ΑΝΑΛΥΣΗ ΑΛΓΟΡΙΘΜΩΝ

- **Η εύρεση των πόρων (resources) που ένας αλγόριθμος απαιτεί να τρέξει**
  - **χρόνος** (time) – πολυπλοκότητα χρόνου (time complexity)
  - **χώρος** (space) – πολυπλοκότητα χώρου (space complexity)

- **Μοντέλα υπολογισμού**
  - Κατάταξη αλγορίθμων σε κλάσεις πολυπλοκότητας (complexity classes) – Πιο θεωρητικό!
    - ***Μηχανή Turing***
  - Ανάλυση πολυπλοκότητας αλγορίθμων
    - ***Μοντέλο μηχανής τυχαίας προσπέλασης*** *(RAM)*
    - *Αυτό θα χρησιμοποιήσουμε*

ΜΟΝΤΕΛΟ RAM

ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ
ΤΜΗΜΑ ΣΤΑΤΙΣΤΙΚΗΣ

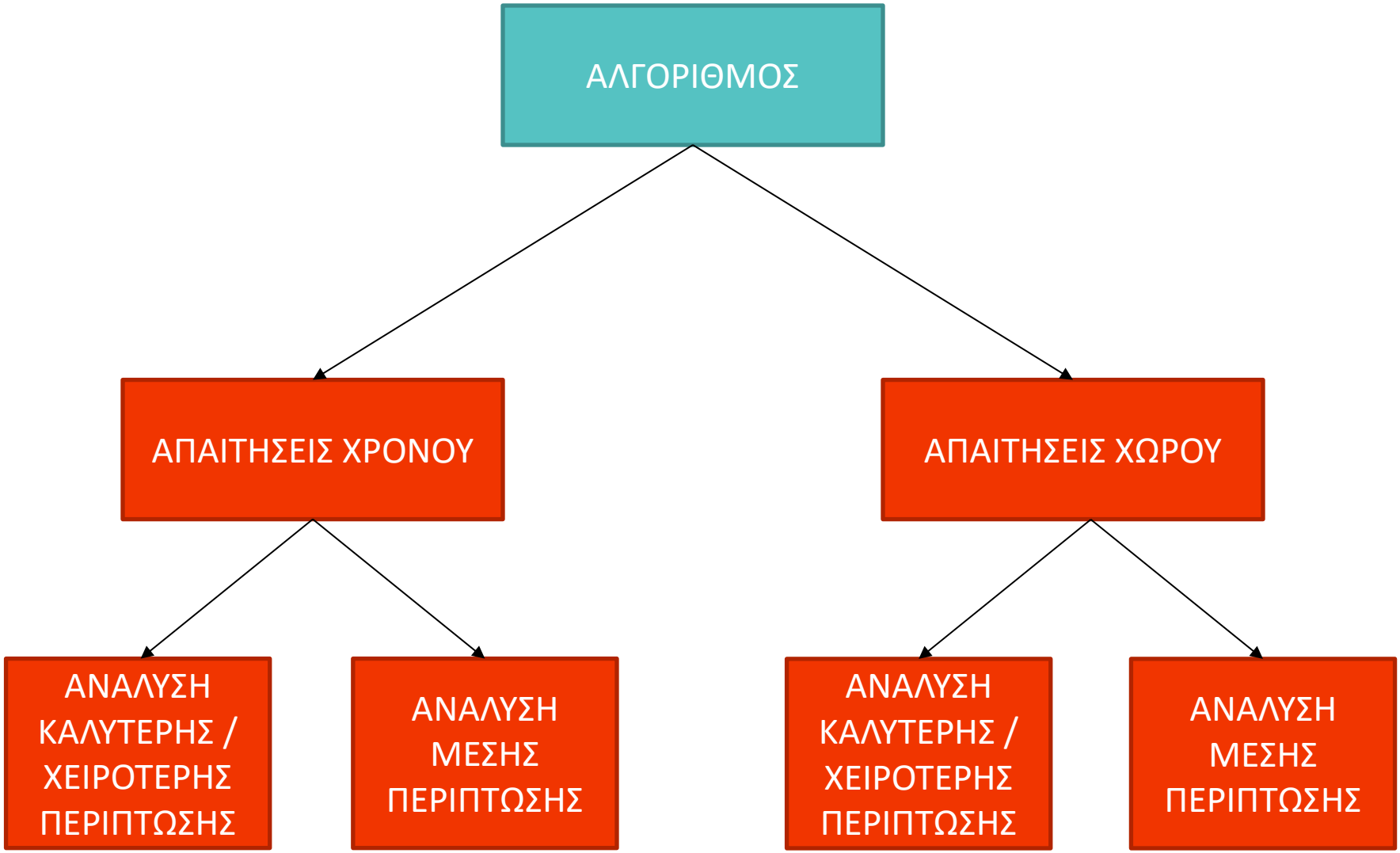
- **1 επεξεργαστής χωρίς παραλληλισμό**
- **Εκτέλεσης αριθμητικών πράξεων, λήψης αποφάσεων διακλάδωσης και εγγραφής / διαβάσματος μνήμης**
- **Ανάλυση συναρτήσει του μεγέθους της εισόδου!**

