ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΑΛΓΟΡΙΘΜΟΙ

ΕΒΔΟΜΑΔΑ 1

ΔΙΑΔΙΚΑΣΤΙΚΑ

Διαδικαστικά του Μαθήματος

Τρόπος Εξέτασης

- Τελική Εξέταση
- Ασκήσεις

Στόχοι του Μαθήματος

- Ανάπτυξη αναλυτικής σκέψης, συνθετικής και κριτικής ικανότητας
- Ανάπτυξη και καλλιέργεια της δημιουργικότητας στο σχεδιασμό λύσεων και στην αναζήτηση εναλλακτικών λύσεων
- Εισαγωγή στις θεμελιώδεις αλγοριθμικές έννοιες και τεχνικές
- Παρουσίαση σημαντικών αλγορίθμων και ανάλυσή τους

Περιεχόμενο Μαθήματος

Εισαγωγή στους Αλγορίθμους

- Αλγοριθμική Επίλυση Προβλημάτων, Σημαντικά Προβλήματα

Ανάλυση Αλγορίθμων

- Πλαίσιο Ανάλυσης Αλγορίθμων, Συμβολισμοί και Πολυπλοκότητα, Εμπειρική Ανάλυση Αλγορίθμων
- Μαθηματική Ανάλυση μη Αναδρομικών Αλγορίθμων, Μαθηματική Ανάλυση Αναδρομικών Αλγορίθμων

Μεθοδολογίες

- Δυναμικός Προγραμματισμός, Άπληστες Μέθοδοι
- Brute Force and Exhaustive Search
- Decrease and Conquer, Divide and Conquer, Transform and Conquer
- Travelling Salesman Problem, Knapsack Problem, Assignment Problem

Αλγόριθμοι Ταξινόμησης

- Selection Sort, Bubble Sort, Insertion Sort, Median Sort, Quicksort, Heap Sort, Counting Sort, Bucket Sort

Αλγόριθμοι Αναζήτησης

- Σειριακή Αναζήτηση, Δυαδική Αναζήτηση, Hash-based Αναζήτηση

Αλγόριθμοι Γράφων

- Shortest Paths, Minimum Spanning Tree, Maximum Flow

Αλγόριθμοι Δένδρων

- Depth-first Search, Breadth-first Search, A* Search

Χωρο-Χρονική Ανάλυση

- Ταξινόμηση με Καταμέτρηση, Ταίριασμα Συμβολοσειρών, Hashing, B-Δένδρα

Επαναληπτική Βελτιστοποίηση

- Γραμμικός Προγραμματισμός, Μέθοδος Simplex, Πρόβλημα Μέγιστης Ροής

Περιορισμοί Αλγορίθμων

- P, NP και NP-Complete Προβλήματα, Προσεγγιστικοί Αλγόριθμοι

Βιβλιογραφία

- Cormen, T., Leiserson, C., Rivest, R., Stein, C., Introduction to Algorithms', 3rd Edition, The MIT Press, 2009
  - Το βιβλίο έχει εκδοθεί στα Ελληνικά από το ΙΤΕ-Πανεπιστημιακές Εκδόσεις Κρήτης (2012)
- Μποζάνης, Π., 'Αλγόριθμοι', Εκδόσεις Τζιόλα, 2006
- Levitin, A., 'Introduction to the Design and Analysis of Algorithms', 3rd Edition, Pearson Education Inc. publishing, 2012

ΕΙΣΑΓΩΓΗ

Αλγόριθμοι

- Τι είναι αλγόριθμοι;
- Σε τι μας χρησιμεύει η μελέτη τους;
- Ποιος είναι ο ρόλος τους σε σχέση με άλλες τεχνολογίες που χρησιμοποιούνται στους υπολογιστές;

Ορισμός Αλγορίθμου (1/2)

Knuth

- Ένας αλγόριθμος είναι μια πεπερασμένη, συγκεκριμένη, αποτελεσματική διαδικασία, με μια είσοδο και κάποια έξοδο

