Κεφάλαιο 1ο

Εισαγωγή

1.1 Αλγόριθμοι

Η λέξη "αλγόριθμος" προέρχεται από μια μελέτη του Πέρση μαθηματικού Abu Ja'far Mohammed ibn al Khowarizmi με τίτλο "Κανόνες σύνθεσης και αναγωγές", ο οποίος έζησε περί το 825 μ.Χ. στην πόλη Χίβα του σημερινού Ουζμπεκιστάν. Πέντε αιώνες αργότερα η μελέτη αυτή μεταφράστηκε στα λατινικά και άρχιζε με τη φράση "Algoritmi dixit ..." (ο αλγόριθμος λέει ...). Η μελέτη του al Khowarizmi υπήρξε η πρώτη πραγματεία άλγεβρας (όρος που κι αυτός προέρχεται από αραβικό al-jabr (= αποκατάσταση), γιατί ένας από τους σκοπούς της άλγεβρας είναι η αποκατάσταση της ισότητας μέσα σε μια εξίσωση. Η λέξη επέζησε επί χίλια χρόνια ως σπάνιος όρος, που σήμαινε κάτι σαν "συστηματική διαδικασία αριθμητικών χειρισμών". Τη σημερινή του χρήση απόκτησε από την αρχή του 20ου αιώνα με την ανάπτυξη της ομώνυμης θεωρίας και φυσικά με την επικαιρότητα των ηλεκτρονικών υπολογιστών. Η έννοια του αλγόριθμου είναι βασική στον προγραμματισμό των ΗΥ. Ένας ορισμός που θα μπορούσε να δοθεί είναι ο ακόλουθος:

Αλγόριθμος είναι η ακριβής περιγραφή μιας αυστηρά καθορισμένης σειράς ενεργειών (βημάτων) για τη λύση ενός προβλήματος. Κάθε αλγόριθμος έχει κανένα, ένα ή περισσότερα δεδομένα εισόδου και τουλάχιστον ένα αποτέλεσμα στην έξοδο.

Αξίζει να αναφερθεί ότι προκειμένου να επιτύχουμε την «ακριβή περιγραφή» σε ένα αλγόριθμο χρησιμοποιούμε γλώσσα που μπορεί να περιγράφει σειρές ενεργειών με τρόπο αυστηρό, χωρίς ασάφειες και διφορούμενα. Τέτοιες γλώσσες είναι οι γλώσσες προγραμματισμού (με την προϋπόθεση ότι η σημασιολογία τους είναι αυστηρά διατυπωμένη), τα μαθηματικά μοντέλα, κάποιες συμβολικές γλώσσες που χρησιμοποιούν αυστηρά καθορισμένους κανόνες περιγραφής, καθώς και κατάλληλα διαμορφωμένα υποσύνολα των φυσικών (ομιλούμενων) γλωσσών.

Ιστορικά, ένας από τους πρώτους αλγόριθμους είναι ο αλγόριθμος του Ευκλείδη, για την εύρεση του Μέγιστου Κοινού Διαιρέτη (ΜΚΔ) δύο θετικών ακεραίων x και y. Ο αλγόριθμος ΜΚΔ περιγράφεται σε ομιλούμενη γλώσσα ως εξής: *"Διαίρεσε το x με το y, και έστω z το υπόλοιπο. Αν z = 0, τότε ΜΚΔ είναι ο y. Αν z ≠ 0, τότε επανάλαβε το βήμα με τους ακέραιους y και z αντί για x και y".*

Πιο αυστηρά, ο αλγόριθμος αυτός μπορεί να εκφραστεί ως εξής:

Αλγόριθμος Ευκλείδης
Δεδομένα // x, y //
z ← y
Όσο z ≠ 0 **επανάλαβε**
    z ← x **mod** y
    x ← y
    y ← z
Τέλος_επανάληψης
Αποτελέσματα // x //
Τέλος Ευκλείδης

Παρατηρούμε ότι για την περιγραφή του αλγορίθμου χρησιμοποιείται μια σαφής γλώσσα στην οποία το όνομα του αλγορίθμου (Ευκλείδης) καθορίζει την αρχή και το τέλος του, ενώ τα δεδομένα (x, y) βρίσκονται μεταξύ των συμβόλων //.../. όπου ο ΜΚΔ είναι η τιμή που παίρνει τελικά η μεταβλητή x. Το αποτέλεσμα βρίσκεται και αυτό εντός των συμβόλων //.../.

Η εύρεση του ΜΚΔ είναι μια επαναληπτική διαδικασία που συνεχίζεται όσο το υπόλοιπο της διαίρεσης z και x διά του y είναι διάφορο του μηδενός. Η επαναληπτική διαδικασία έχει την έννοια «όσο ισχύει η συνθήκη επαναλάμβανε τη διαδικασία, αλλιώς μην εκτελείς άλλο τη διαδικασία και συνέχισε στο επόμενο βήμα» και περιγράφεται με το ακόλουθο γενικό σχήμα:

Όσο *συνθήκη* **επανάλαβε**
    .
    . Διαδικασία
    .
Τέλος_επανάληψης

Οι εντολές του τύπου x ← y δεν έχουν την έννοια της ισότητας, αλλά της *εκχώρησης* τιμής του δεξιού μέλους στη μεταβλητή του αριστερού μέλους. Αυτό σημαίνει ότι μετά την εκτέλεση της εντολής η μεταβλητή του αριστερού μέλους θα έχει τιμή ίση με αυτή του δεξιού μέλους. Το δεξί μέλος μπορεί να είναι μια σταθερά, μια μεταβλητή ή γενικότερα μια παράσταση. Αξίζει να σημειωθεί ότι σε κάποια βιβλία για την ενέργεια αυτή χρησιμοποιείται το ίσον '=' ή και ο συνδυασμός ':='.

Με βάση τα παραπάνω, ο αλγόριθμος του Ευκλείδη περιγράφεται πλήρως με ακρίβεια και σαφήνεια. Επομένως μπορούμε να εμπιστευθούμε την εκτέλεσή του σε κάποιον, ο οποίος μπορεί να εκτελεί τα βήματα του αλγορίθμου και ο οποίος δε χρειάζεται να γνωρίζει τι ακριβώς επιδιώκει ο αλγόριθμος ή ακόμα και σε ένα υπολογιστή. Αξίζει βέβαια να αναφερθεί ότι θα ήταν παράλογο να υποθέσουμε την ύπαρξη μιας υποθετικής εντολής ΜΚΔ, που θα υπολόγιζε κατ' ευθείαν το μέγιστο κοινό διαιρέτη, εφόσον τότε δε θα είχε έννοια η ύπαρξη του αλγορίθμου.

