Αλγόριθμοι Ι

Κάθε καλώς ορισμένη υπολογιστική διαδικασία, η οποία καλείται να επιλύσει ένα συγκεκριμένο πρόβλημα εντός πεπερασμένου χρόνου.

Αλγόριθμος είναι μια **πεπερασμένη σειρά ενεργειών**, **αυστηρά καθορισμένων** και **εκτελέσιμων σε πεπερασμένο χρόνο**, που στοχεύουν στην επίλυση ενός προβλήματος.

Αλγόριθμοι ΙΙ

Ένας αλγόριθμος πρέπει να διαθέτει:

- είσοδο: καμία, μία ή περισσότερες τιμές δεδομένων
- έξοδο: τουλάχιστο μία τιμή δεδομένων
- καθοριστικότητα: κάθε ενέργεια να προσδιορίζεται σαφώς για τον τρόπο εκτέλεσής της
- περατότητα: να τελειώνει μετά από πεπερασμένα βήματα εκτέλεσης των εντολών του
- αποτελεσματικότητα: κάθε μεμονωμένη εντολή του αλγορίθμου να είναι απλή και εκτελέσιμη

Αλγόριθμοι III

Κάθε είσοδος που ικανοποιεί τις προδιαγραφές του προβλήματος καλείται **νόμιμη** και λέμε πως ορίζει ένα συγκεκριμένο **στιγμιότυπο** του προβλήματος.

Ένας αλγόριθμος **επιλύει ένα πρόβλημα**, όταν **για κάθε στιγμιότυπο** του προβλήματος **τερματίζει** μετά από πεπερασμένο χρόνο, **παράγοντας σωστή έξοδο**.

Αλγορίθμοι IV

Τεχνικές σχεδίασης αλγορίθμων:

- αλγόριθμοι brute-force: διακρίνονται από το γεγονός ότι επιλύουν το πρόβλημα με τον πιο απλό, άμεσο ή προφανή τρόπο. Η λειτουργία τους βασίζεται στον εξαντλητικό έλεγχο όλων των πιθανών περιπτώσεων.
- άπληστοι αλγόριθμοι: προχωρούν με βάση σταδιακές επιλογές, που αφορούν στο βέλτιστο κάθε βήματος.
- αλγόριθμοι οπισθοδρόμησης: σε κάθε βήμα εξετάζουν συστηματικά όλα τα πιθανά αποτελέσματα κάθε απόφασης, εκτός και αν αποφασίσουν ότι για κάποιες περιπτώσεις δεν είναι απαραίτητος ένας τέτοιος εξαντλητικός έλεγχος.
- αλγόριθμοι διαίρει και βασίλευε: το πρόβλημα υποδιαιρείται σε πολλά μικρά υποπροβλήματα πανομοιότυπα με το αρχικό. Κάθε ένα από αυτά επιλύεται ανεξάρτητα και οι επιμέρους λύσεις συνδυάζονται σε μία τελική λύση του αρχικού προβλήματος.

Αλγόριθμοι V

- αλγόριθμοι δυναμικού προγραμματισμού: το πρόβλημα υποδιαιρείται σε πολλά μικρά προβλήματα, τα οποία επιλύονται σταδιακά ως σύνθεση των λύσεων των μικρότερων και άρα απλούστερων υποπροβλημάτων. Είναι ιδιαίτερα διαδεδομένοι για την επίλυση προβλημάτων βελτιστοποίησης.
- τυχαιοποιημένοι αλγόριθμοι: αλγόριθμοι που φαίνεται ότι συμπεριφέρονται τυχαία

- αλγόριθμοι γράφων
- αλγόριθμοι αναγνώρισης συμβολοσειρών
- αριθμητικοί αλγόριθμοι
- κατανεμημένοι αλγόριθμοι
- αλγόριθμοι συντακτικής – σημασιολογικής ανάλυσης (μεταγλωττιστές)

Ανάλυση Αλγορίθμων I

Ποιος είναι ο καλύτερος αλγόριθμος;

Συγκρίνουμε τους αλγορίθμους με βάση την **αποδοτικότητά** τους ως συνάρτηση του **μεγέθους** των περιπτώσεων του προβλήματος για τον υπολογισμό **χρόνου** ή **χώρου.** Διακρίνουμε την

- πρακτική ή εκ των υστέρων προσέγγιση
- την θεωρητική ή εκ των προτέρων προσέγγιση

Η αποδοτικότητα χαρακτηρίζεται ως συνάρτηση του μεγέθους του προβλήματος

Με τον όρο **μέγεθος** αναφερόμαστε σε οποιοδήποτε ακέραιο που με κάποιο τρόπο «μετράει» τον αριθμό των συστατικών μερών μιας περίπτωσης του προβλήματος.

