Εισαγωγικές Έννοιες

Δημήτρης Φωτάκης

Τμήμα Μηχανικών Πληροφοριακών και Επικοινωνιακών Συστημάτων
Πανεπιστήμιο Αιγαίου, 83200 Καρλόβασι, Σάμος
Email: fotakis@aegean.gr

1 Αλγόριθμος, Υπολογιστικό Πρόβλημα, και Στιγμιότυπο

Αλγόριθμος. Με απλά λόγια, οι αλγόριθμοι είναι οι "συνταγές" που ακολουθούν οι υπολογιστές για να λύνουν προβλήματα. Πιο συγκεκριμένα, αλγόριθμος είναι μια διαδικασία ή ένα σύνολο κανόνων με σκοπό το μηχανιστικό υπολογισμό. Η εκτέλεση ενός αλγορίθμου δεν πρέπει να απαιτεί τη χρήση της διαίσθησης ή της δημιουργικότητας[1]. Ένας αλγόριθμος αποτελεί το "εργαλείο" για την επίλυση ενός *υπολογιστικού προβλήματος*. Ο ορισμός του υπολογιστικού προβλήματος καθορίζει την σχέση μεταξύ των *δεδομένων εισόδου* (input) και των *δεδομένων εξόδου* (output) και ο αλγόριθμος πραγματοποιεί τον επιθυμητό μετασχηματισμό. Με βάση τα παραπάνω, θα λέμε ότι *αλγόριθμος* είναι κάθε καλώς ορισμένη, πεπερασμένη υπολογιστική διαδικασία για την επίλυση ενός υπολογιστικού προβλήματος.

Περίπου στο 1930, διάφοροι μαθηματικοί ορισμοί της έννοιας μιας "πεπερασμένης υπολογιστικής διαδικασίας" εδόθησαν από τους Church, Godel, Herbrand, Kleene, Post, Turing, και άλλους. Οι ορισμοί αυτοί είχαν ως βάση μεθόδους που επιφανειακά εμφανίζονταν διαφορετικές μεταξύ τους (π.χ. τυπικά συστήματα επεξεργασίας συμβόλων, ιδανικά μοντέλα υπολογιστικών μηχανών). Όμως, όλοι αυτοί οι ορισμοί καταλήγουν να είναι ισοδύναμοι, με την έννοια ότι όλες οι "μορφές" αλγορίθμων υπολογίζουν την ίδια κατηγορία συναρτήσεων, συγκεκριμένα την κλάση των των μερικώς αναδρομικών συναρτήσεων (partial recursive functions). Μέχρι τώρα, και παρά τις προόδους της τεχνολογίας, η εμπειρίας μας καταδεικνύει ότι η αυτή η κλάση των συναρτήσεων είναι ακριβώς ότι μπορεί να υπολογισθεί συστηματικά (από υπολογιστές, προγράμματα, ή πεπερασμένες διαδικασίες).

Τυπικά παραδείγματα αλγορίθμων είναι οι μέθοδοι πρόσθεσης, αφαίρεσης, πολλαπλασιασμού και διαίρεσης αριθμών που μαθαίνουμε στο σχολείο. Ένα άλλο παράδειγμα είναι ο αλγόριθμος του Ευκλείδη για τον υπολογισμό του Μέγιστου Κοινού Διαιρέτη δύο ακεραίων αριθμών.

Αναπαράσταση Αλγορίθμων. Εκτός από τη φυσική γλώσσα, για την αναπαράσταση των αλγορίθμων θα χρησιμοποιήσουμε μια μορφή δομημένου ψευδοκώδικα που θυμίζει τις γλώσσες C και Pascal.

Όλες οι εντολές έχουν την ίδια σημασία με τις αντίστοιχες εντολές στις γλώσσες C και Pascal, όμως ο ψευδοκώδικας δεν έχει την αυστηρότητα που απαιτούν οι συνηθισμένες γλώσσες προγραμματισμού (π.χ. απουσία δηλώσεων μεταβλητών, χρήση ρουτινών που δεν έχουν οριστεί, περιστασιακή χρήση φυσικής γλώσσας αντί για εντολές, κλπ.). Κάθε σύνολο εντολών με κοινή

[1] Πάντως τα όρια μεταξύ αλγορίθμων και ανθρώπινων σχεδίων προς εκτέλεση γίνονται όλο και περισσότερο δυσδιάκριτα, βλέπε για παράδειγμα τυχαιότητα στους αλγορίθμους, fuzzy λογική, κλπ.

στοίχιση θεωρείται σαν ομάδα (block) εντολών που (θα έπρεπε να) βρίσκεται ανάμεσα σε αγκύλες $\{\ldots\}$ στη C (ή σε begin ... end στην Pascal).

