ΤΜΗΜΑ ΕΦΑΡΜΟΣΜΕΝΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΑΝΑΛΥΣΗ ΑΛΓΟΡΙΘΜΩΝ

Εισαγωγή

Κωνσταντίνος Γιαννουτάκης
Επίκουρος Καθηγητής

ΣΥΝΟΨΗ ΔΙΑΛΕΞΗΣ

- Εισαγωγικά / Διαδικαστικά
- Εισαγωγή στους Αλγορίθμους
- Εισαγωγή στην Πολυπλοκότητα

ΕΙΣΑΓΩΓΙΚΑ / ΔΙΑΔΙΚΑΣΤΙΚΑ

- *Διαλέξεις*
- *Περίγραμμα Μαθήματος*
- *Μαθησιακά Αποτελέσματα*
- *Περιεχόμενο Μαθήματος*
- *Συγγράμματα*
- *Επικοινωνία*
- *Σελίδα Μαθήματος*

ΠΕΡΙΓΡΑΜΜΑ ΜΑΘΗΜΑΤΟΣ

ΚΩΔΙΚΟΣ ΜΑΘΗΜΑΤΟΣ	CSC401	**ΕΞΑΜΗΝΟ ΣΠΟΥΔΩΝ**	Δ
ΤΙΤΛΟΣ ΜΑΘΗΜΑΤΟΣ	ΑΝΑΛΥΣΗ ΑΛΓΟΡΙΘΜΩΝ		
ΕΒΔΟΜΑΔΙΑΙΕΣ ΩΡΕΣ ΔΙΔΑΣΚΑΛΙΑΣ		**ΠΙΣΤΩΤΙΚΕΣ ΜΟΝΑΔΕΣ**	
3		5	
ΤΥΠΟΣ ΜΑΘΗΜΑΤΟΣ	Ειδικού υποβάθρου		
ΠΡΟΑΠΑΙΤΟΥΜΕΝΑ ΜΑΘΗΜΑΤΑ:	Αλγόριθμοι, Διακριτά μαθηματικά		
ΗΛΕΚΤΡΟΝΙΚΗ ΣΕΛΙΔΑ ΜΑΘΗΜΑΤΟΣ (URL)	https://openeclass.uom.gr/courses/DAI183/		

ΜΑΘΗΣΙΑΚΑ ΑΠΟΤΕΛΕΣΜΑΤΑ

Κατανόηση των ιδιαίτερων χαρακτηριστικών και στοιχείων της ανάλυσης αλγορίθμων, εξοικείωση με τις **βασικές στρατηγικές επίλυσης προβλημάτων**. Απόκτηση ικανότητας **εφαρμογής γνώσεων ανάλυσης αλγορίθμων** σε γνωστούς αλγόριθμους αλλά και σε **νέους αλγόριθμους**.

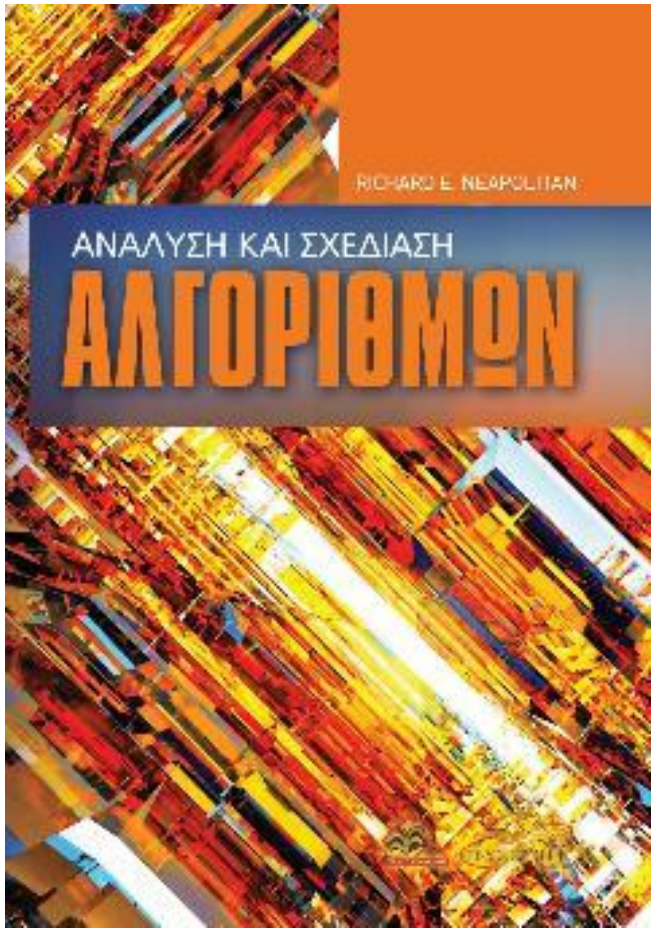
Συγκεκριμένα:

- Εξήγηση της χρήσης του ασυμπτωτικού συμβολισμού για να περιγράψουν το χρόνο εκτέλεσης ενός αλγόριθμου σε σχέση με το μέγεθος του προβλήματος.
- Χρήση των ασυμπωτικών συμβολισμών για τον προσδιορισμό με τυπικό τρόπο των ασυμπτωτικών φραγμάτων (άνω, άνω και κάτω) της χρονικής πολυπλοκότητα των αλγόριθμων.
- Χρήση αναδρομικών σχέσεων για να προσδιορίσουν το χρονική πολυπλοκότητα αναδρομικών αλγόριθμων. Επίλυση βασικών αναδρομικών σχέσεων, χρήση του Master Theorem.
- Κατανόηση βασικών κλάσεων πολυπλοκότητας αλγορίθμων.
- Κατανόηση βασικών αρχών και μεθόδων σχεδίασης αλγορίθμων.