Σαν παράδειγμα, ας υποθέσουμε ότι θέλουμε να βρούμε το ΜΚΔ των αριθμών 27 και 78. Χρησιμοποιώντας τον ανωτέρω αλγόριθμο Ευκλείδης, σε κάθε επανάληψη παίρνουμε για τα χ, y και z, τις τιμές που παρουσιάζονται στον επόμενο πίνακα. από τον οποίο εύκολα συνάγεται ότι ο ΜΚΔ των αριθμών 27 και 78 είναι ο 3, που βρίσκεται κατά την τέταρτη επανάληψη.

Επανάληψη	**x**	**y**	**z**
	27	78	78
1η	78	27	27
2η	27	24	24
3η	24	3	3
4η	3	0	0

Είναι σημαντικό να αναφερθεί ότι ο αλγόριθμος που παρουσιάσαμε, μπορεί να απαντήσει όχι μόνο στη συγκεκριμένη ερώτηση, "να βρεθεί ο ΜΚΔ των 27 και 78", αλλά σε όλες τις παρόμοιες ερωτήσεις. Λύνει, δηλαδή, ένα πρόβλημα (problem). Ένα πρόβλημα είναι ένα σύνολο από παραπλήσιες ερωτήσεις με δεδομένα (π.χ. αριθμούς, γραφήματα, κ.λπ.) που διαφέρουν ποσοτικά από ερώτηση σε ερώτηση. Κάθε μία από τις ερωτήσεις αυτές λέγεται *στιγμιότυπο* (instance) του προβλήματος. Έτσι, η εύρεση του ΜΚΔ των 27 και 78 είναι ένα στιγμιότυπο του προβλήματος της εύρεσης του ΜΚΔ δύο θετικών ακεραίων.

Δηλαδή, αν ακολουθήσουμε τα βήματα του αλγόριθμου, θα τερματίσουμε έχοντας πάρει τη σωστή απάντηση για οποιοδήποτε ζευγάρι θετικών ακεραίων. Ωστόσο, ένα ιδιαίτερα ενδιαφέρον θεωρητικό ερώτημα είναι: «πώς μπορούμε να είμαστε βέβαιοι ότι ο αλγόριθμος λύνει οποιοδήποτε στιγμιότυπο του προβλήματος;» Συνήθως, για να βεβαιωθούμε ότι ένας αλγόριθμος πραγματικά λύ-

νει ένα (καλώς διατυπωμένο μαθηματικό) πρόβλημα, πρέπει να μπορούμε να αποδείξουμε την ορθότητά του με αυστηρό τρόπο. Αξίζει να αναφερθεί, ότι η ορθότητα του αλγορίθμου του Ευκλείδη που περιγράψαμε παραπάνω, αποδεικνύεται από τον ίδιο τον Ευκλείδη στο 7ο βιβλίο των «Στοιχείων» του.

1.2 Ιδιότητες αλγορίθμων

Από τα προηγούμενα μπορεί κανείς να συμπεράνει ότι κάθε αλγόριθμος πρέπει να διαθέτει κάποιες συγκεκριμένες ιδιότητες. Από αυτές, αξίζει να αναφερθούν οι ακόλουθες:

- είσοδος
- έξοδος
- περατότητα
- καθοριστικότητα και
- αποτελεσματικότητα.

Από τις ιδιότητες αυτές, η είσοδος και η έξοδος σχετίζονται με την ύπαρξη δεδομένων και αποτελεσμάτων, αντίστοιχα, στον αλγόριθμο. Στις υπόλοιπες ιδιότητες αναφερόμαστε στη συνέχεια.

Η ***περατότητα*** ενός αλγορίθμου αναφέρεται στην ικανότητά του να ολοκληρώνεται μετά από πεπερασμένα βήματα εκτέλεσης των εντολών του. Όταν μια διαδικασία δεν τελειώνει μετά από ένα συγκεκριμένο αριθμό βημάτων δεν αποτελεί αλγόριθμο, αλλά απλά *υπολογιστική διαδικασία*. Για παράδειγμα, ας θεωρήσουμε την εξής ακολουθία εντολών:

Αλγόριθμος Ευκλείδης
Δεδομένα // x, y //
z ← y
Όσο z ≠ 0 **επανάλαβε**
    x ← y
    y ← z
Τέλος_επανάληψης
Αποτελέσματα // x //
Τέλος Ευκλείδης

Είναι εύκολο να διαπιστώσουμε ότι η ακολουθία αυτή δε θα ολοκληρωθεί ποτέ, εφόσον η τιμή του z δε μεταβάλλεται, επομένως η συνθήκη τερματισμού δε θα ικανοποιηθεί ποτέ.

Η ***καθοριστικότητα*** σε ένα αλγόριθμο σχετίζεται την ικανότητα εκτέλεσης κάθε εντολής. Ας θεωρήσουμε την επόμενη ακολουθία εντολών.

Αλγόριθμος Ευκλείδης
Δεδομένα // x, y //
z ← y
y ← 0
Όσο z ≠ 0 **επανάλαβε**
        z ← x **mod** y
        x ← y
        y ← z
Τέλος_επανάληψης
Αποτελέσματα // x //
Τέλος Ευκλείδης

Η ακολουθία αυτή δεν μπορεί να εκτελεστεί, εφόσον η εντολή z ← x **mod** y θα προσπαθήσει να εκτελέσει διαίρεση με το μηδέν, που βέβαια δεν είναι επιτρεπτό.

Τέλος, η ***αποτελεσματικότητα*** έχει να κάνει με το γεγονός ότι κάθε εντολή δεν αρκεί να έχει ορισθεί, αλλά πρέπει να είναι και εκτελέσιμη.

Αλγόριθμος Ευκλείδης
Δεδομένα // x, y //
y ← z
Όσο z ≠ 0 **επανάλαβε**
        z ← x **mod** y
        x ← y
        y ← z
Τέλος_επανάληψης
Αποτελέσματα // x //
Τέλος Ευκλείδης

Στην ανωτέρω ακολουθία εντολών, η εντολή y ← z δεν μπορεί να εκτελεστεί, εφόσον το z δεν έχει κάποια τιμή, επομένως το y δεν μπορεί να πάρει τιμή.