Παράδειγμα: πρόβλημα ταξινόμησης
μέγεθος περίπτωσης είναι ο αριθμός n των προς ταξινόμηση δεδομένων

Ανάλυση Αλγορίθμων ΙΙ

Μοντέλο Μηχανής Τυχαίας Προσπέλασης (RAM)

Αφαιρετική θεώρηση υπολογιστικού συστήματος που:

- αναφέρεται σε συστήματα του ενός επεξεργαστή
- δεν έχει τη δυνατότητα τελέσεως ταυτόχρονων πράξεων
- διαθέτει τους αναγκαίους καταχωρητές, ένα συσσωρευτή και μία ακολουθία απθηκευτικών θέσεων με διευθύνσεις οι οποίες συνιστούν την κύρια μνήμη
- είναι σε θέση να εκτελεί τις αριθμητικές πράξεις {+, -, *, /, mod}, να παίρνει αποφάσεις τύπου if και να γράφει – διαβάζει από και προς τις θέσεις μνήμης

Ανάλυση Αλγορίθμων ΙΙΙ

Οι στοιχειώδεις πράξεις που μπορεί να εκτελέσει μια μηχανή RAM έχουν κάποιο κόστος σε χρόνο:

- μέτρηση μοναδιαίου κόστους: σταθερό, πεπερασμένο κόστος ανεξαρτήτως του μήκους της δυαδικής αναπαράστασης των τελεστέων
- μέτρηση λογαριθμικού κόστους: η πράξη παίρνει χρόνο ανάλογο με το μήκος της δυαδικής αναπαράστασης

Εμείς θα χρησιμοποιήσουμε την πρώτη προσέγγιση.

Θεώρημα: Κάθε υπολογιστική διαδικασία μιας μηχανής Turing, με κόστος πολυωνυμικό στο μέγεθος της εισόδου, μπορεί να εξομοιωθεί από μία υπολογιστική, πολυωνυμική στο μέγεθος της εισόδου, διαδικασία μιας μηχανής RAM και αντίστροφα.

Ανάλυση Αλγορίθμων IV

Η ανάλυση θα γίνεται βάση του μοναδιαίου μέτρου ως συνάρτηση του μεγέθους της εισόδου που:

- για ένα πρόβλημα ταξινόμησης είναι το πλήθος n των προς διάταξη αντικειμένων
- για έναν αλγόριθμο εύρεσης ελάχιστου επικαλυπτόμενου δένδρου σε γράφο με σύνολο κορυφών V και σύνολο ακμών E, η είσοδος έχει μέγεθος $n=|V|+|E|$
- για την τέλεση πολλαπλασιασμών μεταξύ μεγάλων ακεραίων το μέγεθος εισόδου είναι το συνολικό μήκος σε αριθμό bit της δυαδικής αναπαράστασής τους (όχι το πλήθος τους)

Ανάλυση Αλγορίθμων V

Υπολογισμός κόστους ενός αλγορίθμου:

- σε κάθε εντολή ανάθεσης τιμής ή δήλωσης θα χρεώνουμε σταθερό χρόνο
- οι βρόχοι **for** και **while** θα χρεώνονται το πλήθος των επαναλήψεων επί το πλήθος των εντολών που εκτελούνται σε κάθε επανάληψη

Συχνά θα επικεντρωνόμαστε στις **κυρίαρχες πράξεις**, δηλαδή αυτές που παίζουν καταλυτικό ρόλο στη συμπεριφορά του αλγορίθμου:

π.χ. στους αλγορίθμους ταξινόμησης οι κυρίαρχες πράξεις είναι οι συγκρίσεις και οι ανταλλαγές στοιχείων μεταξύ δύο θέσεων του πίνακα εισόδου

Ανάλυση Αλγορίθμων VI

Παράδειγμα ανάλυσης:

Algorithm selectionSort(A)
Input: Πίνακας A *n* αριθμών
Output: Διατεταγμένος πίνακας

	κόστος	#εκτελέσεων
1. **for** (i=0;i<n;i++) {	c_1	$n+1$
2. min=i;	c_2	n
3. **for** (j=i+1;j<=n;j++)	c_3	$\sum_{j-1}^{n} j$
4. **if** (a[j]<a[min]) min=j;	c_4	$\sum_{j-1}^{n} (j-1)$
5. temp=a[i];	c_5	n
6. a[i]=a[min];	c_6	n
7. a[min]=temp;	c_7	n
8. }		

Ανάλυση Αλγορίθμων VII

Συνολικός χρόνος αλγορίθμου:

$$T(n) = (c_3 + c_4)\frac{n^2}{2} + (c_1 + c_2 + \frac{c_3}{2} - \frac{c_4}{2} + c_5 + c_6 + c_7)n + c_1$$

- ένας αλγόριθμος είναι **γρηγορότερος** από έναν άλλο αν

$$T_1(n) \leq T_2(n), \forall n$$

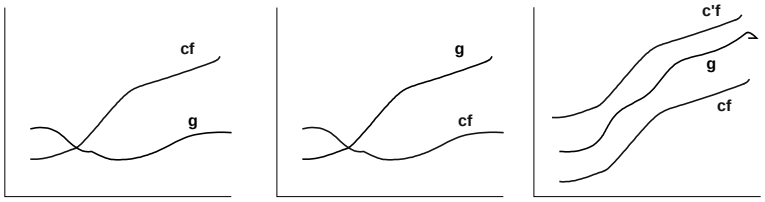
- ένας αλγόριθμος είναι **ασυμπτωτικά γρηγορότερος** από έναν άλλο αν

$$\lim_{n\to\infty} \frac{T_1(n)}{T_2(n)} = 0$$

Ανάλυση Αλγορίθμων VIII

Στις περισσότερες περιπτώσεις είτε δεν είμαστε σε θέση είτε δε μας ενδιαφέρει να βρούμε ακριβείς τύπους και απλά ασχολούμαστε με το **ρυθμό αύξησης** ή την **τάξη αύξησης** των συναρτήσεων που εκφράζουν την πολυπλοκότητα του αλγορίθμου.

