Φροντιστήρια Αλγορίθμων 1-3

Συγγραφή: Φοιτητής του Τμήματος

19 Μαρτίου 2020

Περίληψη

Το συγκεκριμένο pdf περιέχει τις λυμένες ασκήσεις των φροντιστηρίων 1-3(δεν περιέχεται το 4ο γιατί είχε κάνει ασκήσεις προηγούμενων φροντ και δεν έχει λύσει κάποια απ το 4 επί του παρόντος) που έκανε η κα Παπακωνσταντινοπούλου. Δημιουργήθηκε με σκοπό να βοηθήσει όσους δεν μπορούν να παρευρίσκονται στα φροντιστήρια, καθώς δεν ανεβαίνουν στο e-class οι λύσεις των ασκήσεων. Έγινε προσπάθεια να λυθούν πιο αναλυτικά κάποιες ασκήσεις και περιέχει και λύσεις ασκήσεων που δεν έγιναν.Χρησιμοποιήθηκε το LaTeX.

Πηγές: Διαφάνειες και Κανάλι κ. Ψούνη

1 Φροντιστήριο Ι

ΑΣΚΗΣΗ 1

Να δείξετε ότι ισχύει η παρακάτω σχέση με την αρχή της μαθηματικής επαγωγής:

$$1 \cdot 1! + 2 \cdot 2! + ... + n \cdot n! = (n+1)! - 1, \forall n \in \mathbb{N}^+.$$

Λύση:

1ο Βήμα: Βάση Επαγωγής: $n = 1$

Βρίσκουμε έναν αριθμό για τον οποίο ισχύει η ισότητα.

Έχουμε: $1 \cdot 1! = 1$ και $(1+1)! - 1 = 2! - 1 = 1$, άρα η σχέση ισχύει για $n = 1$

2ο Βήμα: Επαγωγική Υπόθεση

Έστω πως ισχύει για $n = k$, οπότε έχουμε:

$1 \cdot 1! + 2 \cdot 2! + ... + k \cdot k! = (k+1)! - 1$ **(1)**

3ο Βήμα: Επαγωγικό Βήμα

Θέλω να αποδείξω πως η σχέση ισχύει και για $n = k + 1$. Δηλαδή ότι:

$1 \cdot 1! + 2 \cdot 2! + ... + (k+1) \cdot (k+1)! = (k+2)! - 1$

Σημείωση: Στο επαγωγικό βήμα θέλω πάντα να εμφανίζω με κάποιο τρόπο την επαγωγική υπόθεση, δηλαδή την (1) σε αυτή την περίπτωση.

Ξεκινάμε από το αριστερό μέλος της παραπάνω σχέσης προσπαθώντας να καταλήξουμε στο δεξί και ταυτόχρονα να εμφανίσουμε την (1). Επομένως:

$1 \cdot 1! + 2 \cdot 2! + ... + (k+1) \cdot (k+1)! = \underline{1 \cdot 1! + 2 \cdot 2! + ... + k \cdot k!} + (k+1) \cdot (k+1)! =$

Αντικαθιστούμε το υπογραμμισμένο κομμάτι με την (1) καθώς εμφανίστηκε η επαγωγική υπόθεση.

$= (k+1)! - 1 + (k+1) \cdot (k+1)! = (k+1)! \cdot (k+2) - 1 = (k+2)! - 1,$

δηλαδή ισχύει η σχέση και για $n + 1$,άρα αποδείχθηκε.

ΑΣΚΗΣΗ 2

Να δείξετε ότι ισχύει η παρακάτω σχέση χρησιμοποιώντας την αρχή της μαθηματικής επαγωγής:

$$2^1 + 2^2 + ... + 2^n = 2^{n+1} - 2, \forall n \in \mathbb{N}^+.$$

Λύση:

1)Βάση Επαγωγής: $n = 1$,έχουμε:

$2^1 = 2$ και $2^{1+1} - 2 = 4 - 2 = 2$, άρα η σχέση ισχύει για $n = 1$.

2)Επαγωγική Υπόθεση: Έστω πως ισχύει για $n = k$, δηλαδή

$2^1 + 2^2 + ... + 2^k = 2^{k+1} - 2$ **(1)**

3)Επαγωγικό Βήμα: Θα δείξουμε πως ισχύει για $n = k + 1$, δηλαδή ότι

$2^1 + 2^2 + ... + 2^{k+1} = 2^{k+2} - 2$

Αναπτύσσουμε το αριστερό μέλος αυτής της σχέσης με σκοπό να χρησιμοποιήσουμε την επαγωγική υπόθεση και να καταλήξουμε στο δεξί μέλος.

$2^1 + 2^2 + ... + 2^{k+1} = \underline{2^1 + 2^2 + ... + 2^k} + 2^{k+1} =$

Αντικαθιστώ το υπογραμμισμένο κομμάτι με την επαγωγική υπόθεση, την (1) δηλαδή

$= 2^{k+1} - 2 + 2^{k+1} = 2 \cdot 2^{k+1} - 2 = 2^{k+2} - 2$, άρα αποδείξαμε τη σχέση της άσκησης.

ΑΣΚΗΣΗ 3

Να λυθεί η $T(n) = T(n-1) + 2$ για $n \geq 1$ και $T(1) = 1$.

Σημείωση: Όταν λέμε λύση αναδρομικής εξίσωσης εννοούμε να βρούμε την $T(n)$ χωρίς να είναι πια αναδρομική, δηλαδή να είναι απλώς συναρτήσει του n. Επίσης όταν η αναδρομή είναι της μορφής $aT(n-b) + f(n)$ με το $a = 1$ μπορούμε να χρησιμοποιήσουμε μια εύκολη παραλλαγή της μεθόδου επανάληψης που εξηγείται παρακάτω.

