ΑΛΓΟΡΙΘΜΟΙ & ΣΤΟΙΧΕΙΑ ΠΟΛΥΠΛΟΚΟΤΗΤΑΣ

Ηλίας Κ. Σάββας
Επίκουρος Καθηγητής
Τμήμα: Τεχνολογίας Πληροφορικής & Τηλεπικοινωνιών
Email: savvas@teilar.gr

Περίγραμμα

- Εισαγωγή
- Στοιχεία Πολυπλοκότητας
- Αλγόριθμοι
  - (Brute-Force, Αναδρομικοί, Διαίρει και Βασίλευε, Δυναμικός Προγ/μός, Άπληστοι, Ευρεστικοί,…)
- Κλάσεις Πολυπλοκότητας

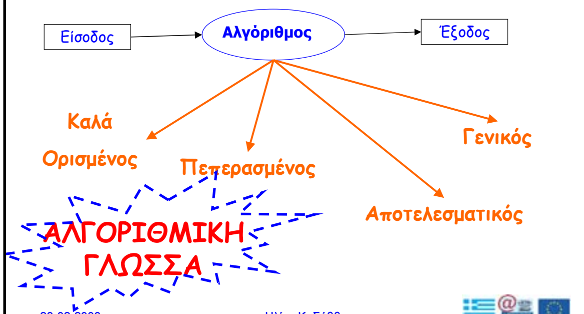
Περίγραμμα

- **Εισαγωγή**
- Στοιχεία Πολυπλοκότητας
- Αλγόριθμοι
  - (Brute-Force, Αναδρομικοί, Διαίρει και Βασίλευε, Δυναμικός Προγ/μός, Άπληστοι, Ευρεστικοί,…)
- Κλάσεις Πολυπλοκότητας

Γενικά περί αλγορίθμων

- **Αλγόριθμος** ← Περσία 825 μΧ: Abu Ja'far Mohammed ibn Musa al Khowarizmi
- Σύνολο (απλών) οδηγιών οι οποίες επιλύουν ένα συγκεκριμένο πρόβλημα
  - *3x+5=0*
- ή μία **κλάση προβλημάτων**.
  - ***ax+b=0***

Αλγόριθμος



Τεχνικές αλγορίθμων

- Brute-Force,
- Αναδρομικοί,
- «Διαίρει και Βασίλευε»,
- Δυναμικού Προγραμματισμού,
- Άπληστοι,
- Παράλληλοι,
- Ευρεστικοί (Εύρηκα!!!)
- Και πολλές άλλες τεχνικές…

Περίγραμμα

- *Εισαγωγή*
- **<u>Στοιχεία Πολυπλοκότητας</u>**
- Αλγόριθμοι
  - (Brute-Force, Αναδρομικοί, Διαίρει και Βασίλευε, Δυναμικός Προγ/μός, Άπληστοι, Ευρεστικοί,…)
- Κλάσεις Πολυπλοκότητας

Πολυπλοκότητα

- Αξιολόγηση αλγόριθμου:
  - **<u>Ταχύτητα,</u>**
  - Χρήση μνήμης,
  - Χρήση πόρων δικτύου,
  - Χρήση οποιονδήποτε υπολογιστικών πόρων.
- ΤΑΧΥΤΗΤΑ αλγόριθμου (σε συνάρτηση με το πλήθος δεδομένων εισόδου, έστω N):
  - **<u>Χειρίστη περίπτωση,</u>**
  - Μέση περίπτωση,
  - Καλύτερη περίπτωση.

Μέτρηση πολυπλοκότητας f(N)

- Μεγάλο Όμικρον (big Oh):
  - f(N)=O(g(N))➔ δεν αυξάνει πιο γρήγορα από την g
- Μικρό όμικρον (little oh):
  - f(N)=o(g(N))➔ αυξάνει πιο αργά από την g
- Θήτα (theta):
  - f(N)=θ(g(N))➔ f και g αυξάνουν με τον ίδιο (περίπου) ρυθμό
- Ασυμπτωτικά:
  - f(N)~g(N)➔ f και g αυξάνουν με τον ίδιο ρυθμό
- Ωμέγα (omega):
  - f(N)=Ω(g(N))➔ Άρνηση του μικρού όμικρον, η g αποτελεί κάτω όριο της f.