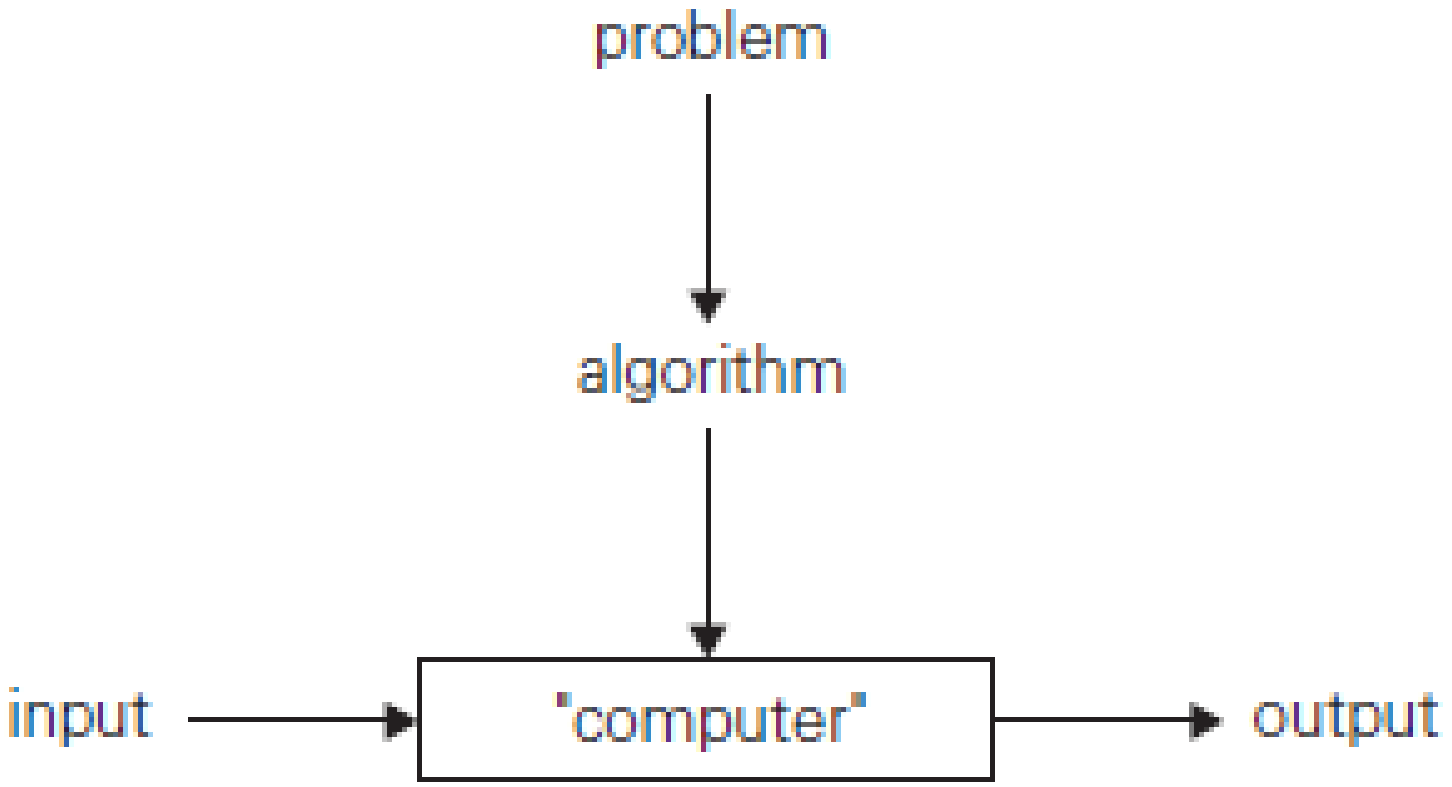
Schneider, M. and J. Gersting (1995), *An Invitation to Computer Science*, West Publishing Company, New York, NY, p. 9.

- Ένας αλγόριθμος είναι μια καλά ορισμένη ακολουθία σαφών και υπολογιστικά αποδοτικών λειτουργιών / βημάτων οι οποίες όταν εκτελεστούν παράγουν ένα αποτέλεσμα και τερματίζουν σε πεπερασμένο χρονικό διάστημα

Ο όρος ***αλγόριθμος*** αναφέρεται σε οποιαδήποτε καλά ορισμένη υπολογιστική διαδικασία που δέχεται κάποια τιμή ή κάποιο σύνολο τιμών ως ***είσοδο*** και δίνει κάποια τιμή ή κάποιο σύνολο τιμών ως ***έξοδο***.

Ένας αλγόριθμος είναι μια ακολουθία υπολογιστικών βημάτων που μετασχηματίζει την είσοδο στην έξοδο.