Ο ψευδοκώδικας χρησιμοποιεί τη συνάρτηση σαν βασική δομή προγράμματος και υποστηρίζει αναδρομή. Χρησιμοποιούμε τους πίνακες (arrays), τις δομές (structures ή εγγραφές - records στην Pascal) και περιστασιακά τα σύνολα (sets) για την οργάνωση των δεδομένων. Θεωρούμε επίσης δεδομένες τις βασικές δομές δεδομένων (π.χ. διασυνδεδεμένες λίστες, πίνακες, ουρές προτεραιότητας, δέντρα αναζήτησης) και βασικούς αλγόριθμους (π.χ. αλγόριθμοι ταξινόμησης και αναζήτησης, αλγόριθμοι διαχείρισης ξένων συνόλων) που διδάχθηκαν στο προηγούμενο εξάμηνο.

Υπολογιστικό Πρόβλημα. Ένα *υπολογιστικό πρόβλημα* (computational problem) ορίζει έναν επιθυμητό μετασχηματισμό ενός συνόλου δεδομένων εισόδου (input) σε ένα σύνολο δεδομένων εξόδου (output). Ο ορισμός του υπολογιστικού προβλήματος ορίζει τόσο τη μορφή όσο και τους περιορισμούς που πρέπει να ικανοποιούν τα δεδομένα εισόδου και τα δεδομένα εξόδου.

Για παράδειγμα, στο πρόβλημα του πολλαπλασιασμού δύο αριθμών, τα δεδομένα εισόδου είναι δύο αριθμοί (x, y), και τα δεδομένα εξόδου είναι το γινόμενό τους $x \times y$. Στο πρόβλημα της ταξινόμησης n ακεραίων αριθμών σε αύξουσα σειρά, τα δεδομένα εισόδου είναι μία ακολουθία $X = (x_1, x_2, \ldots, x_n)$ από n ακέραιους αριθμούς. Τα δεδομένα εξόδου είναι μια μετάθεση αυτών των αριθμών $X' = (x'_1, x'_2, \ldots, x'_n)$ ώστε κανένας αριθμός να μην είναι μεγαλύτερος από τον επόμενό του, δηλαδή $x'_1 \leq x'_2 \leq \cdots \leq x'_n$.

Στιγμιότυπο και Λύση. Κάθε σύνολο δεδομένων εισόδου που συμφωνεί με τους περιορισμούς που θέτει ο ορισμός ενός υπολογιστικού προβλήματος αποτελεί ένα *στιγμιότυπο* (instance) του προβλήματος. Για παράδειγμα, οι αριθμοί $(15, 32)$ αποτελούν ένα στιγμιότυπο για το πρόβλημα του πολλαπλασιασμού δύο αριθμών. Η ακολουθία $(8, 1, 7, 5, 3, 6, 2, 4)$ αποτελεί ένα στιγμιότυπο για το πρόβλημα της ταξινόμησης ακεραίων αριθμών.

Κάθε στιγμιότυπο ενός προβλήματος έχει καμία, μία ή περισσότερες *λύσεις* (solutions). Η λύση ενός στιγμιότυπου για ένα πρόβλημα είναι κάθε σύνολο δεδομένων εξόδου που σε συνδυασμό με το στιγμιότυπο ικανοποιούν τους περιορισμούς που θέτει ο ορισμός του προβλήματος. Για παράδειγμα το 480 αποτελεί τη λύση του στιγμιότυπου $(15, 32)$ για το πρόβλημα του πολλαπλασιασμού. Η ακολουθία $(1, 2, 3, 4, 5, 6, 7, 8)$ αποτελεί τη λύση του στιγμιότυπου $(8, 1, 7, 5, 3, 6, 2, 4)$ για το πρόβλημα της ταξινόμησης.

Ορθότητα Αλγορίθμων. Τα προβλήματα που θα μελετήσουμε έχουν άπειρο αριθμό στιγμιοτύπων. Για να θεωρηθεί ένας αλγόριθμος *σωστός* (ή *ορθός* - correct), πρέπει να υπολογίζει σωστά τις λύσεις για όλα τα στιγμιότυπα του προβλήματος που λύνει. Για να τεκμηριώσουμε ότι την ορθότητα ενός αλγόριθμου για κάποιο πρόβλημα πρέπει να αποδείξουμε (με τυπική μαθηματική απόδειξη) ότι πράγματι ο αλγόριθμος υπολογίζει μία λύση για κάθε στιγμιότυπο του προβλήματος. Για να τεκμηριώσουμε τη μη-ορθότητα ενός αλγόριθμου για κάποιο πρόβλημα, αρκεί να περιγράψουμε ένα στιγμιότυπο του προβλήματος για το οποίο ο αλγόριθμος δεν υπολογίζει λύση.