Προηγμένες Υπηρεσίες Ηλεκτρονικής Μάθησης στο ΤΕΙ Λάρισας

Περίγραμμα

- *Εισαγωγή*
- *Στοιχεία Πολυπλοκότητας*
- **Αλγόριθμοι**
  - **(Brute-Force, Αναδρομικοί, Διαίρει και Βασίλευε, Δυναμικός Προγ/μός, Άπληστοι, Ευρεστικοί,...)**
- Κλάσεις Πολυπλοκότητας

Προηγμένες Υπηρεσίες Ηλεκτρονικής Μάθησης στο ΤΕΙ Λάρισας

Τεχνική (?): **Brute-Force…**

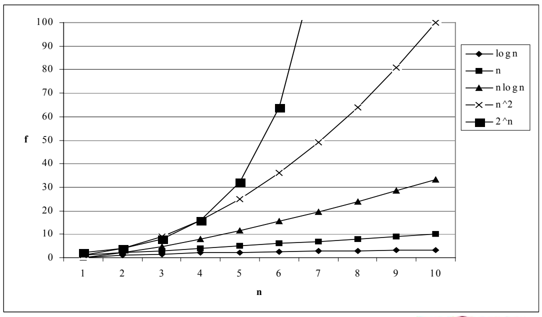
- Επίλυση με τον πιο ΑΠΛΟ και ΠΡΟΦΑΝΗ τρόπο (χωρίς να σημαίνει ότι είναι και ο πιο καλός, που σχεδόν σε όλες τις περιπτώσεις **δεν** είναι)
- Παράδειγμα: A^{16}
  - Brute-Force: A^{16} = A x A x A x … x A (15 πολ/μοί)
  - αλλά
  - $A^{16} = (((A^2)^2)^2)^2$ (4 πολ/μοί)

Προηγμένες Υπηρεσίες Ηλεκτρονικής Μάθησης στο ΤΕΙ Λάρισας

… **Brute-Force**

- Ταξινομήσεις
  - Brute-Force:
    - Φυσαλίδα,
    - Επιλογή, → $O(N^2)$
    - Εισαγωγή…
  - Πιο «έξυπνες τεχνικές»:
    - Quick sort,
    - Merge sort, → $O(N\log(N))$
    - Heap sort…

Συγκρίσεις



Τεχνική: **Αναδρομικοί αλγόριθμοι**

- Επιλύει ένα πρόβλημα επιλύοντας ένα ή περισσότερα μικρότερα κομμάτια του ίδιου προβλήματος.
- Τυπικό παράδειγμα: N! = N(N-1)! , 0!=1

Ακέραια συνάρτηση Παραγοντικό(Ακέραια παράμετρος Ν)
Αρχή
 Εάν Ν<2 Τότε
  Επέστρεψε 1
 Αλλιώς
  Επέστρεψε Ν x Παραγοντικό(Ν-1)
 Τέλος_Εάν
Τέλος

Κριτήριο τέλους

Αναδρομικοί αλγόριθμοι…

Πολλά προβλήματα επιλύονται με αναδρομή!!!

Παράδειγμα: Εύρεση του k-οστού άρτιου

Ακέραια Συνάρτηση Άρτιος_k(Ακέραια παράμετρος k)
Εάν k=1 Τότε
 Επέστρεψε 0
Αλλιώς
 Επέστρεψε 2 + Άρτιος_k(k-1)
Τέλος_Εάν

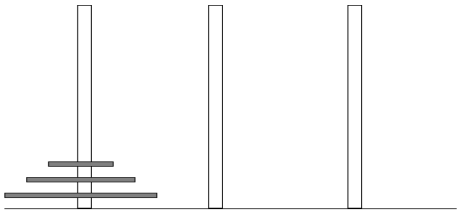
…Αναδρομικοί αλγόριθμοι…

Αναδρομικός αλγόριθμος αναζήτησης σε αταξινόμητο πίνακα

Ακέραια συνάρτηση Α_Αναζήτηση(Πίνακας Π, αρχή, τέλος, Χ)
Εάν τέλος < αρχή Τότε
 Επέστρεψε -1
Αλλιώς
 Εάν Π[αρχή] = Χ Τότε
  Επέστρεψε αρχή
 Αλλιώς
  Επέστρεψε Α_αναζήτηση(Π, αρχή+1, τέλος, Χ)
 Τέλος_Εάν
Τέλος_Εάν

Πύργοι Hanoi

- Μετακίνηση δίσκων: 1) Επιτρέπεται η μεταφορά ενός δίσκου κάθε φορά, και 2) Απαγορεύετε να βρεθούν δύο διαδοχικού δίσκοι στη ίδια κολόνα με τον μικρότερο κάτω από τον μεγαλύτερο.