Ορισμός Αλγορίθμου (2/2)



Παράδειγμα Αλγορίθμου (1/2)

Ταξινόμηση μιας ακολουθίας αριθμών

- Είσοδος: η ακολουθία των αριθμών ($\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_N$)
- Έξοδος: μια αναδιάταξη των αριθμών της εισόδου ($\alpha'_1, \alpha'_2, \alpha'_3, \ldots, \alpha'_N$) τέτοια ώστε ($\alpha'_1 <= \alpha'_2 <= \alpha'_3 <= \ldots <= \alpha'_N$) για αύξουσα σειρά ή ($\alpha'_1 >= \alpha'_2 >= \alpha'_3 >= \ldots >= \alpha'_N$) για φθίνουσα σειρά
- Για την ακολουθία (12, 3, 24, 35, 6, 32) θα πάρουμε (3, 6, 12, 24, 32, 35)

Παράδειγμα Αλγορίθμου (2/2)

Αλγόριθμος του Ευκλείδη

- Εύρεση μέγιστου κοινού διαιρέτη
- Είσοδος: δύο θετικοί ακέραιοι αριθμοί
- Έξοδος: ο μεγαλύτερος ακέραιος που διαιρεί ακριβώς τους δύο αριθμούς που παίρνουμε σαν είσοδο
- Βασίζεται στην εφαρμογή πολλαπλών εκτελέσεων της ισότητας

$$\gcd(m, n) = \gcd(n, m \bmod n)$$

- Μέχρι να έχουμε m mod n = 0
- Παράδειγμα

$$\gcd(60, 24) = \gcd(24, 12) = \gcd(12, 0) = 12.$$

ALGORITHM *Euclid*(m, n)

//Computes gcd(m, n) by Euclid's algorithm
//Input: Two nonnegative, not-both-zero integers m and n
//Output: Greatest common divisor of m and n
while $n \neq 0$ **do**
    $r \leftarrow m \bmod n$
    $m \leftarrow n$
    $n \leftarrow r$
return m

Ορθοί Αλγόριθμοι

Στιγμιότυπο του προβλήματος δηλώνει την είσοδο η οποία απαιτείται για να υπολογιστεί μια λύση του προβλήματος (εννοείται ότι η είσοδος αυτή θα ικανοποιεί όλους τους περιορισμούς που επιβάλλει η διατύπωση του προβλήματος)

Ένας αλγόριθμος θεωρείται **σωστός / ορθός** να για κάθε στιγμιότυπο εισόδου, ο αλγόριθμος τερματίζει δίνοντας σωστή έξοδο

Ένας **μη ορθός** αλγόριθμος μπορεί να μην τερματίζει καν σε ορισμένα στιγμιότυπα εισόδου, ή μπορεί να τερματίζει δίνοντας αποτέλεσμα διαφορετικό από το ζητούμενο

Διάσταση Προβλήματος

Παραδείγματα

- Πλήθος αριθμών n
- Πλήθος αντικειμένων n

Επηρεάζει την επίδοση ενός αλγορίθμου

Εφαρμογές Αλγορίθμων

Οι αλγόριθμοι βρίσκονται στην καρδιά κάθε υπολογιστικής διαδικασίας

- Δίκτυα επικοινωνιών
- Διαδίκτυο και Παγκόσμιος Ιστός
- Τεχνητή Νοημοσύνη
- Μεταγλωττιστές
- Βάσεις Δεδομένων
- Ανάλυση Δεδομένων
- Γραφικά Υπολογιστών
- Επεξεργασία Σημάτων
- Επιστημονικός Υπολογισμός
- Ηλεκτρονικό Εμπόριο
- ……..