ΠΟΛΥΠΛΟΚΟΤΗΤΑ (1)

- **Υπάρχουν τρεις διαφορετικές προσεγγίσεις για την ανάλυση της υπολογιστικής συμπεριφοράς των αλγορίθμων. Οι προσεγγίσεις αυτές είναι**
  - **Ανάλυση καλύτερης – χειρότερης περίπτωσης** (worst – best case analysis)
  - **Ανάλυση μέσης περίπτωσης** (average case analysis)
  - **Πειραματική ανάλυση** (experimental analysis)

Theoretical results can not tell the full story about real-world algorithmic performance

ΠΟΛΥΠΛΟΚΟΤΗΤΑ (2)



ΠΟΛΥΠΛΟΚΟΤΗΤΑ (3)

- **Η χρονική πολυπλοκότητα είναι συνάρτηση η οποία καθορίζει πώς ο χρόνος εκτέλεσης εξαρτάται από το μέγεθος της εισόδου**

- **Μια συνάρτηση που αντιστοιχίζει το "μέγεθος" της εισόδου με το "χρόνο" T(n) εκτέλεσης**

ΑΝΑΛΥΣΗ ΑΛΓΟΡΙΘΜΩΝ

- **Επιθυμίες στην μελέτη αλγορίθμων**
  - Κατάταξη ως **αποτελεσματικών** ή όχι
  - Επιλογή **ταχύτερου** αλγόριθμου (σύγκριση αλγορίθμων)
  - Προσδιορισμός άνω ορίου στο χρόνο επίλυσης (**ανάλυση χειρότερης περίπτωσης**)
  - Ανάπτυξη θεωρίας διαχρονικής που ισχύει για όλους τους Η/Υ

ΑΣΥΜΠΤΩΤΙΚΗ ΑΝΑΛΥΣΗ

- **Άνω όριο – Ο**
  - Ο μέγιστος χρόνος που θα εκτελεστεί ο αλγόριθμος
- **Κάτω όριο – Ω**
  - Ο ελάχιστος χρόνος που θα εκτελεστεί ο αλγόριθμος

ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ
ΤΜΗΜΑ ΣΤΑΤΙΣΤΙΚΗΣ

- **Εστίαση στα μεγάλα προβλήματα. Τα μικρά λύνονται γρήγορα. Δηλαδή n →∞**

- **Γι' αυτό ασυμπτωτική ανάλυση**

- **Επικεντρώνουμε μόνο στις μεγαλύτερες τιμές!**
  - **Αν έχουμε n^2 δεν μας νοιάζει το n**
    - ***$T(n) = 2n^2 + n$, τότε είναι $O(n^2)$***
  - **Δεν επικεντρώνουμε σε σταθερές**
    - ***$T(n) = n-5$, τότε είναι $O(n)$***

ΑΣΥΜΠΤΩΤΙΚΗ ΑΝΑΛΥΣΗ

- **Ορισμοί ασυμπτωτικών ορίων**
  - **Ο (συμβολισμός μεγάλου όμικρον): άνω όριο / το πολύ**
  - **Ω (συμβολισμός μεγάλου ωμέγα): κάτω όριο / τουλάχιστον**
  - **Θ (συμβολισμός θήτα): ίδια τάξη / ακριβώς**
  - **ο (συμβολισμός μικρού όμικρον): αυστηρό άνω όριο**
  - **ω (συμβολισμός μικρού ωμέγα): αυστηρό κάτω όριο**