…Αναδρομικοί αλγόριθμοι

- Η αναδρομή δεν αποτελεί πάντα την βέλτιστη λύση, αλλά:
- Πολλά προβλήματα επιλύονται με μαθηματική αναδρομή και κατά συνέπεια μπορούν να μεταφερθούν εύκολα σε έναν αλγόριθμο.

Τεχνική: **«Διαίρει και Βασίλευε»**

- Πάγια τακτική των Ρωμαίων αυτοκρατόρων:
  - Διαίρεσε τους εχθρούς σου,
  - Νικησέ τους έναν – έναν.

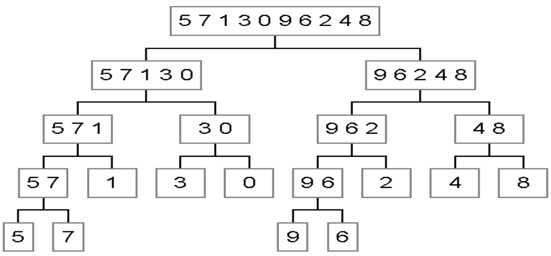
1. Διαίρει (το πρόβλημα σε μικρότερα επιμέρους προβλήματα),
2. Βασίλευε (λύνοντας το κάθε επιμέρους πρόβλημα εφαρμόζοντας αναδρομικά την ίδια τακτική), και τέλος
3. Συνένωσε (τις επιμέρους λύσεις σε μία καθολική λύση του προβλήματος).

Διαίρει και Βασίλευε: Merge Sort

```
Διαδικασία MergeSort(πίνακας A, ακέραιος p, ακέραιος r)
Αρχή
    Εάν (p < r) Τότε
        q = (p+r) / 2
        MergeSort(A, p, q)
        MergeSort(A, q+1, r)
        Merge(A, p, q, r)
    Τέλος Εάν
Τέλος Διαδικασίας «MergeSort»
```

(1) Διαίρει,

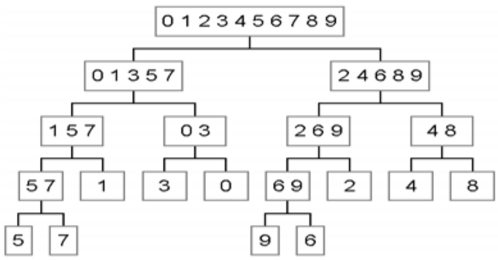
Είσοδος



Προηγμένες Υπηρεσίες Ηλεκτρονικής Μάθησης στο ΤΕΙ Λάρισας

(2) Συγχώνευσε, (3) Βασίλευε.

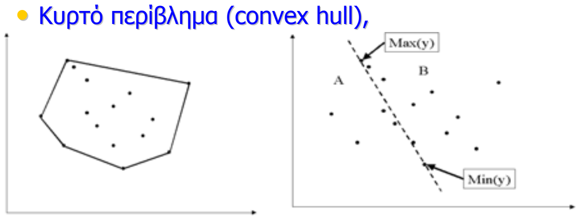
Έξοδος



Προηγμένες Υπηρεσίες Ηλεκτρονικής Μάθησης στο ΤΕΙ Λάρισας

…Διαίρει και Βασίλευε (εφαρμογές)

- Quick Sort,
- Κυρτό περίβλημα (convex hull),



Πολλά γεωμετρικά προβλήματα (πχ ελάχιστες αποστάσεις), και πολλές άλλες.