Τότε μιλάμε για ασυμπτωτικές εκτιμήσεις *O*, *Θ* ή *Ω*.



***O*(*f*)** ***Ω*(*f*)** ***Θ*(*f*)**

Ανάλυση Αλγορίθμων IX

Αλγόριθμος που εκτελείται σε χρόνο τάξης

- n εμφανίζει **γραμμική πολυπλοκότητα**
- n^2 εμφανίζει **τετραγωνική πολυπλοκότητα**
- n^3 εμφανίζει **κυβική πολυπλοκότητα**
- n^k **πολυωνυμική πολυπλοκότητα**
- c^n **εκθετική πολυπλοκότητα**

- Αποδοτικότητα χώρου μνήμης

για παράλληλους αλγορίθμους:

- Αποδοτικότητα επικοινωνίας

Ανάλυση Αλγορίθμων X

n	10	20	50	100
n^2	10^{-4} sec	4 10^{-4} sec	25 10^{-4} sec	10^{-2} sec
n^5	10^{-1} sec	3.2 min	5.2 min	2.8 ώρες
2^n	10^{-3} sec	1 sec	35.7 χρόνια	400 τρις αιώνες
n^n	2.8 ώρες	3.3 τρις χρόνια		

$$O(1) \subset O(\log n) \subset O(poly \log n) \subset O(n^k) \subset O(n^k poly \log n)$$
$$\subset O(2^n) \subset O(n!) \subset O(n^n)$$

Ανάλυση Αλγορίθμων XI

ΟΡΙΣΜΟΣ

Θεωρούμε ότι η συνάρτηση **f(n) είναι μη αρνητική** για όλους τους ακεραίους $n \geq 0$. Λέμε ότι f(n)=O(g(n)), αν υπάρχει ακέραιος n_0 και μία σταθερά c>0 έτσι ώστε για όλους τους ακέραιους $n \geq n_0$ $f(n) \leq cg(n)$.

ΠΑΡΑΔΕΙΓΜΑ

Έστω f(n)=8n+128. Ισχύει ότι f(n)=O(n^2)?

Ας θεωρήσουμε c=1. Τότε, $f(n) \leq cn^2 \Rightarrow 8n+128 \leq n^2$

$$\Rightarrow 0 \leq n^2 - 8n - 128$$
$$\Rightarrow 0 \leq (n-16)(n+8)$$

Αφού για κάθε $n \geq 0$ ισχύει (n+8)>0, θέλουμε $n_0 - 16 \geq 0 \Rightarrow n_0 \geq 16$

Άρα ισχύει ότι f(n)=0(n^2)

Ανάλυση Αλγορίθμων XII

ΟΡΙΣΜΟΣ

Θεωρούμε τη συνάρτηση f(n)=O(g(n)). Αν για κάθε συνάρτηση h(n) έτσι ώστε f(n)=O(h(n)), ισχύει επίσης ότι g(n)=O(h(n)), τότε λέμε ότι η g(n) είναι ένα **στενό ασυμπτωτικό όριο** της f(n).

Παράδειγμα: Δείξαμε ότι για f(n)=8n+128 ισχύει ότι f(n)=O(n^2). Παρόλα αυτά, το n^2 δεν είναι ένα στενό ασυμπτωτικό όριο, γιατί ισχύει επίσης f(n)=O(n).

Βέβαια, για να αποδειχθεί ότι η g(n)=n είναι ένα στενό ασυμπτωτικό όριο πρέπει να δείξουμε ότι για κάθε h(n) που f(n)=O(h(n)), ισχύει ότι g(n)=O(h(n)). Για το συγκεκριμένο παράδειγμα μπορεί αυτό να γίνει με εις άτοπο απαγωγή.

Ανάλυση Αλγορίθμων XIII

ΣΥΜΒΑΣΕΙΣ:

- Συνηθίζεται σε μία έκφραση κεφαλαίου «O» να διατηρούμε μόνο τον πιο σημαντικό όρο. Έτσι, αντί για O(n^2+nlogn+n) γράφουμε απλά O(n^2).
- Συνηθίζεται να αγνοούνται οι συντελεστές. Έτσι, αντί για O(3n^2), απλά γράφουμε O(n^2).
- Προτιμούμε την εύρεση στενού ασυμπτωτικού ορίου. Έτσι, αντί να γράφουμε f(n)=n=O(n^3) προτιμούμε να γράφουμε f(n)=O(n).
- Σε κάποια βιβλία αντί για f(n)=O(g(n)) γράφουνε f(n)$\in$O(g(n)). (βλ. Brassard & Bratley).

Ανάλυση Αλγορίθμων XIV

ΑΣΥΜΠΤΩΤΙΚΟ ΚΑΤΩ ΟΡΙΟ (ΣΥΜΒΟΛΙΣΜΟΣ Ω):

Θεωρούμε ότι η συνάρτηση **f(n) είναι μη αρνητική** για όλους τους ακεραίους $n \geq 0$. Λέμε ότι f(n)=Ω(g(n)), αν υπάρχει ακέραιος n_0 και μία σταθερά c>0 έτσι ώστε για όλους τους ακέραιους $n \geq n_0$, $f(n) \geq cg(n)$.
Όλες οι συμβάσεις που εισάγαμε στο συμβολισμό κεφαλαίου «Ο» ισχύουν και για το συμβολισμό κεφαλαίου «Ω».