ΠΑΡΑΔΕΙΓΜΑ

Παραδείγματα

Εάν $T(n) = 3n^2$ τότε $T(n)$ είναι $O(n^2)$

Εάν $T(n) = 13n^2 + 1000$ τότε η $T(n)$ είναι $O(n^2)$

Εάν $T(n) = 100n^3 + 3n^2$ τότε η $T(n)$ είναι $O(n^3)$

Άσκηση

Να βρεθεί η τάξη μεγέθους των συναρτήσεων:

α) $T(n) = 5n\log n + 0.5n^2 + 10$

β) $T(n) = \dfrac{3n + 5\log n + 1}{2n}$

Καλύτερη, και χειρότερη περίπτωση?

ΠΑΡΑΔΕΙΓΜΑ

Αλγόριθμος για εύρεση μέγιστου αριθμού σε πίνακα

1. Είσοδος: πίνακας Α με n στοιχεία
2. max <- A[1]
3. Για i από 2 μέχρι n
    1. Αν A[i] > max τότε
        1. max <- A[i]
    2. Τέλος_αν
4. Τέλος_Για
5. Επιστροφή max // μέγιστη τιμή και θέση της

Καλύτερη, και χειρότερη περίπτωση?

ΠΑΡΑΔΕΙΓΜΑ

Αλγόριθμος για αναζήτηση στοιχείου σε πίνακα

1. Είσοδος: πίνακας Α με n στοιχεία, ζητούμενο x
2. Για i από 1 μέχρι n
    1. Αν A[i] = x τότε
        1. Επιστροφή i // θέση που βρέθηκε
    2. Τέλος_αν
3. Τέλος_Για
4. Επιστροφή -1 // δεν βρέθηκε

Καλύτερη, και χειρότερη περίπτωση?

ΠΑΡΑΔΕΙΓΜΑ

Αλγόριθμος για ταξινόμηση πίνακα

1. Είσοδος: πίνακας A με n στοιχεία
2. Για i από 2 μέχρι n:
    1. key ← A[i]
    2. j ← i – 1
    3. Όσο j > 0 και A[j] > key:
        1. A[j+1] ← A[j]
        2. j ← j – 1
    4. Τέλος_Όσο
    5. A[j+1] ← key
3. Τέλος_Για

Καλύτερη, και χειρότερη περίπτωση?

ΠΑΡΑΔΕΙΓΜΑ

Αλγόριθμος δυαδικής αναζήτησης

1. Είσοδος: Ταξινομημένος πίνακας A[1...n], ζητούμενο x
2. left ← 1, right ← n
3. Όσο left ≤ right:
    1. mid ← (left + right) // 2
    2. Αν A[mid] = x τότε
        1. Επιστροφή mid // η θέση του x
    3. Αλλιώς αν A[mid] < x τότε
        1. left ← mid + 1
    4. Αλλιώς
        1. right ← mid - 1
4. Επιστροφή -1 // δεν βρέθηκε

ΜΟΝΤΕΛΑ ΚΟΣΤΟΛΟΓΗΣΗΣ (1)

- **1 – μοναδιαίο κόστος: σταθερός χρόνος εκτέλεσης**
  - **Οι περισσότερες εντολές**

- **$logn$ – λογαριθμικό κόστος: η εκτέλεση του προγράμματος καθυστερεί ελαφρώς καθώς αυξάνεται το n**
  - **Προγράμματα που λύνουν κάποιο μεγάλο πρόβλημα μετασχηματίζοντάς το σε μια σειρά από μικρότερα, μειώνοντας σε κάθε βήμα το μέγεθος του προβλήματος**

ΜΟΝΤΕΛΑ ΚΟΣΤΟΛΟΓΗΣΗΣ (2)

- **n – γραμμικό κόστος: διπλασιάζεται η είσοδος, διπλασιάζεται και ο χρόνος εκτέλεσης**
  - **Ένας βρόγχος επανάληψης**