ΠΕΡΙΕΧΟΜΕΝΟ ΜΑΘΗΜΑΤΟΣ

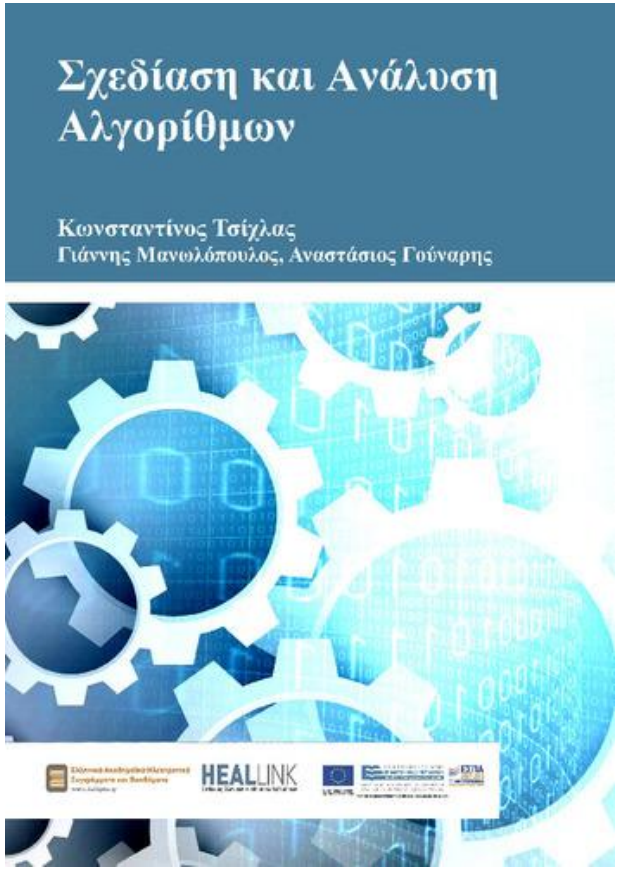
- Η έννοια του αλγορίθμου και της πολυπλοκότητας
- Ασυμπτωτικοί συμβολισμοί (ο, Ο, ω, Ω, Θ) και ασυμπτωτική ανάλυση αλγορίθμων
- Αναδρομικοί αλγόριθμοι και αναδρομικές εξισώσεις
- Οι κλάσεις P και NP, προβλήματα NP-complete και NP-hard
- Τεχνικές σχεδίασης αλγορίθμων: ωμή βία, εξαντλητική αναζήτηση, διαίρει και βασίλευε, μείωση και κυριαρχία, απληστία, δυναμικός προγραμματισμός
- Αλγόριθμοι δένδρων: Δυαδικά δένδρα αναζήτησης, δένδρα σωροί, AVL δένδρα, Β δένδρα
- Αλγόριθμοι γράφων: Βασικοί ορισμοί και έννοιες γράφων, αναπαράσταση, διερεύνηση κατά πλάτος και βάθος, ελάχιστα συνδετικά δένδρα, συντομότερες διαδρομές

ΣΥΓΓΡΑΜΜΑ



Βιβλίο [112690793]: Ανάλυση και Σχεδίαση Αλγορίθμων, Neapolitan Richard

ΒΟΗΘΗΤΙΚΑ ΣΥΓΓΡΑΜΜΑΤΑ



https://repository.kallipos.gr/handle/11419/4005

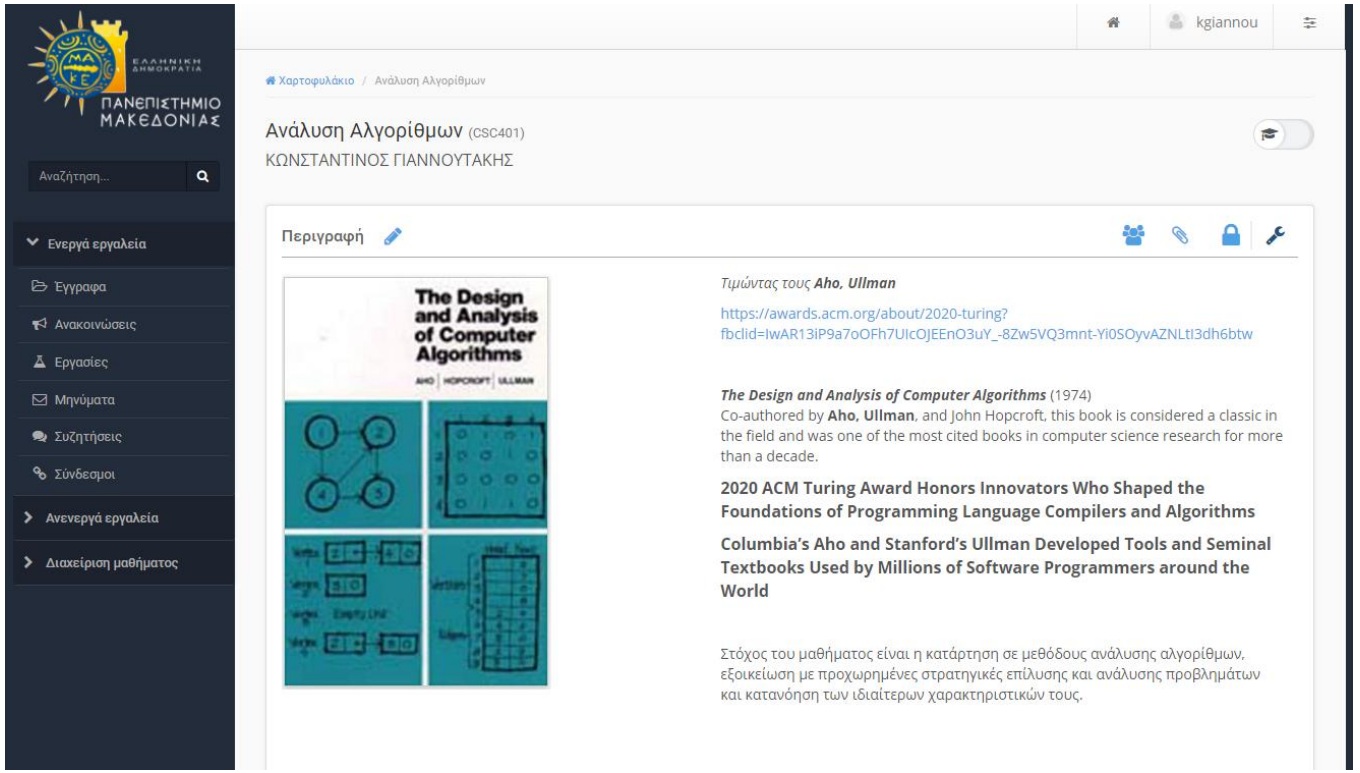
https://repository.kallipos.gr/handle/11419/2067