Λύση:

$T(n) = \cancel{T(n-1)} + 2$
$\cancel{T(n-1)} = \cancel{T(n-2)} + 2$
$\cancel{T(n-2)} = \cancel{T(n-3)} + 2$
$\cancel{T(n-3)} = \cancel{T(n-4)} + 2$
.....
.....
$\cancel{T(4)} = \cancel{T(3)} + 2$
$\cancel{T(3)} = \cancel{T(2)} + 2$
$\cancel{T(2)} = T(1) + 2 \qquad \oplus$
$T(n) = T(1) + 2 \cdot (n-1)$

Παραπάνω προσθέσαμε το 2 $(n-1)$ φορές όσες και οι σχέσεις.

Βρίσκουμε τις τιμές της T για κάθε αριθμό-βήμα αναδρομής μέχρι να φτάσουμε σε σχέση η οποία περιέχει τη βάση της αναδρομής(εδώ $n = 1$, $T(1) = 1$. Μετά αθροίζουμε κατά μέλη όλες τις σχέσεις και απαλείφουμε όσους όρους μπορούμε όπως παραπάνω. Με την πρόσθεση κατά μέλη τελικά μένει:

$T(n) = 1 + 2n - 2 \Rightarrow T(n) = 2n - 1.$

ΑΣΚΗΣΗ 4

Να λύσετε την αναδρομική εξίσωση $T(n) = 2T(n-1) + 2$ για $n \geq 1$ με $T(1) = 1$.

Λύση:

Στη συγκεκριμένη περίπτωση χρησιμοποιούμε πάλι μέθοδο επανάληψης αλλά με κάποια παραπάνω βήματα.

1. Κάνουμε 3 εφαρμογές της αναδρομικής σχέσης(μέχρι να φτάσουμε στη μορφή: $T(n) = a^3T(n-bk)$.
2. Εκτίμηση της σειράς που προκύπτει μετά απο k επαναλήψεις(μας καθοδηγεί ο όρος: $T(n-bk)$).
3. Υπολογίζουμε πότε σταματάει η αναδρομή(Θέτουμε $n - bk = n_0$ και λύνουμε ως προς k).Π.χ. αν $n_0 = 0$, τότε $k = \frac{n}{b}$
4. Αντικατάσταση του k στον τύπο του βήματος 2.
5. Υπολογισμός της σειράς που προκύπτει. Χρήσιμοι οι τύποι $\sum_{i=0}^{n} x^i = \frac{x^{n+1}-1}{x-1}$

$T(n) = 2T(n-1) + 2 = \quad |T(n-1) = 2T(n-2) + 2$
$= 2 \cdot (2T(n-2) + 2) + 2 = \quad |T(n-2) = 2T(n-3) + 2$
$= 2^2 \cdot T(n-2) + 2 \cdot 2 + 2 =$
$= 2^2 \cdot (2T(n-3) + 2) + 2 \cdot 2 + 2 =$
$= 2^3T(n-3) + 2^2 \cdot 2 + 2^1 \cdot 2 + 2^0 \cdot 2 =$
$= 2^kT(n-k) + 2^{k-1} \cdot 2 + ... + 2^2 \cdot 2 + 2^1 \cdot 2 + 2^0 \cdot 2 =$

Η αναδρομή σταματά όταν $n - k = 1 \Rightarrow k = n - 1$

$= 2^{n-1} \cdot T(1) + 2^{n-2} \cdot 2 + ... + 2^2 \cdot 2 + 2^1 \cdot 2 + 2^0 \cdot 2 =$
$= 2^{n-1} + 2 \cdot [2^{n-2} + ... + 2^2 + 2^1 + 2^0] =$
$= 2^{n-1} + 2 \cdot [2^0 + 2^1 + 2^2 + ... + 2^{n-2}] =$
$= 2^{n-1} + 2 \cdot \sum_{i=0}^{n-2} 2^i = 2^{n-1} + 2 \cdot (\frac{2^{n-1}-1}{2-1}) =$
$= 2^{n-1} + 2 \cdot (2^{n-1} - 1) = 2^{n-1} + 2 \cdot 2^{n-1} - 2 = 3 \cdot 2^{n-1} - 2$

Άρα $T(n) = 3 \cdot 2^{n-1} - 2$.

ΑΣΚΗΣΗ 5

Να δείξετε με τη βοήθεια της μαθηματικής επαγωγής, ότι όταν το n είναι ακριβής δύναμη του 2, η λύση της αναδρομής

$$T(n) = \begin{cases} 2, \text{αν } n = 2 \\ 2T(\frac{n}{2}) + n), \text{αν } n = 2^k \text{ με } k > 1. \end{cases} \quad (1)$$

είναι $T(n) = n \log n$

Λύση:

Για ευκολία θεωρούμε πως το n είναι δύναμη του 2, δηλαδή $n = 2^k, k > 1$ και θα εφαρμόσουμε μαθηματική επαγωγή στο k (δηλαδή έμμεση επαγωγή στο n).

1. Βάση Επαγωγής $k = 1, (n = 2)$ άρα έχουμε $T(2^1) = 2^1 \cdot \log 2^1 = 2$, οπότε ισχύει η σχέση.

2. Επαγωγική Υπόθεση: Έστω πως η πρόταση μας ισχύει για $k = m, (n = 2^m)$ δηλαδή:

$$2T(\frac{2^m}{2}) + 2^m = 2^m \cdot \log 2^m = 2^m \cdot m$$

3. Επαγωγικό Βήμα: Θα δείξουμε ότι ισχύει για $k = m + 1$ δηλαδή ότι:

$$2T(\frac{2^{m+1}}{2}) + 2^{m+1} = 2^{m+1} \cdot \log 2^{m+1} = 2^{m+1} \cdot (m + 1)$$

Ακολουθώντας ακριβώς τον ίδιο τρόπο με τις προηγούμενες έχουμε:

$2T(\frac{2^m+1}{2}) + 2^{m+1} = 2\underline{T(2^m)} + 2^{m+1} =$

Αντικαθιστούμε το υπογραμμισμένο μέλος με $2T(\frac{2^m}{2}) + 2^m$ γιατί από αρχική σχέση ισχύει $T(2^m) = 2T(\frac{2^m}{2}) + 2^m$

$= 2 \cdot 2^m \cdot \log 2^m + 2^{m+1} = 2 \cdot (2^m \cdot m) + 2^{m+1} = 2^{m+1} \cdot (m + 1)$, άρα αποδείχθηκε.