Συγγραφή Αλγορίθμων

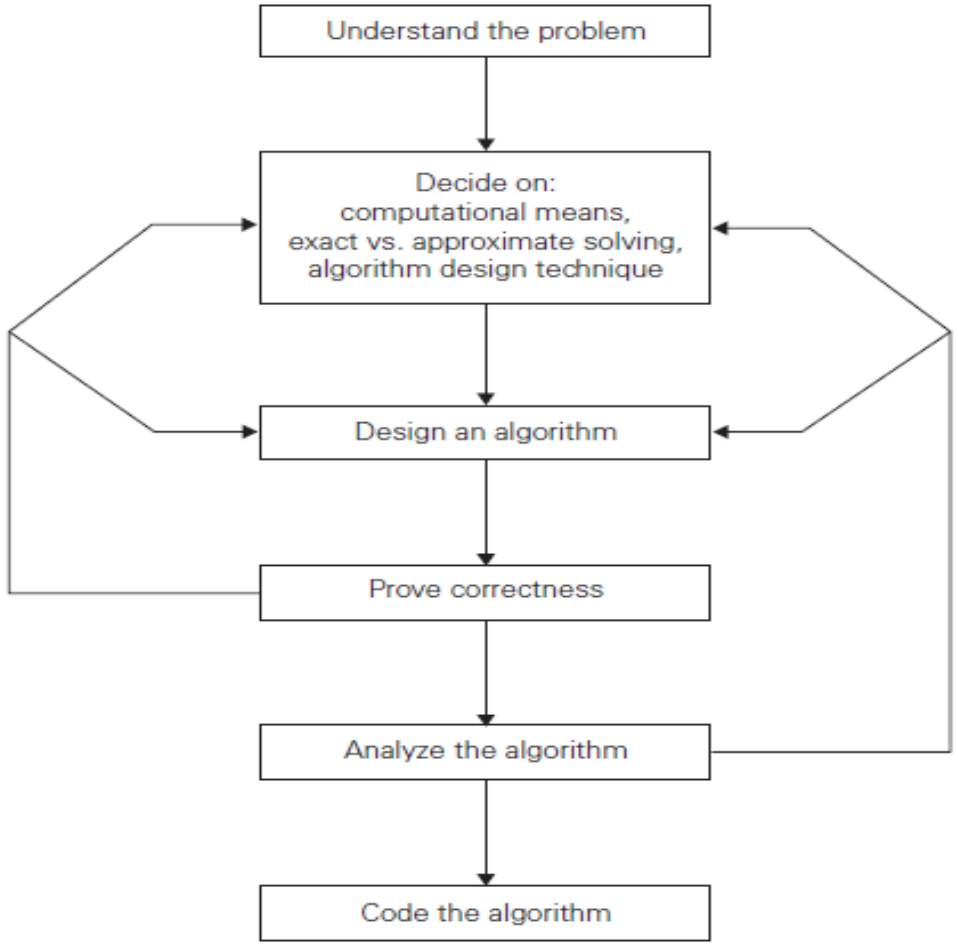
Πολλαπλές επιλογές

- Χρήση φυσικής γλώσσας (Ελληνικά – Αγγλικά - …)
  - Μειονέκτημα: χρήση πολλών λέξεων
- Χρήση γλωσσών προγραμματισμού
  - Μειονέκτημα: απαιτείται η γνώση της γλώσσας προγραμματισμού στην οποία έχει γραφτεί ο αλγόριθμος
- Ψευδοκώδικας
- Διαγράμματα ροής δεδομένων

Βασικές Έννοιες Επίλυσης Προβλημάτων

Βήματα επίλυσης:

- Πλήρης κατανόηση του προβλήματος
- Επιλογή μεθόδου επίλυσης
- Σχεδιασμός αλγορίθμου
- Απόδειξη ορθότητας / εγκυρότητας
- Ανάλυση του Αλγορίθμου
- Κωδικοποίηση



Υπολογιστικές Μηχανές - Υλικό

- Μετά την κατανόηση του προβλήματος θα πρέπει να διαπιστώσουμε τις δυνατότητες του Υλικού
- Οι περισσότεροι αλγόριθμοι έχουν σχεδιαστεί για μηχανές αρχιτεκτονικής von Neumann
- Βασική θέση στις αρχιτεκτονικές αυτές έχει η λεγόμενη **Random Access Machine (RAM)**
- Βασική υπόθεση: οι εντολές εκτελούνται **ακολουθιακά**
- Οι αντίστοιχοι αλγόριθμοι ονομάζονται **ακολουθιακοί**
- Οι σύγχρονες εξελίξεις στο Υλικό βοηθούν τη δημιουργία παράλληλων αλγορίθμων (οι λειτουργίες εκτελούνται παράλληλα)

Επιλογή Μεθόδου Επίλυσης

Επακριβής επίλυση του προβλήματος

- exact algorithms

Προσεγγιστική επίλυση

- approximation algorithms

Υπάρχει πλήθος προβλημάτων που δεν λύνονται επακριβώς (π.χ. τετραγωνικές ρίζες, μη γραμμικά συστήματα, ολοκληρώματα)

