Αλγόριθμοι

1η Σειρά Ασκήσεων

Ονοματεπώνυμο:	Κεχριώτη Ελένη
Αριθμός Μητρώου:	3210078
Email:	p3210078@aueb.gr

2023

Ασκήσεις

Άσκηση 1

$$\sqrt{\log_9 n} < \log(\sqrt{n}) < 2^{\frac{logn}{2}} < \log(4^n) < \log(n^n) < (\log(2^n))^2 < 8^{logn} < n^{logn} < 2^n < 2^{nlogn} < n!$$

Δηλαδή:

$f_1 = \sqrt{\log_9 n}$ $f_2 = \log(\sqrt{n})$ $f_3 = 2^{\frac{logn}{2}}$

$f_4 = \log(4^n)$ $f_5 = \log(n^n)$ $f_6 = (\log(2^n))^2$

$f_7 = 8^{logn}$ $f_8 = n^{logn}$ $f_9 = 2^n$

$f_{10} = 2^{nlogn}$ $f_{11} = n!$

Άσκηση 2

α) $n * 2^n = O(3^n)$

Για να ισχύει η παραπάνω ισότητα πρέπει να ισχύει:

$$n * 2^n \leq c * 3^n, \forall n > n0$$

Πράγματι,

$$n * 2^n \leq c * 3^n \leftrightarrow n \leq c * \frac{3^n}{2^n} \leftrightarrow n \leq c * \left(\frac{3}{2}\right)^n \leftrightarrow c * \left(\frac{3}{2}\right)^n - n \geq 0$$

Έστω $c = 1$ και $f(n) = \left(\frac{3}{2}\right)^n - n$.

$H\ f'(n) = \log\frac{3}{2} * \left(\frac{3}{2}\right)^n - 1.$ Προφανώς η f' είναι γνησίως αύξουσα, αφού $\log\frac{3}{2} > 0$ και $\frac{3}{2} > 1$ και είναι εκθετική.

$f'(3) \approx 0.97 > 0.$ Άρα, $f'(x) > 0\ \forall x \geq 3.$ Άρα, η f είναι γνησίως αύξουσα για $x > 3.$

Για $x = 3,\ \ f(3) \approx 0.37 > 0.$ Λόγω μονοτονίας της f ισχύει ότι $f(x) > 0\ \forall\ x > 3.$ Άρα, για $c = 1$ και $n > 3$ ισχύει ότι $f(n) > 0,$ δηλαδή ισχύει η ανισότητα (2).

Επομένως, ισχύει και η αρχική ανισότητα (1). Άρα αποδείχθηκε οτι $n * 2^n = O(3^n).$

β) $n^2 * 2^n + 10 = O(3^n)$

Για να ισχύει η παραπάνω ισότητα πρέπει να ισχύει:

$$n^2 * 2^n + 10 \leq c * 3^n, \forall n > n0$$

Πράγματι,

$$n^2 * 2^n + 10 \leq c * 3^n \ \leftrightarrow n^2 * 2^n \leq c * 3^n - 10 \ και \ ξέρω \ ότι \ c * 3^n - 10 \leq c * 3^n$$

Άρα έχω ότι $n^2 * 2^n \leq c * 3^n$. Επομένως, $n^2 \leq c * \frac{3^n}{2^n} \leftrightarrow n^2 \leq c * \left(\frac{3}{2}\right)^n$

$Έστω \ c = 1 \ και \ f(n) = \left(\frac{3}{2}\right)^n - n^2.$

$'Εχω \ ότι \quad f'(n) = \log\frac{3}{2} * \left(\frac{3}{2}\right)^n - 2n \quad και \quad f''(n) = \left(\log\frac{3}{2}\right)^2 * \left(\frac{3}{2}\right)^n - 2.$

Προφανώς η f'' είναι γνησίως αύξουσα για τους ίδιους λόγους με το α) ερώτημα.

$$f''(10) \approx 17.7 > 0. \ Άρα \ η \ f' \ είναι \ γνησίως \ αύξουσα \ για \ x > 10.$$

