ΑΠΟΔΟΤΙΚΟΤΗΤΑ ΑΛΓΟΡΙΘΜΩΝ

Αν ένα πρόβλημα λύνεται από δύο ή περισσότερους αλγόριθμους, ποιος θα είναι ο καλύτερος;

Με ποια κριτήρια θα τους συγκρίνουμε;

Πως θα υπολογίσουμε το χρόνο εκτέλεσης ενός αλγόριθμου;

ΑΠΟΔΟΤΙΚΟΤΗΤΑ ΑΛΓΟΡΙΘΜΩΝ

Για να απαντήσουμε στα προηγούμενα ερωτήματα πρέπει να εντοπίζουμε για κάθε αλγόριθμο:

- Την χειρότερη περίπτωση εκτέλεσης
- Το μέγεθος εισόδου του.
- Να υπολογίζουμε το χρόνο εκτέλεσης.

ΑΠΟΔΟΤΙΚΟΤΗΤΑ ΑΛΓΟΡΙΘΜΩΝ
Τρόποι σύγκρισης

Έχουμε δύο τρόπους για να συγκρίνουμε και τέλος να αποφασίσουμε για τον αποδοτικότερο αλγόριθμο:

- Εμπειρικός ή εκ των υστέρων
- Θεωρητικός ή εκ των προτέρων (υπολ/μός. Πολυπλοκότητας)

Τα κριτήρια σύγκρισης για τους παραπάνω τρόπους είναι με βάση:

- Το χρόνο εκτέλεσης
- Τη μνήμη που απαιτείται στην εκτέλεση.

ΑΠΟΔΟΤΙΚΟΤΗΤΑ ΑΛΓΟΡΙΘΜΩΝ
Τρόποι σύγκρισης

Ο χρόνος εκτέλεσης εξαρτάτε:

- Από τον τύπο του υπολογιστή που εκτελεί το αντίστοιχο πρόγραμμα.
- Από τη γλώσσα προγραμματισμού.
- Τη δομή του προγράμματος και τις δομές δεδομένων.
- Το χρόνο προσπέλασης στον σκλ. Δίσκο και τις ενέργειες εισόδου-εξόδου.
- Το λειτουργικό σύστημα (ενός χρήστη ή πολλών χρηστών).

ΑΠΟΔΟΤΙΚΟΤΗΤΑ ΑΛΓΟΡΙΘΜΩΝ
Τρόποι σύγκρισης

Η μνήμη που απαιτείται εξαρτάται:

- Από τη γλώσσα προγραμματισμού και τους τύπους δεδομένων που υποστηρίζει.

- Τις δομές δεδομένων, τις μεταβλητές και τις σταθερές που χρησιμοποιούμε.

ΑΠΟΔΟΤΙΚΟΤΗΤΑ ΑΛΓΟΡΙΘΜΩΝ
Τρόποι σύγκρισης

Στον εμπειρικό τρόπο οι συνθήκες εκτέλεσης αλγόριθμων από αντίστοιχα προγράμματα πρέπει να είναι ίδιες:

- Ίδια γλώσσα προγραμματισμού.
- Ο ίδιος μεταφραστής(της γλώσ. προγ/μού)
- Η ίδια υπολογιστική πλατφόρμα
- Ακριβώς τα ίδια δεδομένα εισόδου.

ΑΠΟΔΟΤΙΚΟΤΗΤΑ ΑΛΓΟΡΙΘΜΩΝ
Χειρότερη περίπτωση

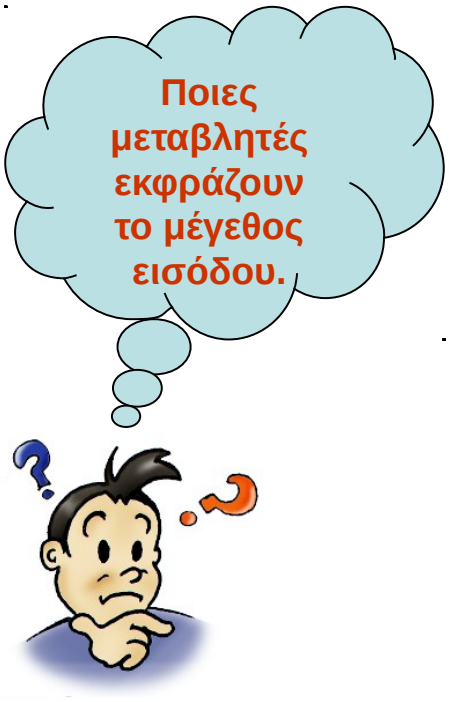
Είναι η περίπτωση όπου με είσοδο συγκεκριμένων τιμών έχουμε το μέγιστο κόστος σε υπολογιστικούς πόρους. Για τον υπολογισμό τους κόστους μετράμε **βασικές πράξεις** όπως:

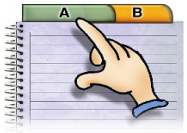
- Ανάθεσης τιμής.
- Σύγκριση μεταξύ δύο μεταβλητών.
- Οποιαδήποτε αριθμητική πράξη μεταξύ δύο μεταβλητών.


```
Αλγόριθμος Παράδειγμα_1
n ←10
Αρχή_Επανάληψης
    Διάβασε m
        n←n – 1
Μέχρις_ότου (m=0) ή (n=0)
Εκτύπωσε m
Τέλος Παράδειγμα_1
```

ΑΠΟΔΟΤΙΚΟΤΗΤΑ ΑΛΓΟΡΙΘΜΩΝ
Μέγεθος εισόδου

Εκφράζεται από τις μεταβλητές που απεικονίζουν τους συνδυασμούς τιμών εισόδου(τα δεδομένα) που κρίνουν την συμπεριφορά εκτέλεσης. Γενικά τα δεδομένα εισόδου συνιστούν το μέγεθος εισόδου ενός αλγόριθμου.






```
Αλγόριθμος Παράδειγμα_1
n ←10
Αρχή_Επανάληψης
    Διάβασε m
        n←n – 1
Μέχρις_ότου (m=0) ή (n=0)
Εκτύπωσε m
Τέλος Παράδειγμα_1
```

ΑΠΟΔΟΤΙΚΟΤΗΤΑ ΑΛΓΟΡΙΘΜΩΝ
Μέγεθος εισόδου

Αλγόριθμος	Μέγεθος εισόδου	Βασική πράξη
ΤΑΞΙΝΟΜΗΣΗ	Πλήθος στοιχείων προς ταξινόμηση	σύγκριση
ΠΟΛΛΑΠΛΑΣΙΑΣΜΟΣ	Το πλήθος των ψηφίων που θα πολλαπλασιαστούν	Αριθμητικές πράξεις
ΑΝΑΖΗΤΗΣΗ	Πλήθος στοιχείων πίνακα	σύγκριση

ΑΠΟΔΟΤΙΚΟΤΗΤΑ ΑΛΓΟΡΙΘΜΩΝ
Χρόνος εκτέλεσης αντίστοιχου προγράμματος


```
Αλγόριθμος Παράδειγμα_2
x ←123
y ←234
Για  i από 0 μέχρι 4
    Εκτύπωσε i
        z←x * y
Τέλος_επανάληψης
Εκτύπωσε x
Εκτύπωσε y
Εκτύπωσε z
Τέλος Παράδειγμα_2
```

ΒΑΣΙΚΕΣ ΠΡΑΞΕΙΣ	Αριθμός πράξεων (1πράξη/1 μsec)			
Ανάθεσης τιμής στα x,y	2	2	2	2
Βρόχος n επαναλήψεων	n=5	n=10	n=100	n=1000000
Ανάθεση αρχικής τιμής στο i	1	1	1	1
έλεγχος i	6	11	101	1000001
Αύξηση i	5	10	100	1000000
Εκύπωση i	5	10	100	1000000
z←x * y	(2x5)10	20	200	2000000
Εκτύπωση x,y,z	3	3	3	3
ΣΥΝΟΛΟ σε μsec	32	57	507	5000007 Περίπου 5sec

ΠΟΛΥΠΛΟΚΟΤΗΤΑ ΑΛΓΟΡΙΘΜΩΝ

Για τον υπολογισμό της αποδοτικότητας με τον θεωρητικό τρόπο χρησιμοποιούμε μία συνάρτηση F(n) με την οποία υπολογίζουμε την χρονική και χωρητική πολυπλοκότητα.

Το n είναι η μεταβλητή που εκφράζει το μέγεθος του προβλήματος(το μέγεθος εισόδου του αντίστοιχου αλγόριθμου). πχ. Αν το πρόβλημα είναι η ταξινόμηση το μέγεθος n = πλήθος των στοιχείων που πρόκειται να ταξινομηθούν.