Οι διαθέσιμοι αλγόριθμοι για την επίλυση προβλημάτων μπορεί να είναι πολύ αργοί λόγω **υψηλής πολυπλοκότητας**

Τεχνικές Σχεδίασης Αλγορίθμων

Κρίσιμο ερώτημα: Πως σχεδιάζουμε ένα αλγόριθμο για ένα δοσμένο πρόβλημα;

Τεχνική σχεδίασης αλγορίθμων

- *μια γενική προσέγγιση για την αλγοριθμική επίλυση προβλημάτων που είναι εφαρμόσιμη σε μια ποικιλία προβλημάτων σε διάφορες περιοχές της Επιστήμης των Υπολογιστών*

Ένα σύνολο τεχνικών θα παρουσιαστούν στα πλαίσια του μαθήματος

Κάποιες τεχνικές μπορεί να είναι ανεφάρμοστες

Συνδυασμός τεχνικών

Αλγόριθμοι και Δομές Δεδομένων

- Δίνουμε έμφαση στην επιλογή των απαραίτητων δομών δεδομένων
- Παράδειγμα: ο αλγόριθμος του Ερατοσθένη για την εξαγωγή των πρώτων αριθμών που δεν ξεπερνούν ένα δοσμένο n > 1 (***sieve of Eratosthenes***) διαρκεί περισσότερο αν υιοθετηθούν συνδεδεμένες λίστες αντί για απλούς πίνακες
- Μεγαλύτερη σημασία λόγω των αντικειμενοστραφών γλωσσών προγραμματισμού

Sieve of Eratosthenes (1/2)

- Αρχικοποιεί μια λίστα από υποψήφιους πρώτους αριθμούς με συνεχόμενους αριθμούς από 2 μέχρι n
- Στο πρώτο iteration εξαλείφει όλα τα πολλαπλάσια του 2, 4, 6, …
- Προχωρά στο επόμενο στοιχείο 3 και εξαλείφει τα πολλαπλάσιά του
- Δεν χρειάζεται εξέταση του 4
- Το επόμενο στοιχείο είναι το 5
- Συνεχίζει μέχρι να μην υπάρχει αριθμός που να μπορεί να εξαλειφθεί

2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
2	3		5		7		9		11		13		15		17		19		21		23		25
2	**3**		5		7				11		13				17		19				23		25
2	3		**5**		7				11		13				17		19				23		

Sieve of Eratosthenes (2/2)

ALGORITHM *Sieve*(n)

```
//Implements the sieve of Eratosthenes
//Input: A positive integer n > 1
//Output: Array L of all prime numbers less than or equal to n
for p ← 2 to n do A[p] ← p
for p ← 2 to ⌊√n⌋ do     //see note before pseudocode
    if A[p] ≠ 0          //p hasn't been eliminated on previous passes
        j ← p * p
        while j ≤ n do
            A[j] ← 0     //mark element as eliminated
            j ← j + p
//copy the remaining elements of A to array L of the primes
i ← 0
for p ← 2 to n do
    if A[p] ≠ 0
        L[i] ← A[p]
        i ← i + 1
return L
```

Απόδειξη Ορθότητας

- Όταν οριστεί ένας αλγόριθμος θα πρέπει να αποδείξουμε την ***ορθότητα του*** (***correctness***)
- Ο αλγόριθμος επιστρέφει αποτέλεσμα για κάθε είσοδο σε πεπερασμένο χρονικό διάστημα
- Μπορεί να είναι πολύπλοκη διαδικασία
- Τυπική προσέγγιση: **μαθηματική επαγωγή**
- Οι επαναλήψεις ενός αλγορίθμου προσφέρουν μια ‘φυσική’ αλληλουχία βημάτων για την υιοθέτηση της επαγωγής
- Στους προσεγγιστικούς αλγορίθμους αποδεικνύουμε ότι το σφάλμα είναι κάτω από ένα συγκεκριμένο όριο

Ανάλυση Αλγορίθμων

Αποδοτικότητα (efficiency)

- Μπορεί να οριστεί με μαθηματικό τρόπο

Time efficiency

- Πόσο γρήγορα ‘τρέχει’ ένας αλγόριθμος

Space efficiency

- Πόση επιπλέον μνήμη χρειάζεται

Απλότητα (simplicity)

