Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

Αλγόριθμοι και Πολυπλοκότητα

Διδάσκοντες: Σ. Ζάχος, Δ. Φωτάκης

1η Σειρά Προτεινόμενων Ασκήσεων

Άσκηση 1: Ασυμπτωτική Εκτίμηση

Να ταξινομήσετε τις παρακάτω συναρτήσεις σε αύξουσα σειρά τάξης μεγέθους. Βρείτε δηλαδή διάταξη $g_1, g_2, g_3, \ldots$ τέτοια ώστε $g_1 = \mathrm{O}(g_2)$, $g_2 = \mathrm{O}(g_3)$, κ.ο.κ.

2^{2^n}	$n!$	$n2^n$	$10n$
$\log(n!)$	$\log^{10} n$	$n^{\log\log n}$	$\left(\frac{\log n}{\log\log n}\right)^{\frac{\log n}{\log\log n}}$
$\log n^3$	$\frac{n}{\log n}$	$\frac{n^2}{\log^{10} n}$	$\frac{\log n}{n}$
$3n^6$	e^n	$\sqrt{n!}$	$\binom{n}{4}$

Να επισημάνετε ακόμη τις συναρτήσεις που έχουν ίδια τάξη μεγέθους.

Λύση. Η σειρά κατάταξης είναι:

$\frac{\log n}{n}$	$\log n^3$	$\log^{10} n$	$\left(\frac{\log n}{\log\log n}\right)^{\frac{\log n}{\log\log n}}$
$\frac{n}{\log n}$	$10n$	$\log(n!)$	$\frac{n^2}{\log^{10} n}$
$\binom{n}{4}$	$3n^6$	$n^{\log\log n}$	$n2^n$
e^n	$\sqrt{n!}$	$n!$	2^{2^n}

Για την κατάταξη μπορούμε να χρησιμοποιήσουμε ότι:

$$\text{Αν } \lim_{n\to+\infty} \frac{f(n)}{g(n)} = L \text{ με } 0 < L \leq \infty \text{ τότε } f(n) = \Omega(g(n)). \text{ Αν } 0 < L < \infty \text{ τότε } f(n) = \Theta(g(n)).$$

Ακόμα χρήσιμες είναι οι ακόλουθες σχέσεις:

$$\begin{aligned}
\log(n!) &= \Theta(n\log n)\\
\binom{n}{4} &= \frac{n!}{4!(n-4)!} = \frac{n(n-1)(n-2)(n-3)}{4!} = \Theta(n^4)\\
n! &= \Theta\left(\sqrt{2\pi n}\left(\frac{n}{e}\right)^n\right) \quad \text{(Stirling's approximation)}\\
n^{\log\log n} &= e^{\log n \log\log n} \quad \text{(αν θεωρήσουμε φυσικούς λογάριθμους)}\\
\left(\frac{\log n}{\log\log n}\right)^{\frac{\log n}{\log\log n}} &= e^{\frac{\log n}{\log\log n}(\log\log n - \log\log\log n)} = e^{\log n(1-\frac{\log\log\log n}{\log\log n})}
\end{aligned}$$

΄Ετσι έχουμε ότι:

$$\frac{n}{\log n} = e^{\log n - \log\log n} >_{\text{ασυμπτωτικά}} e^{\log n(1-\frac{\log\log\log n}{\log\log n})} = \left(\frac{\log n}{\log\log n}\right)^{\frac{\log n}{\log\log n}}$$

και συνεπώς $\left(\frac{\log n}{\log\log n}\right)^{\frac{\log n}{\log\log n}} = o(\frac{n}{\log n})$. Σημειώνουμε ότι για κάθε σταθερά $\varepsilon > 0$, $\left(\frac{\log n}{\log\log n}\right)^{\frac{(1+\varepsilon)\log n}{\log\log n}} = \omega(n)$, δηλαδή αν πολλαπλασιάσουμε τον εκθέτη με οποιαδήποτε σταθερά μεγαλύτερη της μονάδας, η τάξη μεγέθους της συνάρτησης αλλάζει από υπο-γραμμική σε υπερ-γραμμική. □