2 Ανάλυση Αλγορίθμων

Για να συγκρίνουμε διαφορετικούς αλγόριθμους που λύνουν το ίδιο πρόβλημα απαιτείται να προσδιορίσουμε την ποσότητα των υπολογιστικών πόρων που απαιτούνται για την εκτέλεση του

αλγόριθμου (π.χ. υπολογιστικός χρόνος, θέσεις μνήμης, αριθμός επεξεργαστών, επικοινωνία μέσω δικτύου, κλπ.). Η διαδικασία της μαθηματικής εκτίμησης των υπολογιστικών πόρων που απαιτεί η εκτέλεση ενός αλγόριθμου ονομάζεται *ανάλυση του αλγόριθμου*.

Για να αναλύσουμε έναν αλγόριθμο, πρέπει να θεωρήσουμε ένα μοντέλο του υπολογιστή στον οποίο θα εκτελεστεί ο αλγόριθμος. Για τους σκοπούς του μαθήματος, θεωρούμε ότι κάθε αλγόριθμος θα υλοποιηθεί σε μία Υπολογιστική Μηχανή Άμεσης Προσπέλασης Μνήμης (Random Access Machine, RAM) με έναν επεξεργαστή. Στη Μηχανή Άμεσης Προσπέλασης Μνήμης, οι εντολές εκτελούνται ακολουθιακά και η κάθε στοιχειώδες υπολογιστικό βήμα έχει μοναδιαίο κόστος (για όλα τα είδη υπολογιστικών πόρων). Σαν στοιχειώδη υπολογιστικά βήματα θεωρούνται η προσπέλαση μιας θέσης μνήμης για ανάγνωση ή εγγραφή, οι βασικές εντολές (π.χ. ανάθεση τιμής σε μεταβλητή, συγκρίσεις αριθμών ή χαρακτήρων, εντολές ελέγχου ροής, κλπ.), και οι βασικές αριθμητικές και λογικές πράξεις (π.χ. πρόσθεση, πολλαπλασιασμός, λογικό και, λογική ή, κλπ.) εφόσον αφορούν αντικείμενα που μπορούν να αποθηκευτούν σε μικρό αριθμό θέσεων μνήμης. Το μοντέλο της Μηχανή Άμεσης Προσπέλασης Μνήμης είναι πολύ κοντά στα σύγχρονα μονο-επεξεργαστικά υπολογιστικά συστήματα.

Η πιο συνηθισμένη κατηγορία ανάλυσης ενός αλγορίθμου αφορά το χρόνο εκτέλεσης. Ο χρόνος εκτέλεσης ενός αλγορίθμου προσδιορίζεται από τον αριθμό των στοιχειωδών βημάτων που εκτελούνται. Το είδος των στοιχειωδών βημάτων τα οποία προσμετρώνται ποικίλει ανάλογα με το πρόβλημα (π.χ. για τα προβλήματα ταξινόμησης και αναζήτησης, ο χρόνος εκτέλεσης προσδιορίζεται από τις συγκρίσεις μεταξύ στοιχείων της συλλογής, για τα προβλήματα αριθμητικών πράξεων, ο χρόνος εκτέλεσης προσδιορίζεται από τις αριθμητικές πράξεις μεταξύ μονοψήφιων αριθμών, κλπ.)

Είναι εύλογο ότι η ποσότητα των υπολογιστικών πόρων εξαρτάται άμεσα από το μέγεθος του στιγμιότυπου που πρέπει να επιλυθεί. Για παράδειγμα, ο υπολογιστικός χρόνος για την επίλυση ενός στιγμιότυπου του Προβλήματος της Ταξινόμησης με δέκα αριθμούς δεν μπορεί να είναι συγκρίσιμος με τον υπολογιστικό χρόνο για την επίλυση ενός στιγμιότυπου με εκατό, χίλιους ή ένα εκατομμύριο αριθμούς.

Τα αποτελέσματα της ανάλυσης ενός αλγόριθμου εκφράζονται σαν συνάρτηση του μεγέθους του στιγμιότυπου που επιλύεται. Το *μεγέθος ενός στιγμιότυπου* είναι ο αριθμός των δυαδικών ψηφίων (bits) που απαιτούνται για να αναπαρασταθεί το στιγμιότυπο στη μνήμη του υπολογιστή. Για να διευκολυνθούμε, συνήθως υποθέτουμε ότι κάθε "βασική συνιστώσα" του στιγμιότυπου χρειάζεται περίπου τον ίδιο αριθμό δυαδικών ψηφίων για να αποθηκευθεί στη μνήμη του υπολογιστή και χρησιμοποιούμε τον αριθμό των "βασικών συνιστωσών" σαν μέτρο του μεγέθους του. Για παράδειγμα, στο Πρόβλημα της Ταξινόμησης βασικές συνιστώσες είναι οι αριθμοί που θα ταξινομηθούν.