- Είναι υποκειμενική (απλοί αλγόριθμοι μπορούν να γίνουν αντιληπτοί πιο εύκολα)

Γενικότητα (Generality)

- Γενικότητα του προβλήματος που επιλύει ο αλγόριθμος
- Το σύνολο των δεδομένων που δέχεται

Κωδικοποίηση

Μεγάλη προσοχή στο πέρασμα από τον αλγόριθμο στο πρόγραμμα

- Κίνδυνος λανθασμένης κωδικοποίησης
- Κίνδυνος μη αποδοτικής κωδικοποίησης

Επαλήθευση (verification)

Εγκυρότητα (validity)

- Πολλαπλά και εξαντλητικά τεστ
- Τεχνικές testing και debugging

Σημαντικά Προβλήματα

- Ταξινόμηση
- Αναζήτηση
- Επεξεργασία Αλφαριθμητικών
- Προβλήματα Γράφων
- Συνδυαστικά Προβλήματα
- Γεωμετρικά Προβλήματα
- Αριθμητικά Προβλήματα

Αλγόριθμοι Ταξινόμησης

Αναδιάταξη αντικειμένων μιας δοσμένης λίστας

Τα αντικείμενα πρέπει να επιδέχονται μιας διάταξης

Για εγγραφές πρέπει να επιλέξουμε ένα πεδίο ως προς το οποίο θα γίνεται η ταξινόμηση: **κλειδί (key)**

Stability

- Ο αλγόριθμος διατηρεί τη σχετική σειρά οποιονδήποτε δύο ίσων αντικειμένων στην είσοδο του

In place

- Ο αλγόριθμος δεν απαιτεί επιπλέον μνήμη

Αλγόριθμοι Αναζήτησης

- Εύρεση μιας τιμής (**search key**) σε ένα σύνολο αντικειμένων
- Σειριακή, Δυαδική αναζήτηση, αλγόριθμοι αναπαράστασης των δεδομένων σε διαφορετική μορφή εύκολα ‘αναζητήσιμη’
- Δεν υπάρχει το πρόβλημα της σταθερότητας
- Σε δυναμικά περιβάλλοντα με συνεχείς αλλαγές των δεδομένων οι αλγόριθμοι αναζήτησης πρέπει να συνδυάζονται με ενέργειες προσθαφαίρεσης δεδομένων
- Προσεκτική επιλογή δομών δεδομένων

Επεξεργασία Αλφαριθμητικών

- Ένα αλφαριθμητικό (string) είναι μια ακολουθία από χαρακτήρες ενός αλφαβήτου
- Ακολουθίες χαρακτήρων, δυαδικών ψηφίων, ακολουθίες γονιδίων, κ.λπ.
- **String matching**
  - Παράδειγμα: Αναζήτηση μιας λέξης μέσα σε ένα κείμενο

Αλγόριθμοι Γράφων

- Ένας **γράφος (graph)** είναι ένα σύνολο **κόμβων (vertices)** από τους οποίους κάποιοι είναι συνδεδεμένοι μεταξύ τους μέσω **ακμών (edges)**
- **Κατευθυνόμενοι – Μη κατευθυνόμενοι**
- Μοντελοποιούν πλήθος προβλημάτων
- Βασικοί αλγόριθμοι
  - Διάσχιση γράφων
  - Συντομότερο μονοπάτι
  - Τοπολογική ταξινόμηση
- Κάποια προβλήματα είναι υπολογιστικά πολύπλοκα
  - Travelling salesman problem (shortest tour through n cities that visits every city exactly once)
  - Graph coloring problem (assign the smallest number of colors to the vertices of a graph so that no two adjacent vertices have the same color)

Συνδυαστικά – Γεωμετρικά Προβλήματα

Συνδυαστικά προβλήματα

- Αναζητούν ένα συνδυαστικό αντικείμενο (π.χ., αναδιάταξη, συνδυασμό) το οποίο ικανοποιεί ένα σύνολο περιορισμών
- Πολύ δύσκολα προβλήματα
  - Επηρεάζονται από το μέγεθος του προβλήματος
  - Απαιτούν υψηλό χρόνο εκτέλεσης

Γεωμετρικά προβλήματα

- Σχετίζονται με σχήματα (γραμμές, σημεία, πολύγωνα)
- The closest-pair problem is self-explanatory: given n points in the plane, find the closest pair among them
- The convex-hull problem asks to find the smallest convex polygon that would include all the points of a given set

