5.

Ανάλυση αλγορίθμων

Εισαγωγή

Η καταγραφή των μεγεθών που επηρεάζουν την επίδοση ενός αλγορίθμου είναι μία σημαντική ενέργεια για την κατανόηση της αποδοτικότητας ενός αλγορίθμου. Για το σκοπό αυτό πρέπει να βρεθεί ποιά είναι η βασική λειτουργία και ποιά η δομή του αλγορίθμου, όπου δαπανώνται κυρίως οι υπολογιστικοί πόροι (δηλαδή, ο χρόνος). Η πολυπλοκότητα των αλγορίθμων, λοιπόν, περιγράφεται αναλυτικά και τυποποιούνται οι ορισμοί για τις κατηγορίες (τάξεις) πολυπλοκότητας των περισσοτέρων ειδών αλγορίθμων. Για μερικούς από τους αλγορίθμους, που γνωρίσαμε στα προηγούμενα κεφάλαια, γίνεται ανάλυση και εκφράζεται η πολυπλοκότητά τους και τέλος, γίνεται μια παρουσίαση των ειδών των αλγορίθμων.

Διδακτικοί στόχοι

Στόχοι του κεφαλαίου αυτού είναι οι μαθητές :

- να περιγράφουν τις τεχνικές ανάλυσης των αλγορίθμων,
- να μετρούν τη επίδοση των αλγορίθμων,
- να αντιπαραβάλλουν αλγόριθμους με βάση τυποποιημένους κανόνες επιλογής,
- να μπορούν να ελέγχουν την ορθότητα των αλγορίθμων,
- να διατυπώνουν την έννοια της πολυπλοκότητας των αλγορίθμων,
- να συνοψίζουν τα κυριότερα είδη αλγορίθμων.

Προερωτήσεις

- ✓ 'Εχεις αναρωτηθεί πόσο γρήγορα μπορεί να γίνει η αναζήτηση ενός στοιχείου ή η ταξινόμηση κάποιων αριθμών;
- ✓ Ξέρεις αν υπάρχουν προβλήματα άλυτα;

5.1 Επίδοση αλγορίθμων

Στα προηγούμενα κεφάλαια του βιβλίου παρουσιάσθηκαν πολλοί αλγόριθμοι για την επίλυση ενός προβλήματος, όπως για παράδειγμα σχετικά με την αναζήτηση και την ταξινόμηση. Στο σημείο αυτό θα παρουσιάσουμε έναν τρόπο εκτίμησης της *επίδοσης* (performance) ή της *αποδοτικότητας* (efficiency) των αλγορίθμων.

Για την κατανόηση της επίδοσης ενός αλγορίθμου χρειάζεται να απαντηθεί ένα σύνολο ερωτημάτων. Τα πρωταρχικά ερωτήματα που προκύπτουν είναι:

1. πώς υπολογίζεται ο χρόνος εκτέλεσης ενός αλγορίθμου;
2. πώς μπορούν να συγκριθούν μεταξύ τους οι διάφοροι αλγόριθμοι;
3. πώς μπορεί να γνωρίζει κανείς, αν ένας αλγόριθμος είναι "βέλτιστος";

Η απάντηση στα ερωτήματα αυτά προαπαιτεί την καταγραφή ουσιωδών πληροφοριών για το πρόβλημα. Οι πληροφορίες αυτές αφορούν κυρίως στην αναγνώριση της χειρότερης περίπτωσης του αλγορίθμου και στην αποτύπωση του μεγέθους του προβλήματος με βάση το πλήθος των δεδομένων.

5.1.1 Χειρότερη περίπτωση ενός αλγορίθμου

Η χειρότερη περίπτωση ενός αλγορίθμου αφορά στο μέγιστο κόστος εκτέλεσης του αλγορίθμου, κόστος που μετράται σε υπολογιστικούς πόρους. Το κόστος αυτό πολλές φορές κρίνει την επιλογή και το σχεδιασμό ενός αλγορίθμου. Για να εκφρασθεί αυτή η χειρότερη περίπτωση χρειάζεται κάποιο μέγεθος σύγκρισης και αναφοράς που να χαρακτηρίζει τον αλγόριθμο. Η πλέον συνηθισμένη πρακτική είναι η μέτρηση του αριθμού των *βασικών πράξεων* που θα πρέπει να εκτελέσει ο αλγόριθμος στη χειρότερη περίπτωση. Για παράδειγμα, μία βασική πράξη μπορεί να είναι :

- ✓ ανάθεση τιμής,
- ✓ σύγκριση μεταξύ δύο μεταβλητών, ή
- ✓ οποιαδήποτε αριθμητική πράξη μεταξύ δύο μεταβλητών.

Η χειρότερη περίπτωση αντιπροσωπεύει τις τιμές εκείνες, που όταν δίνονται ως είσοδος στον αλγόριθμο, οδηγούν στην εκτέλεση μέγιστου αριθμού βασικών πράξεων.

Παράδειγμα: έστω ότι δίνεται ο επόμενος απλός αλγόριθμος :

```
Αλγόριθμος Παράδειγμα1
n ← 10
Αρχή_επανάληψης
    Διάβασε m
    n ← n - 1
Μέχρις_ότου (m=0) ή (n=0)
Εκτύπωσε m
Τέλος Παράδειγμα1
```

Είναι προφανές ότι η χειρότερη περίπτωση για αυτόν τον αλγόριθμο προκύπτει όταν γίνουν 10 επαναλήψεις (δηλαδή μέχρι να ισχύει το n=0).

5.1.2 Μέγεθος εισόδου ενός αλγορίθμου

Πρέπει να υπάρχει κάποια ή κάποιες μεταβλητές που να εκφράζουν το *μέγεθος* (size) του αλγορίθμου. Οι μεταβλητές αυτές απεικονίζουν τους διαφορετικούς συνδυασμούς τιμών που κρίνουν τη συμπεριφορά ενός αλγορίθμου και δίνονται ως δεδομένα στον αλγόριθμο. Γενικά, τα δεδομένα συνιστούν το μέγεθος της εισόδου ενός αλγορίθμου. Για παράδειγμα, στον αλγόριθμο του προηγούμενου παραδείγματος, το μέγεθος του αλγορίθμου μπορεί να εκφρασθεί από τη μεταβλητή n, που στην ουσία εκφράζει το πλήθος των επαναλήψεων του βασικού βρόχου του αλγορίθμου.