ΕΠΙΚΟΙΝΩΝΙΑ

- Email
  - kgiannou@uom.edu.gr
- Ώρες Γραφείου (331):
  - Τρίτη 09:00-12:00 και Τετάρτη 12:00-16:00, καθώς και οποιαδήποτε άλλη μέρα και ώρα κατόπιν συνεννόησης.

ΣΕΛΙΔΑ ΜΑΘΗΜΑΤΟΣ

- https://openeclass.uom.gr/courses/DAI183/



ΑΞΙΟΛΟΓΗΣΗ

- Τελική γραπτή εξέταση

ΕΙΣΑΓΩΓΗ ΣΤΟΥΣ ΑΛΓΟΡΙΘΜΟΥΣ

- *Προβλήματα και Στιγμιότυπα*
- *Αλγόριθμοι και Χαρακτηριστικά*
- *Θεμελιώδεις Δομές Δεδομένων*

ΠΡΟΒΛΗΜΑΤΑ ΚΑΙ ΣΤΙΓΜΙΟΤΥΠΑ

- **Πρόβλημα**: Έννοια βασική, π.χ. το οικονομικό πρόβλημα
- **Υπολογιστικά Προβλήματα (ΥΠ)**: Αυτά που **λύνονται με χειρισμούς αριθμών** (αριθμητικές πράξεις, συγκρίσεις, εγγραφές και διαγραφές αριθμών, κ.τ.λ.)
- **Μαθηματική περιγραφή ΥΠ**: **Γίνεται συνήθως με παραμέτρους εισόδου και άγνωστες μεταβλητές**
  - Συνήθως, ζητείται υπολογισμός των αγνώστων ως συνάρτηση των παραμέτρων

ΣΗΜΑΝΤΙΚΟΙ ΤΥΠΟΙ ΠΡΟΒΛΗΜΑΤΩΝ

- Ταξινόμηση (sorting problems)
- Αναζήτηση (searching problems)
- Επεξεργασία συμβολοσειρών (string processing problems)
- Προβλήματα γράφων (graph problems)
- Προβλήματα συνδυαστικής (combinatorial problems)
- Γεωμετρικά προβλήματα (geometric problems)
- Προβλήματα απόφασης (decision problems)
- Προβλήματα βελτιστοποίησης (optimization problems)
- Αριθμητικά προβλήματα (numerical problems)

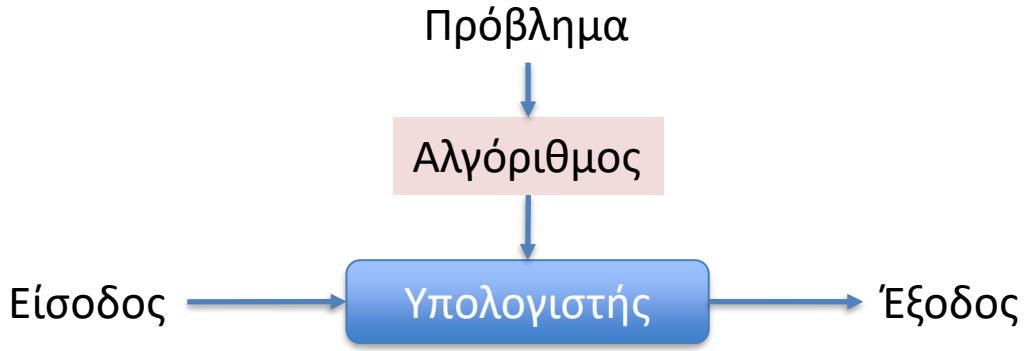
ΠΡΟΒΛΗΜΑΤΑ ΚΑΙ ΣΤΙΓΜΙΟΤΥΠΑ

- **Στιγμιότυπο ή περίπτωση ΥΠ:** Προκύπτει **αναθέτοντας συγκεκριμένες τιμές στις παραμέτρους** του προβλήματος
  - Να βρεθεί ο ΜΚΔ των αριθμών 28 και 15
  - Να βρεθεί αν το προσανατολισμένο γράφημα με V={1,2,3,4,5,6,7} και E={ (1,2), (1,5), (1,7), (2,4), (2,6), (3,2), (3,6), (4,3), (5,1), (7,2)}, έχει προσανατολισμένους κύκλους

ΑΛΓΟΡΙΘΜΟΣ

Οι αλγόριθμοι (algorithms) ή αλγοριθμική (algorithmics) είναι κάτι παραπάνω από ένας απλός κλάδος της πληροφορικής.
Είναι ο πυρήνας της και, αν θέλουμε να είμαστε δίκαιοι, μπορούμε να πούμε ότι άπτεται των περισσοτέρων τομέων της επιστήμης, της επιχειρηματικότητας και της τεχνολογίας.
David Harel: Algorithmics: the spirit of computing

Αλγόριθμος είναι μια πεπερασμένη ακολουθία ξεκάθαρων και ρητών εντολών για την επίλυση ενός προβλήματος.



ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΑΛΓΟΡΙΘΜΩΝ

1. Είναι **πεπερασμένος**. Τελειώνει ύστερα από έναν πεπερασμένο αριθμό διεργασιών ή βημάτων.
2. Είναι **καλά ορισμένος**. Κάθε κανόνας του ορίζεται επακριβώς και η αντίστοιχη διεργασία είναι συγκεκριμένη.
3. Έχει **είσοδο και έξοδο**. Έχει μηδέν ή περισσότερα μεγέθη εισόδου. Δίνει τουλάχιστον ένα μέγεθος σαν αποτέλεσμα που εξαρτάται κατά κάποιο τρόπο από τις αρχικές εισόδους.
4. Είναι **αποτελεσματικός**. Όλες οι διαδικασίες που περιλαμβάνει μπορούν να πραγματοποιηθούν με ακρίβεια και σε πεπερασμένο χρόνο "με μολύβι και χαρτί".

ΠΑΡΑΔΕΙΓΜΑΤΑ ΑΛΓΟΡΙΘΜΩΝ

Υπολογισμός **μέγιστου κοινού διαιρέτη** (greatest common divisor) δυο μη αρνητικών ακεραίων *m* και *n* (gcd*(m,n)*).

Αλγόριθμος του Ευκλείδη (Euclid's algorithm)

Βήμα 1: Εάν *n*=0, επέστρεψε την τιμή του *m* ως απάντηση και σταμάτησε. Διαφορετικά, πήγαινε στο Βήμα 2

Βήμα 2: Διαίρεσε τον *m* με τον *n* και ανάθεσε την τιμή του υπολοίπου στον *r*.

Βήμα 3: Ανάθεσε την τιμή του *n* στον *m* και του *r* στον *n*. Πήγαινε στο Βήμα 1.

gcd(1071,462)			
Βήμα	***m***	***n***	***r***
Αρχικοποίηση	1071	462	0
Βήμα 2	1071	462	147
Βήμα 3	462	147	147
Βήμα 2	462	147	21
Βήμα 3	147	21	21
Βήμα 2	147	21	0
Βήμα 3	21	0	0
Βήμα 1	Επέστρεψε το *m* (*m*=21)		

ΠΑΡΑΔΕΙΓΜΑΤΑ ΑΛΓΟΡΙΘΜΩΝ

Εύρεση **πρώτων αριθμών** μέχρι ένα συγκεκριμένο ακέραιο (primes(*n*)).

Κόσκινο του Ερατοσθένη (Sieve of Eratosthenes algorithm)

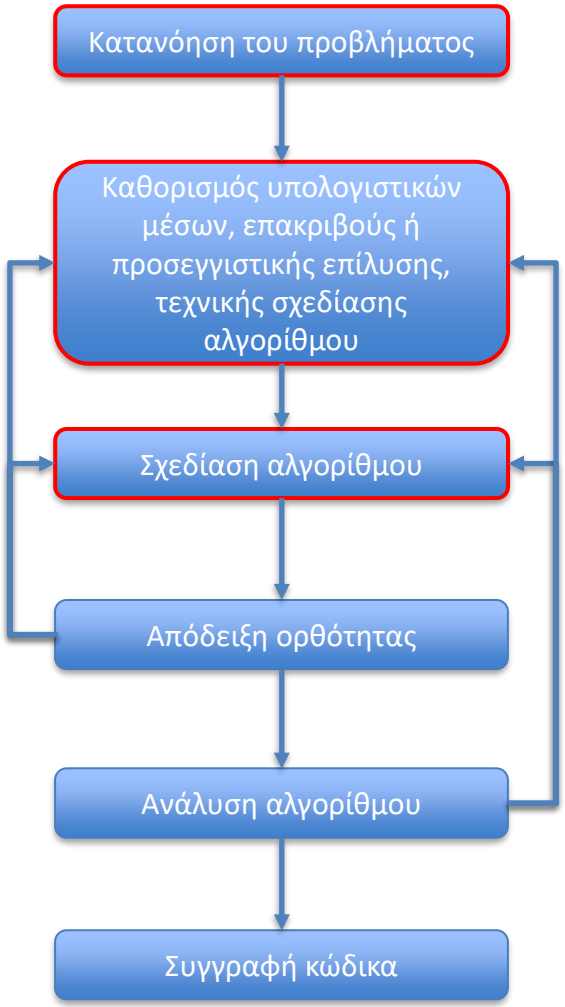
Βήμα 1: Δημιούργησε μια λίστα από διαδοχικούς ακέραιους από το 2 έως το *n*.
Βήμα 2: Θέσε ως *p* τον πρώτο αριθμό της λίστας, δηλαδή *p*=2.
Βήμα 3: Διέγραψε από τη λίστα όλα τα πολλαπλάσια του *p* που είναι μικρότερα ή ίσα του *n*.
Βήμα 4: Βρες τον επόμενο αριθμό της λίστας μετά τον *p*, και αντικατέστησε τον *p* με αυτόν τον αριθμό.
Βήμα 5: Επανάλαβε τα βήματα 3 και 4 έως ότου το p^2 να είναι μεγαλύτερο από *n*.
Βήμα 6: Οι αριθμοί που απέμειναν στη λίστα είναι όλοι πρώτοι αριθμοί.