2.1 Αρχή του Αναλλοίωτου και Ασυμπτωτική Εκτίμηση

Έχοντας καταλήξει σε ένα υπολογιστικό μοντέλο για την ανάλυση των αλγορίθμων και σε ένα μέτρο του μεγέθους των στιγμιότυπων ενός προβλήματος, ανακύπτει το ερώτημα ποια είναι η υλοποίηση του αλγόριθμου για την οποία θα γίνει η ανάλυση.

Η *αρχή του αναλλοίωτου* (principle of invariance) απαντάει ότι ουσιαστικά αυτό είναι αδιάφορο. Η αρχή του αναλλοίωτου λέει ότι δύο διαφορετικές υλοποιήσεις του ίδιου αλγόριθμου δεν

διαφέρουν σε αποτελεσματικότητα παρά μόνο κατά μια σταθερή αναλογία (π.χ. αυτή εξαρτάται από την τεχνολογία του υλικού και του λογισμικού του υπολογιστικού συστήματος στο οποίο θα εκτελεστεί ο αλγόριθμος). Συνεπώς, οι πολλαπλασιαστικές σταθερές δεν παίζουν ιδιαίτερα σημαντικό ρόλο στη *θεωρητική εκτίμηση* της αποδοτικότητας των αλγορίθμων και μπορούν να αγνοηθούν (δεν πρέπει να αγνοούνται όμως στην πράξη).

Ορισμός 1 (Αρχή του Αναλλοίωτου). *Ένας αλγόριθμος απαιτεί χρόνο της τάξεως $T(n)$ για να εκτελεσθεί, όπου n είναι το μέγεθος του στιγμιότυπου προς επίλυση, εάν υπάρχει θετική σταθερά c και υλοποίηση του αλγορίθμου ικανή να λύσει κάθε στιγμιότυπο μεγέθους n σε χρόνο όχι περισσότερο από $cT(n)$ βήματα.*

Με χρήση της αρχής του αναλλοίωτου, κάθε άλλη υλοποίηση του ίδιου αλγορίθμου θα έχει τον ίδιο χρόνο εκτέλεσης (με ενδεχόμενη αλλαγή της σταθεράς c). Αυτή η ιδιαίτερα σημαντική έννοια είναι γνωστή ως *ασυμπτωτική εκτίμηση*. Η ασυμπτωτική εκτίμηση καταδεικνύει ότι η συμπεριφορά του αλγόριθμου για μεγάλα στιγμιότυπα είναι αυτή που χαρακτηρίζει την αποδοτικότητά του.

Με απλά λόγια, το σημαντικό στη (θεωρητική) ανάλυση ενός αλγόριθμου είναι ο προσδιορισμός της *τάξης μεγέθους* του χρόνου εκτέλεσης σαν συνάρτηση του μεγέθους του στιγμιότυπου[2]. Όταν το μέγεθος του στιγμιότυπου n γίνει αρκετά μεγάλο, οι τιμές της συνάρτησης που καθορίζει την τάξη μεγέθους είναι σημαντικά μεγαλύτερες από οποιουσδήποτε άλλους όρους. Για παράδειγμα, όταν το n γίνει αρκετά μεγάλο, το 2^n είναι σημαντικά μεγαλύτερο από το $1000n^2$ και το n^2 είναι σημαντικά μεγαλύτερο από το $100\,n\log n$. Στη θεωρία λοιπόν, ένας αλγόριθμος με χρόνο εκτέλεσης μικρότερης τάξης μεγέθους είναι προτιμότερος από έναν αλγόριθμο με χρόνο εκτέλεσης μεγαλύτερης τάξης μεγέθους. Στην πράξη, ο πρώτος αλγόριθμος είναι γρηγορότερος από τον δεύτερο μόνο όταν τα στιγμιότυπα που επιλύονται είναι αρκετά μεγάλα.

Μερικές τάξεις μεγέθους είναι ιδιαίτερα συνηθισμένες. Όταν το $T(n)$ είναι ανεξάρτητο του n λέμε ότι ο χρόνος εκτέλεσης είναι *σταθερός* (σε αυτή την περίπτωση μπορούμε να θεωρήσουμε ότι $T(n) = 1$ με βάση την αρχή του αναλλοίωτου). Όταν $T(n) = n$, n^2, ή n^3 λέμε ότι ο χρόνος εκτέλεσης είναι *γραμμικός*, *τετραγωνικός*, και *κυβικός* αντίστοιχα. Όταν $T(n) = n^k$ για κάποια συγκεκριμένη σταθερά k, λέμε ότι ο χρόνος εκτέλεσης είναι *πολυωνυμικός*, ενώ όταν $T(n) = d^n$ για κάποια συγκεκριμένη σταθερά $d > 1$, λέμε ότι ο χρόνος εκτέλεσης είναι *εκθετικός*.