Ο πίνακας 5.1 περιλαμβάνει παραδείγματα από συνήθεις κατηγορίες προβλημάτων, με δήλωση του μεγέθους που βασικά εκφράζει την είσοδό τους και των αντίστοιχων βασικών πράξεων:

Πιν. 5.1. *Μέγεθος εισόδου και βασική πράξη αλγορίθμων*

Αλγόριθμος	Μέγεθος εισόδου αλγορίθμου (n)	Βασική Πράξη
ΤΑΞΙΝΟΜΗΣΗ	το πλήθος των αντικειμένων που θα ταξινομηθούν	σύγκριση
ΠΟΛΛΑΠΛΑΣΙΑΣΜΟΣ	το πλήθος των ψηφίων των αριθμών που θα πολλαπλασιασθούν	αριθμητικές πράξεις
ΑΝΑΖΗΤΗΣΗ	το πλήθος των στοιχείων του πίνακα	σύγκριση

5.1.3 Χρόνος εκτέλεσης προγράμματος ενός αλγορίθμου

Έστω ότι έχουμε τον εξής αλγόριθμο:

```
Αλγόριθμος Παράδειγμα2
x ← 123
y ← 234
Για i από 0 μέχρι 4
    Εκτύπωσε i
    z ← x * y
Τέλος_επανάληψης
Εκτύπωσε x
Εκτύπωσε y
Εκτύπωσε z
Τέλος Παράδειγμα2
```

Θα υπολογισθεί η επίδοσή του με βάση τον αριθμό των πράξεων που θα εκτελεσθούν. Με δεδομένο ότι ο βρόχος του προγράμματος θα εκτελεσθεί 5 φορές, προκύπτει η παρακάτω ανάλυση :

Εντολή αλγορίθμου	Αριθμός πράξεων
ανάθεση τιμών στα x και y	2
Βρόχος επανάληψης αρχική τιμή i έλεγχος i αύξηση i εκτύπωση i υπολογισμός z (2X5)	 1 6 5 5 10
Εκτύπωση x, y, z	3
ΣΥΝΟΛΟ	***32***

Οι βρόχοι επανάληψης αποτελούν το κρίσιμο σημείο για τον χαρακτηρισμό της επίδοσης ενός αλγορίθμου Έτσι αν ο αλγόριθμος αυτός γενικευθεί ώστε ο βρόχος να εκτελεσθεί n φορές, ο χρόνος εκτέλεσης θα εξαρτάται από το μέγεθος του n. Ο πίνακας 5.2 παρουσιάζει τους χρόνους εκτέλεσης του αλγορίθμου αυτού για διαφορετικά μεγέθη του n, θεωρώντας ότι ο χρόνος είναι ο αριθμός των πράξεων σε μικρο-δευτερόλεπτα :

Πιν. 5.2. *Χρόνοι εκτέλεσης αλγορίθμου ανάλογα με το μέγεθος*

μέγεθος n	Χρόνος εκτέλεσης
5	42 μικρο-δευτερόλεπτα
10	77 μικρο-δευτερόλεπτα
100	707 μικρο-δευτερόλεπτα
1.000.000	7 δευτερόλεπτα (περίπου)

5.1.4 Αποδοτικότητα αλγορίθμων

Αν η επίλυση ενός προβλήματος επιτυγχάνεται με τη χρήση δύο ή περισσοτέρων αλγορίθμων, χρειάζεται να γίνει η επιλογή του καταλληλότερου με βάση την αποδοτικότητά τους. Έτσι, αν ο αλγόριθμος Β έχει το ίδιο αποτέλεσμα με τον αλγόριθμο Α, αλλά δίνει τα αποτελέσματα σε λιγότερο χρόνο, τότε είναι αποδοτικότερος του Α. Με παρόμοιο τρόπο όταν ο αλγόριθμος Β έχει το ίδιο αποτέλεσμα με έναν αλγόριθμο Α, αλλά έχει τα αποτελέσματα με χρήση λιγότερης μνήμης, τότε είναι αποδοτικότερος του Α. Βέβαια όταν συγκρίνονται δύο αλγόριθμοι, θα πρέπει να συγκρίνονται με χρήση των ίδιων δεδομένων και κάτω από τις ίδιες συνθήκες. Γενικά, ο χρόνος εκτέλεσης κάθε αλγορίθμου εξαρτάται από ένα σύνολο παραγόντων που μπορούν να συνοψισθούν στους εξής:

- Τύπος ηλεκτρονικού υπολογιστή που θα εκτελέσει το πρόγραμμα του αλγορίθμου,
- Γλώσσα προγραμματισμού που θα χρησιμοποιηθεί,
- Δομή προγράμματος και δομές δεδομένων που χρησιμοποιούνται,
- Χρόνος για πρόσβαση στο δίσκο και στις ενέργειες εισόδου-εξόδου,
- Είδος συστήματος, ενός χρήστη ή πολλών χρηστών.

Επομένως, για να έχει έννοια κάθε σύγκριση μεταξύ δύο προγραμμάτων αλγορίθμων, θα πρέπει να ικανοποιούνται οι παρακάτω προϋποθέσεις:

- ✓ και τα δύο προγράμματα να έχουν συνταχθεί στην ίδια γλώσσα προγραμματισμού,
- ✓ να έχει χρησιμοποιηθεί ο ίδιος μεταφραστής της γλώσσας προγραμματισμού,
- ✓ να χρησιμοποιείται η ίδια υπολογιστική πλατφόρμα,

✓ ακριβώς τα ίδια δεδομένα να αποτελούν είσοδο κατά τον έλεγχο των δύο αλγορίθμων.

5.2 Ορθότητα αλγορίθμων

Η επίλυση ενός προβλήματος με τη χρήση κάποιου αλγορίθμου, έχει μεγαλύτερη ισχύ όταν υπάρχει κάποια τυποποιημένη ένδειξη ή απόδειξη για την ορθότητα του προτεινόμενου αλγορίθμου. Είναι ιδιαίτερα σημαντικό να επικυρωθεί η ορθότητα και εγκυρότητα ενός αλγορίθμου με χρήση κάποιων απλών μαθηματικών τύπων ή μεθόδων. Οι τεχνικές απόδειξης της ορθότητας ενός αλγορίθμου συνδέονται άμεσα με τον τρόπο της αρχικής σχεδίασης και ανάλυσης του συγκεκριμένου αλγορίθμου. Είναι λάθος να επιχειρείται ο έλεγχος των σφαλμάτων και της ορθότητας ενός αλγορίθμου μετά τη σχεδίαση και την τυποποίησή του. Είναι, δηλαδή, απαραίτητο να συνοδεύεται το στάδιο της ανάλυσης και της σχεδίασης του αλγορίθμου από τον αντίστοιχο έλεγχο της ορθότητάς του. Στη συνέχεια θα παρουσιασθούν οι περιορισμοί, οι τεχνικές και οι μέθοδοι προσέγγισης των αποδείξεων της ορθότητας των αλγορίθμων.