Προηγμένες Υπηρεσίες Ηλεκτρονικής Μάθησης στο ΤΕΙ Λάρισας

Δυναμικός Προγραμματισμός

- Αυτή η τεχνική προϋποθέτει μία αναδρομική επίλυση προβλημάτων αλλά με μία bottom-up εκτίμηση των λύσεων. **Οι υπό-λύσεις συνήθως αποθηκεύονται για την επαναχρησιμοποίησή τους.** Ο δυναμικός προγραμματισμός χρησιμοποιείται κυρίως σε προβλήματα βελτιστοποίησης τα οποία σχετίζονται με πολλές λύσεις αλλά και μια τιμή κόστους για το κάθε ένα από αυτά. Τυπικό παράδειγμα αποτελεί η επίλυση της συνάρτησης Fibbonaci η οποία ορίζεται ως εξής:

…Δυναμικός Προγραμματισμός…

$$f(n)=\begin{cases} f(n-1)+f(n-2), \forall n>2 \\ f(1)=f(2)=1 \end{cases}$$

…Δυναμικός Προγραμματισμός…

$$f(6)\begin{cases} f(5)\begin{cases} f(4)\begin{cases} f(3)\begin{cases} f(2) \\ f(1) \end{cases} \\ f(2) \end{cases} \\ f(3)\begin{cases} f(2) \\ f(1) \end{cases} \end{cases} \\ f(4)\begin{cases} f(3)\begin{cases} f(2) \\ f(1) \end{cases} \\ f(2) \end{cases} \end{cases}$$

…Δυναμικός Προγραμματισμός…

- Διαίρεση του προβλήματος σε μικρότερα και απλούστερα μέρη.
- Αποθήκευση των λύσεων σε ένα πίνακα. Αυτό γίνεται γιατί πολλές αν όχι όλες από τις λύσεις των υπό-προβλημάτων πρόκειται να ξαναχρησιμοποιηθούν και να συνδυασθούν και δεν ενδείκνυται το να επιλυθούν ξανά και ξανά.
- Συνδυασμός των επιμέρους λύσεων του προβλήματος ώστε να οδηγήσουν στην συνολική λύση.

…Δυναμικός Προγραμματισμός

- Πολλαπλασιασμός αλυσίδας πινάκων: $A_{3x4}B_{4x5}C_{5x1}D_{1x9}\ldots X_{IxJ}$
- String editing,
- Βιοπληροφορική,
- Χρονοπρογραμματισμός εργασιών,
- Και πολλές άλλες.

Άπληστοι Αλγόριθμοι…

- Δυναμικός Προγ/μός: Ψάχνει ΟΛΕΣ τις πιθανές λύσεις ➔ Βέλτιστη λύση (μερικές φορές ασύμφορη).
- Απληστία: Επιλέγει με καθαρά τοπικά κριτήρια τοπικά βέλτιστες λύσεις που **ίσως** οδηγήσουν σε καθολικά βέλτιστη λύση (όχι πάντα).
- Χαρακτηριστικό παράδειγμα: Επιστροφή ρέστων→ Επιστροφή **μεγαλύτερου** κέρματος κάθε φορά και αφαίρεσέ το από το σύνολο ρέστων

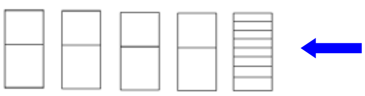
…Άπληστοι Αλγόριθμοι

Κάθε κλάσμα αναλύεται σε πεπερασμένο άθροισμα κλασμάτων με αριθμητή μονάδα (Αιγυπτιακά κλάσματα)

$$\frac{2}{3} = \frac{1}{2} + \frac{1}{6}$$

$$\frac{87}{110} = \frac{1}{2} + \frac{1}{4} + \frac{1}{25} + \frac{1}{1100}$$

Αιγυπτιακά κλάσματα…

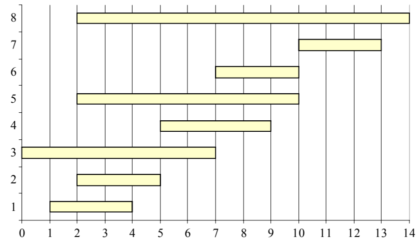


$$\frac{5}{8} = \frac{1}{2} + \frac{1}{8}$$

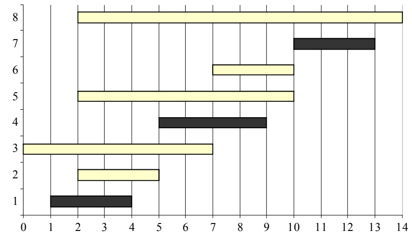
…Αιγυπτιακά κλάσματα (a/r)

```
Ακέραιος i ←2 {πρώτο πιθανό κλάσμα= ½, απληστία}
Εφόσον (a ≠ 0)
        Εάν (a*i ≠ r) Τότε
                Εμφάνισε «1/i»
                a ← a*i-r
                r ← r*i
        Τέλος_Εάν
        i ← i + 1
Τέλος_Εφόσον
```