ΑΣΚΗΣΗ 6

α)Δείξτε ότι ένα πλήρες δυαδικό δέντρο με n κόμβους έχει ύψος $O(\log n)$.
β)Υπολογίστε το πλήθος φύλλων ενός πλήρους δυαδικού δέντρου ύψους h.

Λύση:

- Παρατηρούμε πως ο αριθμός των φύλλων ενός πλήρους δυαδικού δέντρου σχετίζεται με το ύψος του(ύψος h = η απόσταση της ρίζας από τα φύλλα του δέντρου). Όταν έχουμε τη ρίζα μόνο, το ύψος $h = 0$ και έχουμε μόνο 1 φύλλο(2^0), όταν το ύψος είναι 1 έχουμε 2 φύλλα(2^1) κ.ο.κ., οπότε για ύψος h θα έχουμε 2^h φύλλα.
- Οι κόμβοι μας προφανώς θα ισούνται με :

  $$n = 2^0 + 2^1 + 2^2 + ... + 2^h$$

  Από την Άσκηση 2 έχουμε $2^0 + 2^1 + ... + 2^h = 2^{h+1} - 1$
  Άρα $n = 2^{h+1} - 1$ και λύνουμε ως προς h για να βρούμε την πολυπλοκότητα του ύψους για n κόμβους.

  $2^{h+1} = n + 1 \Rightarrow 2^h = \frac{n+1}{2} \Rightarrow h = \log(\frac{n+1}{2})$, άρα όντως $h = O(\log n)$.

ΑΣΚΗΣΗ 7

Το τμήμα Πληροφορικής δίνει κάθε χρόνο υποτροφία στο φοιτητή με το μέγιστο βαθμό πτυχίου και βραβείο σε αυτόν με το δεύτερο μεγαλύτερο βαθμό. Δίνεται πίνακας Α που περιέχει το όνομα και το βαθμό πτυχίου για καθένα από τους ν φοιτητές που αποφοίτησαν τη χρονιά που πέρασε. Δώστε έναν αποδοτικό αλγόριθμο που βρίσκει τους φοιτητές που θα πάρουν βραβείο. Εξηγήστε γιατί είναι σωστός ο αλγόριθμος. Αποδείξτε την πολυπλοκότητα του.

Λύση:

1. Μια πρώτη περίπτωση(μη βέλτιστη) είναι να ταξινομήσουμε τον πίνακα(αλγόριθμος τάξεως $O(n \log n)$) και να επιστρέψουμε το 2ο στοιχείο του φθίνοντα πίνακα οπότε συνολικά η πολυπλοκότητα παραμένει και πάλι $O(n \log n)$.
2. Η βέλτιστη περίπτωση είναι να διατρέξουμε τον πίνακα μια φορά επαναληπτικά, να βρούμε το μεγαλύτερο στοιχείο και να το αφαιρέσουμε $\Rightarrow$ επανάληψη για n στοιχεία άρα τάξη $O(n)$, και στη συνέχεια να επαναδιατρέξουμε τον πίνακα με τα $n - 1$ πλέον στοιχεία και να βρούμε το μεγαλύτερο που θα αποτελεί το 2ο προφανώς του αρχικού $\Rightarrow$ επανάληψη για $n - 1$ στοιχεία άρα τάξη $O(n)$. Οπότε είναι τάξεως $O(n)$.

ΑΣΚΗΣΗ 8

Ποια είναι η πολυπλοκότητα του παρακάτω τμήματος προγράμματος;

for $i = 1$ to n do

    for $j = 1$ to $\frac{i+1}{2}$ do
    $x = x + 1$
    end_for
end_for

Λύση:

$T(n) = \sum_{i=1}^{n} [\sum_{j=1}^{\frac{i+1}{2}} 2] = \sum_{i=1}^{n} 2 \cdot (\frac{i+1}{2}) = \sum_{i=1}^{n} i+1 = \frac{n \cdot (n+1)}{2} + n = \frac{n^2+n}{2} + n =$
$= O(n^2)$

Σημείωση: Τα άκρα των for μπαίνουν στα δύο άκρα των αθροισμάτων

ΑΣΚΗΣΗ 9

Μια ομάδα μπάσκετ έχει συνολικά 12 παίκτες και στον αγωνιστικό χώρο κάθε φορά βρίσκονται 5. Πόσες είναι οι δυνατές πεντάδες που μπορούμε να φτιάξουμε;

Λύση:

Ζητούμενος αριθμός : $C(\binom{12}{5}) = \frac{12!}{7! \cdot 5!} = 792$ από το γνωστό τύπο Συνδυαστικής $C(\binom{n}{k}) = \frac{n!}{(n-k)! \cdot k!}$

2 Φροντιστήριο ΙΙ

ΑΣΚΗΣΗ 1

Να δείξετε χρησιμοποιώντας τον ορισμό του O-συμβολισμού, ότι:
$1047n^2 + 52n = O(n^2)$.

Λύση:

Σύμφωνα με τον ορισμό του O χρειαζόμαστε μια σταθερά $c > 0$ και n_0 τέτοια ώστε $1047n^2 + 52n \leq cn^2$ για κάθε $n \geq n_0$. Επιλέγουμε $n_0 = 1, c = 1099$

$1047n^2 + 52n \leq cn^2 \Rightarrow 1047n^2 + 52n \leq 1099n^2 \Rightarrow n \geq 1$ που ισχύει εφόσον έχουμε επιλέξει για n_0 το 1. Άρα αποδείχθηκε.