Περιορισμοί ελέγχου ορθότητας ενός αλγορίθμου

Η πρόταση ενός αλγορίθμου καταλήγει στη σύνταξη κώδικα για την επίλυσή του στον υπολογιστή. Αν ένας προγραμματιστής έχει υλοποιήσει κώδικα για την υλοποίηση ενός αλγορίθμου, μπορεί να ελέγξει το πρόγραμμα "τρέχοντας" τον κώδικα μία, δύο, τρεις κ.λπ. φορές. Έστω ότι σε όλες τις περιπτώσεις καταλήγει σε σωστά αποτελέσματα. Είναι όμως βέβαιο ότι, ο αλγόριθμος του προγράμματος δίνει πάντοτε τη σωστή λύση; Η απάντηση είναι όχι, διότι υπάρχει πιθανότητα κάποια φορά να καταλήξει σε λάθος αποτέλεσμα λόγω κάποιου λάθους στον αλγόριθμο. Το παράδειγμα που ακολουθεί παρουσιάζει μία τέτοια περίπτωση :

Παράδειγμα: Να βρεθεί ο μικρότερος αριθμός από το σύνολο των θετικών αριθμών που βρίσκονται αποθηκευμένοι σε ένα πίνακα 10 θέσεων.

Στη συνέχεια δίνεται ο σχετικός αλγόριθμος για την εύρεση του μικρότερου στοιχείου σε πίνακα:

```
Αλγόριθμος Παράδειγμα3
Δεδομένα  // a //
low ← a[1]
i ← 2
Όσο i < 10 επανάλαβε
    Αν a[i] < low τότε low ← a[i]
    i ← i+1
Τέλος_επανάληψης
Αποτελέσματα // low //
Τέλος Παράδειγμα3
```

Από τον ψευδοκώδικα αυτό μπορεί να θεωρήσει κανείς ότι, ο αλγόριθμος είναι σωστός και παράγει το σωστό αποτέλεσμα. Η ορθότητα του αλγορίθμου θα ελεγχθεί με ένα σύνολο από πίνακες 10 θέσεων, που έχουν τυχαίους αριθμούς. Έστω ότι προκύπτουν οι επόμενοι πίνακες από 3 διαφορετικές εκτελέσεις του προγράμματος του αλγορίθμου :

11	3	2	56	32	69	81	90	222	2

2	444	1	65	51	51	99	98	1	1

90	11	333	38	224	61	73	80	7	59

Η εφαρμογή του αλγορίθμου σε κάθε έναν από τους πίνακες αυτούς, θα δώσει το σωστό αποτέλεσμα για το μικρότερο στοιχείο του πίνακα. Αν όμως δοθεί ο πίνακας με τα εξής δεδομένα:

6	8	3	4	3	7	8	9	5	2

τότε ο αλγόριθμος θα δώσει ως αποτέλεσμα το 3, ενώ το μικρότερο στοιχείο είναι το 2. Άρα είναι σαφές ότι, υπάρχει κάποιο σφάλμα στον αλγόριθμο, που εδώ παρουσιάστηκε με την τέταρτη εκτέλεση του αλγορίθμου. Θα μπορούσε να είχε παρουσιασθεί το σφάλμα πολύ αργότερα, δηλαδή μετά από μεγάλο αριθμό επαναληπτικών εκτελέσεων του αλγορίθμου.

Γενικότερα, λοιπόν, δεν υπάρχει καμμία εγγύηση ή επιβεβαίωση του αριθμού των επαναλήψεων εκτέλεσης ενός αλγορίθμου, μέχρι να βρεθεί κάποιο πιθανό σφάλμα του. Το σφάλμα στον προηγούμενο αλγόριθμο είναι η συνθήκη (i < 10), που δεν ελέγχει ποτέ την τελευταία θέση του πίνακα.

Έτσι αν το μικρότερο στοιχείο βρίσκεται στην τελευταία θέση του πίνακα, τότε ο αλγόριθμος θα επιστρέψει λανθασμένη τιμή. Βέβαια, υπάρχει περίπτωση να επιστρέψει το σωστό αποτέλεσμα, μόνο αν το μικρότερο στοιχείο που είναι στην τελευταία θέση, έχει εμφανισθεί ήδη και σε κάποια άλλη μικρότερη θέση του πίνακα.

Για τον υπολογισμό της πιθανοτήτας να υπάρξει λάθος κατά την εκτέλεση του αλγορίθμου, παρατηρούμε ότι η πιθανότητα αυτή προκύπτει, όταν το μικρότερο στοιχείο βρίσκεται στην τελευταία θέση του πίνακα. Άρα η πιθανότητα είναι 1/10, αφού υπάρχουν 10 θέσεις στον πίνακα. Έτσι η πιθανότητα να μη βρεθεί το λάθος με μία προσπάθεια είναι 0,9. Αν γίνουν δύο προσπάθειες, λόγω της τυχαιότητας των στοιχείων του πίνακα η πιθανότητα αυτή γίνεται 0,9*0,9=0,81. Όσο επαναλαμβάνεται η εκτέλεση του αλγορίθμου για κάποια επιπλέον τυχαία ακολουθία δεδομένων οι πιθανότητες διαμορφώνονται ως εξής :

Αριθμός Εκτελέσεων Αλγορίθμου	Πιθανότητα Ανεύρεσης Σφάλματος	Πιθανότητα Μη-Ανεύρεσης Σφάλματος
1	0,1	0,9
2	0,19	0,81
3	0,271	0,729
4	0,344	0,656
5	0,410	0,590
6	0,469	0,531
7	0,522	0,478

Από αυτόν τον πίνακα πιθανοτήτων, είναι φανερό ότι χρειάζονται τουλάχιστον 7 προσπάθειες εκτέλεσης του αλγορίθμου για να υπάρχει πιθανότητα της τάξης του 0,5 να βρεθεί ένα σφάλμα. Η παραπάνω καταγραφή των πιθανοτήτων αποτελεί έναν τρόπο εκτίμησης για τον εντοπισμό λαθών ενός αλγορίθμου. Ωστόσο, δεν μπορεί να υπάρξει απόλυτη ασφάλεια στην εκτίμηση των πιθανών σφαλμάτων ενός αλγορίθμου. Γενικότερα δεν μπορεί να υπάρξει απόλυτη επιβεβαίωση της εγκυρότητας ενός αλγορίθμου με χρήση κάποιας πιθανολογικής εκτίμησης των πιθανών σφαλμάτων του.

Η απόδειξη της ορθότητας ενός αλγορίθμου θα πρέπει να περιλαμβάνει τις εξής δύο συνθήκες :

- απόδειξη ότι σε κάθε περίπτωση ο τερματισμός της εκτέλεσης του αλγορίθμου οδηγεί σε αποδεκτά αποτελέσματα, και
- απόδειξη ότι θα υπάρξει τερματισμός της εκτέλεσης του αλγορίθμου

Γενικά, η απόδειξη της ορθότητας ενός αλγορίθμου δεν είναι εύκολη υπόθεση. Στο σημείο αυτό δίνουμε μόνο ένα απλό παράδειγμα.