1.3 Παράσταση αλγορίθμων

Για την παράσταση των αλγορίθμων έχουν χρησιμοποιηθεί πλήθος μεθόδων, μερικές από τις οποίες είναι η ψευδογλώσσα, τα διαγράμματα ροής, τα διαγράμματα Nassi-Shneiderman (N-S), και τα διαγράμματα HIPO. Οι τρόποι μέθοδοι αυτές δεν παρουσιάζουν τα ίδια πλεονεκτήματα. Συνήθως χρησιμοποιούνται εκείνες αυτές που είναι σύμφωνες με το πνεύμα του *δομημένου προγραμματισμού*, ο οποίος χρησιμοποιεί τις τρεις βασικές αλγοριθμικές δομές της ακολουθίας, της επιλογής και της επανάληψης.

Δομημένη σχεδίαση αλγορίθμων

Με τον όρο δομημένη σχεδίαση αλγορίθμων αναφερόμαστε στη σχεδίαση αλγορίθμων χρησιμοποιώντας τρεις βασικές δομές ελέγχου: ***ακολουθία, επιλογή*** και ***επανάληψη***.

Πρώτοι οι Bohm και Jacopini απέδειξαν, το 1966, ότι οποιοσδήποτε αλγόριθμος μπορεί να περιγραφεί με τις τρεις αυτές βασικές δομές ελέγχου.

Στην ακολουθία (sequence), χρησιμοποιούμε απλές εντολές που εκτελούνται η μια μετά την άλλη ακολουθιακά.

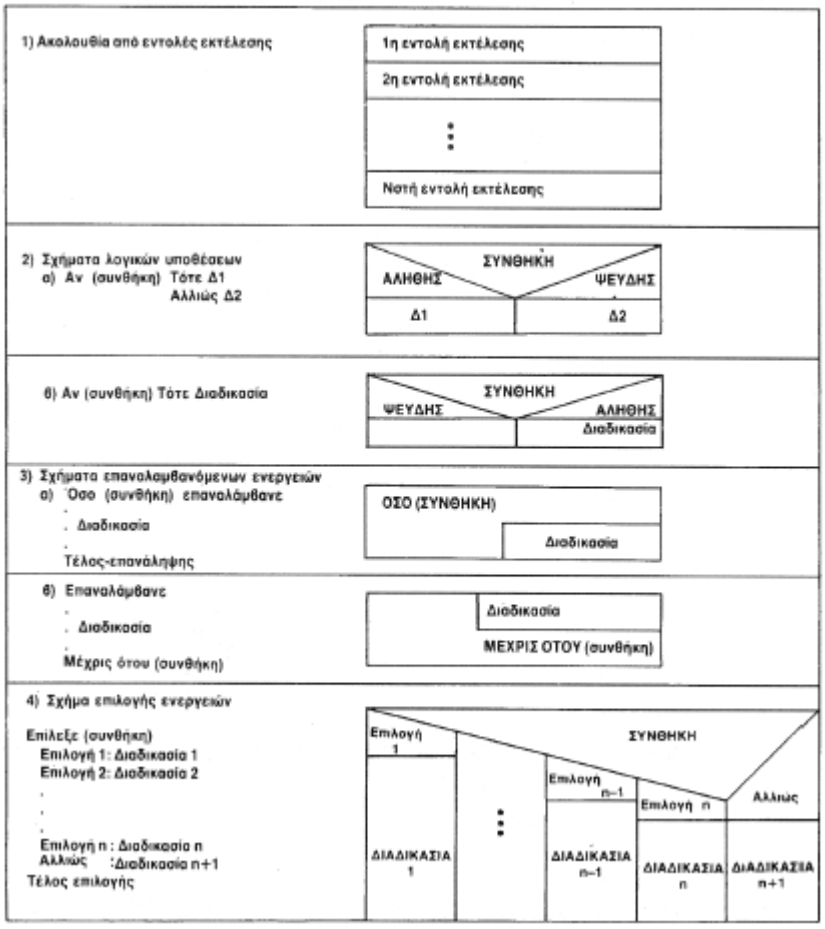
Στην επιλογή (πιο συνηθισμένη της μορφή είναι η αν-τότε-αλλιώς ή if-then-else), χρησιμοποιούμε σχήματα λογικών υποθέσεων. Το ακόλουθο παράδειγμα σχήματος, έχει την εξής σημασία : «αν ισχύει η συνθήκη, τότε εκτέλεσε τη Δ1, ενώ αν δεν ισχύει η συνθήκη, τότε εκτέλεσε τη Δ2».

Αν συνθήκη **τότε**
    Δ1
αλλιώς
    Δ2
Τέλος_αν

Στην επανάληψη (μια γνωστή μορφή της είναι η όσο-επανάλαβε ή while-do) χρησιμοποιούμε σχήματα όπως το ακόλουθο, που έχει την έννοια ότι όσο ισχύει η συνθήκη, η διαδικασία επαναλαμβάνεται.

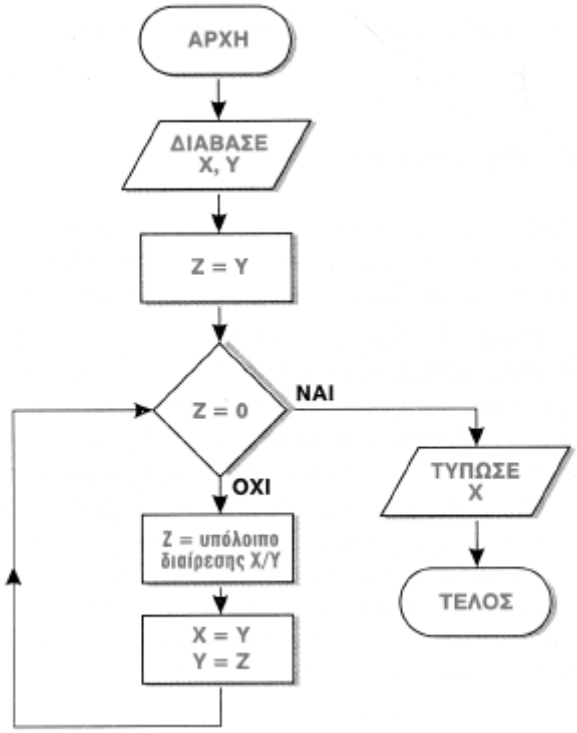
Όσο *συνθήκη* **επανάλαβε**
    .
    . Διαδικασία
    .
Τέλος_επανάληψης

Μια από τις πιο αποτελεσματικές δομημένες διαγραμματικές τεχνικές είναι η τεχνική Nassi-Shneiderman, οι βασικές συνιστώσες της οποίας περιγράφονται στο σχήμα 1.1.