ΠΑΡΑΔΕΙΓΜΑ: Αποδείξτε ότι αν $f(n)=5n^2-64n+256$, τότε $f(n)=\Omega(n^2)$.

ΑΣΥΜΠΤΩΤΙΚΟΣ ΣΥΜΒΟΛΙΣΜΟΣ Θ:

Θεωρούμε ότι η συνάρτηση **f(n) είναι μη αρνητική** για όλους τους ακεραίους $n \geq 0$. Λέμε ότι f(n)=Θ(g(n)), αν και μόνο αν f(n)=O(g(n)) και f(n)=Ω(g(n)).

Τότε λέμε ότι η f(n) είναι **ακριβώς τάξης** g(n) και αν μπορεί να αποδειχθεί κάτι τέτοιο για έναν αλγόριθμο είναι πιο ισχυρό από ότι αν απλά ο αλγόριθμος είναι τάξης g(n).

Ανάλυση Αλγορίθμων XV

ΑΣΥΜΠΤΩΤΙΚΟΣ ΣΥΜΒΟΛΙΣΜΟΣ ΠΕΖΟΥ «ο»:

Θεωρούμε ότι η συνάρτηση **f(n) είναι μη αρνητική** για όλους τους ακεραίους $n \geq 0$. Λέμε ότι f(n)=o(g(n)), αν και μόνο αν f(n)=O(g(n)), αλλά δεν ισχύει $f(n) \notin \Theta(g(n))$.

ΑΣΥΜΠΤ. ΣΥΜΒΟΛΙΣΜΟΣ ΜΕ ΠΟΛΛΕΣ ΠΑΡΑΜΕΤΡΟΥΣ:

Χρήσιμος για περιπτώσεις όπου για παράδειγμα ο χρόνος εκτέλεσης εξαρτάται από περισσότερες από μία παραμέτρους της κάθε περίπτωσης π.χ. αλγόριθμοι γράφων που εξαρτώνται από τον αριθμό των ακμών και τον αριθμό των κορυφών. Λέμε τότε ότι f(m,n)=O(g(m,n)), αν υπάρχουν ακέραιοι n_0 και m_0 και μία σταθερά c>0, έτσι ώστε για όλους τους ακέραιους $n \geq n_0$ και $m \geq m_0$ ισχύει $f(m,n) \leq cg(m,n)$.

Ανάλυση Αλγορίθμων XVI

ΑΣΥΜΠΤ. ΣΥΜΒΟΛΙΣΜΟΣ ΥΠΟ ΣΥΝΘΗΚΗ:

Μία συνάρτηση t(n) είναι O(f(n)|P(n)) αν υπάρχει ένας ακέραιος n_0 και μία σταθερά c>0, έτσι ώστε για όλους τους ακέραιους $n \geq n_0$ να συνάγεται P(n)⇒t(n)≤cf(n).

Έτσι, αν αυτό βολεύει, μπορούμε να μελετήσουμε την αποδοτικότητα ενός αλγορίθμου για περιπτώσεις με μεγέθη π.χ. που προκύπτουν ως δυνάμεις του 2. Συνήθως βέβαια, στη συνέχεια, αναζητείται κάποιος τρόπος για να αρθεί η συνθήκη της οποίας ο ρόλος περιορίζεται στη διευκόλυνση της ανάλυσης της του αλγορίθμου.

ΠΡΑΞΕΙΣ ΜΕ ΑΣΥΜΠΤ. ΣΥΜΒΟΛΙΣΜΟ:

Τι σημαίνει ο συμβολισμός O(f(n))+O(g(n)); Ο συμβολισμός αυτός εκφράζει το σύνολο των συναρτήσεων που προκύπτει με την πρόσθεση της τιμής οποιασδήποτε συνάρτησης ανήκει στο O(f(n)) στην τιμή οποιασδήποτε συνάρτησης ανήκει στο O(g(n)).
Γενικά, μία συνάρτηση t(n) είναι O(f(n)) op O(g(n)), αν υπάρχουν συναρτήσεις f και g, που ανήκουν αντίστοιχα στα σύνολα συναρτήσεων O(f(n)) και O(g(n)) και ένας ακέραιος n_0 έτσι ώστε για όλους τους ακέραιους $n \geq n_0$
t(n)=f(n) op g(n)

Ανάλυση Αλγορίθμων XVII

ΘΕΩΡΗΜΑ

Αν θεωρήσουμε ότι $f_1(n)=O(g_1(n))$ και $f_2(n)=O(g_2(n))$, τότε

$$f_1(n)+f_2(n)=O(\max(g_1(n),g_2(n)))$$

ΑΠΟΔΕΙΞΗ: Εξ ορισμού υπάρχουν δύο ακέραιοι n_1 και n_2 και δύο σταθερές c_1 και c_2, έτσι ώστε $f_1(n)\leq c_1g_1(n)$ για $n\geq n_1$ και $f_2(n)\leq c_2g_2(n)$ για $n\geq n_2$.
Έστω ότι $n_0=\max(n_1,n_2)$ και $c_0=2\max(c_1,c_2)$. Τότε, για $n\geq n_0$

$$\begin{aligned} f_1(n)+f_2(n) &\leq c_1g_1(n)+c_2g_2(n) \\ &\leq c_0(g_1(n)+g_2(n))/2 \\ &\leq c_0\max(g_1(n),g_2(n)) \end{aligned}$$

ΘΕΩΡΗΜΑ

Αν $f(n)=f_1(n)+f_2(n)$ και οι $f_1(n)$ και $f_2(n)$ είναι και οι δύο μη αρνητικές για όλους τους ακεραίους $n\geq 0$, έτσι ώστε

$$\lim_{n\to\infty}\frac{f_2(n)}{f_1(n)}=L$$

για κάποιο $L\geq 0$, τότε $f(n)=0(f_1(n))$.