Έστω ότι δίνονται δύο μεταβλητές a και b με τις αντίστοιχες τιμές τους και ότι πρέπει να παρουσιασθεί ένας αλγόριθμος ανταλλαγής των τιμών τους. Αν υπάρχει εκ των προτέρων πρόβλεψη για την ορθότητα ενός αλγορίθμου ανταλλαγής των τιμών δύο μεταβλητών, τότε θα πρέπει να ακολουθηθεί κάποια τυποποίηση και στην ονοματολογία των μεταβλητών. Αυτό είναι απαραίτητο, γιατί είναι σαφές ότι η ανταλλαγή των μεταβλητών όπως: $a \leftarrow b$ και $b \leftarrow a$, δεν οδηγεί πράγματι στην ανταλλαγή των τιμών τους. Επομένως, οι αρχικές τιμές των μεταβλητών πρέπει να αποθηκευθούν σε κάποιες προσωρινές μεταβλητές a_0 και b_0 αντίστοιχα. Έτσι, ο στόχος του προβλήματος είναι να ισχύει: $a=b_0$ και $b=a_0$. Με αυτήν την ονοματολογία το πρόβλημα αντιμετωπίζεται στη γενική του μορφή. Ωστόσο, η προηγούμενη τυποποίηση είναι απλά ενδεικτική, γιατί τελικά επαρκεί μία επιπλέον μεταβλητή που θα λειτουργήσει ως ενδιάμεσος χώρος για την ανταλλαγή των τιμών των μεταβλητών. Ο προτεινόμενος αλγόριθμος λοιπόν είναι:

$t \leftarrow a$, $a \leftarrow b$ και $b \leftarrow t$

Σύμφωνα με τα προηγούμενα, είναι προφανές ότι, ο αλγόριθμος που στηρίζεται στην τεχνική αυτή είναι ορθός.

5.3 Πολυπλοκότητα αλγορίθμων

Ο απλούστερος τρόπος μέτρησης της επίδοσης ενός αλγορίθμου είναι ο *εμπειρικός* (empirical) ή αλλιώς ο λεγόμενος *εκ των υστέρων* (a posteriori). Δηλαδή, ο αλγόριθμος υλοποιείται και εφαρμόζεται σε ένα σύνολο δεδομένων, ώστε να υπολογισθεί ο απαιτούμενος *χρόνος επεξεργασίας* (processing time) και η *χωρητικότητα μνήμης* (memory space). Ο τρόπος όμως αυτός παρουσιάζει δύο κύρια μειονεκτήματα.

- με τη μέθοδο αυτή είναι δύσκολο να προβλεφθεί η συμπεριφορά του αλγορίθμου για κάποιο άλλο σύνολο δεδομένων.
- ο χρόνος επεξεργασίας εξαρτάται από το υλικό, τη γλώσσα προγραμματισμού και το μεταφραστή και προ πάντων από τη δεινότητα του προγραμματιστή.

'Ετσι μπορεί να συναχθούν λανθασμένες εκτιμήσεις για την επίδοση του αλγορίθμου.

Ο δεύτερος τρόπος εκτίμησης της επίδοσης ενός αλγορίθμου είναι ο *θεωρητικός* (theoretical) ή αλλιώς ο λεγόμενος *εκ των προτέρων* (a priori). Για το σκοπό αυτό εισάγεται μία μεταβλητή n, που εκφράζει το μέγεθος (size) του προβλήματος, ώστε η μέτρηση της αποδοτικότητας του αλγόριθμου να ισχύει για οποιοδήποτε σύνολο δεδομένων και ανεξάρτητα από υποκειμενικούς παράγοντες, όπως αυτοί που αναφέρθηκαν. Η σημασία της μεταβλητής αυτής εξαρτάται από το πρόβλημα, που πρόκειται να επιλυθεί. Για παράδειγμα, αν το πρόβλημα είναι η ταξινόμηση k στοιχείων τότε n=k. Στη συνέχεια ο χρόνος επεξεργασίας και ο απαιτούμενος χώρος μνήμης εκτιμώνται με τη βοήθεια μίας συνάρτησης f(n) που εκφράζει τη *χρονική πολυπλοκότητα* (time complexity) ή την *πολυπλοκότητα χώρου* (space complexity). Σε πολλές περιπτώσεις όμως δεν ενδιαφέρουν οι επακριβείς τιμές αλλά μόνο η γενική συμπεριφορά των αλγορίθμων, δηλαδή η τάξη του αλγορίθμου. Για το λόγο αυτό εισάγεται ο λεγόμενος *συμβολισμός Ο* (O-notation), όπου το Ο είναι το αρχικό γράμμα της αγγλικής λέξης order και διαβάζεται "*όμικρον κεφαλαίο*". Ακολουθεί όμως ένας τυπικός ορισμός.

Ορισμός: Αν η πολυπλοκότητα ενός αλγορίθμου είναι f(n), τότε λέγεται ότι είναι τάξης O(g(n)), αν υπάρχουν δύο θετικοί ακέραιοι c και n_0, έτσι ώστε για κάθε $n \geq n_0$ να ισχύει:

$$|f(n)| \leq c\,|g(n)|$$

Παράδειγμα. Έστω ότι η πολυπλοκότητα ενός αλγορίθμου δίνεται από το πολυώνυμο $f(n)=2n^3+5n^2-4n+3$. Ο πιο σημαντικός όρος του πολυωνύμου είναι η τρίτη δύναμη, αρκεί να σκεφτούμε ότι καθώς το x αυξάνεται (και τείνει στο άπειρο) ο όρος αυτός έχει μεγαλύτερο «ειδικό βάρος» και συνεχώς καθίσταται σημαντικότερος, ενώ η σημασία των υπολοίπων όρων σταδιακά μειώνεται. Επιπλέον, αγνοείται ο συντελεστής 2 της τρίτης δύναμης και έτσι προκύπτει ότι $g(n)=n^3$. Επομένως, τελικά η πολυπλοκότητα του αλγορίθμου είναι $O(n^3)$. Πλέον, η έκφραση αυτή εκφράζει την ποιοτική και όχι την ποσοτική συμπεριφορά του αλγορίθμου. Κατά τον ίδιο τρόπο αν δοθεί η έκφραση $f(n)=5\cdot 2^n+4n^2-4\log n$, τότε προκύπτει ότι $g(n)=2^n$, καθώς αγνοούνται οι μη σημαντικοί όροι του f(n) και ο συντελεστής 5 του όρου 2^n. Στο τελευταίο παράδειγμα είναι περισσότερο σαφές, γιατί αγνοούμε το δεύτερο και τρίτο όρο του f(n), αν απλά δώσουμε τιμές 1, 2, 3 κλπ στη μεταβλητή n και υπολογίσουμε το αποτέλεσμα του κάθε όρου της έκφρασης. Θα παρατηρήσουμε ότι πολύ σύντομα (λόγου χάριν, για n=10), το τελικό αποτέλεσμα της έκφρασης κατά βάση προέρχεται από τη δύναμη 2^n.