ΑΣΚΗΣΗ 2

Να δοθεί ο καλύτερος O-συμβολισμός για τις ακόλουθες συναρτήσεις:

1. $2\log n - 4n + 3n\log n$
2. $2 + 4 + 6 + + ... + 2n$
3. $2 + 4 + 8 + ... + 2^n$

Λύση:

1. $\log n < n < n\log n$ ασυμπτωτικά οπότε $2\log n - 4n + 3n\log n = O(n\log n)$
2. $2 + 4 + 6 + ... + 2n = 2 \cdot (1 + 2 + 3 + ... + n) = 2 \cdot \sum_{i=1}^{n} i = 2 \cdot \frac{n(n+1)}{2} =$
   $= n^2 + n = O(n^2)$
3. $2 + 4 + 8 + ... + 2^n = 2 \cdot (1 + 2 + 4 + ... + 2^{n-1}) = 2 \cdot \sum_{i=0}^{n-1} 2^i =$
   $= 2 \cdot (\frac{2^{n-1+1}-1}{2-1}) = 2 \cdot (2^n - 1) = 2^{n+1} - 2 = 2 \cdot 2^n - 2 =$
   $= O(2^n)$

ΑΣΚΗΣΗ 3

Έστω $T_1(n) = O(f(n))$ και $T_2(n) = O(g(n))$ οι πολυπλοκότητες δύο τμημάτων P_1 και P_2 ενός προγράμματος P. Αν το P_2 εκτελείται αμέσως μετά το P_1, ποια είναι η πολυπλοκότητα του προγράμματος P;

Λύση:

$P_1 : T_1(n) = O(f(n)), P_2 : T_2(n) = O(g(n))$ και $T(n) = O(f(n) + O(g(n)$

Από τον ορισμό του ασυμπτωτικού συμβολισμού,εφόσον έχω άθροισμα, ασυμπτωτικά με ενδιαφέρει μόνο ο μεγαλύτερος από τους 2 όρους. Επομένως έχουμε:

$$T(n) = O(max\{T_1(n), T_2(n)\}) = max\{O(f(n), O(g(n)\}$$

ΑΣΚΗΣΗ 4

Έστω f, g, h θετικές συναρτήσεις. Αποφασίστε αν οι παρακάτω παραστάσεις είναι αληθείς ή ψευδείς.

1. $f(n) = \Omega(g(n)) \Rightarrow g(n) = O(f(n))$
2. $f(n) = \omega(g(n)) \Rightarrow f(n) = \Omega(g(n))$
3. $f(n) = O(g(n)) \wedge f(n) = \Omega(h(n)) \Rightarrow g(n) = \Theta(h(n))$
4. $f(n) + g(n) = \Theta(min\{g(n), f(n)\})$
5. $f(n) = O(g(n)) \wedge g(n) = \Omega(f(n)) \Rightarrow f(n) = \Theta(g(n))$

Λύση:

1. Αληθής, από ορισμό. Αφού $f(n) = \Omega(g(n))$ υπάρχουν c, n_0 τέτοια ώστε $f(n) \geq cg(n)$ για κάθε $n \geq n_0 \Rightarrow g(n) \leq \frac{1}{c} f(n)$ Θέτω $c_1 = \frac{1}{c}$. Άρα $g(n) \leq c_1 f(n)$. Συνεπώς υπάρχουν c_1, n_0 τέτοια ώστε $g(n) \leq c_1 f(n)$ για κάθε $n \geq n_0$
2. Αληθής, από ορισμό. Αφού $f(n) = \omega(g(n))$, για κάθε $c > 0$ υπάρχει $n_0 > 0$ τέτοιο ώστε $f(n) > cg(n)$ για κάθε $n \geq n_0$. Όμως $f(n) > cg(n) \Rightarrow f(n) \geq cg(n)$. Επομένως $f(n) \geq cg(n)$ για κάθε $n \geq n_0$,οπότε $f(n) = \Omega(g(n))$
3. Ψευδής. Για να αποδείξουμε οτι μια σχέση είναι ψευδής αρκεί να βρούμε ένα αντιπαράδειγμα. Έστω $f(n) = n^2, g(n) = n^3, h(n) = n$. Ισχύει ότι $f(n) = O(g(n)) \Rightarrow n^2 = O(n^3)$ και $f(n) = \Omega(h(n)) \Rightarrow n^2 = \Omega(n)$ αλλά $g(n) \neq \Theta(h(n)) \Rightarrow n^3 \neq \Theta(n)$

4. Ψευδής, αρκεί να βρούμε ένα αντιπαράδειγμα. Έστω $f(n) = n^2, g(n) = 2^n$ Έχουμε ότι $f(n) + g(n) = n^2 + 2^n = \Theta(2^n)$. Άρα όχι $\Theta(n^2)$.

5. Ψευδής, αρκεί να βρούμε ένα αντιπαράδειγμα. Έστω $f(n) = n, g(n) = n^2$. Ισχύει $f(n) = O(n^2)$ και $g(n) = \Omega(n)$ αλλά $n \neq \Theta(n^2)$

ΑΣΚΗΣΗ 5

Δείξτε τις ακόλουθες προτάσεις:

1. Αν $a > 1$ και $f(n) = O(\log_a(n))$, τότε $f(n) = O(\log n)$
2. $\log(n!) = O(n \log n)$
3. $3^n \neq O(2^n)$
4. Αν $f(n) = O(n)$, τότε η $2^f(n)$ δεν είναι $O(2^n)$.