Παρατήρηση 1. Στη συζήτηση σχετικά με την ασυμπτωτική εκτίμηση, η έννοια του χρόνου εκτέλεσης μπορεί να αντικατασταθεί από οποιονδήποτε υπολογιστικό πόρο (π.χ. μνήμη, αριθμό επεξεργαστών, μηνύματα δικτύου, κλπ.) ως προς την οποία υπολογίζουμε την αποδοτικότητα του αλγόριθμου. □

2.2 Ανάλυση Χειρότερης Περίπτωσης

Η ασυμπτωτική εκτίμηση χρησιμοποιείται για την εξαγωγή συμπερασμάτων σχετικά με το χρόνο εκτέλεσης των αλγορίθμων, και γενικότερα σχετικά με τις απαιτήσεις τους σε διάφορα είδη υπολογιστικών πόρων. Στην ιδανική περίπτωση, θα θέλαμε ο χρόνος εκτέλεσης ενός αλγόριθμου να δίνεται από μία συνάρτηση του n, δηλαδή του μεγέθους του στιγμιότυπου εισόδου.

[2] Στο εξής το μέγεθος του στιγμιότυπου εισόδου θα συμβολίζεται με n εκτός αν ρητά αναφέρεται κάτι διαφορετικό. Επίσης, θα συμβολίζουμε με $\log n$ το λογάριθμο με βάση 2 και με $\ln n$ το φυσικό λογάριθμο του n.

Είναι όμως πολύ συνηθισμένο, ένας αλγόριθμος να παρουσιάζει εντελώς διαφορετικό χρόνο εκτέλεσης για διαφορετικά στιγμιότυπα του ίδιου μεγέθους. Υπάρχουν δηλαδή σημαντικές αποκλίσεις στο χρόνο εκτέλεσης, οι οποίες εξαρτώνται εκτός από το μέγεθος, και από τη δομή στιγμιότυπου εισόδου.

Για παράδειγμα, ας θεωρήσουμε τον αλγόριθμο γραμμικής αναζήτησης σε έναν πίνακα με n στοιχεία. Ο χρόνος εκτέλεσης του αλγορίθμου εξαρτάται από τη θέση του στοιχείου που αναζητούμε. Αν το στοιχείο βρίσκεται στις πρώτες θέσεις, ο χρόνος εκτέλεσης του αλγόριθμου είναι σταθερός, ενώ αν το στοιχείο βρίσκεται στις τελευταίες θέσεις ή δεν υπάρχει στον πίνακα, ο χρόνος εκτέλεσης είναι γραμμικός. Με άλλα λόγια, ο αλγόριθμος γραμμικής αναζήτησης έχει σταθερό χρόνο εκτέλεσης *καλύτερης περίπτωσης* και γραμμικό χρόνο εκτέλεσης *χειρότερης περίπτωσης*.

Τίθεται λοιπόν το ερώτημα ποια από τις δύο περιπτώσεις μπορεί να θεωρηθεί αντιπροσωπευτική για τη συμπεριφορά του αλγόριθμου (στη θεωρία και στην πράξη). Η απλούστερη και ασφαλέστερη επιλογή είναι να θεωρήσουμε μόνο τα στιγμιότυπα που μεγιστοποιούν το χρόνο εκτέλεσης του αλγόριθμου, δηλαδή να εφαρμόσουμε τη λεγόμενη *ανάλυση χειρότερης περίπτωσης* (worst-case analysis).

Από θεωρητικής άποψης, η ανάλυση χειρότερης περίπτωσης είναι ιδιαίτερα επιθυμητή επειδή παρέχει ένα άνω φράγμα στο χρόνο εκτέλεσης που ισχύει για κάθε στιγμιότυπο εισόδου. Όμως και στην πράξη, οι περισσότεροι αλγόριθμοι έχουν χρόνους εκτέλεσης που δεν διαφοροποιούνται σημαντικά από τα αποτελέσματα της ανάλυσης χειρότερης περίπτωσης (π.χ. βλ. αλγόριθμους ταξινόμησης merge-sort και heap-sort ή αλγόριθμο δυαδικής αναζήτησης). Για κάποιους αλγόριθμους που τα στιγμιότυπα εισόδου χειρότερης περίπτωσης είναι η εξαίρεση και όχι ο κανόνας (π.χ. πιθανοτική εκδοχή της quicksort, αναζήτηση με παρεμβολή), εφαρμόζουμε ανάλυση *μέσης περίπτωσης* (avegage-case analysis) για να έχουμε σαφή και ασφαλή εικόνα για το χρόνο εκτέλεσης του αλγόριθμου στην πράξη.

2.3 Ασυμπτωτικός Συμβολισμός

Η ασυμπτωτική εκτίμηση αγνοεί τις σταθερές και εξετάζει μόνο την τάξη μεγέθους του χρόνου εκτέλεσης ενός αλγόριθμου. Ο *ασυμπτωτικός συμβολισμός* χρησιμοποιείται για να εκφραστούν τα αποτελέσματα της ασυμπτωτικής εκτίμησης.