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		

Γιατί όταν $p^2>n$ έχουμε βρει όλους τους πρώτους αριθμούς;

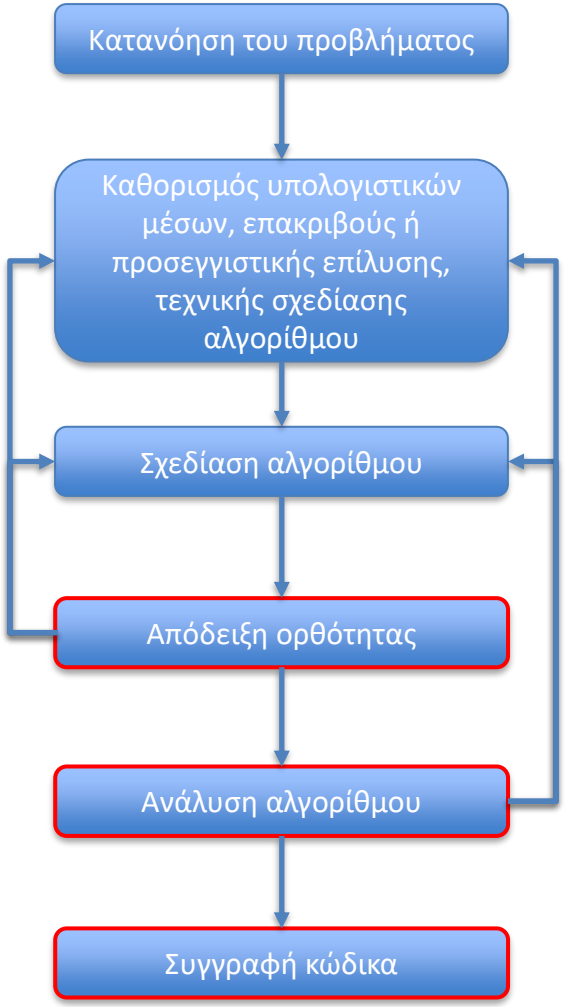
ΒΑΣΙΚΕΣ ΕΝΝΟΙΕΣ ΑΛΓΟΡΙΘΜΙΚΗΣ ΕΠΙΛΥΣΗΣ

- **Κατανόηση προβλήματος:** Πλήρης κατανόηση του δεδομένου προβλήματος, της εισόδου και εξόδου του.
- **Υπολογιστικό μέσο:** Καθορισμός των δυνατοτήτων της υπολογιστικής μηχανής που θα εκτελεστεί ο αλγόριθμος.
- **Επιλογή ακριβούς ή προσεγγιστικής λύσης:** Ανάπτυξη ακριβούς αλγόριθμου (exact algorithm) ή προσεγγιστικού (approximate algorithm) ανάλογα με το πρόβλημα επίλυσης.
- **Σχεδίαση αλγορίθμου:** Υπάρχουν προκαθορισμένες στρατηγικές για την παροχή καθοδήγησης στη σχεδίαση ενός αλγορίθμου ανάλογα με την φύση του κάθε προβλήματος. Βέλτιστη επιλογή της δομής δεδομένων.



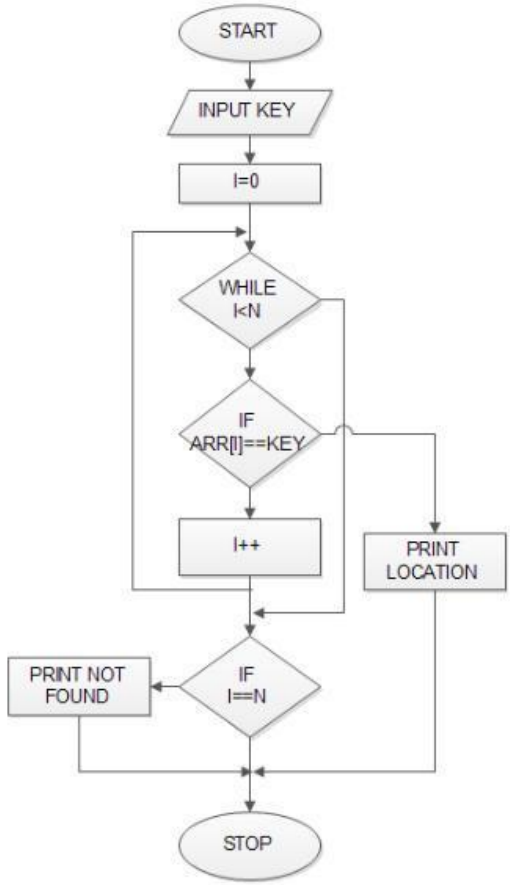
ΒΑΣΙΚΕΣ ΕΝΝΟΙΕΣ ΑΛΓΟΡΙΘΜΙΚΗΣ ΕΠΙΛΥΣΗΣ

- **Απόδειξη ορθότητας:** Απόδειξη ότι ο αλγόριθμος παράγει το επιθυμητό αποτέλεσμα για κάθε αποδεκτή είσοδο.
- **Ανάλυση αλγορίθμου:** Υπολογισμός χρονικής αποδοτικότητας (time efficiency), χωρικής αποδοτικότητας (space efficiency), γενικότητας (generality) και απλότητας (simplicity).
- **Συγγραφή κώδικα:** Ορθή και βέλτιστη υλοποίηση σε γλώσσα προγραμματισμού που υποστηρίζει της επιθυμητές λειτουργίες.



ΑΝΑΠΑΡΑΣΤΑΣΗ ΑΛΓΟΡΙΘΜΩΝ

Διάγραμμα Ροής



Ψευδοκώδικας

```
Algorithm Linear-Search (S,x)
begin
    index ← 1
    while (S[index]≠x and index≤n) do
        index ← index+1
    end
    if (index > n) then
        index ← 0
end
```

ΝΤΕΤΕΡΜΙΝΙΣΤΙΚΟΙ ΑΛΓΟΡΙΘΜΟΙ

- Ένας ντετερμινιστικός (deterministic) αλγόριθμος θα βρει πάντοτε **την ίδια λύση με τον ίδιο αριθμό υπολογισμών** της αντικειμενικής συνάρτησης (objective-function evaluations) ανεξάρτητα από το χρόνο που άρχισε, **εάν το αρχικό σημείο και οι συνθήκες τερματισμού είναι ίδιες.**