Άσκηση 2: Επίλυση Αναδρομικών Σχέσεων

(α) Να λύσετε την παρακάτω αναδρομική σχέση χωρίς χρήση του Master Theorem. Για απλότητα, να υποθέσετε ότι το n είναι δύναμη του 3.

$$T(n) = 2\,T(n/3) + 29n\,, \quad T(1) = 7$$

(β) Η πολυπλοκότητα τριών αλγορίθμων για το ίδιο πρόβλημα δίνεται από τις παρακάτω αναδρομικές σχέσεις:

1. $T_1(n) = 5\,T_1(n/5) + n/\log_5 n$, $T_1(1) = \Theta(1)$
2. $T_2(n) = 2\,T_2(n/4) + n^2\sqrt{n}$, $T_2(1) = \Theta(1)$
3. $T_3(n) = T_3(n-1) + 1/n$, $T_3(1) = \Theta(1)$

Να λύσετε τις αναδρομικές σχέσεις και να ταξινομήσετε τους τρεις αλγορίθμους ως προς την αποδοτικότητά τους. Ποιοι από αυτούς χρησιμοποιούν τη μέθοδο "διαίρει-και-βασίλευε" και γιατί;

Λύση. (α) Εργαζόμαστε με την μέθοδο της επανάληψης:

$$\begin{aligned}
T(n) &= 2\,T(n/3) + 29n \\
&= 2(2\,T(n/3^2) + 29n/3) + 29n \\
&= 2(2(2\,T(n/3^3) + 29n/3^2) + 29n/3) + 29n = \ldots \\
&= 2^{i+1}\,T(n/3^{i+1}) + 29n\sum_{k=0}^{i}(2/3)^k \\
&= 2^{i+1}\,T(n/3^{i+1}) + 87n(1-(2/3)^{i+1})
\end{aligned}$$

Για $i = \log_3 n - 1$, έχουμε ότι $n/3^{i+1} = 1$, και:

$$\begin{aligned}
T(n) &= 2^{\log_3 n}\,T(1) + 87n(1 - n^{\log_3 2/3}) \\
&= 7n^{\log_3 2} + 87n(1 - n^{-1+\log_3 2}) \\
&= 7n^{\log_3 2} + 87n - 87n^{\log_3 2} \\
&= 87n - 80n^{\log_3 2} = \Theta(n)
\end{aligned}$$

(β.1) Κατ' αρχάς να πούμε ότι δεν μπορούμε να χρησιμοποιήσουμε το Master Theorem γιατί η $n/\log n$ είναι ασυμπτωτικά μικρότερη της n **αλλά όχι πολυωνυμικά!** Θεωρούμε για απλότητα ότι το n είναι δύναμη του 5, δηλ. ότι $n = 5^m$ για κάποιον ακέραιο m, και έχουμε:

$$T_1(5^m) = 5\,T_1(5^{m-1}) + \frac{5^m}{m}$$

Αν θέσουμε $T_1(5^m) = S(m)$, με $S(0) = \Theta(1)$, έχουμε ότι:

$$\begin{aligned} S(m) &= 5\,S(m-1) + \frac{5^m}{m} \\ &= 5^2\,S(m-2) + \frac{5^m}{m-1} + \frac{5^m}{m} = \ldots \\ &= 5^m\,S(0) + 5^m \sum_{k=0}^{m-1} \frac{1}{m-k} \\ &= 5^m\,\Theta(1) + 5^m(\ln m + \Theta(1)) \\ &= \Theta(5^m \ln m) \end{aligned}$$

Με αντίστροφη αντικατάσταση, έχουμε ότι: $T_1(n) = \Theta(n \log\log n)$.

(β.2) Εφαρμόζεται η τρίτη περίπτωση του Master Theorem, και έχουμε ότι $T_2(n) = \Theta(n^{5/2})$.