Για διευκόλυνση, ορίζουμε τις διάφορες μορφές ασυμπτωτικού συμβολισμού για συναρτήσεις με πεδίο ορισμού το σύνολο των φυσικών αριθμών $\mathbb{N} = \{0, 1, 2, \ldots\}$. Ο λόγος είναι ότι το μέγεθος ενός στιγμιότυπου είναι φυσικός αριθμός. Οι ορισμοί μπορούν εύκολα να επεκταθούν σε πραγματικές συναρτήσεις. Σε όλους τους ορισμούς υποθέτουμε ότι οι συναρτήσεις είναι θετικές (δηλ. παίρνουν τιμές στο $\mathbb{R}_+^*$).

Συμβολισμός Θ. Το Θ χρησιμοποιείται για τον *ακριβή προσδιορισμό* της τάξης μεγέθους μίας συνάρτησης. Συγκεκριμένα, για κάθε συνάρτηση $g(n)$, η κλάση συναρτήσεων $\Theta(g(n))$ αποτελείται από όλες τις συναρτήσεις με την ίδια τάξη μεγέθους. Τυπικά, δεδομένης μιας συνάρτησης $g(n)$, συμβολίζουμε με $\Theta(g(n))$ το σύνολο συναρτήσεων

$$\Theta(g(n)) = \{f(n) : \mathbb{N} \mapsto \mathbb{R}_+ \mid (\exists c_1, c_2 \in \mathbb{R}_+^*)\ (\exists n_0 \in \mathbb{N})\ (\forall n \geq n_0)\ c_1\, g(n) \leq f(n) \leq c_2\, g(n)\}$$

Με άλλα λόγια, μια συνάρτηση $f(n)$ ανήκει στην κλάση συναρτήσεων $\Theta(g(n))$ αν υπάρχουν θετικές σταθερές c_1, c_2 τέτοιες ώστε το $f(n)$ να είναι μεταξύ $c_1\, g(n)$ και $c_2\, g(n)$ για όλες τις τιμές του n που ξεπερνούν ένα κατώφλι n_0 (διαισθητικά, για όλες τις *μεγάλες* τιμές του n).

Ο συμβολισμός $\Theta(g(n))$ δηλώνει κλάση συναρτήσεων. Μολαταύτα, γράφουμε $f(n) = \Theta(g(n))$ (και όχι $f(n) \in \Theta(g(n))$) για να δηλώσουμε ότι η $f(n)$ ανήκει στην κλάση συναρτήσεων $\Theta(g(n))$. Το ίδιο συμβαίνει και με όλες τις μορφές ασυμπτωτικού συμβολισμού.

Εξ' ορισμού ο ασυμπτωτικός συμβολισμός αγνοεί τις σταθερές. Επιπλέον, ο ασυμπτωτικός συμβολισμός προσδιορίζει την τάξη μεγέθους των συναρτήσεων και αγνοεί όρους μικρότερης τάξης μεγέθους. Καθώς το n τείνει στο άπειρο, οι όροι μικρότερης τάξης μεγέθους καθίστανται αμελητέοι σε σχέση με τον κυρίαρχο όρο.

Για παράδειγμα, για κάθε πολυώνυμο δευτέρου βαθμού $f(n) = an^2 + bn + c$, $a > 0$, ισχύει ότι $f(n) = \Theta(n^2)$. Πράγματι, καθώς το n τείνει στο άπειρο, ο κυρίαρχος όρος n^2 καθορίζει τη συμπεριφορά της συνάρτησης $f(n)$. Γενικότερα, κάθε πολυώνυμο βαθμού d ανήκει στην κλάση $\Theta(n^d)$. Ομοίως, είναι $106n^2 + 104n^3 + 10^{-2}n^3 \log n = \Theta(n^3 \log n)$, αφού το $n^3 \log n$ είναι ο κυρίαρχος όρος. Ακόμη, κάθε σταθερή συνάρτηση ανήκει στην κλάση $\Theta(1)$, αφού το σημαντικό είναι ότι η συνάρτηση έχει την ίδια τιμή για κάθε n και όχι ποια ακριβώς είναι αυτή η τιμή.

Συμβολισμός O. Δεδομένης μιας συνάρτησης $g(n)$, συμβολίζουμε με $\mathrm{O}(g(n))$ το σύνολο των συναρτήσεων

$$\mathrm{O}(g(n)) = \{f(n) : \mathbb{N} \mapsto \mathbb{R}_+ \mid (\exists c \in \mathbb{R}_+^*)\, (\exists n_0 \in \mathbb{N})\, (\forall n \geq n_0)\ f(n) \leq c\, g(n)\}$$

Με άλλα λόγια, μια συνάρτηση $f(n)$ ανήκει στην κλάση συναρτήσεων $\mathrm{O}(g(n))$ αν υπάρχει σταθερά c τέτοια ώστε το $f(n)$ να μην ξεπερνά το $c\, g(n)$ για όλες τις μεγάλες τιμές του n.