Λύση:

1. Αφού ισχύει$f(n) = O(\log_a n)$ από τον ορισμό προκύπτει ότι υπάρχουν c και n_0 τέτοια ώστε για κάθε $n \geq n_0$ είναι $f(n) \leq c \log_a n$. Όμως $\log_a n = \frac{\log n}{\log a}$ οπότε $f(n) \leq \frac{c}{\log a} logn$ ή $f(n) \leq c_1 \log n$ με $c_1 = \frac{c}{\log a}$ για κάθε $n \geq n_0$. Άρα $f(n) = O(\log n)$.

2. Γνωρίζουμε ότι ασυμπτωτικά $n! < n^n \Rightarrow \log(n!) < \log(n^n)$ Η φορά έμεινε ίδια γιατι η λογαριθμική είναι γνησίως αύξουσα συνάρτηση. Οπότε έχουμε $\log(n!) < n \log n \Rightarrow \log(n!) \leq n \log n$ Οπότε αν πάρουμε ως $c = 1, n_0 = 1$ βάσει ορισμού καταλήγουμε ότι $\log(n!) = O(n \log n)$.

3. Έχουμε $\lim_{n\to\infty} \frac{2^n}{3^n} = 0$ οπότε $2^n = O(3^n)$, $3^n = \Omega(2^n)$ και $2^n \neq \Theta(3^n)$. Επομένως $3^n \neq O(2^n)$.

4. Έστω $f(n) = 2n$. Τότε είναι $f(n) = O(n)$ αλλά $2^{f(n)} = 2^{2n} \neq O(2^n)$

ΑΣΚΗΣΗ 6

Απαντήστε στις ακόλουθες ερωτήσεις δικαιολογώντας την απάντησή σας.

- Ισχύει $2^{n+1} = O(2^n)$;
- Ισχύει $2^{2n} = O(2^n)$;

Λύση:

- Έστω $f(n) = 2^{n+1}$ και $g(n) = 2^n$ Έχουμε $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{2^n \cdot 2}{2^n} = 2 \neq 0$ Άρα $f(n) = \Theta(g(n))$. Από θεωρία αν $f(n) = \Theta(g(n) \Rightarrow f(n) = O(g(n)) = O(2^n)$, άρα ισχύει.

- Έχουμε $\lim_{n\to\infty} \frac{2^{2n}}{2^n} = \lim_{n\to\infty} \frac{\cancel{2^n}\cdot 2^n}{\cancel{2^n}} = \infty$, άρα $f(n) = \Omega(2^n)$ και ο παραπάνω ισχυρισμός δεν ισχύει.

ΑΣΚΗΣΗ 7

Δείξτε ότι για πραγματικούς αριθμούς a, b με $b > 0$ ισχύει $(n + a)^b = \Theta(n^b)$

Λύση:

Το a είναι σταθερά οπότε όταν αναπτυχθεί η ταυτότητα ο μόνος όρος που μας ενδιαφέρει ασυμπτωτικά(και είναι ο μεγαλύτερος από τους άλλους προφανώς), είναι ο n^b. Οπότε ισχύει ότι $(n + a)^b = \Theta(n^b)$.

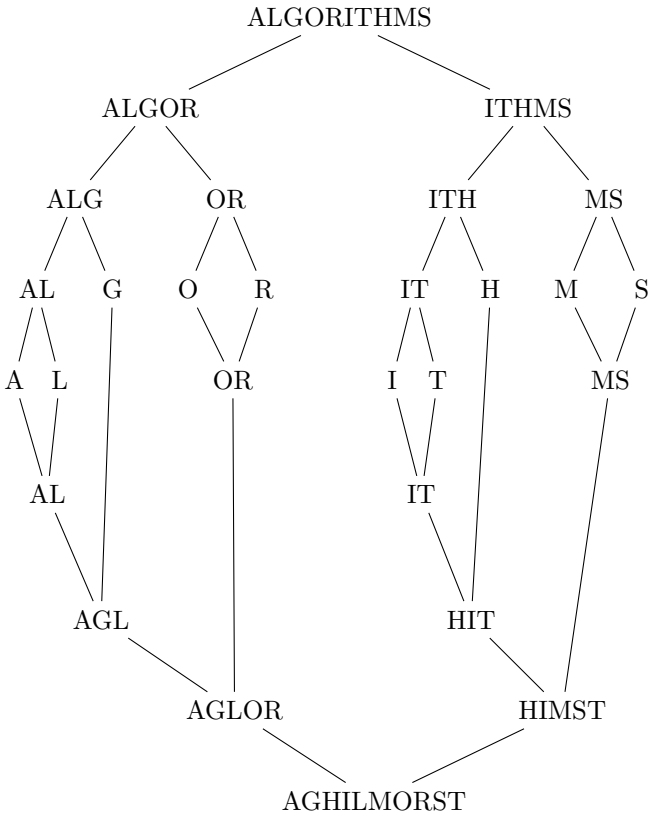
.

3 Φροντιστήριο III

ΑΣΚΗΣΗ 1

Να ταξινομήσετε τα γράμματα της λέξης 'Algorithms'. χρησιμοποιώντας τον αλγόριθμο Mergesort.

Λύση:



Ο Αλγόριθμος Mergesort χωρίζει το πρόβλημα σε δύο υποπροβλήματα κάθε φορά αναδρομικά(στην περίπτωσή μας δηλαδή χωρίζει τη λέξη στα δύο) μέχρι να φτάσει στο ένα στοιχείο σε καθένα απ τα κομμάτια που χωρίζονται. Έπειτα κάνει συγχώνευση των ταξινομημένων υπο-λέξεων καταλήγοντας στην ταξινομημένη λέξη που θέλουμε.

ΑΣΚΗΣΗ 2

Χρησιμοποιήστε τον Αλγόριθμο Quicksort για να ταξινομήσετε τη συμβολοακολουθία 'Algorithms'.