Ανάλυση Αλγορίθμων XVIII

ΣΗΜΑΝΤΙΚΟ ΘΕΩΡΗΜΑ, διότι χάρη σε αυτό συμπεραίνουμε π.χ. ότι $f_1(n)+f_2(n)=n^3+n^2=O(n^3)$. ΓΙΑΤΙ??

ΕΠΕΙΔΗ: $\lim_{n\to\infty}\frac{f_2(n)}{f_1(n)}=\lim_{n\to\infty}\frac{n^2}{n^3}=\lim_{n\to\infty}\frac{1}{n}=0$

ΘΕΩΡΗΜΑ

Αν θεωρήσουμε ότι $f_1(n)=O(g_1(n))$ και $f_2(n)=O(g_2(n))$, τότε

$$f_1(n)Xf_2(n)=O(g_1(n)Xg_2(n))$$

ΑΠΟΔΕΙΞΗ: Εξ ορισμού υπάρχουν δύο ακέραιοι n_1 και n_2 και δύο σταθερές c_1 και c_2, έτσι ώστε $f_1(n)\le c_1g_1(n)$ για $n\ge n_1$ και $f_2(n)\le c_2g_2(n)$ για $n\ge n_2$. Επιπλέον, οι $f_1(n)$ και $f_2(n)$ είναι μη αρνητικές για όλους τους ακεραίους $n\ge 0$.
Θεωρούμε $n_0=\max(n_1,n_2)$ και $c_0=c_1c_2$. Τότε για $n\ge n_0$

$$\begin{aligned} f_1(n)\times f_2(n) &\le c_1g_1(n)\times c_2g_2(n)\\ &\le c_0(g_1(n)+g_2(n)) \end{aligned}$$

Ανάλυση Αλγορίθμων XIX

ΣΗΜΑΝΤΙΚΟ ΘΕΩΡΗΜΑ, διότι χάρη σε αυτό συμπεραίνουμε π.χ. ότι $f_1(n)Xf_2(n)=(n^3+n^2+n+1)X(n^2+n+1)=0(n^3Xn^2)=O(n^5)$.

ΘΕΩΡΗΜΑ

Αν θεωρήσουμε ότι $f_1(n)=O(g_1(n))$ και η $g_2(n)$ είναι μη αρνητική για όλους τους ακεραίους $n\ge 0$, τότε

$$f_1(n)Xf_2(n)=O(g_1(n)Xg_2(n))$$

ΘΕΩΡΗΜΑ

Αν θεωρήσουμε ότι $f(n)=O(g(n))$ και η $g(n)= O(h(n))$, τότε $f(n)=O(h(n))$.

ΣΗΜΑΝΤΙΚΟ ΘΕΩΡΗΜΑ, διότι χάρη σε αυτό συμπεραίνουμε π.χ. ότι αν $f_1(n)=5n^3$, που είναι $O(n^3)$ και $f_2(n)=3n^2$, τότε
$f_1(n)+f_2(n)=0(f_1(n))$, γιατί $\lim_{n\to\infty}\frac{f_2(n)}{f_1(n)}=0$. Το γεγονός όμως ότι $f_1(n)=O(n^3)$,

δίνει με μεταβατική ιδιότητα ότι $f_1(n)+f_2(n)= O(n^3)$

Ανάλυση Αλγορίθμων XX

ΘΕΩΡΗΜΑ

Αν θεωρήσουμε ένα πολυώνυμο της μορφής

$$f(n) = \sum_{i=0}^{m} a_i n^i$$
$$= a_m n^m + a_{m-1} n^{m-1} + \ldots + a_2 n^2 + a_1 n + a_0$$

όπου $a_m > 0$. Τότε f(n)=O(n^m).

ΑΠΟΔΕΙΞΗ: Αφού το μέγεθος n είναι εξ ορισμού μη αρνητικός αριθμός, για να είναι ένας όρος του πολυωνύμου αρνητικός πρέπει $a_i < 0$. Άρα για κάθε όρο του αθροίσματος ισχύει $a_i n^i \le |a_i| n^i$. Επειδή θεωρήσαμε ότι $a_m > 0$

$$f(n) \le \sum_{i=0}^{m} |a_i| n^i$$
$$\le n^m \sum_{i=0}^{m} |a_i| n^{i-m}, n \ge 1$$
$$\le n^m \sum_{i=0}^{m} |a_i| \frac{1}{n^{m-i}}$$

Επειδή $n \ge 1, \; 1/(n^{m-i}) \le 1, \gamma\iota\alpha \cdot 0 \le i \le m$

$$f(n) \le n^m \sum_{i=0}^{m} |a_i|, n \ge 1$$

Ανάλυση Αλγορίθμων XXI

ΘΕΩΡΗΜΑ

Αν θεωρήσουμε ένα πολυώνυμο της μορφής

$$f(n) = \sum_{i=0}^{m} a_i n^i$$
$$= a_m n^m + a_{m-1} n^{m-1} + \ldots + a_2 n^2 + a_1 n + a_0$$

όπου $a_m > 0$. Τότε f(n)=Ω(n^m).