Επειδή μας ενδιαφέρει η γενική συμπεριφορά του αλγόριθμου Υπάρχει ένας συμβολισμός τάξης **O**(order) στην οποία ανήκει η F(n) (***F(n) Є O(g(n))***)

Αν η πολυπλοκότητα ενός αλγόριθμου είναι f(n), τότε λέγεται ότι είναι τάξης O(g(n)), αν υπάρχουν δύο θετικοί ακέραιοι c, n_0 έτσι ώστε για κάθε $n \geqslant n_0$ *να ισχύει* $|f(n)| \leq c\,|g(n)|$

$f(n)=2n^3+5n^2-4n+3$ *Τότε* $g(n)=n^3$ *και πολυπλοκότητα τάξης* $O(n^3)$

$f(n)=5\cdot 2^n+4n^2-4\log(n)$ *πολυπλοκότητα τάξης* $O(2^n)$

Ποιος είναι ο σημαντικότερος όρος της δεύτερης συνάρτηση f(n) για n=1,2,3,4,5

$f(1)=5\cdot 2^{(1)}+4\cdot 1^2-4\log(1)=5\cdot 2+4\cdot 1-4\cdot 0$

$f(2)=5\cdot 2^{(2)}+4\cdot 2^2-4\log(2)=5\cdot 4+4\cdot 4-4\cdot 0{,}30$

$f(3)=5\cdot 2^{(3)}+4\cdot 3^2-4\log(3)=5\cdot 8+4\cdot 9-4\cdot 0{,}48$

$f(4)=5\cdot 2^{(4)}+4\cdot 4^2-4\log(4)=5\cdot 16+4\cdot 16-4\cdot 0{,}60$

$f(5)=5\cdot 2^{(5)}+4\cdot 5^2-4\log(5)=5\cdot 32+4\cdot 25-4\cdot 0{,}67$

ΠΟΛΥΠΛΟΚΟΤΗΤΑ ΑΛΓΟΡΙΘΜΩΝ
Συνήθεις κατηγορίες χρονικής πολυπλοκότητας

- O(1) Σταθερή πολυπλοκότητα: Κάθε εντολή του προγράμματος εκτελείται μία ή λίγες μόνο φορές.
- O(n) Γραμμική πολυπλοκότητα: Είναι η καλύτερη για αλγόριθμο που πρέπει να εξετάσει ή να δώσει έξοδο n στοιχεία.
- O(log n) Λογαριθμική: Με log θα συμβολίζουμε τον δυαδικό λογάριθμο, με ln τον φυσικό λογάριθμο.
- O(n log(n)) Σ` αυτή ανήκουν μια σπουδαία οικογένεια αλγόριθμων ταξινόμησης.
- $O(n^2)$:Τετραγωνική: Κατάλληλη για προβλήματα μικρού μεγέθους.
- $O(n^3)$ Κυβική: Κατάλληλη για προβλήματα μικρού μεγέθους.
- $O(2^n)$ Εκθετική: Χρησιμοποιείται πολύ σπάνια

Πολυπλοκότητα	Μέγεθος προβλήματος (Μία βασική πράξη / μικροδευτερόλεπτο)		
	n=20	n=40	n=60
O(n)	0,00002 sec	0,00004 sec	0,00006 sec
$O(n^2)$	0,0004 sec	0,0016 sec	0,0036 sec
$O(n^3)$	0,008 sec	0,064 sec	0,216 sec
$O(2^n)$	1 sec	12,7 ημέρες	366 αιώνες
O(n!)	77146 έτη	$3X10^{32}$ αιώνες	$3X10^{66}$ αιώνες

ΠΟΛΥΠΛΟΚΟΤΗΤΑ ΑΛΓΟΡΙΘΜΩΝ
Ταξινόμηση ευθείας ανταλλαγής

Για τον υπολογισμό της πολυπλοκότητας χρόνου και χώρου στην ταξινόμηση πρέπει να αθροίσουμε τις βασικές πράξεις της σύγκρισης και της αντιμετάθεσης.

```
Αλγόριθμος Φυσσαλίδα
Δεδομένα  // table, n //
Για i από 2 μέχρι n
   Για j από n μέχρι i με_βήμα -1
      Αν table[ j -1] > table[ j ] τότε
        αντιμετάθεσε table[ j -1 ], table[ j ]
      Τέλος_αν
   Τέλος_επανάληψης
Τέλος_επανάληψης
Αποτέλεσμα // table //
Τέλος Φυσσαλίδα
```