Λύση:

Η πολυπλοκότητα του Αλγορίθμου QuickSort όπως αναφέρεται και στις διαφάνειες εξαρτάται κυρίως από το ποιο pivot θα επιλεχθεί. Συνήθως επιλέγεται ως pivot είτε το πρώτο είτε το μεσαίο είτε το τελευταίο στοιχείο. Στην περίπτωση αυτή επιλέχθηκε το πρώτο στοιχείο για να είναι πιο απλός ο ψευδοκώδικας, παρόλα αυτά δεν είναι το βέλτιστο pivot όσον αφορά την πολυπλοκότητα. Ο ψευδοκώδικας:

Αλγόριθμος **QuickSort**(A,start,finish)

```
if (start<finish) then
        pos=Partition(A,start,finish)
        QuickSort(A,start,pos-1)
        QuickSort(A,pos+1,finish)
```

procedure **Partition(A,start,finish)**

```
pivot=A[start]
i=start
j=finish
for(k=start+1 to finish)
        if(A[k]>pivot)
            B[j]=A[k]                ⇒ Χρησιμοποιούμε ένα νέο πίνακα
            j = j − 1
         else
            B[i]=A[k]
            i = i + 1
```

Ο αλγόριθμος QuickSort αρχικά καλεί την Partition. Η Partition αυτό που κάνει είναι να βάλει το pivot στη σωστή θέση δηλαδή βάζει όλα τα μεγαλύτερα στοιχεία απ το pivot στις επόμενες θέσεις και όλα τα μικρότερα στις προηγούμενες θέσεις απ αυτό. Μετά την επιστροφή της partition γίνεται αναδρομή για τα μικρότερα στοιχεία του pivot που βρίσκονται αριστερά του και για τα μεγαλύτερα στοιχεία του που βρίσκονται δεξιά του. Με αυτή τη λογική για τη λέξη Algorithms η εκτέλεση της QuickSort γίνεται ως εξής:

0	1	2	3	4	5	6	7	8	9
A	L	G	O	R	I	T	H	M	S
A	S	M	H	T	I	R	O	G	L

⇒ Λέξη μετά το partition

Οπότε το A μετά την partition έχει μπει στη σωστή θέση μιας και είναι το μικρότερο στοιχείο. Τώρα θα καλεστεί η QuickSort για όλα τα μεγαλύτερα στοιχεία(για τα μικρότερα είναι $T(0)$ καθώς δεν υπάρχουν) ως εξής:

1	2	3	4	5	6	7	8	9
S	M	H	T	I	R	O	G	L
M	H	I	R	O	G	L	S	T

⇒ Λέξη μετά το partition

Τώρα το γράμμα S έχει μπει στη σωστή θέση και με την ίδια λογική καλείται πάλι η QuickSort για τα μικρότερα και τα μεγαλύτερα στοιχεία απ αυτό(Το μόνο μεγαλύτερο είναι το T οπότε αυτό παραμένει στη θέση του). Παρουσιάζεται η Quicksort για τα μικρότερα στοιχεία:

1	2	3	4	5	6	7
M	H	I	R	O	G	L
H	I	G	L	M	O	R

⇒ Λέξη μετά το partition

Οπότε μετά το parition το M έχει μπει στη σωστή θέση και με την ίδια λογική αναδρομικά καλείται η QuickSort για τα μικρότερα και τα μεγαλύτερα στοιχεία. Ας δούμε πρώτα την εκτέλεση της QuickSort για τα μεγαλύτερα στοιχεία:

6	7
O	R
O	R

⇒ Λέξη μετά το partition

Το γράμμα O έχει μπει στη σωστή θέση και τώρα γίνεται η αναδρομή μόνο για το γράμμα R που επειδή είναι το μοναδικό μένει στη θέση του. Άρα κι αυτό είναι στη σωστή θέση και τελειώσαμε με αυτό τον υποπίνακα. Θα ασχοληθούμε τώρα με τα μικρότερα γράμματα:

1	2	3	4
H	I	G	L
G	H	L	I

⇒ Λέξη μετά το partition

Το H έχει μπει στη σωστή θέση και μένει πάλι αναδρομικά να καλέσουμε την QuickSort για τα μικρότερα και τα μεγαλύτερα. Το μόνο μικρότερο είναι το G οπότε θα μείνει κι αυτό στη θέση του και έχει ταξινομηθεί. Οπότε ας δούμε την QuickSort που γίνεται για τα δύο μεγαλύτερα στοιχεία:

3	4
L	I
I	L

⇒ Λέξη μετά το partition

Το L έχει μπει στη σωστή θέση οπότε καλείται η QuickSort μόνο για το I που επειδή είναι μοναδικό είναι ήδη ταξινομημένο. Άρα η διαδικασία τελείωσε και όλα τα γράμματα έχουν ταξινομηθεί ως εξής:

0	1	2	3	4	5	6	7	8	9
A	G	H	I	L	M	O	R	S	T

ΑΣΚΗΣΗ 3

Ποιος είναι ο χρόνος εκτέλεσης του Αλγορίθμου Quicksort

- όταν όλα τα στοιχεία του πίνακα A είναι ίσα;
- όταν ο πίνακας A είναι ανάποδα ταξινομημένος και όλα τα στοιχεία του είναι άνισα;

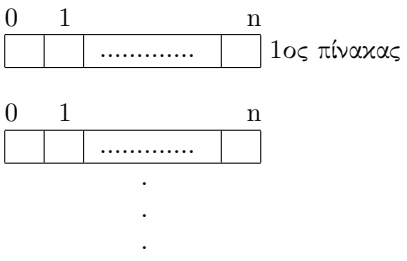
Λύση:

- Σε κάθε βήμα γίνεται 1 κλήση της Quicksort για πίνακα κατά 1 θέση μικρότερο από πριν.Το partition σε κάθε βήμα απαιτεί χρόνο $O(n)$. Έχουμε συνολικά n αναδρομικές κλήσεις όπου $O(n)$ η πολυπλοκότητα της κάθε κλήσης, άρα η συνολική πολυπλοκότητα είναι $n \cdot O(n) = O(n^2)$
- Το pivot αν ο πίνακας είναι σε φθίνουσα ταξινόμηση και εμείς θέλουμε να κάνουμε αύξουσα ταξινόμηση, είναι πάντα το μικρότερο στοιχείο, οπότε στην επόμενη κλήση θα κάνουμε partition στον πίνακα $n-1$ στοιχείων με pivot πάλι το μικρότερο στοιχείο όπως είπαμε. Ομοίως, στη μεθεπόμενη περίπτωση θα έχουμε partition σε πίνακα $n-2$ κ.ο.κ. Συνεπώς θα γίνουν n συνολικά αναδρομικές κλήσεις που σε αυτές το partition επίσης απαιτεί χρόνο $O(n)$, άρα η πολυπλοκότητα θα είναι $n \cdot O(n) = O(n^2)$. Μπορούμε επίσης να λύσουμε την αναδρομική εξίσωση του αλγορίθμου για πίνακα n που προκύπτει από ταξινόμηση $n-1$. Δηλαδή $T(n) = T(n-1) + n$ Ακολουθώντας τα βήματα που εξηγούνται στην Άσκηση 3 του Φροντιστηρίου Ι καταλήγουμε ότι $T(n) = O(n^2)$.

ΑΣΚΗΣΗ 4

Δίνονται k ταξινο$μημ$ένοι πίνακες n στοιχείων ο καθένας. Να σχεδιάσετε αλγόριθ$μο$ διαίρει-και-βασίλευε που να κατασκευάζει έναν ταξινο$μημ$ένο πίνακα $με$ όλα τα στοιχεία και να υπολογίσετε την πολυπλοκότητά του.

Λύση:



.

0	1	.....	n	
				κ-ος πίνακας

Ο Αλγόριθμος που θα χρησιμοποιηθεί είναι ο MergeSort και δουλεύει αναδρομικά ως εξής:

-Χωρίζει τους πίνακες σε 2 σύνολα πινάκων $\frac{k}{2}$ και $\frac{k}{2}$
-Καλεί τον εαυτό του 2 φορές, δηλαδή ταξινομεί τον υποπίνακα που προκύπτει από το κάθε σύνολο(κάθε σύνολο μπορεί να θεωρηθεί ως ενιαίος πίνακας $\frac{kn}{2}$ στοιχείων). Χρόνος: $2T(\frac{k}{2})$
-Συγχωνεύει τους 2 πίνακες(merge) τους 2 υποπίνακες. Χρόνος: kn

Η βάση της αναδρομής μας θα είναι να έχουμε έναν πίνακα σε κάθε σύνολο, οπότε τότε απλώς συγχωνεύουμε τους 2 πίνακες. Η αναδρομική εξίσωση είναι το άθροισμα των δύο παραπάνω χρόνων επομένως:

$$T(k) = 2T(\frac{k}{2}) + k$$

με $a = 2, b = 2, f(n) = k$(Εφόσον λύνουμε ως προς k το n μπορεί να θεωρηθεί σταθερά οπότε ασυμπτωτικά δε μας ενδιαφέρει και δεν το συμπεριλαμβάνουμε). Από Master Theorem τελικά $T(k) = \Theta(k \log k)$

ΑΣΚΗΣΗ 5

Δίνεται πίνακας n ακεραίων με την εξής ιδιότητα: Υπάρχει δείκτης i^* τέτοιος ώστε $1 \leq i^* \leq n-1$, τα $A[1], A[2], ..., A[i^*]$ είναι σε αύξουσα σειρά και τα $A[i^*], A[i^*+1], ..., A[n]$ είναι σε φθίνουσα σειρά. Να δώσετε αλγόριθμο που υπολογίζει το i^* σε χρόνο $O(\log n)$.

Λύση:

Από την εκφώνηση της άσκησης καταλαβαίνουμε ότι ο πινακας θα είναι της παρακάτω μορφής:

0	1			i^*				n
12	16	35	.....	50	47	29	.....	9

Ένας γνωστός Αλγόριθμος που είναι τάξεως $O(\log n)$ είναι ο BinarySearch(Δυαδική Αναζήτηση). Θα χρησιμοποιήσουμε μια παραλλαγή αυτού με τον παρακάτω ψευδοκώδικα:

Αλγόριθμος **IndexSearch** (A,start,finish)

```
if start>finish then ⇒ Εδώ φτάνουμε αν δεν υπάρχει τέτοιος δείκτης
        return 0
else
        middle=(start+finish) div 2
        if (A[middle-1]≤A[middle] && A[middle]≥A[middle+1]) then (1)
                return middle
        else if(A[middle] ≤ A[middle+1]) then (2)
                pos=IndexSearch(A,middle+1,finish)
                return pos
        else (3)
                pos=IndexSearch(A,start,middle−1)
                return pos
```

Αρχικά βρίσκουμε το μεσαίο στοιχείο(middle). Έπειτα στην (1) συνθήκη ελέγχουμε αν αυτό το στοιχείο είναι ο δείκτης i^* για τον οποίο ισχύει $A[i^* - 1] \leq A[i^*] \geq A[i^* + 1]$ Αν ισχύει σημαίνει ότι ο μεσαίος δείκτης είναι αυτός που ψάχνουμε και τον επιστρέφουμε. Σε διαφορετική περίπτωση πάμε στη συνθήκη (2) όπου ελέγχουμε αν το μεσαίο στοιχείο είναι μικρότερο του επόμενου του. Αν ισχύει, τότε το πρώτο μισό του πίνακα δε μας ενδιαφέρει καθώς αποκλείεται να βρίσκεται ο δείκτης που θέλουμε εκεί. Επομένως κάνουμε αναδρομή απ το δείκτη middle+1 ως το finish, που μας επιστρέφει το δείκτη i^*. Αν η (2) συνθήκη δεν ισχύει τότε πάμε στην (3) που σημαίνει ότι βρισκόμαστε στο πρώτο μισό του πίνακα και με αντίστοιχο τρόπο κάνουμε αναδρομή για να βρούμε το δείκτη i^*.