ΘΕΩΡΗΜΑ

Για κάθε ακέραιο k≥1, log^{k}n=O(n).

Ανάλυση Αλγορίθμων XXII

Για να βρεθούν ασυμπτωτικά όρια:

- Όταν ο αλγόριθμος περιέχει επαναληπτικές δομές ελέγχου (while, for κ.λ.π.), τότε η αποδοτικότητα εκφράζεται ως άθροισμα όρων, π.χ. $\sum_{i=2}^{n} i$. Έτσι, η ανάλυση ανάγεται στην δυνατότητα χειρισμού τέτοιων αθροισμάτων και εύρεσης ασυμπτωτικών ορίων.

  Πρέπει να γνωρίζετε πολύ καλά τον υπολογισμό σειρών:
  - αριθμητικών
  - γεωμετρικών
  - αρμονικών

  για τις οποίες προσπαθείτε να αποδείξετε την ισχύ ενός ασυμπτωτικού ορίου με τη χρήση επαγωγής.

Ανάλυση Αλγορίθμων XXIII

- Όταν ο αλγόριθμος περιέχει αναδρομικές κλήσεις, τότε η αποδοτικότητά του εκφράζεται από αναδρομικές εξισώσεις ή ανισοϊσότητες, π.χ.

$$T(n) = 2T(\lfloor n/2 \rfloor) + 1$$

  τότε προσπαθούμε να βρούμε την αποδοτικότητα με κάποιο από το εξής τρόπους:
  - προσπαθούμε να «μαντέψουμε»
  - με επαναληπτική αντικατάσταση
  - με τη γενική μέθοδο

Ανάλυση Αλγορίθμων XXIV

… προσπαθώντας να «μαντέψουμε»

Έστω ότι θέλουμε να δείξουμε ότι αν το κόστος είναι

$$T(n) = 2T(\lfloor n/2 \rfloor) + 1$$

η λύση είναι $O(n)$. Άρα πρέπει να δείξουμε ότι για μία σταθερά c, $T(n) \le cn$. Εφαρμόζουμε ισχυρά επαγωγή και αντικαθιστούμε στην αναδρομική εξίσωση:

$$T(n) = 2c(\lfloor n/2 \rfloor) + 1 \le 2c(n/2) + 1$$

$$= cn + 1$$

αλλά εμείς θέλουμε σύμφωνα με την υπόθεση της επαγωγής να δείξουμε ότι $T(n) \le cn$. Για το λόγο αυτό δοκιμάζουμε για cn-b, όπου b σταθερά:

$$T(n) \le 2c(n/2) - 2b + 1 = cn + 1 - 2b$$

$$\le cn - b, \forall b \ge 1$$

Ανάλυση Αλγορίθμων XXV

… προσπαθώντας να «μαντέψουμε»

εάν τυχόν υπάρχουν αρχικές συνθήκες για την αναδρομή όπως π.χ. $T(1)=4$, τότε αυτές μπορεί να καθορίζουν τη σταθερά c

$$T(1) = 4 \le c \cdot 1 - b \overset{b \ge 1}{\Rightarrow} c \ge b + 4 \Rightarrow c \ge 5$$

ένα «κόλπο» για να «μαντέψουμε» είναι με χρήση «δένδρου αναδρομής» (βλ. Αλγόριθμοι – Σχεδιασμός και Ανάλυση, Παναγιώτης Δ. Μποζάνης).

Ανάλυση Αλγορίθμων XXVI

... με επαναληπτική αντικατάσταση

έστω ότι $T(n) = 2T(n/3) + n$

και T(1)=1. Δίχως βλάβη της γενικότητας ας υποθέσουμε ότι το n είναι δύναμη του 3. Τότε,

$$\begin{aligned} T(n) &= 2T(\frac{n}{3}) + n \\ &= 2\left(2T(\frac{n}{3^2}) + \frac{n}{3}\right) + n = 2^2 T\left(\frac{n}{3^2}\right) + \left(n + \frac{2n}{3}\right) \\ &= 2^2\left(2T(\frac{n}{3^3}) + \frac{n}{3^2}\right) + \left(n + \frac{2n}{3}\right) = 2^3 T\left(\frac{n}{3^3}\right) + \left(n + \frac{2n}{3} + \frac{2^2 n}{3^2}\right) \\ &= \text{.....} = \\ &= 2^i T(\frac{n}{3^i}) + \sum_{k=0}^{i-1} \frac{2^k n}{3^k} = 2^i T(\frac{n}{3^i}) + n\sum_{k=0}^{i-1}\left(\frac{2}{3}\right)^k \end{aligned}$$