Σχήμα 1.1. *Δομές ελέγχου του δομημένου προγραμματισμού και αναπαραστάσεις στη μορφή N-S.*

Τα ***διαγράμματα ροής*** (flow charts) ή λογικά διαγράμματα αποτελούν ένα εναλλακτικό τρόπο παρουσίασης αλγορίθμων χρησιμοποιώντας γεωμετρικά σχήματα για να καθορίσουν τα βήματα επεξεργασίας καθώς και τα διάφορα μέσα που χρησιμοποιούνται και δεν εντάσσονται στις δομημένες τεχνικές. Ως παράδειγμα, στο σχήμα 1.2 δίνουμε τον αλγόριθμο Ευκλείδης με τη χρήση διαγράμματος ροής.



Σχήμα 1.2. *Ο αλγόριθμος Ευκλείδης με διάγραμμα ροής.*

Στο βιβλίο αυτό χρησιμοποιείται για την παράσταση των αλγορίθμων η ***ψευδογλώσσα*** (ή ***ψευδοκώδικας*****)** αναλυτική περιγραφή της οποίας γίνεται στο επόμενο κεφάλαιο.

1.4 Τα δεδομένα

1.4.1. Πληροφορίες και δεδομένα

Όπως έχει αναφερθεί, ένας αλγόριθμος παίρνει στην είσοδο κάποια δεδομένα, τα επεξεργάζεται μέσα από μια σειρά βήματα και δίνει ως έξοδο τα αποτελέσματα. Σχετικά με τον όρο δεδομένα, είναι χρήσιμος ο ακόλουθος ορισμός:

Δεδομένο (datum-πληθ. data) είναι μια παράσταση γεγονότων, εννοιών ή εντολών σε τυποποιημένη μορφή που είναι κατάλληλη για επικοινωνία, ερμηνεία ή επεξεργασία από τον άνθρωπο ή από αυτόματα μέσα.

Η ***επεξεργασία δεδομένων*** (data processing), η οποία στην πράξη πραγματοποιείται μέσω αλγορίθμων, αναφέρεται στην εκτέλεση διαφόρων πράξεων/ λειτουργιών πάνω στα δεδομένα. Το αποτέλεσμα της επεξεργασίας δεδομένων

είναι η ***πληροφορία.*** Έτσι, θα μπορούσε κανείς γενικεύοντας να πει ότι ένας αλγόριθμος μετατρέπει τα δεδομένα σε πληροφορία. Για παράδειγμα, ένας τηλεφωνικός αριθμός και ένα ονοματεπώνυμο αποτελούν δεδομένα. Δεν παρέχουν όμως καμία πληροφορία. Πληροφορία παράγεται μόνο αν σχετισθούν μεταξύ τους, ότι δηλαδή κάποιο τηλέφωνο ανήκει σε κάποιο συγκεκριμένο συνδρομητή. Με βάση τις πληροφορίες λαμβάνονται αποφάσεις και γίνονται διάφορες ενέργειες. Στη συνέχεια οι ενέργειες αυτές παράγουν νέα δεδομένα, αυτές νέες πληροφορίες, οι τελευταίες νέες αποφάσεις και ενέργειες κ.ο.κ. όπως φαίνεται στο σχήμα 1.3. Η διεργασία αυτή μπορεί να επαναλαμβάνεται, οι πληροφορίες που παρήχθησαν, να επανα-τροφοδοτούν μέσω ανάδρασης την είσοδο για επανάληψη του κύκλου κ.ο.κ.

Σχήμα 1.3 *Σχέση δεδομένων και πληροφοριών*

1.4.2 Μελέτη των δεδομένων

Τα δεδομένα ενδιαφέρουν το συγκεκριμένο βιβλίο (αλλά και το γενικότερο πεδίο της αλγοριθμικής) για μια σειρά από λόγους, και μελετώνται από διαφορετικές σκοπιές. Πιο συγκεκριμένα, η αλγοριθμική μελετά τα δεδομένα κυρίως από τις σκοπιές του *υλικού* και των *γλωσσών προγραμματισμού*.

Το ***υλικό*** επιτρέπει τα δεδομένα ενός προγράμματος να αποθηκεύονται στην κύρια μνήμη ή και στις περιφερειακές συσκευές ενός υπολογιστή με διάφορες μορφές. Όπως είναι γνωστό, η πρόσβαση στην κύρια μνήμη του υπολογιστή είναι ταχύτερη από ότι στη δευτερεύουσα. Επομένως, για την πρόσβαση σε δεδομένα που βρίσκονται στη βοηθητική μνήμη είναι πιθανό (και πράγματι συμβαίνει) να χρησιμοποιούνται διαφορετικοί αλγόριθμοι για την επεξεργασία τους. Επομένως, το υλικό του υπολογιστή έχει επίδραση στο είδος των αλγορίθμων που θα χρησιμοποιηθούν.

Οι ***γλώσσες προγραμματισμού*** μας ενδιαφέρουν διότι κάθε γλώσσα μπορεί να υποστηρίζει τη χρήση διαφόρων ***τύπων δεδομένων*** (data types). Οι πιο συνή-

θεις τύποι δεδομένων είναι οι ακέραιοι και οι πραγματικοί αριθμοί, τα λογικά δεδομένα και οι συμβολοσειρές. Οι ***πραγματικοί αριθμοί*** διακρίνονται σε ***σταθερής υποδιαστολής*** (fixed point) και ***κινητής υποδιαστολής*** (floating point). Τα ***λογικά δεδομένα*** έχουν μόνο δύο τιμές, αληθής (true) ή ψευδής (false). Οι ***συμβολοσειρές*** (strings) χρησιμοποιούνται για την παράσταση αλφαριθμητικών δεδομένων. Σε κάθε τύπο δεδομένων μπορούμε να εφαρμόσουμε διαφορετικές πράξεις. Για παράδειγμα, μεταξύ δύο ακεραίων μπορούμε να εφαρμόσουμε ακέραια διαίρεση, ενώ μεταξύ δύο πραγματικών δεν μπορούμε να κάνουμε κάτι τέτοιο. Επομένως, όταν σχεδιάζουμε ένα αλγόριθμο μας ενδιαφέρει άμεσα το είδος των τύπων δεδομένων που υποστηρίζονται, ώστε να μπορούμε να εκτελέσουμε τις κατάλληλες πράξεις. Αξίζει να αναφερθεί ότι για το χειρισμό διαφορετικών τύπων δεδομένων αναπτύσσονται διαρκώς νέοι αλγόριθμοι με σκοπό τον αποτελεσματικό και ταχύ χειρισμό τους.