(β.3) Έχουμε ότι:

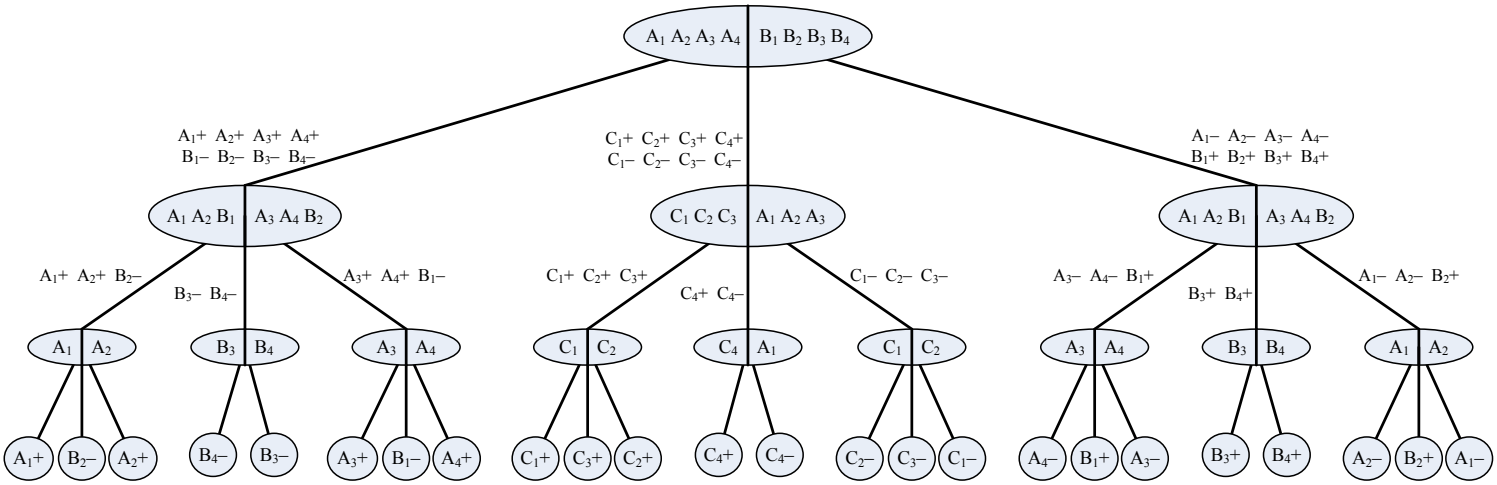
$$\begin{aligned} T_3(n) &= T_3(n-1) + \frac{1}{n} \\ &= T_3(n-2) + \frac{1}{n-1} + \frac{1}{n} = \ldots \\ &= T_3(1) + \sum_{k=0}^{n-2} \frac{1}{n-k} \\ &= \Theta(\log n) \end{aligned}$$

Μόνο οι δύο πρώτες χρησιμοποιούν "διαίρει-και-βασίλευε" αφού σπάνε το αρχικό πρόβλημα σε επιμέρους προβλήματα υποπολλαπλάσιου μεγέθους. □

Άσκηση 3: Βρείτε το Κάλπικο Νόμισμα

Έχουμε 12 νομίσματα, το ένα από τα οποία είναι κάλπικο. Όλα τα γνήσια νομίσματα έχουν το ίδιο ακριβώς βάρος. Το κάλπικο νόμισμα διαφέρει από τα γνήσια μόνο ως προς το βάρος, αλλά δεν γνωρίζουμε αν είναι ελαφρύτερο ή βαρύτερο. Για τον εντοπισμό του κάλπικου νομίσματος, χρησιμοποιούμε μια ζυγαριά ακριβείας που ελέγχει αν ένα σύνολο νομισμάτων είναι ελαφρύτερο, ίδιου βάρους, ή βαρύτερο από ένα άλλο σύνολο νομισμάτων (φυσικά, τα δύο σύνολα πρέπει να είναι ξένα μεταξύ τους), αλλά δεν δίνει καμία άλλη πληροφορία για το βάρος των νομισμάτων. Οι ζυγίσεις μας πρέπει να αφορούν μόνο υποσύνολα των 12 νομισμάτων που έχουμε στη διάθεσή μας (δηλ. δεν μπορεί να συμμετέχουν στις ζυγίσεις νομίσματα εκτός των 12 αρχικών που γνωρίζουμε ότι είναι γνήσια ή κάλπικα).