ΧΕΙΡΟΤΕΡΗ ΠΕΡΙΠΤΩΣΗ

Μέγεθος Προβλήματος n=4	j=4	j=3	j=2	table[j -1] > table[j]	αντιμετάθεσε
	200	200	200	3 ή n-1	3 ή n-1
i=2	100	100	10		
	50	10	100		
	10	50	50		
	10	10		2 ή n-2	2 ή n-2
	200	200			
i=3	100	50			
	50	100			
	10			1	1
	50				
	200				
i=4	100				

Με γενίκευση του αθροίσματος των συγκρίσεων έχουμε αριθμητική πρόοδο:

$$\sum_{j=2}^{n} \text{συγκρίσεων} = (n-1)+(n-2)+(n-3)+\ldots+2+1 = (n-1+1)(n-1)\frac{1}{2} = \frac{n(n-1)}{2} = \frac{1}{2}n^2 - \frac{1}{2}n$$

$$f(n) = 2\times\left(\frac{n^2}{2} - \frac{n}{2}\right) = n^2 - n, \qquad O(n^2)$$

ΠΟΛΥΠΛΟΚΟΤΗΤΑ ΑΛΓΟΡΙΘΜΩΝ
Σειριακή αναζήτηση σε ταξινόμητο πίνακα

Για τον υπολογισμό της πολυπλοκότητας χρόνου στη σειριακή αναζήτηση αρκεί να μετρήσουμε τις συγκρίσεις κλειδιού με τα στοιχεία του πίνακα (**table[i] = key**).

```
Αλγόριθμος Sequential_search
Δεδομένα //n, table, key//
        done ← ψευδής
        position ← 0
        i ← 1
Όσο (done=ψευδής) και (i≤n) Επανάλαβε
        Αν table[i] = key τότε
                done ← αληθής
                position ← I
        Τέλος_αν
      i ← i +1
Τέλος_επανάληψης
Αποτελέσματα //done, position//
Τέλος Sequential_search
```

Μέγεθος Προβλήματος n=5	key=100	key=0	key=50	key=10	Key=-20
	100	100	100	100	100
	0	0	0	0	0
Πλήθος συγκρίσεων	50	50	50	50	50
Χειρότερη περίπτωση	10	10	10	10	10
	-20	-20	-20	-20	-20
Συγκρίσεις table[i] = key	1	2	3	4	5
Γενίκευση	1	2	n-2	n-1	n

Στη χειρότερη περίπτωση θα έχουμε **n** συγκρίσεις.

$$f(n)=n \Rightarrow O(n)$$

Η πολυπλοκότητα με επιτυχή αναζήτηση είναι ο μέσος όρος των συγκρίσεων κάθε κλειδιού(key:100, 0, 50, ...).

$$f(n)=\frac{\sum_{i=1}^{n} \text{συγκρίσεων}}{n} = \frac{1+2+...+(n-1)+n}{n} = \frac{n(n+1)}{2n}$$

$$f(n) \Rightarrow O(n)$$

ΠΟΛΥΠΛΟΚΟΤΗΤΑ ΑΛΓΟΡΙΘΜΩΝ
Δυαδική αναζήτηση

Στη δυαδική αναζήτηση στη αρχή συγκρίνουμε **key=μεσαίο στοιχείο.** Αν δεν είναι αυτό που αναζητούμε διαιρούμε το n(πλήθος στοιχείων πίνακα) δια δύο και περιορίζουμε την αναζήτησή μας στο **n/2**:
Επαναλαμβάνουμε τις συγκρίσεις και τις υποδιαιρέσεις στο n:

$$\frac{n}{2}, \frac{n}{4}, \frac{n}{8} \dots$$

ή

$$\frac{n}{2^1}, \frac{n}{2^2}, \frac{n}{2^3}, \dots \frac{n}{2^k}$$

Στη χειρότερη περίπτωση αλγόριθμου οι επαναλήψεις συγκρίσεων θα γίνονται

μέχρι να γίνει : $\frac{n}{2^k} \leqslant 1$

ή

$$2^k \geqslant n \Rightarrow k \geqslant \log(n)$$

Χειρότερη Περίπτωση			
Μέγεθος Προβλήματος n=5	n	n/2	n/4
key=58	0	0	0
Συγκρίσεις	10	10	10
Τμήμα που απορρίπτουμε	20	20	20
	24	24	24
	50	50	50
	55	55	55
	56	56	56

$$f(n) = \log(n) \Rightarrow O(\log(n))$$