1.4.3 Οι Δομές Δεδομένων

Δομή δεδομένων είναι ένα σύνολο αποθηκευμένων δεδομένων, τα οποία είναι έτσι οργανωμένα, ώστε να υπόκεινται σε συγκεκριμένες απαιτούμενες επεξεργασίες. Ο όρος αναφέρεται σε ένα σύνολο δεδομένων μαζί με ένα σύνολο λειτουργιών που επιτρέπονται στα δεδομένα αυτά. Πρέπει να αναφερθεί ότι οι δομές δεδομένων είναι πολύ στενά συνδεδεμένες με την έννοια του αλγορίθμου. Είναι πολύ χαρακτηριστική η ακόλουθη «σχέση» που διατύπωσε ο N. Wirth, δημιουργός της γλώσσας Pascal.

Αλγόριθμοι + Δομές Δεδομένων = Προγράμματα

Η ανωτέρω σχέση έχει την έννοια ότι αν κανείς διαθέτει τον κατάλληλο αλγόριθμο και τις δομές δεδομένων, οι οποίες θα χρησιμοποιηθούν, είναι εντελώς άμεση η μετατροπή και υλοποίηση σε πρόγραμμα σε γλώσσα υπολογιστή.

Κάθε δομή δεδομένων αποτελείται, στην πιο γενική της μορφή, από ένα σύνολο στοιχείων ή κόμβων. Οι βασικές λειτουργίες ή πράξεις επί των δομών δεδομένων είναι οι ακόλουθες:

- ✓ **Προσπέλαση** (access), πρόσβαση σε έναν κόμβο με σκοπό να εξεταστεί ή να τροποποιηθεί το περιεχόμενό του.
- ✓ **Ανάκτηση** (retrieval), η με οποιονδήποτε τρόπο λήψη (ανάγνωση) του περιεχομένου ενός κόμβου.
- ✓ **Αναζήτηση** (searching) ενός συνόλου στοιχείων δεδομένων προκειμένου να εντοπιστούν ένα ή περισσότερα στοιχεία, που έχουν μια δεδομένη ιδιότητα.
- ✓ **Εισαγωγή** (insertion), η προσθήκη ή δημιουργία νέων κόμβων σε μια υπάρχουσα δομή.

- ✓ **Διαγραφή** (deletion) που συνιστά το αντίθετο της εισαγωγής.
- ✓ **Ταξινόμηση** (sorting), όπου τα στοιχεία μιας δομής διατάσσονται κατά αύξουσα ή φθίνουσα τάξη.
- ✓ **Συγχώνευση** (merging), κατά την οποία δύο ή περισσότερες δομές συνενώνονται σε μια ενιαία δομή.
- ✓ **Διαχωρισμός** (separation), που αποτελεί την αντίστροφη πράξη της συγχώνευσης.
- ✓ **Προσάρτηση** (append), κατά την οποία μία δομή επικολλάται στο τέλος μιας άλλης.
- ✓ **Αντιγραφή** (copying), όπου κάποιοι οι κόμβοι μιας δομής αντιγράφονται σε μια άλλη.

Πρέπει να αναφερθεί ότι οι ανωτέρω λειτουργίες στην πράξη σπάνια υπάρχουν όλες σε μια δομή. Συνήθως παρατηρείται το φαινόμενο μια δομή δεδομένων να είναι αποδοτικότερη από μια άλλη για κάποια λειτουργία π.χ. την αναζήτηση, αλλά λιγότερο αποδοτική για κάποια άλλη λειτουργία, όπως π.χ. την εισαγωγή. Αυτό εξηγεί τόσο την ύπαρξη διαφορετικών δομών όσο και τη σπουδαιότητα της επιλογής της κατάλληλης δομής κάθε φορά.

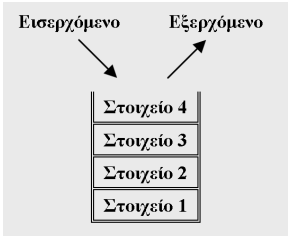
Οι πιο ευρέως χρησιμοποιούμενες δομές δεδομένων είναι ο πίνακας (table), η στοίβα (stack), η ουρά (queue), η συνδεδεμένη λίστα (linked list), το δέντρο (tree) και ο γράφος (graph).

Ο ***Πίνακας*** αποτελείται από ένα σύνολο ομοειδών απλών στοιχείων, καθένα από τα οποία καθορίζεται με τη βοήθεια ενός ή περισσοτέρων ***δεικτών*** *(index)*. Ένας πίνακας μπορεί να είναι μίας, δύο ή περισσοτέρων διαστάσεων, ανάλογα με το πλήθος δεικτών που χρειάζονται για να καθοριστεί η θέση του. Στο ακόλουθο σχήμα, παρουσιάζουμε ένα δισδιάστατο πίνακα Α με 4 γραμμές και 3 στήλες.

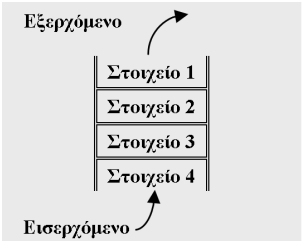
Δισδιάστατος Πίνακας Α 3x4

Στοιχείο 1,1	Στοιχείο 1,2	Στοιχείο 1,3
Στοιχείο 2,1	Στοιχείο 2,2	Στοιχείο 2,3
Στοιχείο 3,1	Στοιχείο 3,2	Στοιχείο 3,3
Στοιχείο 4,1	Στοιχείο 4,2	Στοιχείο 4,3

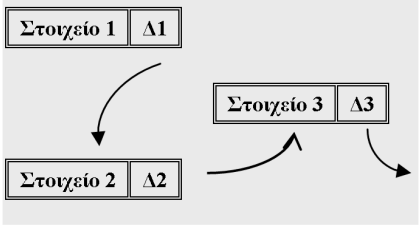
Μία ***στοίβα*** *(stack)* είναι μια γραμμική διάταξη στοιχείων, στην οποία εισάγονται και εξάγονται στοιχεία μόνο από το ένα άκρο. Η λειτουργία της εισαγωγής αποκαλείται ***ώθηση*** *(push)* και της εξαγωγής ***απώθηση*** *(pull ή pop)*. Η φιλοσοφία εισαγωγής και εξαγωγής των στοιχείων ονομάζεται ***LIFO*** *(Last In, First Out)*, δηλαδή το τελευταίο εισαγόμενο δεδομένο εξέρχεται και πρώτο.