Ακόμη, $f'(10) \approx 13.7 > 0$. $Άρα \ η \ f \ είναι \ γνησίως \ αύξουσα \ για \ x > 10.$

$$f(13) \approx 25.6 > 0. \ Άρα \ f(n) > 0 \ \forall n > 13.$$

$Άρα, για \ c = 1 \ και \ n > 13 \ ισχύει \ ότι \ f(n) > 0, \quad δηλαδή \ ισχύει \ η \ ανισότητα \ (2).$

$Επομένως, ισχύει \ και \ η \ αρχική \ ανισότητα \ (1). \ Άρα \ αποδείχθηκε \ οτι \ n^2 * 2^n = O(3^n).$

Άσκηση 3

α) Αν το n ≥ 1 τότε σε κάθε κλήση της συνάρτησης f, εκτυπώνονται n αριθμοί. Επίσης καλείται αναδρομικά η συνάρτηση 2 φορές για n = n/2. Με βάση αυτή τη ανάλυση καταλήγω στη

$$T(n) = 2T\left(\frac{n}{2}\right) + n$$

Χρησιμοποιώντας το Master Theorem έχω:

$$a = 2\,, b = 2, d = 1 \ και \ f(n) = n$$

$$b^d = a \ \leftrightarrow \ 2^1 = 2 \ \leftrightarrow 1 = \log_2 2$$

Ακόμη $n = \Theta\left(n^{\log_2 2}\right) = \Theta(n)$, που ισχύει.

Αυτή είναι η 2η περίπτωση του Master Theorem άρα

$$T(n) = O(nlogn) \quad \text{και} \quad T(n) = \Theta(nlogn)$$

β) Για κάθε κλήση της f εκτυπώνεται μια φορά ο αριθμός 1. Ύστερα, η f καλείται αναδρομικά 2 φορές και το n υποδιπλασίαζεται. Όπως και στο α) ερώτημα η ανάλυση είναι ίδια με την μόνη διαφορά ότι ο αριθμός 1 εκτυπώνεται 1 φορά σε κάθε κλήση.

Με βάση αυτή τη ανάλυση καταλήγω:

$$T(n) = 2T\left(\frac{n}{2}\right) + 1$$

Χρησημοποιώντας το Master Theorem έχω:

$$a = 2\,, b = 2, d = 0 \; \kappa\alpha\iota \; f(n) = 1$$

$$b^d < a \;\leftrightarrow\; 2^0 = 1 < 2 \;\leftrightarrow 0 < \log_2 2$$

Ακόμη, για $\varepsilon = 1 > 0$ ισχύει ότι $f(n) = O(n^{1-1}) = O(1)$.

Αυτή είναι η 3η περίπτωση του Master Theorem άρα $T(n) = O(n)$ και η 1η περίπτωση για Θ notations άρα $T(n) = \Theta(n)$.

Άσκηση 4

α) Έστω ότι το n είναι άρτιος. Αφού το n είναι άρτιος τότε ο πίνακας θα είναι τέλεια χωρισμένος σε ζεύγη, δηλάδη δεν θα μείνει μόνο του κάποιο στοιχείο χωρίς ζεύγος. Γνωρίζουμε ότι το x είναι το πλειοψηφικό στοιχείο του πίνακα Α, άρα το x θα εμφανίζεται τουλάχιστον $\frac{n}{2} + 1$ φορές. (1)

Σύμφωνα με τον αλγόριθμο που δίνεται υπάρχουν δύο τύποι δυνατών ζευγών: 1) και τα δύο στοιχεία είναι διαφορετικά, 2) και τα δύο στοιχεία είναι το ίδιο.

Λόγω του ότι ο n είναι άρτιος θα έχουμε ακριβώς $\frac{n}{2}$ ζεύγη. Από (1) προφανώς πρέπει σε τουλάχιστον 1 από τα $\frac{n}{2}$ ζεύγη να εμπίπτουν στον 2ο τύπο ζεύγων, δηλάδη και τα δύο στοιχεία να είναι το x.

Άρα από αυτό το ένα ζεύγος θα κρατηθεί ως πλειοψηφικό στοιχείο το x.

Αν όλα τα υπόλοιπα ζεύγη εμπίπτουν στη 1η κατηγορία ζευγών, τότε δεν θα κρατηθεί κανένα από τα δύο στοιχεία.