1. Να δείξετε ότι με 3 ζυγίσεις μπορούμε να ανακαλύψουμε το κάλπικο νόμισμα και αν αυτό είναι ελαφρύτερο ή βαρύτερο από τα γνήσια. Να δείξετε ακόμη ότι κάτι τέτοιο δεν είναι εφικτό με 2 μόνο ζυγίσεις.
2. Αν είχαμε στη διάθεσή μας 2 μόνο ζυγίσεις, ποιος είναι ο μέγιστος αριθμός νομισμάτων ανάμεσα στα οποία θα μπορούσαμε να ξεχωρίσουμε το κάλπικο και αν αυτό είναι ελαφρύτερο ή βαρύτερο από τα γνήσια; Ποιος είναι ο αντίστοιχος αριθμός για $n = 4, 5, \ldots$ ζυγίσεις;



Σχήμα 1. Ένας "αλγόριθμος" που με 3 ζυγίσεις βρίσκει το κάλπικο ανάμεσα σε 12 νομίσματα και αν αυτό είναι ελαφρύτερο ή βαρύτερο. Ο "αλγόριθμος" χωρίζει αρχικά τα νομίσματα σε 3 τετράδες: την τετράδα A με τα νομίσματα A_1, A_2, A_3, A_4, την τετράδα B με τα νομίσματα B_1, B_2, B_3, B_4, και την τετράδα C με τα νομίσματα C_1, C_2, C_3, C_4. Η αναπαράσταση του αλγόριθμου βασίζεται σε ένα τριαδικό δέντρο συγκρίσεων. Κάθε κόμβος (εκτός από τα φύλλα) δηλώνει ότι τα νομίσματα στα αριστερά ζυγίζονται με τα νομίσματα στα δεξιά (π.χ. η ρίζα δηλώνει ότι τα νομίσματα της τετράδας A ζυγίζονται με αυτά της τετράδας B). Αν η ζυγαριά γείρει αριστερά (αντ. δεξιά), ακολουθείται ο αριστερός (αντ. δεξιός) κλάδος, και αν η ζυγαριά ισορροπήσει, ακολουθείται ο μεσαίος κλάδος. Σε κάθε κλάδο σημειώνονται τα πιθανά ενδεχόμενα με βάση την διαθέσιμη πληροφορία από τις ζυγίσεις. Η αναφορά σε ένα νόμισμα δηλώνει ότι αυτό μπορεί να είναι το κάλπικο, και το $+$ (αντ. $-$) δηλώνει ότι θα είναι βαρύτερο (αντ. ελαφρύτερο) από τα κανονικά (π.χ. τα ενδεχόμενα που αντιστοιχούν στον ψηλότερο αριστερό κλάδο είναι κάποιο από τα νομίσματα της τετράδας A να είναι κάλπικο και βαρύτερο από τα κανονικά ή κάποιο από τα νομίσματα της τετράδας B να είναι κάλπικο και ελαφρύτερο από τα κανονικά). Περιορίζοντας διαδοχικά τα ενδεχόμενα σε κάθε κλάδο, καταλήγουμε σε μοναδικό αποτέλεσμα σε κάθε φύλλο.