Λέγεται ότι ο συμβολισμός Ο δίνει ένα άνω φράγμα για την πολυπλοκό-

τητα ενός αλγορίθμου, δηλαδή δίνει ένα μέτρο, που ποτέ δεν πρόκειται να ξεπεράσει ο αλγόριθμος προς τα επάνω. Ωστόσο, στη βιβλιογραφία υπάρχουν και άλλοι συμβολισμοί, αλλά δεν θα μας απασχολήσουν στα πλαίσια αυτού του βιβλίου.

Σχεδόν οι περισσότεροι αλγόριθμοι πρακτικού ενδιαφέροντος έχουν χρονική πολυπλοκότητα, που ανήκει σε μία από τις επόμενες κατηγορίες:

- O(1). Κάθε εντολή του προγράμματος εκτελείται μία φορά ή το πολύ μερικές μόνο φορές. Στην περίπτωση αυτή λέγεται ότι ο αλγόριθμος είναι *σταθερής πολυπλοκότητα*ς.
- O(logn). Ο αλγόριθμος είναι *λογαριθμικής πολυπλοκότητα*ς. Ας σημειωθεί ότι με "log" θα συμβολίζεται ο δυαδικός λογάριθμος, ενώ με "ln" θα συμβολίζεται ο φυσικός λογάριθμος. Συνήθως, οι λογάριθμοι που χρησιμοποιούνται στο βιβλίο αυτό είναι δυαδικοί.
- O(n). Η πολυπλοκότητα λέγεται *γραμμική*. Αυτή είναι η καλύτερη επίδοση για έναν αλγόριθμο που πρέπει να εξετάσει ή να δώσει στην έξοδο n στοιχεία.
- O(n logn). Διαβάζεται όπως ακριβώς γράφεται (n logn), δηλαδή χωρίς να χρησιμοποιείται κάποιο επίθετο (όπως για παράδειγμα γραμμολογαριθμική). Στην κατηγορία αυτή ανήκει μία πολύ σπουδαία οικογένεια αλγορίθμων ταξινόμησης.
- $O(n^2)$. *Τετραγωνική πολυπλοκότητα*. Πρέπει να χρησιμοποιείται μόνο για προβλήματα μικρού μεγέθους.
- $O(n^3)$. *Κυβική πολυπλοκότητα*. Και αυτοί οι αλγόριθμοι πρέπει να χρησιμοποιούνται μόνο για προβλήματα μικρού μεγέθους.
- $O(2^n)$. Σπάνια στην πράξη χρησιμοποιούνται αλγόριθμοι *εκθετικής πολυπλοκότητα*ς.

Στον πίνακα 5.3 υπολογίζεται ο χρόνος που απαιτείται από αλγορίθμους διαφόρων πολυωνυμικών και εκθετικών πολυπλοκοτήτων ως συνάρτηση του μεγέθους του προβλήματος (n). Βέβαια υποτίθεται ότι κάθε στοιχειώδης πράξη απαιτεί ένα μικροδευτερόλεπτο. Έτσι αν για έναν αλγόριθμο τάξης $O(n^3)$ διπλασιασθεί το μέγεθος του προβλήματος, τότε απαιτείται οκταπλάσιος (2^3) χρόνος για να περατωθεί ο αλγόριθμος. Επίσης διαπιστώνεται αμέσως ότι, οι αλγόριθμοι εκθετικής πολυπλοκότητας δεν είναι πρακτικής χρησιμότητας ακόμη και για μικρό αριθμό δεδομένων.

Πιν. 5.3. ***Χρόνοι εκτέλεσης αλγορίθμων ανάλογα με την πολυπλοκότητα και το μέγεθος***

Πολυπλοκότητα αλγορίθμου	Μέγεθος προβλήματος		
	n=20	n=40	n=60
O(n)	0.00002 δεύτερα	0.00004 δεύτερα	0.00006 δεύτερα
$O(n^2)$	0.0004 δεύτερα	0.0016 δεύτερα	0.0036 δεύτερα
$O(n^3)$	0.008 δεύτερα	0.064 δεύτερα	216 δεύτερα
$O(2^n)$	1.0 δεύτερο	2.7 ημέρες	366 αιώνες
O(n!)	771 αιώνες	$3\ 10^{32}$ αιώνες	$3\ 10^{66}$ αιώνες

Η ανάλυση αλγορίθμων είναι ιδιαίτερα σημαντικό κεφάλαιο για όσους μελετούν βαθύτερα την επίδοση αλγορίθμων. Σκοπός μας δεν είναι μόνο να σχεδιάσουμε έναν αλγόριθμο, αλλά και να διαπιστώσουμε την επίδοσή του, δηλαδή να εκφράσουμε την πολυπλοκότητά του με το συμβολισμό O. Στη συνέχεια θα εξετάσουμε λεπτομερέστερα δύο από τους αλγορίθμους που αναπτύχθηκαν στο κεφάλαιο 3, ενώ για τους υπόλοιπους θα δώσουμε απλώς την αντίστοιχη έκφραση με βάση το συμβολισμό O.

5.3.1 Ταξινόμηση ευθείας ανταλλαγής

Το πιο βασικό κριτήριο της επίδοσης μίας μεθόδου ταξινόμησης είναι ο αριθμός C, που μετρά τις απαιτούμενες *συγκρίσεις κλειδιών* (key comparisons), που εκτελούνται μέχρι να τελειώσει η ταξινόμηση. Ένα άλλο κριτήριο είναι ο αριθμός M που μετρά τις *μετακινήσεις* (moves) των στοιχείων. Οι αριθμοί C και M είναι συναρτήσεις του αριθμού n των στοιχείων, που πρέπει να ταξινομηθούν.

Με βάση τον αλγόριθμο όπως περιγράφεται στην παράγραφο, εύκολα προκύπτει ότι ο αριθμός των συγκρίσεων στην καλύτερη, τη μέση και τη χειρότερη περίπτωση είναι ο ίδιος. Ο αριθμός αυτός είναι:

$$C = 1 + 2 + \ldots + (n-1)$$

Με βάση τις ιδιότητες της αριθμητικής προόδου προκύπτει ότι

$$C = \frac{n(n-1)}{2}$$

που τελικά σημαίνει, ότι η πολυπλοκότητα της ταξινόμησης αυτής είναι $O(n^2)$.

5.3.2 Γραμμική αναζήτηση

Στην παράγραφο 3.6 μελετήσαμε την απλούστερη μέθοδο αναζήτησης, τη γραμμική ή σειριακή μέθοδο. Όταν αναζητούμε ένα κλειδί που πράγματι υπάρχει στον πίνακα, τότε λέμε ότι η αναζήτηση είναι *επιτυχής* (successful). Στην αντίθετη περίπτωση, λέμε ότι η αναζήτηση είναι *ανεπιτυχής* (unsuccessful). Το κόστος της αναζήτησης μετράται με τον αριθμό των συγκρίσεων κλειδιών. Έτσι το κόστος αυτό για την επιτυχή αναζήτηση συμβολίζεται με Ε, ενώ για την ανεπιτυχή αναζήτηση συμβολίζεται με Α.