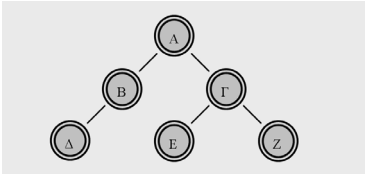
Μια ***ουρά*** *(Queue)* αποτελεί μια γραμμική διάταξη στοιχείων, στην οποία εισάγονται (enqueue) νέα στοιχεία από ένα άκρο και εξάγονται (dequeue) υπάρχοντα στοιχεία από το άλλο άκρο. Ο λειτουργίας της ουράς αποκαλείται ***FIFO*** *(First In, First Out)*, δηλαδή το στοιχείο το οποίο εισάγεται πρώτο στην ουρά εξέρχεται και πρώτο.



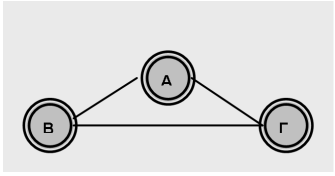
Σε μια ***Δεσμική Λίστα*** *(Linked List)* τα στοιχεία φαίνονται «λογικά» ότι είναι γραμμικά διατεταγμένα, χωρίς όμως αυτό να σημαίνει ότι βρίσκονται σε συνεχόμενες θέσεις της μνήμης του υπολογιστή. Ανεξάρτητα από τη θέση που καταλαμβάνει στη μνήμη ένα δεδομένο, συσχετίζεται με το επόμενό του με τη βοήθεια κάποιου ***δείκτη*** (pointer).



Τα ***Δένδρα*** *(trees)* είναι μη γραμμικές δομές που αποτελούνται από ένα σύνολο κόμβων, οι οποίοι συνδέονται με ακμές. Υπάρχει μόνο ένας κόμβος, από τον οποίο μόνο ξεκινούν ακμές, που ονομάζεται ***ρίζα*** *(root)*. Σε όλους τους άλλους κόμβους καταλήγει μία ακμή και ξεκινούν καμία, μία ή περισσότερες. Οι κόμβοι στους οποίους καταλήγουν μόνο ακμές, ονομάζονται ***φύλλα.***.



Οι ***Γράφοι*** *(Graphs)* αποτελούν τη πιο γενική δομή δεδομένων μια και αποτελούνται από κόμβους και ακμές χωρίς όμως κάποια ιεράρχηση.



Διάκριση των δομών δεδομένων

Υπάρχουν διάφοροι τρόποι διάκρισης των δομών δεδομένων. Κατ' αρχήν διακρίνονται σε *στατικές* (static) και *δυναμικές* (dynamic). Οι στατικές δομές έχουν σταθερό μέγεθος και μπορούν να κατακρατήσουν συγκεκριμένο πλήθος στοιχείων. Αντίθετα οι δυναμικές δομές δεν έχουν σταθερό μέγεθος και το πλήθος των στοιχείων τους μπορεί να μεγαλώνει ή να μικραίνει καθώς στη δομή εισάγονται νέα δεδομένα ή διαγράφονται άλλα.

Οι δομές δεδομένων διακρίνονται επίσης σε *γραμμικές* και *μη γραμμικές*. Στις γραμμικές δομές μπορεί να ορισθεί κάποια σχέση διάταξης για δύο οποιαδήποτε διαδοχικά στοιχεία τους. Αυτό σημαίνει ότι κάποιο στοιχείο θα είναι πρώτο και κάποιο τελευταίο. Οποιοδήποτε από τα υπόλοιπα θα έπεται από το προηγούμενό του και θα προηγείται από το επόμενό του.

Στις μη γραμμικές δομές δεν μπορεί να οριστεί μια σχέση διάταξης όπως η παραπάνω. Τέτοιες δομές είναι τα δένδρα και οι γράφοι. Στα δένδρα ένας κόμβος έχει έναν προηγούμενο αλλά πιθανόν πολλούς επόμενους. Στους γράφους κάθε κόμβος μπορεί να έχει πολλούς προηγούμενους και πολλούς επόμενους.

Τέλος διάκριση των δομών μπορεί να γίνει και ανάλογα με το είδος της χρησιμοποιούμενης μνήμης (κύρια μνήμη, ταινία, δίσκος). Με αυτή τη διάκριση εννοούμε ότι κάποιες δομές είναι καταλληλότερες για την κύρια μνήμη, ενώ κά-

ποιες άλλες βρίσκουν μεγαλύτερη χρήση στη βοηθητική. Ο λόγος γι΄ αυτό είναι ότι οι χρόνοι πρόσβασης στη βοηθητική μνήμη είναι σημαντικά μεγαλύτεροι από την κύρια. Ακόμη υπάρχουν και δομές, που είναι ακατάλληλες για κάποιο είδος βοηθητικής μνήμης, γιατί αυτή η μνήμη δεν επιτρέπει το συγκεκριμένο είδος πρόσβασης (π.χ. ταινία).

Σε αυτό το βιβλίο αναπτύσσονται αλγόριθμοι που χρησιμοποιούν τη δομή του πίνακα (κεφ. 3-6), καθώς και δομές δεδομένων βοηθητικής μνήμης (κεφ. 8-11).