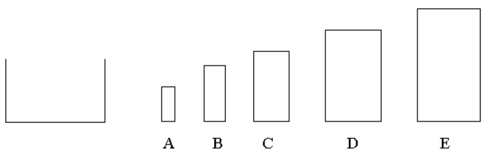
Χρονοπρογραμματισμός εργασιών



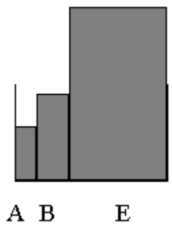
Χρονοπρογραμματισμός εργασιών



Πρόβλημα σακιδίου



Πρόβλημα σακιδίου

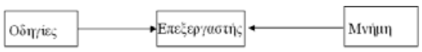


Heuristics

- Neighbor search,
- Simulated annealing,
- Tabu search,
- Artificial neural networks,
- Genetic algorithms,
- Lagrangean relaxation,
- Και άλλοι…

Προηγμένες Υπηρεσίες Ηλεκτρονικής Μάθησης στο ΤΕΙ Λάρισας

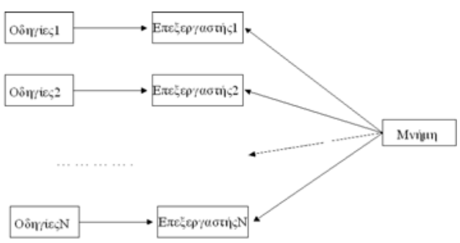
Παράλληλοι αλγόριθμοι (SISD)



28-02-2006 Ηλίας Κ. Σάββας

Προηγμένες Υπηρεσίες Ηλεκτρονικής Μάθησης στο ΤΕΙ Λάρισας

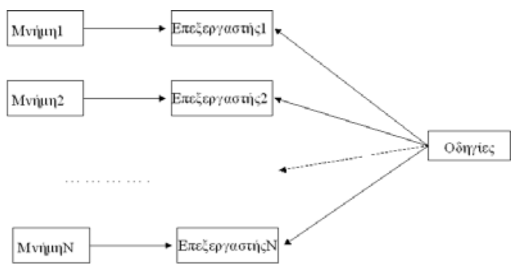
Παράλληλοι αλγόριθμοι (MISD)



28-02-2006 Ηλίας Κ. Σάββας

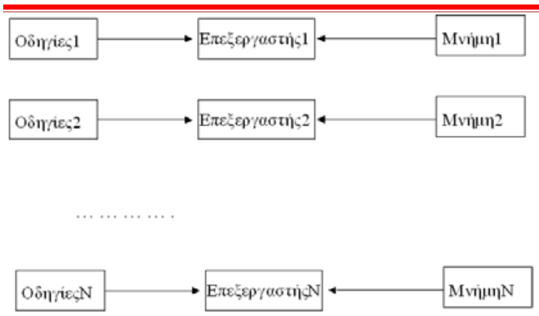
Προηγμένες Υπηρεσίες Ηλεκτρονικής Μάθησης στο ΤΕΙ Λάρισας

Παράλληλοι αλγόριθμοι (SIMD)



28-02-2006 Ηλίας Κ. Σάββας

Παράλληλοι αλγόριθμοι (MIMD)



Παράλληλος αλγόριθμος πρόσθεσης 100 αριθμών (P[100])

Παράλληλη Εκτέλεση (Επεξεργαστές Ε1, Ε2)

(Π) Ε1 ΚΑΙ Ε2: Ακέραιος i

(Π) Ε1: Αθροιστής S1, Ε2: Αθροιστής S2

Ε1: Αθροιστής S

(Π) Ε1: S1 ← 0, Ε2: S2 ← 0

(Π) Ε1: Για i από 1 μέχρι 50, Ε2: Για i από 51 μέχρι 100

(Π) Ε1: S1 ← S1 + P_i, Ε2: S2 ← S2 + P_i

(Π) Τέλος Για (i)