Λύση. Κάθε "αλγόριθμος" που χρησιμοποιεί h ζυγίσεις για να εντοπίσει ένα κάλπικο ανάμεσα σε n νομίσματα μπορεί να αναπαρασταθεί με ένα τριαδικό δέντρο ζυγίσεων ύψους h το οποίο έχει τουλάχιστον $2n$ φύλλα, δύο για κάθε νόμισμα i, που αντιστοιχούν στα ενδεχόμενα το νόμισμα i να είναι κάλπικο και ελαφρύτερο από τα κανονικά (συμβολίζεται με $i-$) και το νόμισμα i να είναι κάλπικο και βαρύτερο από τα κανονικά (συμβολίζεται με $i+$). Το δέντρο είναι τριαδικό γιατί κάθε ζύγιση μπορεί να έχει τρία αποτελέσματα: η ζυγαριά γέρνει αριστερά, ισορροπεί, ή γέρνει δεξιά.

Αφού ένα τριαδικό δέντρο ύψους h έχει το πολύ 3^h φύλλα, ο μέγιστος αριθμός n_h νομισμάτων ανάμεσα στα οποία μπορούμε να εντοπίσουμε ένα κάλπικο νόμισμα (και αν αυτό είναι ελαφρύτερο ή βαρύτερο από τα κανονικά) με h ζυγίσεις είναι $n_h \leq (3^h - 1)/2$. Συνεπώς για 12 νομίσματα χρειαζόμαστε τουλάχιστον 3 ζυγίσεις. Ο πιο γνωστός "αλγόριθμος" για 12 νομίσματα με 3 ζυγίσεις φαίνεται στο Σχήμα 1. Στη βιβλιογραφία αναφέρονται και αρκετοί άλλοι, ένας μάλιστα καθορίζει τις ζυγίσεις που θα κάνει εξ' αρχής, χωρίς αυτές να εξαρτώνται από τα αποτελέσματα των προηγούμενων ζυγίσεων, δείτε στην διεύθυνση:

`http://www.sciencedirect.com/science?_ob=ArticleURL&_udi=B6V0F-483SW0V-4&_user=83473&_rdoc=1&_fmt=&_orig=search&_sort=d&_docanchor=&view=c&_acct=C000059671&_version=1&_urlVersion=0&_userid=83473&md5=48118cfe50b4a4bb78cc17d780e7e8cf`

Ο μέγιστος αριθμός νομισμάτων n_2 ανάμεσα στα οποία μπορούμε να εντοπίσουμε το κάλπικο νόμισμα (και αν αυτό είναι ελαφρύτερο ή βαρύτερο από τα κανονικά) με 2 ζυγίσεις είναι $n_2 = 3$. Ο λόγος που $n_2 < 4$ είναι ότι αν στην πρώτη ζύγιση συμμετέχουν όλα τα 4 νομίσματα, έχουμε

μόνο δύο ενδεχόμενα αποτελέσματα (η ζυγαριά δεν μπορεί να ισορροπήσει). Αν στην πρώτη ζύγιση συμμετέχουν μόνο δύο νομίσματα και η ζυγαριά ισορροπήσει, δεν μπορούμε με μία μόνο ζύγιση να βρούμε ποιό από τα άλλα δύο νομίσματα είναι κάλπικο και αν αυτό είναι ελαφρύτερο ή βαρύτερο από τα κανονικά. Μπορούμε βέβαια να εντοπίσουμε το κάλπικο ανάμεσα σε 4 νομίσματα με 2 μόνο ζυγίσεις είτε αν γνωρίζουμε ένα γνήσιο νόμισμα από τα 4 είτε αν έχουμε 3 επιπλέον νομίσματα που γνωρίζουμε ότι είναι γνήσια (δείτε π.χ. τον μεσαίο κλάδο στο Σχήμα 1).