Άρα στον πίνακα Β θα περαστεί μόνο το x και άρα θα είναι το πλειοψηφικό του στοιχείο.

Ακόμα και αν κάποια από τα ζεύγη εμπίπτουν στη 2η κατηγορία ζευγών, επειδή κρατάμε και περνάμε στον πίνακα Β μόνο όσα ζευγάρια είναι ίδια και επειδή το x είναι πλειοψηφικό θα περαστεί το x περισσότερες από τις μισές φορές. Άρα πάλι στον Β το x θα είναι πλειοψηφικό.

β) Έστω ότι το n είναι περιττός. Αφού το n είναι περιττός τότε ένα στοιχείο θα περισσέυει, δηλαδή δεν θα έχει ζευγάρι. Εφόσον το x είναι πλειοψηφικό στοιχείο, τότε θα εμφανίζεται τουλάχιστον $\frac{n}{2} + 1$ φορές. (1)

Αν το x είναι το περισσευόμενο στοιχείο, τότε στη περίπτωση που εμφανίζεται $\frac{n}{2} + 1$ φορές όλα τα υπόλοιπα ζεύγη θα εμπίπτουν στη $1^{η}$ κατηγορία ζευγών (άρα το x βρίσκεται σε κάθε ζεύγος).

Σε κάθε άλλη περίπτωση, δηλαδή το x να μην είναι το περισσευόμενο ή να εμφανίζεται περισσότερες φορές, κάποια από τα ζεύγη θα εμπίπτουν στη $2^{η}$ κατηγορία ζευγών.

Δεδομένου ότι έχει γίνει ο έλεγχος αν το περισσευόμενο στοιχείο είναι το πλειοψηφικό, θα ελέγξουμε για πλειοψηφία στον υπόλοιπο πίνακα (τα ζεύγη) που είναι άρτιου μεγέθους. Εφόσον το περισσευόμενο δεν είναι το x τότε στον υπόλοιπο πίνακα το x θα εμφανίζεται τουλάχιστον $\frac{n}{2} + 1$ φορές. Από το α ερώτημα αποδείχθηκε ότι αν ο πίνακας είναι άρτιου μεγέθους και το x είναι πλειοψηφικό τότε το x θα είναι πλειοψηφικό και στον B.

Άρα το ζητούμενο αποδείχθηκε.

γ) Συνάρτηση Majority(A):

```
if A.length() = 0: return null
if A.length() = 1: return A[0]
if A.length()mod2 = 1:
        if count(A[A.length -1]) > n/2:
                return A[A.length -1]
B = []
for i = 1 to n/2:
        if A[2*i -1] = A[2*i]:
                B[i] = A[2*i]
p = Majority(B)

if A.count(p) ≥ n/2+1: return p1
else: return null
```

Η συνάρτηση Majority είναι μια αναδρομική συνάρτηση. Αν ο πίνακας A είναι κενός τότε προφανώς δεν υπάρχει πλειοψηγικό στοιχείο και επιστρέφεται null. Αν υπάρχει 1 μόνο στοιχείο τότε επιστρέφεται ως πλειοψηφικό αυτό και αν ο πίνακας είναι περιττού μεγέθους

τότε ελέγχουμε αν το τελευταίο στοιχείο είναι το πλειοψηφικό, αφού δεν θα μπορέσουμε να το ελέγξουμε μετά ως ζευγάρι.

Σε μια επανάληψη ελέγχουμε ανα ζευγάρια αν τα στοιχεία του πίνακα Α είναι μεταξύ τους ίδια, και αν ισχύει τότε προσθέτουμε αυτό το στοιχείο στο κενό πίνακα Β. Ύστερα καλούμε αναδρομικά τη συνάρτηση Majority μέχρι στο p να επιστραφεί είτε ένα στοιχείο του αρχικού πίνακα Α είτε null. Ελέγχουμε τελικά, αν το στοιχείο που επιστρέφεται είναι το πλειοψηφικό στον αρχικό πίνακα και αν ναι τότε επιστρέφουμε αυτό αλλιώς null.