- **$nlogn$ – γραμμολογαριθμικό κόστος: διπλασιάζεται η είσοδος, υπερδιπλασιάζεται ο χρόνος εκτέλεσης**
  - **Προγράμματα που λύνουν κάποιο μεγάλο πρόβλημα μετασχηματίζοντάς το σε μια σειρά από μικρότερα, λύνοντάς τα ανεξάρτητα και συνδυάζοντας κατόπιν τις λύσεις**

ΜΟΝΤΕΛΑ ΚΟΣΤΟΛΟΓΗΣΗΣ (3)

- n^2 – **τετραγωνικό κόστος: διπλασιάζεται η είσοδος, τετραπλασιάζεται ο χρόνος εκτέλεσης**
  - Διπλός βρόγχος επανάληψης
  - Χρήσιμος μόνο για μικρά προβλήματα

- n^3 – **κυβικό κόστος: διπλασιάζεται η είσοδος, οκταπλασιάζεται ο χρόνος εκτέλεσης**
  - Τριπλός βρόγχος επανάληψης
  - Χρήσιμος μόνο σε περιπτώσεις μικρών προβλημάτων

ΜΟΝΤΕΛΑ ΚΟΣΤΟΛΟΓΗΣΗΣ (3)

- 2^n – **εκθετικό** κόστος: όταν το n είναι 20, ο χρόνος εκτέλεσης είναι 1,000,000
  - Υλοποιήσεις ωμής βίας

- $n!$ – **παραγοντικό** κόστος: όταν το n είναι 20, ο χρόνος εκτέλεσης είναι 2.432902e+18
  - Υλοποιήσεις ωμής βίας (εύρεση όλων των μεταθέσεων των υποσυνόλων)

ΠΟΛΥΠΛΟΚΟΤΗΤΑ

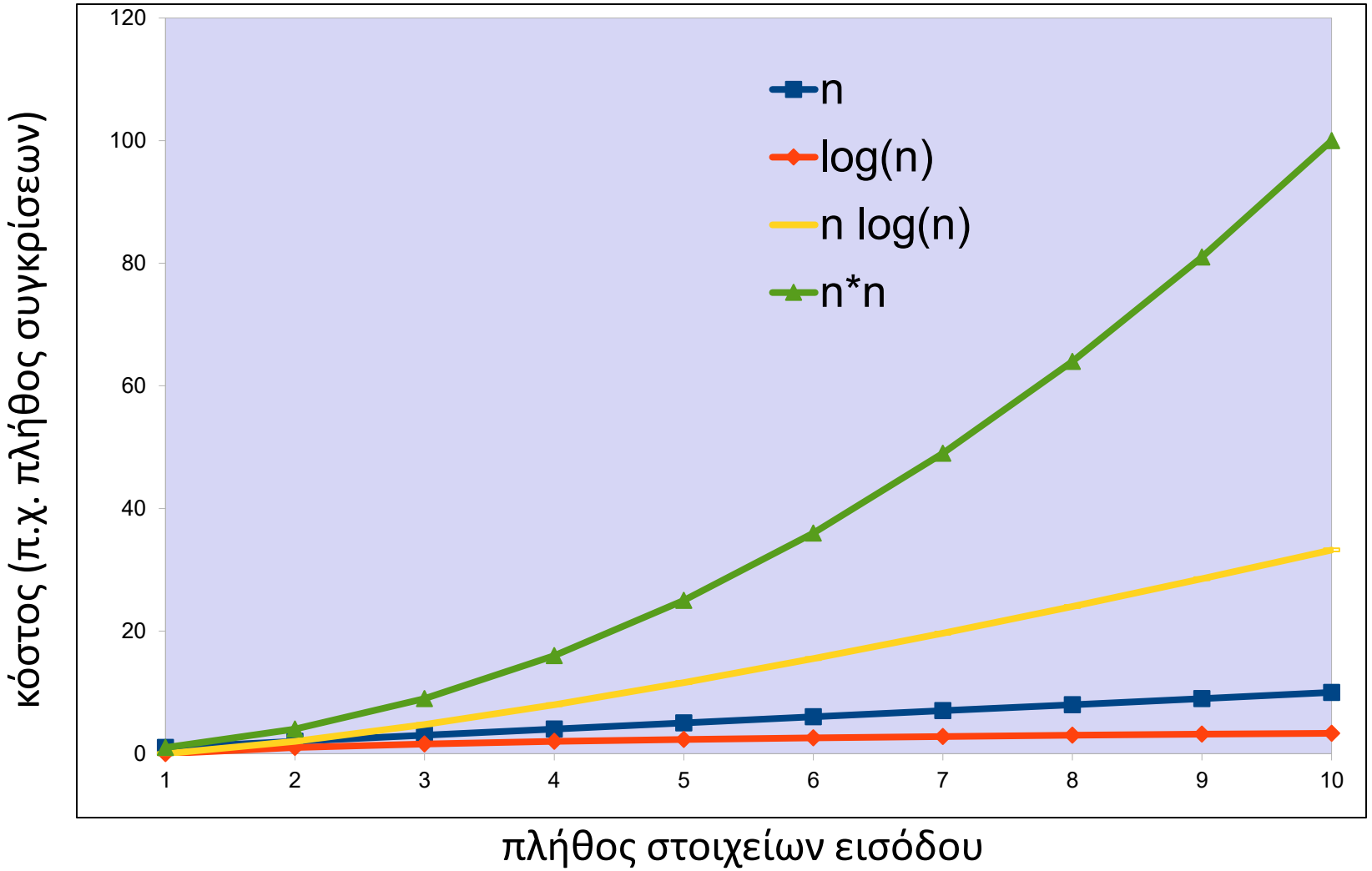
Πολυπλοκότητα χρόνου (time complexity)