Αριθμητικά Προβλήματα

- Επίλυση εξισώσεων, υπολογισμός ολοκληρωμάτων, επίλυση συστημάτων εξισώσεων, κ.λπ.
- Πολλά προβλήματα αυτής της κατηγορίας μπορούν να επιλυθούν μόνο προσεγγιστικά
- Απαιτούν χειρισμό πραγματικών αριθμών οι οποίοι τυχαίνουν προσεγγιστικής επεξεργασίας από τους υπολογιστές

ΑΝΑΛΥΣΗ ΑΛΓΟΡΙΘΜΩΝ

Μέγεθος Εισόδου

Λογική παρατήρηση: η συντριπτική πλειοψηφία αλγορίθμων ‘τρέχει’ περισσότερο για μεγαλύτερο μέγεθος εισόδου

Μελετούμε την απόδοση των αλγορίθμων ως συνάρτηση μιας παραμέτρου **n** που υποδηλώνει το μέγεθος εισόδου

Παραδείγματα

- Μέγεθος λίστας στοιχείων για:
  - Ταξινόμηση
  - Αναζήτηση
  - Εύρεση μικρότερου – μεγαλύτερου
  - …

Μετρικές Μεγέθους Εισόδου

Χρήση μιας τυπικής χρονικής μονάδας (π.χ., second, millisecond)

- Εξάρτηση από το υλικό
- Εξάρτηση από την ποιότητα του αλγορίθμου / προγράμματος

Καλύτερη λύση: Μέτρηση του πλήθους των εκτελέσεων των λειτουργιών των αλγορίθμων

- Αναγνώριση των **βασικών λειτουργιών (basic operations)** – λειτουργίες που συνεισφέρουν περισσότερο στο χρόνο εκτέλεσης
- Μέθοδος: αναγνώριση της λειτουργίας που απαιτεί τον περισσότερο χρόνο στον πιο εσωτερικό βρόχο (loop)

Παράδειγμα – Ταξινόμηση - Βασική λειτουργία: Σύγκριση

Παράδειγμα

Έστω **c_{op}** είναι ο χρόνος εκτέλεσης μιας βασικής λειτουργίας

Έστω **C(n)** είναι το πλήθος των βασικών λειτουργιών που θα εκτελέσει ο αλγόριθμος

Η εκτίμηση του χρόνου εκτέλεσης είναι **$T(n) \sim c_{op} C(n)$**

Προσοχή: θα πρέπει να λάβουμε υπόψιν μας όλες τις λειτουργίες ενός αλγορίθμου

Υπόθεση: $C(n) = \frac{1}{2}n(n-1)$

Αν διπλασιάσουμε το μέγεθος n της εισόδου πόσο περισσότερο θα αυξηθεί ο χρόνος εκτέλεσης?

$$C(n) = \frac{1}{2}n(n-1) = \frac{1}{2}n^2 - \frac{1}{2}n \approx \frac{1}{2}n^2 \qquad \frac{T(2n)}{T(n)} \approx \frac{c_{op}C(2n)}{c_{op}C(n)} \approx \frac{\frac{1}{2}(2n)^2}{\frac{1}{2}n^2} = 4.$$

Ρυθμός Αύξησης

Ρυθμός αύξησης (rate of growth, order of growth) υποδηλώνει το ρυθμό αύξησης του χρόνου εκτέλεσης

Στον ακόλουθο πίνακα απεικονίζονται παραδείγματα ρυθμού αύξησης

n	$\log_2 n$	n	$n \log_2 n$	n^2	n^3	2^n	$n!$
10	3.3	10^1	$3.3 \cdot 10^1$	10^2	10^3	10^3	$3.6 \cdot 10^6$
10^2	6.6	10^2	$6.6 \cdot 10^2$	10^4	10^6	$1.3 \cdot 10^{30}$	$9.3 \cdot 10^{157}$
10^3	10	10^3	$1.0 \cdot 10^4$	10^6	10^9		
10^4	13	10^4	$1.3 \cdot 10^5$	10^8	10^{12}		
10^5	17	10^5	$1.7 \cdot 10^6$	10^{10}	10^{15}		
10^6	20	10^6	$2.0 \cdot 10^7$	10^{12}	10^{18}		

Περιπτώσεις Ανάλυσης

Χειρότερη περίπτωση (worst case efficiency)