1.5 Αλγόριθμοι και είδη προβλημάτων

Όπως έχει αναφερθεί, το ενδιαφέρον για την έννοια του αλγόριθμου χρονολογείται αρκετά πριν από τον πρώτο ηλεκτρονικό υπολογιστή. Στην πρώτη δεκαετία του περασμένου αιώνα κορυφαίοι μαθηματικοί και φιλόσοφοι, όπως ο D. Hilbert και ο B. Russel προβληματίστηκαν πάνω στο αν μπορεί να υπάρξει αλγόριθμος που να αποκαλύπτει τη μαθηματική αλήθεια. Δηλαδή, να αποδεικνύει θεωρήματα και να διαψεύδει λανθασμένες προτάσεις. Η αρνητική απάντηση στο ερώτημα αυτό από τους A. Turing και K. Godel το 1936 έβαλε τα θεμέλια της θεωρίας των αλγορίθμων και οδήγησε στην ορισμό μαθηματικών μοντέλων για τους αλγόριθμους. Αποτέλεσμα αυτής της έρευνας ήταν η διάκριση των προβλημάτων σε τρεις κατηγορίες, ανάλογα με το εάν μπορούν να επιλυθούν ή όχι με τη βοήθεια κάποιου αλγορίθμου: επιλύσιμα, άλυτα και ανοικτά. ***Επιλύσιμο*** ονομάζεται ένα πρόβλημα, όταν υπάρχει ένας αλγόριθμος που να το επιλύει. ***Άλυτο*** ονομάζεται ένα πρόβλημα, όταν έχει αποδειχθεί ότι δεν υπάρχει αλγόριθμος για την επίλυσή του. Τέλος, ***ανοικτό*** ονομάζεται ένα πρόβλημα, όταν δεν έχει αποδειχθεί η μη ύπαρξη αλγορίθμου για τη λύση του προβλήματος, αλλά ούτε έχει βρεθεί μέχρι σήμερα κάτι σχετικό που να επιλύει το πρόβλημα. Είναι αξιοσημείωτο ότι σήμερα τα άλυτα προβλήματα είναι ελάχιστα, τα ανοικτά, όμως, ξεπερνούν τα 500.

1.6 Ανάλυση αλγόριθμων

Όταν λύνουμε ένα πρόβλημα με έναν αλγόριθμο, μας ενδιαφέρει πόσο αποδοτικός είναι ο αλγόριθμος αυτός. Συνήθως ενδιαφερόμαστε για το πόσα βήματα κάνει ο αλγόριθμος αυτός, για να λύσει το πρόβλημα και πόση μνήμη χρησιμοποιεί.

Για τον αλγόριθμο "Ευκλείδης" είναι εύκολο να δούμε ότι χρησιμοποιεί τόση μνήμη όση χρειάζεται για να αποθηκευτούν οι τρεις ακέραιοι x, y, z. Ωστόσο, η εύρεση του πλήθους των εντολών που εκτελούνται δεν είναι τόσο τετριμμένη. Κάθε φορά που πραγματοποιείται η επανάληψη εκτελούνται τέσσερα βήματα (έλεγχος της συνθήκης του όσο, υπολογισμός του z και δύο εντολές εκχώρησης). Επομένως, απαιτούνται 4×α βήματα, όπου α είναι ο αριθμός των επαναλήψεων. Πόσες, όμως, θα είναι οι επαναλήψεις; Το ότι για x = 27 και y = 78

γίνονται τέσσερις επαναλήψεις, δεν είναι πολύ διαφωτιστικό. Αυτό που θα θέλαμε, θα ήταν να μπορούσαμε να υπολογίζουμε τον αριθμό των επαναλήψεων για κάθε ζευγάρι ακεραίων x, y.

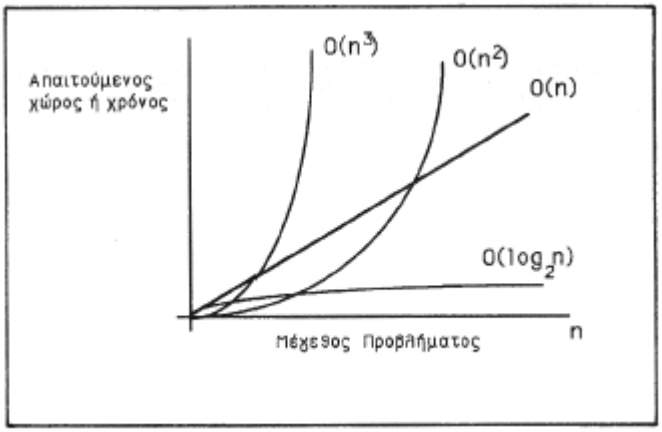
Σχετικά πρόσφατα ο D. Knuth υπολόγισε, ότι ο αριθμός των επαναλήψεων είναι περίπου ίσος με τον λογάριθμο του x (ή του y). Δηλαδή, για να υπολογίσει το ΜΚΔ του x και y, ο αλγόριθμος του Ευκλείδη κάνει 4×logx βήματα. Συνήθως παραλείπουμε σταθερές, όπως το 4 και λέμε ότι η ***πολυπλοκότητα*** του αλγόριθμου είναι O(logx) (διαβάζεται "της τάξης logx").

Η πολυπλοκότητα ενός αλγορίθμου είναι μια ιδιαίτερα σημαντική παράμετρος ενός αλγορίθμου, γιατί μας δίνει ένα μέτρο της χρονικής καθυστέρησης του αλγορίθμου για την επίλυση ενός προβλήματος. Οι περισσότεροι αλγόριθμοι έχουν πολυπλοκότητα χρόνου που ανήκει σε μια από τις κατηγορίες που φαίνονται στον Πίνακα 1.1. Με n συμβολίζουμε το μέγεθος του προβλήματος και εξαρτάται από το πρόβλημα. Για παράδειγμα, αν ζητείται να ταξινομηθούν k αριθμοί, τότε το μέγεθος του προβλήματος είναι k, επομένως n=k.

Πίνακας 1.1: *Κατηγορίες πολυπλοκότητας αλγορίθμων*

Πολυπλοκότητα	**Ονομασία**	**Παρατηρήσεις**
O(1)	Σταθερή	Κάθε βήμα εκτελείται μια φορά ή το πολύ μερικές φορές
O(logn)	Λογαριθμική	Αν δεν σημειώνεται αλλιώς, οι λογάριθμοι είναι δυαδικοί
O(n)	Γραμμική	
O(nlogn)	Δεν υπάρχει επιθετικός προσδιορισμός	
$O(n^2)$	Τετραγωνική	Στην πράξη χρησιμοποιούνται μόνο για προβλήματα μικρού μεγέθους
$O(n^3)$	Κυβική	
$O(2^n)$	Εκθετική	

Στο σχήμα 1.5 έχουν παρασταθεί οι καμπύλες των κυριότερων συναρτήσεων πολυπλοκότητας, ενώ στον Πίνακα 1.2 παρουσιάζονται οι χρόνοι εκτέλεσης για διάφορες πολυπλοκότητες και μεγέθη προβλημάτων.