Ο συμβολισμός O δίνει ένα ένα ασυμπωτικό άνω φράγμα στην τάξη μεγέθους μιας συνάρτησης. Όπως και για το συμβολισμό Θ, γράφουμε $f(n) = \mathrm{O}(g(n))$ αν και το $\mathrm{O}(g(n))$ ορίζει κλάση συναρτήσεων.

Συγκρίνοντας τους ορισμούς των συμβολισμών Θ και O, καταλήγουμε στο συμπέρασμα ότι κάθε συνάρτηση $f(n)$ που ανήκει στο $\Theta(g(n))$, ανήκει και στο $\mathrm{O}(g(n))$. Δηλαδή, $\Theta(g(n)) \subseteq \mathrm{O}(g(n))$.

Συμβολισμός Ω. Δεδομένης μιας συνάρτησης $g(n)$, συμβολίζουμε με $\Omega(g(n))$ το σύνολο των συναρτήσεων

$$\Omega(g(n)) = \{f(n) : \mathbb{N} \mapsto \mathbb{R}_+ \mid (\exists c \in \mathbb{R}_+^*)\, (\exists n_0 \in \mathbb{N})\, (\forall n \geq n_0)\ c\, g(n) \leq f(n)\}$$

Με άλλα λόγια, μια συνάρτηση $f(n)$ ανήκει στην κλάση συναρτήσεων $\mathrm{O}(g(n))$ αν υπάρχει σταθερά c τέτοια ώστε το $f(n)$ να μην υπολείπεται του $c\, g(n)$ για όλες τις μεγάλες τιμές του n.

Ο συμβολισμός Ω δίνει ένα ένα ασυμπωτικό κάτω φράγμα στην τάξη μεγέθους μιας συνάρτησης. Συγκρίνοντας τους ορισμούς των συμβολισμών Θ, O, και Ω, καταλήγουμε στο συμπέρασμα ότι οι συναρτήσεις $f(n)$ και $g(n)$ είναι $f(n) = \Theta(g(n))$ αν και μόνο αν $f(n) = \mathrm{O}(g(n))$ και $f(n) = \Omega(g(n))$. Για παράδειγμα, όταν $f(n) = \Theta(n^2)$, ισχύει επίσης ότι $f(n) = \mathrm{O}(n^2)$ και $f(n) = \Omega(n^2)$. Αντίστροφα, όταν $g(n) = \mathrm{O}(n^3)$ και $g(n) = \Omega(n^3)$, ισχύει ότι $g(n) = \Theta(n^3)$.

Συμβολισμός o. Το ασυμπτωτικό άνω φράγμα του συμβολισμού O δεν είναι πάντα ακριβές (tight), δηλ. δεν είναι πάντα το καλύτερο δυνατό. Για παράδειγμα, το φράγμα $2n^2 = \mathrm{O}(n^2)$ είναι ακριβές, ενώ το φράγμα $2n^2 = \mathrm{O}(n^3)$ δεν είναι. Ο συμβολισμός o δηλώνει ένα άνω φράγμα το οποίο δεν είναι ακριβές.

Δεδομένης μιας συνάρτησης $g(n)$, συμβολίζουμε με $\mathrm{o}(g(n))$ το σύνολο των συναρτήσεων

$$\mathrm{o}(g(n)) = \{f(n) : \mathbb{N} \mapsto \mathbb{R}_+ \mid (\forall c \in \mathbb{R}_+^*)\,(\exists n_0 \in \mathbb{N})\,(\forall n \geq n_0)\ f(n) < c\,g(n)\}\}$$

Ο ορισμός του o απαιτεί για *κάθε* θετική σταθερά c, το $f(n)$ να είναι μικρότερο του $c\,g(n)$ για όλες τις μεγάλες τιμές του n. Μια ισοδύναμη μαθηματική διατύπωση είναι ότι $f(n) = \mathrm{o}(g(n))$ αν $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$. Επομένως, $2n^2 = \mathrm{o}(n^3)$, αλλά $2n^2 \neq \mathrm{o}(n^2)$.

Συμβολισμός ω. Παρόμοια με το o, ο συμβολισμός ω χρησιμοποιείται για ένα κάτω φράγμα που δεν είναι ακριβές. Δεδομένης μιας συνάρτησης $g(n)$, συμβολίζουμε με $\omega(g(n))$ το σύνολο των συναρτήσεων

$$\omega(g(n)) = \{f(n) : \mathbb{N} \mapsto \mathbb{R}_+ \mid (\forall c \in \mathbb{R}_+^*)\,(\exists n_0 \in \mathbb{N})\,(\forall n \geq n_0)\ c\,g(n) < f(n)\}$$

Ο ορισμός του ω απαιτεί για *κάθε* θετική σταθερά c, το $f(n)$ να είναι μεγαλύτερο του $c\,g(n)$ για όλες τις μεγάλες τιμές του n. Μια ισοδύναμη μαθηματική διατύπωση είναι ότι $f(n) = \omega(g(n))$ αν $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$. Επομένως, $2n^2 = \mathrm{o}(n)$, αλλά $2n^2 \neq \mathrm{o}(n^2)$.