Παρόμοια αποδεικνύουμε ότι μέγιστος αριθμός νομισμάτων n_3 (αντ. n_4) ανάμεσα στα οποία μπορούμε να εντοπίσουμε το κάλπικο νόμισμα (και αν αυτό είναι ελαφρύτερο ή βαρύτερο από τα κανονικά) με 3 (αντ. 4) ζυγίσεις είναι $n_3 = 12$ (αντ. $n_4 = 39$). Για να εντοπίσουμε το κάλπικο νόμισμα ανάμεσα σε 39 νομίσματα με 4 ζυγίσεις, χωρίζουμε αρχικά τα νομίσματα σε 3 ομάδες των 13 νομισμάτων και εργαζόμαστε με την λογική του Σχήματος 1.

Γενικότερα, αποδεικνύεται ότι ο μέγιστος αριθμός νομισμάτων n_h ανάμεσα στα οποία μπορούμε να εντοπίσουμε το κάλπικο νόμισμα (και αν αυτό είναι ελαφρύτερο ή βαρύτερο από τα κανονικά) με h ζυγίσεις είναι $n_h = (3^h - 3)/2$. □

Άσκηση 4: Μέτρηση Αντιστροφών

Έστω $a_1, a_2, \ldots, a_n$ μια ακολουθία n διαφορετικών ακεραίων. Ως μέτρο του πόσο απέχει αυτή η ακολουθία από το να είναι ταξινομημένη σε αύξουσα σειρά χρησιμοποιούμε τον *αριθμό των αντιστροφών*. Ο αριθμός των αντιστροφών για την ακολουθία $a_1, a_2, \ldots, a_n$ είναι το πλήθος των ζευγαριών στοιχείων a_i, a_j με $i < j$ και $a_i > a_j$, δηλ. το πλήθος των ζευγαριών στοιχείων που βρίσκονται εκτός διάταξης. Για παράδειγμα, ο αριθμός των αντιστροφών για την ακολουθία $6, 5, 4, 3, 2, 1$ είναι 15 (όλα τα ζεύγη είναι εκτός διάταξης), ο αριθμός των αντιστροφών για την ακολουθία $5, 1, 2, 6, 3, 4$ είναι 6 (τα ζεύγη $(5, 1)$, $(5, 2)$, $(5, 3)$, $(5, 4)$, $(6, 3)$, και $(6, 4)$), και ο αριθμός των αντιστροφών για την ταξινομημένη ακολουθία $1, 2, 3, 4, 5, 6$ είναι 0.

Να διατυπώσετε αλγόριθμο "διαίρει-και-βασίλευε" που υπολογίζει τον αριθμό των αντιστροφών μιας ακολουθίας n διαφορετικών ακεραίων σε χρόνο $\mathrm{O}(n \log n)$. Να αιτιολογήσετε προσεκτικά την ορθότητα και την χρονική πολυπλοκότητα του αλγόριθμου.

Λύση. Η ιδέα είναι ίδια με αυτή της Mergesort. Έστω ότι έχουμε μία ακολουθία με n στοιχεία. Επαγωγικά υποθέτουμε ότι το πρόβλημα έχει λυθεί για την υπακολουθία $A = (a_1, a_2, \ldots, a_{\frac{n}{2}})$ με τα στοιχεία των $n/2$ πρώτων θέσεων του πίνακα (έστω ότι έχουμε μετρήσει r_A αντίστροφα ζεύγη για την υπακολουθία A) και για την υπακολουθία $B = (a_{\frac{n}{2}+1}, a_{\frac{n}{2}+2}, \ldots, a_n)$ με τα στοιχεία των $n/2$ τελευταίων θέσεων του πίνακα (έστω ότι έχουμε μετρήσει r_B αντίστροφα ζεύγη για την υπακολουθία B). Επίσης υποθέτουμε ότι τα στοιχεία των υπακολουθιών A και B είναι ταξινομημένα. Το πλήθος των αντίστροφων ζευγών για όλη την ακολουθία είναι $r = r_A + r_B + r_{AB}$, όπου r_{AB} το πλήθος των αντίστροφων ζευγών (a_i, a_j) με $i \leq n/2$, $j > n/2$, και $a_i > a_j$.