Ανάλυση Αλγορίθμων XXVII

... με επαναληπτική αντικατάσταση

η επαναληπτική διαδικασία σταματάει όταν

$$\frac{n}{3^i} = 1 \Rightarrow i = \log_3 n$$

οπότε με αντικατάσταση και πράξεις

$$\begin{aligned} T(n) &= 2^{\log_3 n} T(1) + n\sum_{k=0}^{\log_3 n - 1}\left(\frac{2}{3}\right)^k \\ &= n^{\log_3 2} + n\frac{1-\left(\frac{2}{3}\right)^{\log_3 n}}{1-\frac{2}{3}} \\ &= n^{\log_3 2} + n\frac{3^{\log_3 n} - n^{\log_3 2}}{\frac{3^{\log_3 n}}{3}} = 3n - 2n^{\log_3 2} \le 2n - 2n^{0.63} = O(n) \end{aligned}$$

Ανάλυση Αλγορίθμων XXVIII

γενική μέθοδος

ουσιαστικά είναι μία μέθοδος για την επίλυση αναδρομικών εξισώσεων της μορφής

$$T(n) = aT(n/b) + f(n), \qquad a, b \geq 1, \quad f(n) > 0, \forall n \geq n_0$$

και βασίζεται στην εφαρμογή κάποιων θεωρημάτων.

Περισσότερες πληροφορίες στο:

Αλγόριθμοι – Σχεδιασμός και Ανάλυση, Παναγιώτης Δ. Μποζάνης, σελ. 19

Μορφές ανάλυσης

ανάλυση μέσης περίπτωσης	μέσο **κόστος** εκτέλεσης του αλγορίθμου
ανάλυση χειρότερης περίπτωσης	μέγιστο **κόστος** εκτέλεσης του αλγορίθμου

Ανάλυση χειρότερης περίπτωσης I

Παράδειγμα ανάλυσης χειρότερης περίπτωσης:

Algorithm quickSort(A,left,right)
Input: Πίνακας Α *n* αριθμών με όρια left και right
Output: Διατεταγμένος πίνακας

```
1.  i=left-1, j=right;
2.  o=a[right];
2.  while (true) {                          //εύρεση θέσης pivot
3.          while (a[++i]<o);
4.          while (o<a[--j]) min=j;
5.                  if (j==left) break;
6.          if (i>=j) break;
7.          temp=a[i]; a[i]=a[j]; a[j]=temp;
8.  }
9.  temp=a[i]; a[i]=a[right]; a[right]=temp;
10. if (left<i-1) quickSort(A, left, i-1);
11. if (i+1<right) quickSort(A, i+1, right);
```

Ανάλυση χειρότερης περίπτωσης II

Πως λειτουργεί το quicksort:

Αν π.χ. έχουμε ένα πίνακα αριθμών [4, 8, 2, 7, 1, 5, 6, 3], τότε επιλέγεται ως στοιχείο pivot αυτό που βρίσκεται στη δεξιότερη θέση και αναζητείται η σωστή του θέση μέσα στον πίνακα. Αυτό τοποθετείται στη σωστή του θέση και αριστερά του τοποθετούνται όλα τα στοιχεία που είναι μικρότερα από το pivot, ενώ δεξιά του αυτά που είναι μεγαλύτερα.

Έτσι, προκύπτει [2, 1, **3**, 4, 8, 7, 5, 6] και η quicksort εφαρμόζεται τόσο στο αριστερό όσο και στο δεξί τμήμα του πίνακα.

Εφαρμογή:

[4, 1, 2, 3]	left=0	right=3		
	i=-1	j=3	pivot=3	
	εύρεση θέσης pivot:			
	i←0	j←2	ανταλλαγή a[i] & a[j]	
[2, 1, 4, 3]	i←2	j←1	θέση pivot: 2	
[2, 1, 3, 4]	κ.λ.π.			

Χειρότερη περίπτωση: να αναζητούμε τη θέση του pivot σε έναν ήδη ταξινομημένο πίνακα

Ανάλυση χειρότερης περίπτωσης III

Σχέση κόστους:

$$T(n) = \max_i \{T(i-1) + T(n-i) + \underbrace{cn}_{\substack{\text{κόστος διάσπασης} \\ \text{σε δύο υποπροβλήματα}}}\}$$

Όταν όμως ο πίνακας είναι ήδη ταξινομημένος (χειρότερη περίπτωση), από τα δύο υποπροβλήματα η quicksort καλείται αναδρομικά μόνο για το ένα. Έτσι, το κόστος δίνεται από τη σχέση:

$$cn + c(n-1) + c(n-2) + \ldots + c = c \cdot \frac{n(n+1)}{2} = O(n^2)$$

Ανάλυση μέσης περίπτωσης I

Μέση περίπτωση:

δοθέντος ενός πίνακα *n* στοιχείων, όλα τα στοιχεία έχουν την ίδια πιθανότητα να είναι pivot (δηλαδή να βρίσκονται στη δεξιότερη θέση του πίνακα)

$$\overline{T}(n) = E[T(n)]$$

$$\overline{T}(n) = \sum_{i=1}^{n} \mathrm{Prob}[\text{το στοιχείο i είναι pivot}] \cdot (\overline{T}(i-1) + \overline{T}(n-i)) + n + 1$$

$$\overline{T}(n) = \frac{1}{n}\sum_{i=1}^{n} (\overline{T}(i-1) + \overline{T}(n-i)) + n + 1 = \frac{2}{n}\sum_{i=1}^{n} \overline{T}(i-1) + n + 1$$

$$n\overline{T}(n) = 2\sum_{i=1}^{n} \overline{T}(i-1) + n(n+1)$$

$$(n-1)\overline{T}(n) = 2\sum_{i=1}^{n} \overline{T}(i-1) + (n-1)\cdot n$$ αφαιρούμε κατά μέλη