Άσκηση 1. Έστω $f(n)$ και $g(n)$ θετικές συναρτήσεις με πεδίο ορισμού τους φυσικούς αριθμούς. Για κάθε μια από τις παρακάτω προτάσεις είτε να αποδείξετε ότι είναι αληθής είτε να εξηγήσετε γιατί είναι ψευδής:

1. $10f(n) + 10^{10} = \mathrm{O}(f(n))$.
2. $f(n) + g(n) = \Theta(\min\{f(n), g(n)\})$.
3. $f(n) + g(n) = \Omega(\min\{f(n), g(n)\})$.
4. $f(n) + g(n) = \mathrm{O}(\max\{f(n), g(n)\})$.
5. Αν $f(n) = \mathrm{O}(g(n))$ και $f(n), g(n) > 1$ για κάθε $n \in \mathbb{N}$, τότε $\log f(n) = \mathrm{O}(\log g(n))$.
6. Αν $f(n) = \Theta(g(n))$, τότε $2^{f(n)} = \Theta(2^{g(n)})$.
7. $f(n) = \Omega(f(n/2))$.

Λύση. Οι 1, 3, 4, και 7 είναι αληθείς και αποδεικνύονται με τους ορισμούς. Οι 2 και 6 είναι ψευδείς και μπορούν να βρεθούν αντιπαραδείγματα. Η 5 είναι ψευδής στη γενική περίπτωση, αλλά αληθεύει αν θεωρήσουμε μόνο αύξουσες συναρτήσεις. □

Άσκηση 2. Να τοποθετηθούν οι παρακάτω συναρτήσεις σε αύξουσα σειρά τάξης μεγέθους:

2^{5n}	$\log^4 n$	$(\log n)^{100} \log\log n$	$n \log\log n$	$n^{0.1} \log\log n$
2^n	$n^{0.6}$	$2^n + n^{2^{100}}$	$n^{1/\log n}$	$\log(n!)$
$n^{\log n}$	$\log\log n$	$2^{\log^3 n}$	$\frac{n}{\log_n 2} + n$	$(\log n)^{\log n}$

Λύση. Είναι

$$n^{1/\log n} = \Theta(1),\ \log\log n,\ \log^4 n,\ (\log n)^{100}\log\log n,\ n^{0.1}\log\log n,$$

$$n^{0.6},\ n\log\log n,\ \log(n!) = \Theta(n\log n),\ \tfrac{n}{\log_n 2} + n = \Theta(n\log n),$$

$$(\log n)^{\log n} = \Theta(n^{\log\log n}),\ n^{\log n},\ 2^{\log^3 n} = \Theta(n^{\log^2 n}),\ 2^n,\ 2^n + n^{2^{100}} = \Theta(2^n),\ 2^{5n}$$

Άσκηση 3. Στον πίνακα που ακολουθεί, σημειώστε σε ποιες περιπτώσεις η συνάρτηση $f(n)$ ανήκει στις κλάσεις $\Theta(g(n)), \mathrm{O}(g(n)), \mathrm{o}(g(n)), \Omega(g(n))$, και $\omega(g(n))$ και σε ποιες όχι.

$f(n)$	$g(n)$	$\Theta(g(n))$	$\mathrm{O}(g(n))$	$\mathrm{o}(g(n))$	$\Omega(g(n))$	$\omega(g(n))$
2^{n+5}	$2^n + 2^5 + n^{100}$					
$n^4 - n^3$	$16^{\log n}$					
5^{4n}	10^{2n}					
$n^{1/\log\log n}$	$n^{0.001}$					
$n!$	n^n					
$n^{\log^{20} n}$	2^n					

Λύση.

$f(n)$	$g(n)$	$\Theta(g(n))$	$\mathrm{O}(g(n))$	$\mathrm{o}(g(n))$	$\Omega(g(n))$	$\omega(g(n))$
2^{n+5}	$2^n + 2^5 + n^{100}$	ναι	ναι	όχι	ναι	όχι
$n^4 - n^3$	$16^{\log n}$	ναι	ναι	όχι	ναι	όχι
5^{4n}	10^{2n}	όχι	όχι	όχι	ναι	ναι
$n^{1/\log\log n}$	$n^{0.001}$	όχι	ναι	ναι	όχι	όχι
$n!$	n^n	όχι	ναι	ναι	όχι	όχι
$n^{\log^{20} n}$	2^n	όχι	ναι	ναι	όχι	όχι