- Η αποδοτικότητα για την είσοδο της χειρότερης περίπτωσης που είναι μια είσοδος μεγέθους n για την οποία ο αλγόριθμος απαιτεί τον **περισσότερο** χρόνο εκτέλεσης για όλες τις πιθανές εισόδους μεγέθους n
- Μας δίνει ένα άνω όριο εκτέλεσης
- Αναγνώριση της εισόδου που απαιτεί τον περισσότερο χρόνο εκτέλεσης

Μέση περίπτωση (average case efficiency)

- Υιοθετούμε υποθέσεις για το μέγεθος εισόδου n
- Πιθανοτικοθεωρητική ανάλυση

Καλύτερη περίπτωση (best case efficiency)

- Η αποδοτικότητα για την είσοδο της καλύτερης περίπτωσης που είναι μια είσοδος μεγέθους n για την οποία ο αλγόριθμος απαιτεί το **λιγότερο** χρόνο εκτέλεσης για όλες τις πιθανές εισόδους μεγέθους n

Σειριακή Αναζήτηση (1/4)

- Αναζητά το κλειδί K σε μια λίστα n στοιχείων
- Ελέγχει σειριακά την ύπαρξη του στοιχείου σε κάθε θέση της λίστας
- Επιστρέφει τη θέση στην οποία έχει βρεθεί το κλειδί αλλιώς επιστρέφει το -1
- Ο χρόνος εκτέλεσης μπορεί να είναι διαφορετικός ακόμα και για το ίδιο n

ALGORITHM *SequentialSearch*$(A[0..n-1], K)$

//Searches for a given value in a given array by sequential search
//Input: An array $A[0..n-1]$ and a search key K
//Output: The index of the first element in A that matches K
// or -1 if there are no matching elements
$i \leftarrow 0$
while $i < n$ **and** $A[i] \neq K$ **do**
    $i \leftarrow i + 1$
if $i < n$ **return** i
else return -1

Σειριακή Αναζήτηση (2/4)

- Χειρότερη περίπτωση:
  - $C_{worst}(n) = n$
- Καλύτερη περίπτωση:
  - $C_{best}(n) = 1$

ALGORITHM *SequentialSearch*$(A[0..n-1], K)$

//Searches for a given value in a given array by sequential search
//Input: An array $A[0..n-1]$ and a search key K
//Output: The index of the first element in A that matches K
// or -1 if there are no matching elements
$i \leftarrow 0$
while $i < n$ **and** $A[i] \neq K$ **do**
  $i \leftarrow i + 1$
if $i < n$ **return** i
else return -1

Σειριακή Αναζήτηση (3/4)

- Μέση περίπτωση:
  - Έστω p η πιθανότητα μιας επιτυχούς αναζήτησης (0 <= p <= 1)
  - Η πιθανότητα επιτυχούς αναζήτησης στη θέση i είναι ίδια για όλα τα i ίση με **p/n** και το πλήθος των αναζητήσεων είναι i
  - Σε περίπτωση ανεπιτυχούς αναζήτησης οι συγκρίσεις θα είναι n με πιθανότητα **1 – p**
  - Άρα

$$C_{avg}(n) = [1 \cdot \frac{p}{n} + 2 \cdot \frac{p}{n} + \cdots + i \cdot \frac{p}{n} + \cdots + n \cdot \frac{p}{n}] + n \cdot (1 - p)$$

$$= \frac{p}{n}[1 + 2 + \cdots + i + \cdots + n] + n(1 - p)$$

$$= \frac{p}{n}\frac{n(n+1)}{2} + n(1 - p) = \frac{p(n+1)}{2} + n(1 - p).$$

Σειριακή Αναζήτηση (4/4)

- Αν p = 1:
  - Συγκρίσεις: **(n+1)/2**
- Αν p = 0:
  - Συγκρίσεις: **n**

$$C_{avg}(n) = [1 \cdot \frac{p}{n} + 2 \cdot \frac{p}{n} + \cdots + i \cdot \frac{p}{n} + \cdots + n \cdot \frac{p}{n}] + n \cdot (1 - p)$$

$$= \frac{p}{n}[1 + 2 + \cdots + i + \cdots + n] + n(1 - p)$$

$$= \frac{p}{n}\frac{n(n+1)}{2} + n(1 - p) = \frac{p(n+1)}{2} + n(1 - p).$$