Ε2: Μεταφορά S2 → Ε1

Ε1: S ← S1 + S2

Ε1: Επέστρεψε S

Κόστος παράλληλων αλγόριθμων

Επιτάχυνση = (Χειρότερος χρόνος του καλύτερου σειριακού αλγόριθμου) / (Χρόνος παράλληλου αλγόριθμου)

Κόστος = (Επιτάχυνση) x (Αριθμός συμμετεχόντων CPU)

Απόδοση = (Χειρότερος χρόνος του καλύτερου σειριακού αλγόριθμου) / (Κόστος παράλληλου αλγόριθμου)

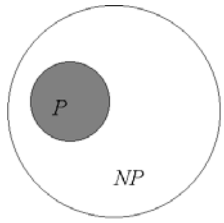
Περίγραμμα

- *Εισαγωγή*
- *Στοιχεία Πολυπλοκότητας*
- *Αλγόριθμοι*
  - *(Brute-Force, Αναδρομικοί, Διαίρει και Βασίλευε, Δυναμικός Προγ/μός, Άπληστοι, Ευρεστικοί,…)*
- **Κλάσεις Πολυπλοκότητας**

Κλάσεις πολυπλοκότητας

- Κλάση P: Ένας αλγόριθμος θεωρείται ότι είναι της κλάσης **P** εάν υπάρχει πολυώνυμο p τέτοιο ώστε f(N) = O(p(N)).
- ΟΡΙΣΜΟΙ:
  - **Το σύνολο των προσδιοριστικών αλγόριθμων απόφασης που επιλύονται σε πολυωνυμικό χρόνο, ανήκουν στην κλάση *P*.**
  - **Το σύνολο των αλγόριθμων απόφασης που επιλύονται σε πολυωνυμικό χρόνο από μη προσδιοριστικούς αλγόριθμους, ανήκουν στην κλάση *NP*.**

P ⊆ NP

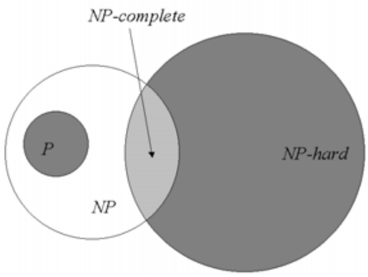


Παραμένει αναπόδεικτη η εικασία. Αυτό δείχνει την άγνοια του ότι ενώ είναι γνωστά προβλήματα της κλάσης *NP* δεν μπορεί να αποδειχθεί ότι δεν μπορεί να βρεθούν κάποιοι καλύτεροι αλγόριθμοι για αυτά τα προβλήματα ώστε να ανήκουν στην κλάση *P*.

NP-hard, NP-complete

- Σε συνέχεια της προηγούμενης άγνοιας → ***NP-complete*** **: ανήκουν στην κλάση NP** **ΚΑΙ** εάν έστω και ένα αποδειχθεί ότι ανήκει στην κλάση P τότε όλα επιλύονται σε πολυωνυμικό χρόνο.
- Ένα πρόβλημα *L* θεωρείται ότι ανήκει στην κλάση ***NP-hard*** εάν η ικανοποίηση του προβλήματος περιορίζει το *L* (ικανοποίηση ∝ *L*)

Σχέσεις P, NP, NP-hard, NP-complete



Περίγραμμα

- *Εισαγωγή*
- *Στοιχεία Πολυπλοκότητας*
- *Αλγόριθμοι*
  - *(Brute-Force, Αναδρομικοί, Διαίρει και Βασίλευε, Δυναμικός Προγ/μός, Άπληστοι, Ευρεστικοί,...)*
- *Κλάσεις Πολυπλοκότητας*

Προηγμένες Υπηρεσίες Ηλεκτρονικής Μάθησης στο ΤΕΙ Λάρισας

ΑΛΓΟΡΙΘΜΟΙ & ΣΤΟΙΧΕΙΑ ΠΟΛΥΠΛΟΚΟΤΗΤΑΣ

Ηλίας Κ. Σάββας
Επίκουρος Καθηγητής
Τμήμα: Τεχνολογίας Πληροφορικής & Τηλεπικοινωνιών
Email: savvas@teilar.gr