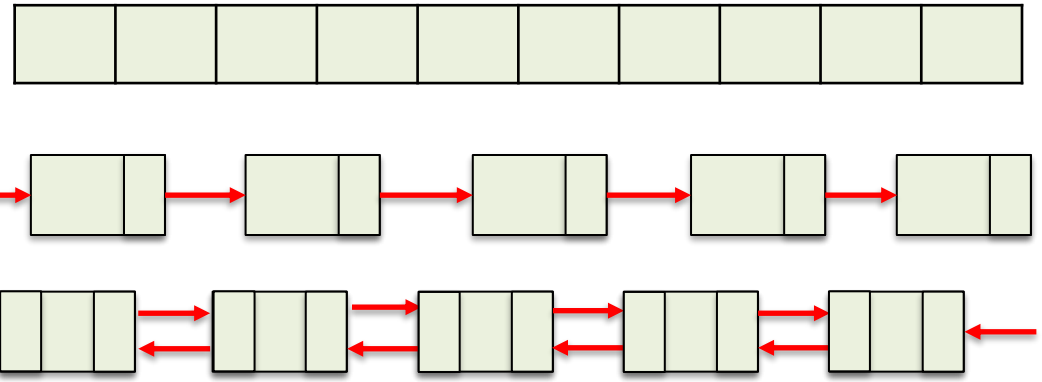
- Εάν ο αλγόριθμος τρέξει πολλές φορές στον ίδιο υπολογιστή ο χρόνος εκτέλεσης για κάθε φορά θα είναι *ακριβώς* ο ίδιος.

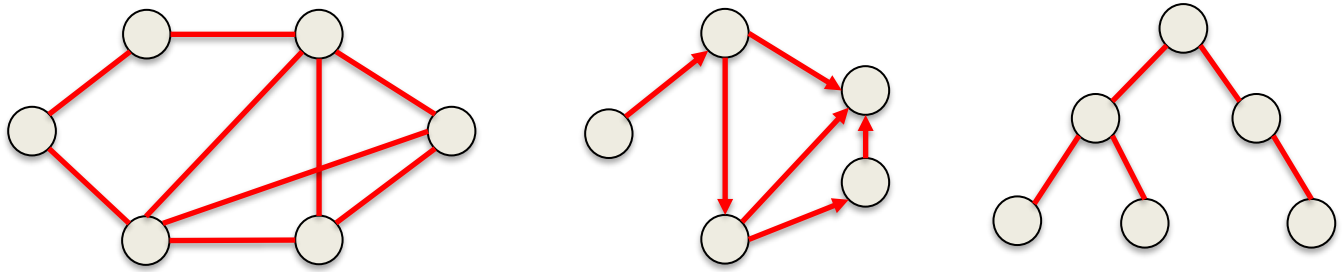
ΣΤΟΧΑΣΤΙΚΟΙ ΑΛΓΟΡΙΘΜΟΙ

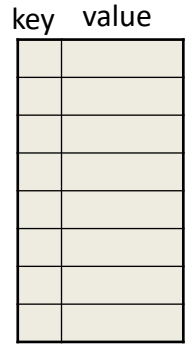
- Οι στοχαστικοί (stochastic) αλγόριθμοι **βασίζονται στη τυχαιότητα**. Σε αυτή την περίπτωση **δεν είναι καθόλου βέβαιο ότι** ένας στοχαστικός αλγόριθμος **θα βρει την ίδια λύση όταν τρέχει πολλές φορές** με τις ίδιες αρχικές συνθήκες.
- Οι στοχαστικοί αλγόριθμοι είναι γενικά απλούστεροι από τους ντετερμινιστικούς αλγόριθμους
- Είναι λιγότερο αποδοτικοί
- Μπορεί να μην βρουν τη βέλτιστη λύση σε ορισμένες περιπτώσεις
  - Μπορούν όμως να βρουν **μια εν-μέρει βέλτιστη λύση** ψάχνοντας σε ένα μεγάλο χώρο αναζήτησης, ενώ οι ντετερμινιστικοί αλγόριθμοι συνήθως αποτυγχάνουν σε τέτοιες περιπτώσεις.
  - Στις περισσότερες εφαρμογές μια τέτοια λύση είναι χρήσιμη.
- Οι στοχαστικοί αλγόριθμοι είναι **ευέλικτοι**. Μπορούν να εφαρμοστούν σε όλων των ειδών τα προβλήματα.

ΘΕΜΕΛΙΩΔΕΙΣ ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ

- Μια **δομή δεδομένων** (data structure) αποτελεί ένα σχήμα οργάνωσης σχετικών μεταξύ τους δεδομένων.
- Ευρέως χρησιμοποιούμενες δομές
  - Πίνακες
  - Απλά/Διπλά συνδεδεμένες λίστες
  - Δένδρα/Γράφοι
  - Σύνολα και λεξικά







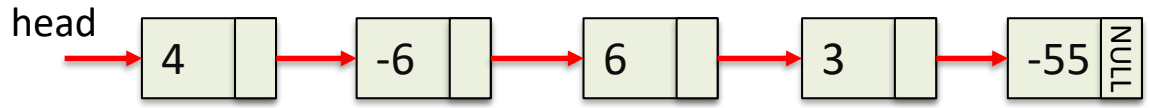
ΓΡΑΜΜΙΚΕΣ ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ

- Πίνακες

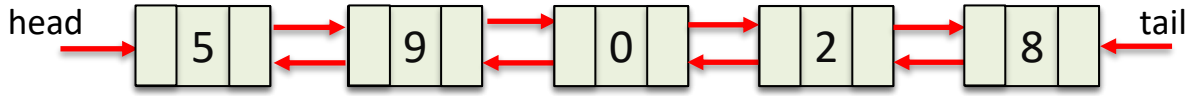
5	3	-4	2	0	0	9	-9	3	2
0	1	2	3	4	5	6	7	8	9

5	3	-4	2
-2	0	65	3
0	0	23	6
1	-3	5	4

- Απλά συνδεδεμένες λίστες (linked list)