Σχήμα 1.5: *Καμπύλες των κυριότερων συναρτήσεων πολυπλοκότητας.*

Πίνακας 1.2. *Χρόνοι εκτέλεσης για διάφορες πολυπλοκότητες και μεγέθη προβλημάτων.*

Τάξη αλγορίθμου	**Μέγεθος Προβλήματος**		
	4	**16**	**64**
O(logn)	6×10^{-10} sec	$1{,}6\times10^{-8}$	$1{,}8\times10^{-8}$
O(nlogn)	2×10^{-8} sec	19×10^{-8}	$1{,}2\times10^{-6}$
O(n)	4×10^{-8} sec	16×10^{-8}	64×10^{-8}
$O(n^2)$	2×10^{-8} sec	$2{,}6\times10^{-6}$	4×10^{-5}
$O(n^3)$	64×10^{-8} sec	4×10^{-5}	3×10^{-3}
$O(2^n)$	16×10^{-8} sec	65×10^{-5}	5×10^{3} years
O(n!)	24×10^{-8} sec	58 hours	4×10^{73} years

Η συνάρτηση O χρησιμοποιείται καλείται πολυωνυμική, αν είναι έκφραση πολυωνύμου ως προς n, διαφορετικά καλείται μη-πολυωνυμική, αν περιέχει όρους όπως 2^n, e^m, n! κ.λπ.

Μετά την επίλυση ενός προβλήματος με ένα αλγόριθμο είναι φυσικό να θέλουμε να σχεδιάσουμε ένα νέο αλγόριθμο με μεγαλύτερη ταχύτητα εύρεσης της λύσης του προβλήματος. Για παράδειγμα ένας αλγόριθμος ταξινόμησης με συγκρίσεις (ο οποίος περιγράφεται στο κεφάλαιο 4) απαιτεί $O(n^2)$ συγκρίσεις. Υπάρχουν όμως ταχύτεροι αλγόριθμοι (όπως ο Quicksort του R. Hoare), που απαιτούν O(nlogn) συγκρίσεις.

Είναι αξιοσημείωτο για το πρόβλημα της ταξινόμησης ενός πίνακα δεν έχει βρεθεί μέχρι σήμερα ταχύτερος αλγόριθμος από τον Quicksort, ενώ για το πρόβλημα του υπολογισμού του ΜΚΔ δεν έχει βρεθεί ταχύτερος αλγόριθμος από εκείνο του Ευκλείδη. Αυτό οδηγεί, μοιραία, στο ερώτημα: Μήπως δεν υπάρχει καλύτερος αλγόριθμος, μήπως δηλαδή τα προβλήματα αυτά έχουν μία εγγενή πολυπλοκότητα;

Φθάνουμε έτσι σε μια σημαντική περιοχή της Επιστήμης των Υπολογιστών που ονομάζεται ***Θεωρία Πολυπλοκότητας*** (Complexity theory). Η έρευνα στην περιοχή αυτή προσπαθεί να βρει τους εγγενείς περιορισμούς στην ταχύτητα των αλγορίθμων για τη λύση συγκεκριμένων προβλημάτων. Έτσι αποδεικνύεται, για παράδειγμα, ότι δεν είναι δυνατόν να ταξινομήσουμε n ακεραίους με λιγότερο nlogn συγκρίσεις και συνεπώς ο Quicksort είναι ουσιαστικά βέλτιστος, όσον αφορά την ταχύτητα ταξινόμησης.

1.7 Είδη αλγορίθμων

Προκειμένου να είναι ευκολότερη η αντιμετώπιση του αντικειμένου των αλγορίθμων έχουν προταθεί διάφορες κατηγοριοποιήσεις τους. Έτσι, αν ένας αλγόριθμος βρίσκει τη βέλτιστη λύση για κάθε στιγμιότυπο του προβλήματος ονομάζεται ***άριστος*** ή ***βέλτιστος*** (optimal) αλγόριθμος. Αντίθετα, όταν η λύση ενός προβλήματος είναι πολύ δύσκολη και η εύρεση του άριστου αλγορίθμου φοβερά πολύπλοκη αν όχι αδύνατη, τότε καταφεύγουμε σε αλγορίθμους που προσεγγίζουν την άριστη λύση του προβλήματος για κάθε στιγμιότυπο. Αυτοί λέγονται ***προσεγγιστικοί*** ή ***ευριστικοί*** (approximation ή heuristic) αλγόριθμοι.

Μια ειδική κατηγορία αλγορίθμων είναι οι ***αναδρομικοί*** αλγόριθμοι. Μία μαθηματική σχέση αποκαλείται αναδρομική αν χρησιμοποιεί την ίδια στον ορισμό της. Οι αναδρομικοί αλγόριθμοι μπορούν να επιλύσουν δύσκολα προβλήματα, ιδιαίτερα προβλήματα που σχετίζονται με τη δομή των γράφων. Μια εισαγωγή στους αναδρομικούς αλγορίθμους γίνεται στο κεφάλαιο 7.

Ο ορισμός του αλγόριθμου που δόθηκε στην αρχή αυτού του κεφαλαίου, συμφωνεί με τη φιλοσοφία των περισσότερων Η/Υ σήμερα, που διαθέτουν μία Κεντρική Μονάδα Επεξεργασίας (ΚΜΕ) στην οποία οι εντολές εκτελούνται με σειρά, η μία μετά την άλλη. Για το λόγο αυτό όλοι οι προηγούμενοι αλγόριθμοι ονομάζονται ***ακολουθιακοί.*** Όμως η ύπαρξη προβλημάτων στα οποία απαιτείται πολύ μεγάλος χρόνος για τον υπολογισμό της λύσης ενός προβλήματος, δημιούργησε την ανάγκη εύρεσης αλγορίθμων, όπου ορισμένα ή μία σειρά από βήματα θα μπορούσαν να εκτελούνται παράλληλα (ταυτόχρονα). Δηλαδή, η εκτέλεση του ενός βήματος δεν εξαρτάται από την εκτέλεση του άλλου. Αλγόριθμοι αυτής της μορφής ονομάζονται ***παράλληλοι*** αλγόριθμοι και βέβαια δεν μπορούν να υλοποιηθούν χωρίς την ύπαρξη πολλαπλών ΚΜΕ στο σύστημα του Η/Υ (multiprocessing systems).

Τέλος, οι αλγόριθμοι, μπορούν να διακριθούν ανάλογα με την περιοχή προβλημάτων όπου έχουν ισχύ. Έτσι έχουμε αλγορίθμους ***αριθμητικούς*** (αν έχουν ισχύ σε προβλήματα αριθμητικής ανάλυσης), ***συνδυαστικούς*** (αν έχουν ισχύ σε προβλήματα συνδυαστικής), ***στοχαστικούς*** (αν έχουν ισχύ σε προβλήματα θεωρίας ουρών) κ.ο.κ.