Για να πετύχουμε πολυπλοκότητα $\mathrm{O}(n \log n)$ πρέπει το merging να γίνει σε χρόνο $\mathrm{O}(n)$ (θυμηθείτε το Master Theorem και την λύση της αναδρομικής εξίσωσης $T(n) = 2T(n/2) + \Theta(n)$.)

Ακολουθούμε την διαδικασία συγχώνευσης της Mergesort. Η μοναδική διαφορά είναι ότι όταν ένα στοιχείο από την υπακολουθία B "συγχωνεύεται", το r_{AB} αυξάνεται κατά το πλήθος των στοιχείων της A που δεν έχουν ακόμη "συγχωνευτεί". Ακολουθεί η διατύπωση αυτής της ιδέας σε ψευδοκώδικα:

Merge&Count(A, B)
    $i \leftarrow 1;\ j \leftarrow 1;\ k \leftarrow 1;\ r_{AB} \leftarrow 0;$
    while $i \leq |A|$ **and** $j \leq |B|$ **do**
        if $A[i] < B[j]$ **then**
            $C[k] \leftarrow A[i];\ k \leftarrow k+1;\ i \leftarrow i+1;$
        else
            $C[k] \leftarrow B[i];\ k \leftarrow k+1;\ j \leftarrow j+1;\ \ r_{AB} \leftarrow r_{AB} + |A| - i + 1;$
    if $i > |A|$ **then**
        Αντέγραψε τα υπόλοιπα στοιχεία της B στην C;
    else
        Αντέγραψε τα υπόλοιπα στοιχεία της A στην C;
    return(r_{AB}, C);

Χρησιμοποιώντας την ρουτίνα Merge&Count, ο παρακάτω ψευδοκώδικας λύνει το πρόβλημα σε χρόνο $\Theta(n \log n)$:

Sort&Count$(a_1, \ldots, a_n)$
    if $n \leq 1$ **then return**(0);
    $A \leftarrow (a_1, \ldots, a_{\frac{n}{2}});\ B \leftarrow (a_{\frac{n}{2}+1}, \ldots, a_n);$
    $(r_A, A) \leftarrow$ Sort&Count(A);
    $(r_B, B) \leftarrow$ Sort&Count(B);
    $(r_{AB}, C) \leftarrow$ Merge&Count(A, B);
    return$(r_A + r_B + r_{AB}, C)$;

Άσκηση 5: Φωλιασμένα Διαστήματα

Δίνονται τα μη κενά κλειστά διαστήματα φυσικών αριθμών $[s_1, t_1], [s_2, t_2], \ldots, [s_n, t_n]$. Υποθέτουμε ότι $s_1 \leq s_2 \leq \cdots \leq s_n$, ενώ κάποια από τα s_i και t_i μπορεί να είναι ίδια. Δύο διαστήματα λέγονται φωλιασμένα όταν το ένα είναι *υποσύνολο* του άλλου. Το ζητούμενο είναι να υπολογίσουμε το *πλήθος* των ζευγών φωλιασμένων διαστημάτων στο $[s_1, t_1], [s_2, t_2], \ldots, [s_n, t_n]$. Π.χ. στο $[1, 8], [2, 3], [3, 4], [5, 8], [6, 8], [7, 12]$ υπάρχουν 5 ζεύγη φωλιασμένων διαστημάτων, τα $[2, 3] \subseteq [1, 8]$, $[3, 4] \subseteq [1, 8]$, $[5, 8] \subseteq [1, 8]$, $[6, 8] \subseteq [1, 8]$, και $[6, 8] \subseteq [5, 8]$. Να διατυπώσετε ένα όσο το δυνατόν πιο αποδοτικό αλγόριθμο για αυτό το πρόβλημα. Να αιτιολογήσετε την ορθότητα και την χρονική πολυπλοκότητα του αλγορίθμου σας.

