ο.κ. (ή απλά θέτουμε $M = M/2$ και συνεχίζουμε αναδρομικά στο κομμάτι που είναι το k). Πρακτικά κάνουμε δυαδική αναζήτηση δηλαδή.
Η πολυπλοκότητα είναι αυτή της δυαδικής αναζήτησης, δηλαδή $\Theta(\log M)$ (υποθέτοντας φυσικά ότι η F_S είναι $\Theta(1)$.

β

Το πρόβλημα λύνεται με τον αλγόριθμο του a ερωτήματος, αρκεί να βρούμε έναν τρόπο να υπολογήσουμε την αντίστοιχη F_S. Για να πετύχουμε αποδοτική υλοποίηση, αρχικά ταξινομούμε τα στοιχεία του συνόλου. Η διαδικασία αυτή παίρνει χρόνο $\Theta(n \log n)$. Ύστερα, για να βρούμε το πλήθος των διαφορών που δεν υπερβαίνουν το ℓ, δηλαδή την $F_S(\ell)$, για κάθε στοιχείο i του ταξινομημένου πίνακα A κάνουμε δυαδική αναζήτηση για να βρούμε το $A[i] + \ell$ και κάνουμε accumulate το πλήθος των στοιχείων μεταξύ του $A[i] + \ell$ και του $A[i]$, παίρνουμε δηλαδή τη διαφορά των δεικτών. Η F_S ύστερα γυρίζει το accumulated άθροισμα.
Η F_S κάνει έτσι χρόνο n για να διατρέξει τον πίνακα και χρόνο $\log n$ για κάθε στοιχείο του λόγω της δυαδικής αναζήτησης. Οπότε συνολικά μαζί με την αρχική ταξινόμηση κάνει χρόνο $\Theta(n \log n + n \log n) = \Theta(n \log n)$. Όταν αυτή η F_S συνδυαστεί με την δυαδική αναζήτηση στον χώρο των τιμών της (βλ. ερώτημα a), ο τελικός αλγόριθμος κάνει χρόνο $\Theta(n \log n \log M)$.