n	$\log_2 n$	n	$n\log_2 n$	n^2	n^3	2^n	n!
10	3.3	10	3.3x10	10^2	10^3	10^3	$3.6x10^6$
10^2	6.6	10^2	$6.6x10^2$	10^4	10^6	$1.3x10^{30}$	$9.3x10^{157}$
10^3	10	10^3	$1x10^4$	10^6	10^9		
10^4	13	10^4	$1.3x10^5$	10^8	10^{12}		
10^5	17	10^5	$1.7x10^6$	10^{10}	10^{15}		
10^6	20	10^6	$2x10^7$	10^{12}	10^{18}		

ΠΟΛΥΠΛΟΚΟΤΗΤΑ

Εντολές ανά δευτερόλεπτο	Μέγεθος προβλήματος 1 εκατομμύριο			Μέγεθος προβλήματος 1 δισεκατομμύριο		
	n	$nlgn$	n^2	n	$nlgn$	n^2
10^6	δευτερόλεπτα	δευτερόλεπτα	εβδομάδες	ώρες	ώρες	ποτέ
10^9	στιγμιαία	στιγμιαία	ώρες	δευτερόλεπτα	δευτερόλεπτα	δεκαετίες
10^{12}	στιγμιαία	στιγμιαία	Δευτερόλεπτα	στιγμιαία	στιγμιαία	εβδομάδες

ΠΟΛΥΠΛΟΚΟΤΗΤΑ



ΠΟΛΥΠΛΟΚΟΤΗΤΑ

Πολυπλοκότητα χρόνου (time complexity) γνωστών αλγορίθμων ταξινόμησης

Αλγόριθμος	Καλύτερη περίπτωση	Αναμενόμενη περίπτωση	Χειρότερη περίπτωση	Χώρος	Ευστάθεια
Bubble sort	$O(n^2)$	$O(n^2)$	$O(n^2)$	Σταθερός	Ναι
Modified bubble sort	$O(n^2)$	O(n)	$O(n^2)$	Σταθερός	Ναι
Insertion sort	$O(n^2)$	$O(n^2)$	$O(n^2)$	Σταθερός	Ναι
Selection sort	$O(n^2)$	$O(n^2)$	$O(n^2)$	Σταθερός	Ναι
Heap sort	$O(nlogn)$	$O(nlogn)$	$O(nlogn)$	Σταθερός	Όχι
Merge sort	$O(nlogn)$	$O(nlogn)$	$O(nlogn)$	Μη σταθερός	Ναι
Quick sort	$O(nlogn)$	$O(nlogn)$	$O(n^2)$	Σταθερός	Ναι