Πολυπλοκότητα: $T(\frac{n}{2}) + \Theta(1)$. Από Master Theorem(2η περίπτωση αφού $a = 1, b = 2, f(n) = n^0 = 1, n^{\log_2 1} = n^0 = 1$) προκύπτει ότι $T(n) = \Theta(\log n)$.

ΑΣΚΗΣΗ 6

Χρησιμοποιώντας τη μέθοδο Master να υπολογίσετε ασυμπτωτικά όρια στις παρακάτω αναδρομές:

(α) $T(n) = 4T(\frac{n}{2}) + n$

(β) $T(n) = 4T(\frac{n}{2}) + n^2$

(γ) $T(n) = 4T(\frac{n}{2}) + n^3$

Λύση:

(α) Έχουμε $a = 4, b = 2, f(n) = n$ και $\log_b a = \log_2 4 = 2$. Ισχύει ότι $f(n) = n = O(n^{2-e})$ για κάποια σταθερά $e > 0$,οπότε βρισκόμαστε στην πρώτη περίπτωση του Master Theorem και $T(n) = \Theta(n^2)$.

(β) Έχουμε και πάλι $a = 4, b = 2$ και $f(n) = n^2$. Οπότε $\log_b a = 2$. Ισχύει ότι $f(n) = n^2 = \Theta(n^{\log_2 4})$ και βρισκόμαστε στη δεύτερη περίπτωση με $T(n) = \Theta(n^2 \log n)$.

(γ) Έχουμε $a = 4, b = 2, f(n) = n^3, \log_2 4 = 2$. Ισχύει ότι $f(n) = n^3 = \Omega(n^{2+e})$ για κάποια σταθερά $e > 0$. Επειδή βρισκόμαστε στην τρίτη περίπτωση του Θεωρήματος πρέπει να ελέγξουμε αν υπάρχει $c < 1$ τέτοιο ώστε:

$af(\frac{n}{b}) \leq cf(n) \Rightarrow 4f(\frac{n}{2}) \leq cf(n) \Rightarrow 4(\frac{n}{2})^3 \leq cn^3 \Rightarrow \frac{4}{8} \leq c \Rightarrow \frac{1}{2} \leq c.$

Άρα ισχύει για $\frac{1}{2} \leq c < 1$. Έπεται $T(n) = \Theta(n^3)$

ΑΣΚΗΣΗ 7

Μπορούμε να εφαρμόσουμε τη μέθοδο Master στην ακόλουθη αναδρομή ;

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

Λύση:

Είναι $a = 2, b = 2$, άρα η $n^{\log_b a} = n$ και $f(n) = \Omega(n)$. Όμως η f δεν απέχει πολυωνυμικά από την n, δηλαδή δεν υπάρχει $e > 0$ τέτοιο ώστε $f(n) = \Omega(n^{1+e})$.

Εναλλακτικά μπορούμε να χρησιμοποιήσουμε όρια για ασυμπτωτικούς συμβολισμούς δηλαδή:

$$\lim_{n\to\infty} \frac{f(n)}{n^{1+e}} = \frac{\log n}{n^e} = 0$$

Άρα $f(n) = O(n^{1+e})$ και $f(n) \neq \Omega(n^{1+e})$ Επομένως η μέθοδος Master δεν μπορεί να εφαρμοσθεί.

ΑΣΚΗΣΗ 8

Ο χρόνος εκτέλεσης ενός Αλγορίθμου A δίνεται από την αναδρομική εξίσωση

$$T(n) = 7T(\frac{n}{2}) + n^2$$

Ένας Αλγόριθμος A' για το ίδιο πρόβλημα έχει χρόνο εκτέλεσης T' που δίνεται από

$$T'(n) = aT(\frac{n}{4}) + n^2$$

Ποια είναι η μέγιστη ακέραια τιμή του a για την οποία ο A' είναι ασυμπτωτικά καλύτερος από τον A;

Λύση:

Αρχικά ας βρούμε την πολυπλοκότητα του Αλγορίθμου A σύμφωνα με το Master Theorem.
Έχουμε $a = 7, b = 2, f(n) = n^2$ οπότε $\log_b a = \log_2 7$ και $2 < log_2 7 < 3$. Άρα υπάρχει σταθερά $e > 0$ τέτοια ώστε $f(n) = n^2 = O(n^{\log_2 7 - e})$. Οπότε σύμφωνα με το θεώρημα η πολυπλοκότητα του Αλγορίθμου A είναι $T(n) = \Theta(n^{\log_2 7})$. Ας δούμε σε πρώτη φάση για ποια τιμή του a οι δύο Αλγόριθμοι θα έχουν την ίδια πολυπλοκότητα. Επειδή έχουν την ίδια $f(n) = n^2$, για να έχουν την ίδια πολυπλοκότητα αρκεί(σύμφωνα και πάλι με το θεώρημα Master:) $\log_2 7 = \log_4 a \Rightarrow \log_2 7 = \frac{\log_2 a}{\log_2 4} \Rightarrow \log_2 7 = \frac{\log_2 a}{2} \Rightarrow 2 \cdot \log_2 7 = \log_2 a \Rightarrow a = 7^2 = 49$
Συνεπώς για $a = 49$ ο Αλγόριθμος A' θα έχει την ίδια πολυπλοκότητα με τον A(αφού $\log_4 49 = \log_2 7$). Καταλαβαίνουμε λοιπόν ότι η τελευταία ακέραια τιμή που μπορεί να πάρει το a ώστε να είναι ασυμπτωτικά καλύτερος ο A' από τον A είναι $a = 48$,αφού $n^{\log_4 48} < n^{\log_2 7}$.