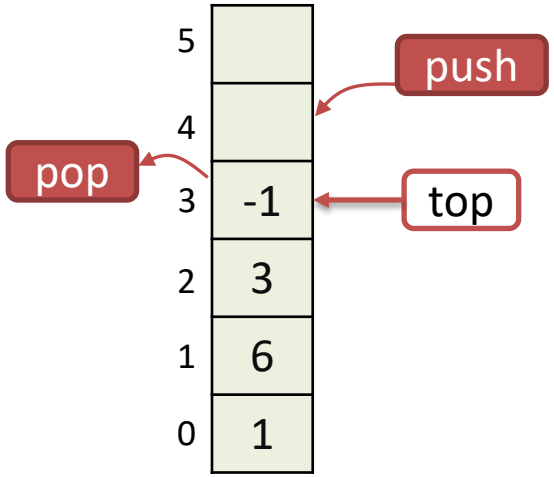


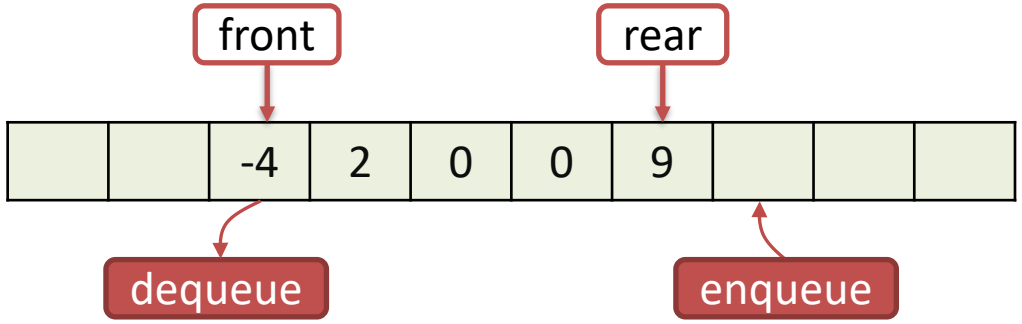
- Διπλά συνδεδεμένες λίστες (double linked list)



ΓΡΑΜΜΙΚΕΣ ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ

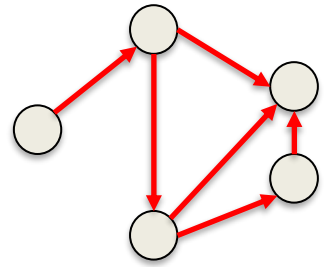
- Στοίβα (stack)
  - isEmpty
  - isFull
  - push
  - pop
- Ουρά (queue)
  - isEmpty
  - enqueue
  - dequeue

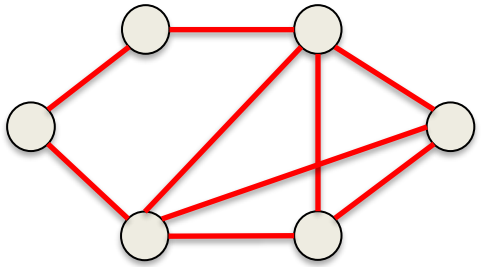


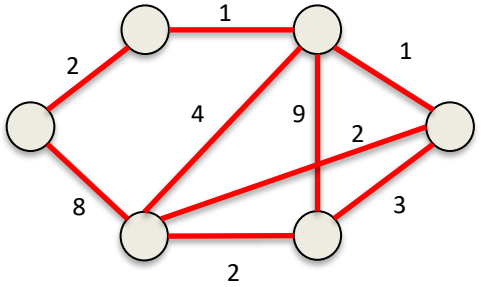


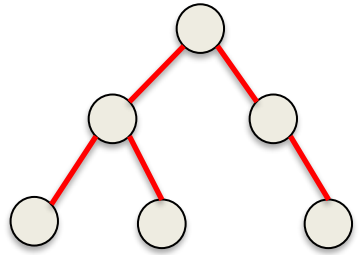
ΓΡΑΦΟΙ/ΔΕΝΔΡΑ

- Κατευθυνόμενοι γράφοι
- Μη κατευθυνόμενοι γράφοι
- Γράφοι με βάρη (σταθμισμένοι γράφοι)
- Δένδρα









ΤΕΧΝΙΚΕΣ ΣΧΕΔΙΑΣΗΣ ΑΛΓΟΡΙΘΜΩΝ

Μια τεχνική σχεδίασης αλγορίθμων (algorithm design technique) είναι **μια γενική προσέγγιση επίλυσης προβλημάτων με αλγοριθμικό τρόπο**, η οποία εφαρμόζεται σε μια πληθώρα προβλημάτων

- Ωμή βία (Brute Force)
- Διαίρει και βασίλευε (Divide and Conquer)
- Μείωση και κυριαρχία (Decrease and Conquer)
- Μετασχηματισμός και κυριαρχία (Transform and Conquer)
- Δυναμικός προγραμματισμός (Dynamic Programming)
- Απληστία (Greedy)
- Οπισθοδρόμηση (Backtracking)
- Διακλάδωση και περιορισμός (Branch and bound)

ΠΟΛΥΠΛΟΚΟΤΗΤΑ

- *Βασικοί Ορισμοί*
- *Βασική Πράξη*

ΕΙΣΑΓΩΓΗ

- Για κάθε επιλύσιμο πρόβλημα Π, υπάρχει ένα σύνολο αλγορίθμων $\{A_1, A_2, \cdots, A_k\}$ που το επιλύουν. Ποιόν αλγόριθμο θα χρησιμοποιήσουμε;

Ποιος αλγόριθμος επιλύει το Π σε λιγότερο χρόνο;

Εάν διπλασιάσω τα δεδομένα εισόδου, πόσο θα αυξηθούν οι χρόνοι εκτέλεσης των αλγορίθμων;

Υπάρχει κάποιος αλγόριθμος που επιλύει το Π σε δεδομένο χρόνο (π.χ. Τ(n));