Ας εξετάσουμε αρχικά, πως προκύπτει η πολυπλοκότητα της επιτυχούς αναζήτησης σε μη ταξινομημένο πίνακα που αποτελείται από n στοιχεία. Αν αναζητάται το πρώτο στοιχείο, τότε η επιτυχής αναζήτηση θα κοστίσει μία σύγκριση, ενώ αν αναζητάται το δεύτερο στοιχείο, τότε το κόστος είναι δύο συγκρίσεις. Με την ίδια λογική, αν αναζητάται το n-οστό στοιχείο, τότε θα εκτελεσθούν n συγκρίσεις κλειδιών μέχρι την περάτωση του αλγορίθμου. Έτσι κατά μέσο όρο, ο απαιτούμενος αριθμός συγκρίσεων κλειδιών για την επιτυχή αναζήτηση είναι:

$$E = \frac{(1 + 2 + \ldots + n)}{n} = \frac{n(n+1)}{2n} = \frac{n+1}{2}$$

Από τη σχέση αυτή, εύκολα καταλήγουμε στο συμπέρασμα ότι η επιτυχής αναζήτηση έχει πολυπλοκότητα της τάξης O(n), με την απαραίτητη προϋπόθεση ότι τα κλειδιά αναζητώνται ισοπίθανα.

Η πολυπλοκότητα της ανεπιτυχούς αναζήτησης είναι επίσης τάξης O(n). Αυτό προκύπτει με βάση την απλή σκέψη ότι, όταν το αναζητούμενο κλειδί δεν υπάρχει στον πίνακα, τότε η αναζήτηση καταλήγει να εξετάσει ένα προς ένα όλα τα κλειδιά μέχρι το τέλος του πίνακα. Αν ο πίνακας είναι ταξινομημένος, τότε η διαδικασία της επιτυχούς και της ανεπιτυχούς αναζήτησης μπορεί να βελτιωθεί, ωστόσο και πάλι η πολυπλοκότητα θα είναι γραμμικής τάξης.

Πιν. 5.4. Πολυπλοκότητες μερικών αλγορίθμων

Αλγόριθμος	Πολυπλοκότητα
'Ωθηση-απώθηση σε στοίβα	O(1)
Εισαγωγή-εξαγωγή σε ουρά	O(1)
Fibonacci επαναληπτική	O(n)
Ταξινόμηση ευθείας ανταλλαγής	$O(n^2)$
Σειριακή αναζήτηση	O(n)
Δυαδική αναζήτηση	O(logn)

5.4 Είδη αλγορίθμων

Όπως αναφέρθηκε ήδη και όπως φάνηκε από τα προηγούμενα, το αντικείμενο των αλγορίθμων έχει μεγάλο πλάτος και βάθος, μεγάλη ιστορία και μεγάλο μέλλον, καθώς είναι η βάση όπου στηρίζονται όλες σχεδόν οι υπόλοιποι τομείς της Πληροφορικής. Έχουν αναπτυχθεί, λοιπόν, πολλά είδη και κατηγορίες αλγορίθμων. Για παράδειγμα, μεταξύ των άλλων στα προηγούμενα κεφάλαια μιλήσαμε για αναδρομικούς και επαναληπτικούς αλγορίθμους. Στη συνέχεια θα αναφερθούμε σε μερικές νέες έννοιες σχετικά με τις κατηγορίες των αλγορίθμων.

Είναι γενικά γνωστό ότι, υπάρχουν σήμερα ακριβοί υπολογιστές με δυνατότητες που ξεπερνούν τα όρια των προσωπικών/οικιακών υπολογιστών. Οι υπολογιστές αυτοί χρησιμοποιούνται από μεγάλες επιχειρήσεις, οργανισμούς και ιδρύματα, και διακρίνονται από τη συνθετότητά τους, αφού αποτελούνται από πολλούς επεξεργαστές, πολλές κύριες μνήμες και πολλές δευτερεύουσες μνήμες (μαγνητικούς δίσκους). Στους υπολογιστές αυτούς είναι δυνατόν ένας αλγόριθμος να κατατμηθεί σε μικρότερα κομμάτια που εκτελούνται παράλληλα, αφού απαιτούν διαφορετικούς υπολογιστικούς πόρους. Σε αυτά τα υπολογιστικά συστήματα ο χρόνος που απαιτείται από έναν αλγόριθμο, είναι κλάσμα του απαιτούμενου χρόνου από τον ίδιο αλγόριθμο σε έναν προσωπικό υπολογιστή, που τυπικά διαθέτει έναν επεξεργαστή, μία κύρια μνήμη και μία δευτερεύουσα μνήμη. Η ανάπτυξη *παράλληλων* (parallel) αλγορίθμων για περιβάλλοντα, όπως αυτό που προαναφέρθηκε, είναι μία σημαντική περιοχή που απασχολεί πλήθος επιστημόνων, καθώς διαφαίνεται ότι στο μέλλον οι υπολογιστές αυτοί θα γίνουν φθηνότεροι και θα επεκταθεί η χρήση τους. Ωστόσο, στα πλαίσια του μαθήματος αυτού δεν θα μας απασχολήσει η σχεδίαση και η ανάλυση παράλληλων αλγορίθμων.

Για να γίνει καλύτερα κατανοητή η έννοια του παράλληλου αλγόριθμου, ας θεωρήσουμε και πάλι το αρχικό παράδειγμα του 2ου κεφαλαίου, όπου πρέπει να δρομολογήσουμε τις ενέργειές μας, ώστε να γευματίσουμε. Όμως τώρα θα θεωρήσουμε ότι εργάζονται δύο άτομα. Έτσι, ο επόμενος πίνακας δίνει τις ενέργειες του κάθε ατόμου.

Πρώτο άτομο	Δεύτερο άτομο
Προετοιμασία σκευών μαγειρικής	Ετοιμασία σαλάτας
Παρασκευή φαγητού	Στρώσιμο τραπεζιού
Γεύμα	Γεύμα
Καθαριότητα τραπεζιού	Πλύσιμο πιάτων και κουζινικών

Παρατηρούμε ότι τώρα η διαδικασία ολοκληρώνεται σε λιγότερο χρόνο, αφού σε κάθε άτομο αναλογούν μόνο τέσσερις εργασίες.

Γενικά, η έννοια του παραληλισμού υπάρχει παντού γύρω μας. Για παράδειγμα, στην τράπεζα πολλοί ταμίες εξυπηρετούν ταυτόχρονα τους πελάτες που δημιουργούν μία ή περισσότερες ουρές, στο θέατρο υπάρχουν πολλές έξοδοι, ώστε ο θεατές να φεύγουν γρήγορα και με ασφάλεια, στο πρατήριο βενζίνης υπάρχουν πολλές αντλίες, ώστε να μειώνεται ο χρόνος αναμονής των αυτοκινήτων κ.λπ.