Ανάλυση μέσης περίπτωσης II

$$n\overline{T}(n)-(n-1)\overline{T}(n-1)=2n+2\overline{T}(n-1)\Rightarrow$$
$$\frac{\overline{T}(n)}{n+1}=\frac{\overline{T}(n-1)}{n}+\frac{2}{n+1}$$

και με επαναληπτική αντικατάσταση

$$\frac{\overline{T}(n)}{n+1}=\frac{\overline{T}(1)}{2}+\sum_{j=3}^{n+1}\frac{2}{j}\Rightarrow$$
$$\overline{T}(n)\le 2(n+1)\sum_{j=1}^{n}\frac{1}{j}\Rightarrow O(n\log n)$$

Ανάλυση ταξινόμησης I

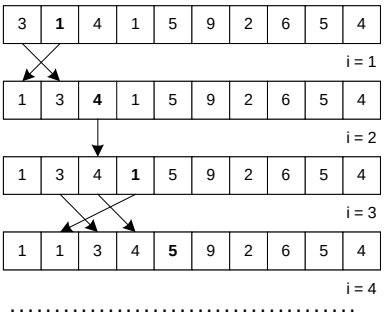
- Αλγόριθμος straight insertion sort (σταδιακή ταξινόμηση)

Algorithm straightInsertSort(A)
Input: Πίνακας A *n* αριθμών
Output: Διατεταγμένος πίνακας

```
for (i=1;i<n;i++) {
        x=T[i];
        j=i-1;
        while (j>=0 and x<T[j]) {
                T[j+1]=T[j];
                j=j-1;
        }
        T[j+1]=x;
}
```

Ανάλυση ταξινόμησης II

- Αλγόριθμος straight insertion sort (εκτέλεση . . .)



Ανάλυση ταξινόμησης III

Αλγόριθμος straight insertion sort (ανάλυση . . .)
πόσες συγκρίσεις (εντολή while) γίνονται για κάθε i?
καλύτερη περίπτωση:
η τιμή στη θέση i είναι μεγαλύτερη από την τιμή στη θέση i-1
(πίνακας ταξινομημένος σε αύξουσα)
μία σύγκριση για κάθε i και άρα συνολικά n-1 **συγκρίσεις**
χειρότερη περίπτωση:
(πίνακας ταξινομημένος σε φθίνουσα)
i-1 συγκρίσεις για κάθε i και άρα συνολικά $\sum_{i=2}^{n}(i-1) = n(n-1)/2$ **συγκρίσεις**

Ανάλυση ταξινόμησης IV

Αλγόριθμος straight insertion sort (ανάλυση . . .)

μέση περίπτωση:

αν θεωρήσουμε μόνο διαφορετικούς μεταξύ τους αριθμούς, τότε η μέση περίπτωση είναι ο μέσος αριθμός συγκρίσεων για όλες τις πιθανές σειρές, που μπορεί να προκύψουν από την {1,2,3,...,n} με αλλαγή της θέσης των στοιχείων της

ονομάζουμε αντιστροφή την εμφάνιση ζεύγους αριθμών που εμφανίζονται με λάθος σειρά π.χ. η σειρά {1,4,3,2} περιέχει τρεις αντιστροφές, τις (4,3,), (4,2) και (3,2)

Τι σχέση έχει η αντιστροφή με τη βασική πράξη, που είναι η σύγκριση??

Ανάλυση ταξινόμησης V

Αλγόριθμος straight insertion sort (ανάλυση . . .)

μέση περίπτωση (συνέχεια):

στον αλγόριθμο straight insertion sort και πιο συγκεκριμένα στο βρόχο while αφαιρείται κάθε φορά μία αντιστροφή με αντιμετάθεση των στοιχείων που την απαρτίζουν. Άρα έχουμε τόσες συγκρίσεις, όσες οι εμφανιζόμενες αντιστροφές.

Θεώρημα: Ο μέσος αριθμός εμφανιζόμενων αντιστροφών σε μία τυχαία επιλεγμένη σειρά n διακριτών αριθμών είναι $n(n-1)/4$

Απόδειξη: Έστω S μία τυχαία σειρά n διακριτών αριθμών και S^R η αντίστροφη σειρά της S. Άρα κάθε ζεύγος αριθμών σχηματίζει μία αντιστροφή που βρίσκεται είτε στην S είτε στην S^R.

Πόσα ζεύγη αριθμών περιέχει η κάθε σειρά? $\binom{n}{2} = n(n-1)/2$

Άρα κατά μέσο όρο η S θα περιέχει τις μισές αντιστροφές.

Ανάλυση ταξινόμησης VI

Συνοψίζοντας:

αλγόριθμος	καλύτερη περίπτωση	μέση περίπτωση	χειρότερη περίπτωση
straight insertion sort	n	n^2	n^2
binary search sort	n log(n)	n^2	n^2
bubble sort	n^2	n^2	n^2
quicksort	n log(n)	n log(n)	n^2
straight selection sort	n^2	n^2	n^2
heapsort		n log(n)	n log(n)
mergesort		n log(n)	n log(n)