Στη ουσία σε κάθε κλήση της συνάρτησης Majority ελέγχουμε αν το περιττό στοιχείο είναι το πλειοψηφικό, και αν όχι τότε κρατάμε από τα ίδια ζευγαράκια το στοιχείο. Ο πίνακας Α, δηλαδή, μειώνεται το πολύ στα 2 και ξανακάνουμε αυτή τη διαδικασία για τα υπόλοιπα στοιχεία. Η αναδρομή τερματίζεται μέχρι να επιστραφεί είτε ένα στοιχείο είτε τίποτα (null). Έτσι στο τέλος θα έχουμε ως απάντηση είτε το πλειοψηφικό στοιχείο αν υπάρχει είτε null αν δεν υπάρχει.

δ)Για τον υπολογισμό της πολυπλοκότητας έχω:

- Σταθερό χρόνο για συγκρίσεις και return, άρα $O(1)$, αλλά έχω n/2 συγκρίσεις
- Για τη μέτρηση (count) έχω $O(n)$
- Στη χειρότερη περίπτωση θα περαστούν στον πίνακα Β $\frac{n}{2}$ στοιχεία, και θα κληθεί η συνάρτηση για το μισό μέγεθος.

Από τα παραπάνω έχω: $O(n) + O\left(\frac{n}{2}\right) + O(1) = O(n)$, άρα $d = 1,\ \ b = 2$

Ακόμη μελετάμε ένα πρόβλημα άρα $a = 1$.

Καταλήγω στη αναδρομική σχέση:

$$T(n) = T\left(\frac{n}{2}\right) + O(n)$$

και έχω $\log_b a = \log_2 1 = 0 < 1 = d$ άρα τελικά από τη 1η περίπτωση του Master Theorem

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

Άσκηση 5

α) Υποθέτω ότι κάθε κουτί είναι ένας πίνακας με n θέσεις, της μορφής [0,0,14,0,6], δηλαδή στο κουτί αυτό έχουν αποθηκευτεί 14 μπάλες του χρώματος 2 και 6 μπάλες του χρώματος 4. Αρχικοποιώ έναν πίνακα με n κουτιά και μηδενίζω τις n θέσεις του κάθε κουτιού για να δηλώσω ότι δεν υπάρχει καμία μπάλα στο κουτί.

Η βασική ιδέα είναι:

Θέλω χ ζευγαράκια χρωμάτων έτσι ώστε συνολικά οι μπάλες και των δύο να είναι τουλάχιστον 20 και να μπορώ να γεμίσω ένα κουτί.

Για να το πετύχω αυτό και να είναι σίγουρο ότι κάθε ζεύγος θα έχει περισσότερες από 20 μπάλες, ταξινομώ τις μπάλες κατά αύξουσα σειρά. Επιλέγω την mergesort που έχει πολυπλοκότητα O(nlogn).

Έτσι, ζευγαρώνω το μικρότερο σετ μπαλών με το μεγαλύτερο.

Σε κάθε κουτί αποθηκεύω όλες τις μπάλες του πρώτου χρώματος και αυτές που χρειάζομαι μόνο από το δεύτερο χρώμα για να γεμίσω το κουτί. Έτσι, στη πρώτη επανάληψη ζευγαρώνω το πρώτο χρώμα με το δεύτερο. Στη δεύτερη επανάληψη ζευγαρώνω το δεύτερο χρώμα με το μεγαλύτερο σετ μπαλών. Ανάλογα τους αριθμούς το ζευγάρωμα θα γίνει είτε με το τελευταίο χρώμα είτε με το πρότελευταιο. Η διαδικασία αυτή συνεχίζεται μέχρι να μείνω με ένα και τελευταίο ζεύγος για το τελευταίο κουτί, είτε με 20 μπάλες ενός χρώματος.