Επειδή η βελτίωση των αλγορίθμων είναι ζωτικής σημασίας, μία ιδιαίτερη περιοχή της Πληροφορικής, η *Υπολογιστική Πολυπλοκότητα* (Computational Complexity), διερευνά από την αναλυτική άποψη τους αλγορίθμους και τους ταξινομεί ανάλογα με την επίδοσή τους. Έτσι συνεχώς νέοι καλύτεροι αλγόριθμοι αναπτύσσονται, ενώ άλλοι εγκαταλείπονται ως μη αποτελεσματικοί. 'Ενας αλγόριθμος λέγεται *άριστος* (optimal), αν αποδειχθεί ότι είναι τόσο αποτελεσματικός, ώστε δεν μπορεί να κατασκευασθεί καλύτερος.

Πολυωνυμικοί (polynomial) λέγονται οι αλγόριθμοι με πολυπλοκότητα που φράσσεται από επάνω με μία πολυωνυμική έκφραση. Για παράδειγμα, πολυωνυμικοί είναι οι αλγόριθμοι τάξης $O(n)$, $O(n^{3/2})$, $O(n^2)$ κ.λπ. Συνήθως αυτοί δεν απαιτούν μεγάλη υπολογιστική προσπάθεια σε αντίθεση με τους αλγορίθμους πολυπλοκότητας τάξης $O(2^n)$, $O(n^2\,2^n)$ ή $O(n^n)$, που ονομάζονται *μη πολυωνυμικοί ή εκθετικοί*.

Πολλές φορές μερικά προβλήματα σε πρώτη εξέταση μοιάζουν παρόμοια, αλλά στη συνέχεια μπορεί να αποδειχθεί ότι έχουν εξαιρετικά διαφορετικό βαθμό δυσκολίας ως προς την εύρεση αποτελεσματικής λύσης. Για παράδειγμα, έστω τα εξής δύο προβλήματα:

- Δίδεται ένα σύνολο διαφορετικών θετικών αριθμών $x_1, x_2, \ldots, x_n$ (όπου το n άρτιος αριθμός). Να βρεθούν δύο υποσύνολα από n/2 αριθμούς, όπου η διαφορά των αθροισμάτων κάθε υποσυνόλου να είναι μέγιστη,
- Δίδεται ένα σύνολο διαφορετικών θετικών αριθμών $x_1, x_2, \ldots, x_n$ (όπου το n άρτιος αριθμός). Να βρεθούν δύο υποσύνολα από n/2 αριθμούς, όπου η διαφορά των αθροισμάτων κάθε υποσυνόλου να είναι ελάχιστη.

Σε σχέση με το πρώτο πρόβλημα, εύκολα μπορούμε να προτείνουμε αποτελεσματικούς αλγορίθμους. Για παράδειγμα, μπορούμε να ταξινομήσουμε τους n αριθμούς και να τους χωρίσουμε σε δύο ομάδες, τους μικρούς και τους μεγάλους. Η λύση αυτή έχει πολυπλοκότητα $O(n^2)$, αν χρησιμοποιηθεί η ταξινόμηση φυσσαλίδας. Μάλιστα σημειώνεται ότι, μπορεί να προταθεί και ακόμη πιο αποτελεσματικός αλγόριθμος γραμμικής πολυπλο-

κότητας. Ωστόσο, σε σχέση με το δεύτερο πρόβλημα δεν υπάρχει αποτελεσματικός αλγόριθμος. Η μοναδική λύση στηρίζεται στην εξαντλητική δοκιμή όλων των δυνατών τρόπων. Ένας τέτοιος αλγόριθμος είναι πολυπλοκότητας $O(2^n)$.

Κατά παρόμοιο τρόπο υπάρχουν πολλά προβλήματα που δεν μπορούν να επιλυθούν με κάποιο αποτελεσματικό (δηλαδή πολυωνυμικό) αλγόριθμο, αλλά πρέπει να δοκιμασθούν όλες οι δυνατές περιπτώσεις ώστε να επιλεχθεί η καλύτερη. Τα προβλήματα αυτά ονομάζονται *δυσχείριστα* (intractable). Ευρύτατα γνωστό ως δυσχείριστο πρόβλημα είναι το πρόβλημα της συντομότερης διαδρομής που πρέπει ένας περιοδεύων πωλητής να ακολουθήσει, ώστε να περάσει μία και μόνο μία φορά από κάθε πόλη και να επιστρέψει στην αρχική. Το πρόβλημα αυτό αναφέρθηκε ως παράδειγμα στην αρχή του κεφαλαίου. Ένα άλλο γνωστό δυσχείριστο πρόβλημα είναι ο χρωματισμός ενός γράφου. Σύμφωνα με το πρόβλημα αυτό, πρέπει να χρωματίσουμε τις κορυφές ενός γράφου χρησιμοποιώνας ένα συγκεκριμένο αριθμό χρωμάτων, έτσι ώστε δύο γειτονικές κορυφές ή περιοχές να μην έχουν το ίδιο χρώμα. Δυσχείριστο επίσης είναι το πρόβλημα του χρωματισμού των συνδέσεων μεταξύ των ακμών ενός γράφου, έτσι ώστε οι συνδέσεις που καταλήγουν σε μία συγκεκριμένη κορυφή να είναι διαφορετικού χρώματος.

Τα δυσχείριστα αυτά προβλήματα, που αποκαλούνται και *ανοικτά* προβλήματα, λέγεται ότι ανήκουν στην κατηγορία των προβλημάτων *NP-πλήρων* προβλημάτων από τα αρχικά των αγγλικών όρων *Nondeterministic Polynomial*. Για κάθε πρόβλημα της κατηγορίας αυτής δεν υπάρχει ή τουλάχιστον δεν έχει βρεθεί ακόμη αλγόριθμος επίλυσής τους σε πολυωνυμικό χρόνο.

Η αδυναμία της επιστήμης να προτείνει αποτελεσματικούς αλγορίθμους για πολλά δυσχείριστα προβλήματα, έχει αναγκαστικά οδηγήσει στην ανάπτυξη *προσεγγιστκών* (approximate) αλγορίθμων για την επίλυσή τους, έτσι ώστε να επιτυγχάνεται μία αποδεκτή λύση σε λογικό χρόνο. Λέγοντας *αποδεκτή* εννοούμε ότι η λύση αυτή, ναι μεν δεν είναι η καλύτερη δυνατή, ωστόσο δίνει στο πρόβλημα μία απάντηση που πλησιάζει την καλύτερη λύση σε μεγάλο βαθμό.