Λύση. Αφού τα s_i είναι ταξινομημένα, αρκεί να υπολογίσουμε για πόσα ζεύγη t_i, t_j, με $i < j$, ισχύει ότι $t_j \leq t_i$. Για κάθε τέτοιο ζεύγος t_i, t_j, έχουμε ότι $s_i \leq s_j$ και $t_j \leq t_i$, άρα $[s_j, t_j] \subseteq [s_i, t_i]$. Το πλήθος αυτών των ζευγών μπορεί να υπολογιστεί σε χρόνο $\mathrm{O}(n \log n)$ όπως στην προηγούμενη άσκηση. □

Άσκηση 6: Βίδες και Παξιμάδια

Δίνονται ένα κουτί με n βίδες και ένα κουτί με n παξιμάδια. Όλες οι βίδες έχουν διαφορετικό μέγεθος, όλα τα παξιμάδια έχουν διαφορετικό μέγεθος, και κάθε βίδα ταιριάζει με ένα μόνο παξιμάδι. Το ζητούμενο είναι να ταιριάξουμε όλες τις βίδες με τα παξιμάδια τους. Υπάρχει όμως ο περιορισμός ότι δεν μπορούμε να συγκρίνουμε τα μεγέθη δύο βιδών ούτε μπορούμε να συγκρίνουμε τα μεγέθη δύο παξιμαδιών. Μπορούμε μόνο να συγκρίνουμε τα μεγέθη μιας βίδας και ενός παξιμαδιού, και

να προσδιορίσουμε αν η βίδα είναι μεγαλύτερη, μικρότερη, ή ταιριάζει με το παξιμάδι (και βέβαια αντίστοιχα, αν το παξιμάδι είναι μικρότερο, μεγαλύτερο, ή ταιριάζει με τη βίδα).

Δεδομένου του παραπάνω περιορισμού, να διατυπώσετε *πιθανοτικό* αλγόριθμο που ταιριάζει όλες τις βίδες με τα παξιμάδια τους και απαιτεί κατά μέση τιμή $O(n \log n)$ συγκρίσεις του μεγέθους μιας βίδας με το μέγεθος ενός παξιμαδιού. *Σημείωση:* Ο αλγόριθμός σας πρέπει οπωσδήποτε να χρησιμοποιεί τυχαιότητα, αφού ο ντετερμινιστικός αλγόριθμος αντίστοιχης πολυπλοκότητας είναι σημαντικά δυσκολότερος και χρησιμοποιεί βαθύτερες τεχνικές που η κατανόησή τους υπερβαίνει τους στόχους του μαθήματος.

Λύση. Ακολουθούμε την λογική του αλγόριθμου Quicksort. Δεν μπορούμε βέβαια να εφαρμόσουμε την Quicksort αυτούσια, γιατί δεν μπορούμε να συγκρίνουμε τις βίδες με τις βίδες και τα παξιμάδια με τα παξιμάδια. Το πρόβλημα εντοπίζεται στην χρήση του pivot για τα δύο σύνολα. Όμως μπορούμε να παρακάμψουμε αυτό το πρόβλημα ως εξής: Διαλέγουμε στην τύχη ένα *παξιμάδι*, το οποίο θα χρησιμοποιηθεί σαν *pivot για το partition των βιδών* (αφού δεν μπορούμε να συγκρίνουμε τις βίδες μεταξύ τους, θα συγκρίνουμε τις βίδες με το παξιμάδι), και βρίσκουμε (σε γραμμικό χρόνο) την βίδα που του αντιστοιχεί, και η οποία θα χρησιμοποιηθεί σαν *pivot για το partition των παξιμαδιών*. Από το partition των δύο συνόλων με αυτά τα pivots (σε γραμμικό χρόνο), προκύπτουν δύο υποσύνολα με βίδες και τα αντίστοιχα υποσύνολα με τα παξιμάδια, στα οποία εφαρμόζουμε την ίδια διαδικασία αναδρομικά. Ο χρόνος εκτέλεσης είναι αντίστοιχος της Quicksort, δηλαδή $\Theta(n \log n)$ στη μέση περίπτωση. □