Τίθεται το παρακάτω παράδειγμα. Έστω ότι έχω 6 κίτρινες, 13 κόκκινες, 28 μπλε και 33 πράσινες μπάλες. Με τον παραπάνω αλγόριθμο έχω:

Κουτί 1-> 6 14	<- 6 13 28 33
Κουτί 2-> 13 7	<- 0 13 28 19
Κουτί 3-> 20	<- 0 0 21 19
Κουτί 4-> 1 19	<- 0 0 1 19

β) Για n=1, δηλαδή για 1 χρώμα, ο αλγόριθμος προφανώς λειτουργεί σωστά και αποθηκεύει τις 20 μπάλες αυτού του χρώματος σε 1 κουτί και επιστρέφει.

Έστω ότι ο αλγόριθμος λειτουργεί σωστά για k χρώματα.

Αρκεί να δείξω ότι ο αλγόριθμος είναι ορθός για k+1 χρώματα.

Όταν εφαρμόσουμε τον αλγόριθμο για τα k+1, μέχρι και το k-1 loop η εφαρμογή αυτή θα ισοδυναμεί με την ολοκλήρωση του αλγορίθμου αυτού αν είχε εφαρμοστεί για τα k χρώματα. Συνεπώς μέχρι αυτό το βήμα θα έχουν γεμίσει πλήρως k από τα κουτιά. Επομένως μένει το γέμισμα του k +1 κουτιού, δηλαδή η τελευταία επανάληψη. Εφόσον εργαζόμαστε με ζευγαράκια και σε κάθε κουτί δεν πρέπει να έχουμε πάνω από 2 χρώματα, έχουμε $\binom{n}{2}$ πιθανούς συνδυασμούς χρωμάτων, δηλαδή $\frac{n^2-n}{2}$. Επίσης, αφού έχουμε αύξουσα σειρά, με βάση των αριθμών των μπαλών, κάθε ζευγαράκι θα έχει ως άθροισμα περισσότερες από 40 μπάλες και επομένως μπορούμε να αποθηκεύσουμε 20 μπάλες σε συνδυασμό αυτών των χρωμάτων σε ένα κουτί. Με βάση αυτές τις δύο προϋποθέσεις, και προφανώς αφού έχουν γεμίσει k κουτιά πλήρως, θα πέσουμε σε 2 περιπτώσεις. Είτε θα έχουν απομείνει 20 μπάλες ακριβώς από αυτό το χρώμα (πιθανώς

επειδή είχαμε έναν μεγάλο αριθμό μπαλών αυτού του χρώματος και πολύ λίγες μπάλες από άλλα χρώματα), είτε θα έχω ένα ακριβώς ζεύγος μπαλών 2 χρωμάτων που και τα 2 χρώματα μαζί θα είναι 20 μπάλες και θα γεμίσει το k+1 κουτί.

Άρα, εφόσον αποδείξαμε την ορθότητα του αλγορίθμου για k+1 χρώματα, ο αλγόριθμος είναι ορθός.

*τα ζευγαράκια γίνονται με ένα χρώμα από τα αριστερά και ένα από τα δεξία, δηλαδή από ένα χρώμα που έχει πολύ λίγες μπάλες και βρίσκεται στην αρχή του πίνακα και από ένα χρώμα με πολλές μπάλες. Αυτό γίνεται για να διασφαλίσουμε ότι χρώμα1+χρώμα2 ≥ 20.

γ) Αρχικά για την ταξινόμηση του πίνακα A, χρησιμοποιούμε την μέθοδο sort, που στη χειρότερη περίπτωση έχει πολυπλοκότητα $O(n\log n)$.

Η αρχικοποίηση του πίνακα Boxes με '0' έχει πολυπλοκότητα $O(n^2)$.

Έχουμε μια επανάληψη for, για κάθε κουτί, άρα n φορές.

Μέσα στη for:

- Η αποθήκευση των μπαλών σε κουτιά -> $O(1)$

Άρα συνολικά μέσα στη for $O(n)$.

Επομένως ο πάραπάνω αλγόριθμος έχει πολυπλοκότητα $O(n^2) + O(n) + O(n\log n) = O(n^2)$.

Ωστόσο, αν εξαιρέσω την αρχικοποίηση των κουτιών με '0', γιατί στην πραγματική ζωή κάτι τέτοιο δεν θα χρειαζόταν τότε η ουσία του αλγορίθμου, δηλαδή από την ταξινόμηση και μετά έχει πολυπλοκότητα $O(n\log n)$.