Ευριστικός είναι ο αλγόριθμος που είτε μπορεί να οδηγήσει σε μία καλή ή ακόμη και βέλτιστη λύση ενός προβλήματος, αν είμαστε τυχεροί, αλλά μπορεί να οδηγήσει και σε μία λύση που απέχει πολύ από τη βέλτιστη ή ακόμη, και να αποτύχει να βρει λύση, αν είμαστε άτυχοι. Οι ευριστικοί αλγόριθμοι δεν είναι τυποποιημένοι και στηρίζονται σε μια εμπειρική παρατήρηση ή ένα τέχνασμα ή μία έμπνευση του προγραμματιστή. Συνήθως οι προσεγγι-

στικοί αλγόριθμοι είναι ευριστικοί αλγόριθμοι, αλλά βέβαια υπάρχουν και πολλοί ευριστικοί αλγόριθμοι που δεν είναι προσεγγιστικοί.

Ανακεφαλαίωση

Αρχικά ορίστηκε και αναλύθηκε η έννοια της επίδοσης των αλγορίθμων, με ειδικότερη αναφορά στη χειρότερη περίπτωση κάθε αλγόριθμου.

'Εγινε καταγραφή των μεγεθών που επηρεάζουν την επίδοση ενός αλγορίθμου και των βασικών πράξεων για την ταξινόμηση, την αναζήτηση και τον πολλαπλασιασμό καθώς και για άλλα είδη προβλημάτων.

Δόθηκε ορισμός για την αποδοτικότητα και για την ορθότητα των αλγορίθμων. Η πολυπλοκότητα αλγορίθμων περιγράφεται αναλυτικά και τυποποιούνται οι ορισμοί για τις κατηγορίες (τάξεις) πολυπλοκότητας των περισσότερων ειδών αλγορίθμων.

Μερικοί από τους πλέον γνωστούς αλγορίθμους αναλύονται και εκφράζεται η πολυπλοκότητα τους με χρήση των αντίστοιχων συμβολισμών των κατηγοριών πολυπλοκότητας.

Τέλος, παρουσιάζεται καταγραφή των ειδών των αλγορίθμων και τυποποιούνται με χρήση ορισμών οι πλέον γνωστοί τύποι αλγορίθμων (πολυωνυμικοί, μη πολυωνυμικοί, παράλληλοι αλγόριθμοι).

Λέξεις κλειδιά

Επίδοση/αποδοτικότητα αλγορίθμου, Χειρότερη περίπτωση αλγορίθμου, Μέγεθος εισόδου, Ορθότητα αλγορίθμου, Πολυπλοκότητα αλγορίθμων, Συμβολισμός Ο, Ανάλυση αλγορίθμων, Είδη αλγορίθμων, Υπολογιστική Πολυπλοκότητα, Προσεγγιστικοί αλγόριθμοι, Ευριστικοί αλγόριθμοι

Ερωτήσεις - Θέματα για συζήτηση

1. Ποιά είναι τα πρωταρχικά ερωτήματα που τίθενται για την κατανόηση της επίδοσης ενός αλγορίθμου;
2. Τι είναι και πώς μπορεί να εκφρασθεί η χειρότερη περίπτωση ενός αλγορίθμου;
3. Να περιγραφεί το μέγεθος της εισόδου ενός αλγορίθμου και να δοθεί ένα παράδειγμα αναγνώρισης και καταγραφής αυτού του μεγέθους.

4. Από ποιους κύριους παράγοντες εξαρτάται ο χρόνος εκτέλεσης ενός αλγορίθμου;
5. Ποιες είναι οι απαραίτητες προϋποθέσεις για να είναι δυνατή η σύγκριση μεταξύ δύο προγραμμάτων αλγορίθμων;
6. Να ορισθεί η αποδοτικότητα αλγορίθμων.
7. Τι είναι ο έλεγχος ορθότητας ενός αλγορίθμου και ποιοι περιορισμοί υπάρχουν για αυτόν τον έλεγχο;
8. Να περιγραφεί ο τρόπος απόδειξης της ορθότητας ενός αλγορίθμου και να δοθεί ένα παράδειγμα απόδειξης ορθότητας αλγορίθμου.
9. Να δοθεί ο ορισμός της πολυπλοκότητας αλγορίθμου.
10. Με τη βοήθεια ενός διαγράμματος να γίνει σύγκριση μεταξύ της λογαριθμικής, της γραμμικής και της τετραγωνικής πολυπλοκότητας.
11. Ποια είναι η πολυπλοκότητα των λειτουργιών σε στοίβες και ουρές;
12. Ποια είναι η πολυπλοκότητα της σειριακής αναζήτησης;
13. Ποια είναι τα κυριότερα είδη αλγορίθμων;

Βιβλιογραφία

1. Φώτω Αφράτη και Γιώργος Παπαγεωργίου, *Αλγόριθμοι – Μέθοδοι Σχεδίασης και Ανάλυση Πολυπλοκότητας*, Εκδόσεις Συμμετρία, Αθήνα, 1993.
2. Χρήστος Κοίλιας, *Δομές Δεδομένων και Οργάνωση Αρχείων*. Εκδόσεις Νέων Τεχνολογιών, Αθήνα, 1993.
3. Μανόλης Λουκάκης, *Δομές Δεδομένων και Αλγόριθμοι*, Εκδόσεις Θεραπευτικής Κοινότητας Ιθάκη, Θεσσαλονίκη, 1988.
4. Ιωάννης Μανωλόπουλος, *Δομές Δεδομένων – μία Προσέγγιση με Pascal*, Εκδόσεις Art of Text, Θεσσαλονίκη, 1998.
5. Ν. Μισυρλής, *Δομές Δεδομένων*, Συμμετρία, Αθήνα, 1993.
6. Niklaus Wirth, *Αλγόριθμοι και Δομές Δεδομένων*, Κλειδάριθμος, Αθήνα, 1990.
7. G. Brassard and P.Bratley, *Fundamentals of Algorithms*, Prentice Hall, 1996.
8. T. Cormen, C. Leiserson and R. Rivest, *Introduction to Algorithms* MIT Press, 1990.

9. E. Horowitz, S. Sahni and S. Rajasekaran, *Computer Algorithms*, Computer Science Press, 1998.

10. I. Oliver, *Programming Classics: Implementing the World's Best Algorithms*, Prentice Hall, 1993.

Διευθύνσεις Διαδικτύου

➨ http://www.csd.auth.gr/~contest/

Κόμβος που ανήκει στο Τμήμα Πληροφορικής του Αριστοτέλειου Πανεπιστήμιου Θεσσαλονίκης. Παρέχει πληροφορίες για τις Βαλκανιάδες Πληροφορικής και συνδέει με άλλους παρεμφερείς κόμβους.

➨ http://olympiads.win.tue.nl/ioi/

Κόμβος Ολυμπιάδων Πληροφορικής με σχετικά προβλήματα αλγορίθμων.

➨ http://acm.baylor.edu/acmicpc/

Κόμβος με πληροφορίες και προβλήματα από μαθητικούς διαγωνισμούς Πληροφορικής που διοργανώνονται από τον οργανισμό Association for Computing Machinery (ACM).