ΠΟΛΥΠΛΟΚΟΤΗΤΑ

- Πρακτικό ενδιαφέρον έχουν οι αλγόριθμοι οι οποίοι είναι αποτελεσματικοί, δηλαδή ελαχιστοποιούν
  - το χρόνο επίλυσης του προβλήματος
  - το χώρο που καταλαμβάνουν στην κύρια μνήμη
- Θα λέμε ότι ένας αλγόριθμος είναι:
  - **Αποτελεσματικός** όταν έχει μικρή πολυπλοκότητα (low complexity)
  - **Μη-αποτελεσματικός** όταν έχει μεγάλη πολυπλοκότητα (high complexity)

ΠΟΛΥΠΛΟΚΟΤΗΤΑ

Πως θα επιλέξουμε τον πιο αποτελεσματικό αλγόριθμο από τους $\{A_1, A_2, \cdots, A_k\}$ για την επίλυση του προβλήματος Π;

Εμπειρική πολυπλοκότητα
Μετρώντας το χρόνο εκτέλεσης του αλγορίθμου σε συγκεκριμένη μηχανή

Θεωρητική πολυπλοκότητα
Καθορίζοντας με τα κατάλληλα μαθηματικά εργαλεία τον χρόνο και χώρο που απαιτεί ο αλγόριθμος, συναρτήσει του μεγέθους του προβλήματος

- Δεν εξαρτάται από τα χαρακτηριστικά του Η/Υ
- Δεν εξαρτάται από τη γλώσσα προγραμματισμού
- Δεν εξαρτάται από τις ικανότητες του προγραμματιστή

ΑΡΧΗ ΤΗΣ ΣΤΑΘΕΡΟΤΗΤΑΣ

Αρχή της σταθερότητας (principle of invariance)

Δυο διαφορετικές υλοποιήσεις του ίδιου αλγορίθμου δεν διαφέρουν στην αποτελεσματικότητά τους περισσότερο από ένα σταθερό πολλαπλάσιο

- Εάν $t_1(n)$ και $t_2(n)$ είναι οι χρόνοι εκτέλεσης δύο υλοποιήσεων του ίδιου αλγόριθμου Α, τότε υπάρχει πάντα σταθερά c τέτοια ώστε:

$$t_1(n) = ct_2(n)$$

όπου n το μέγεθος του προβλήματος που επιλύουν.

ΘΕΜΕΛΙΩΔΗ Η ΒΑΣΙΚΗ ΠΡΑΞΗ

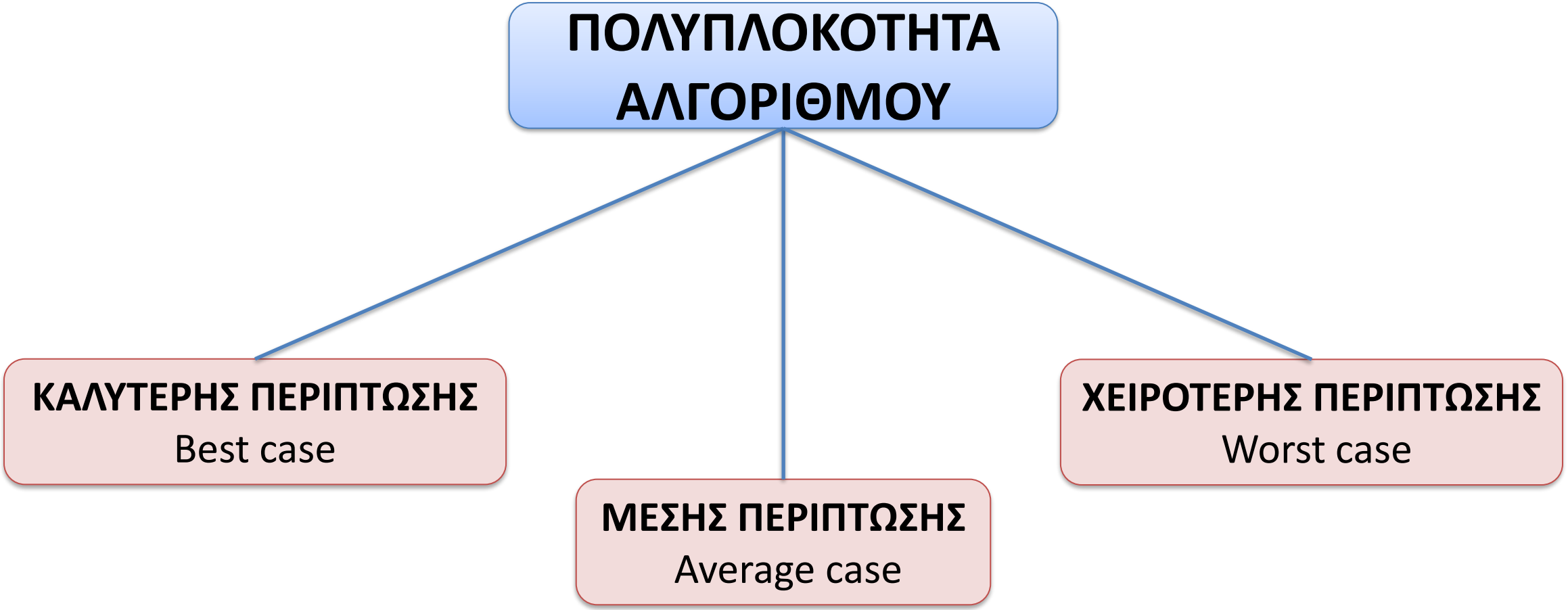
Ποια μονάδα μέτρησης θα χρησιμοποιήσουμε για να εκφράσουμε την αποτελεσματικότητα ενός αλγορίθμου;

Βασική πράξη

Η πράξη της οποίας ο χρόνος εκτέλεσης φράσσεται άνω από μια σταθερά η οποία εξαρτάται μόνο από τη χρησιμοποιούμενη εφαρμογή (όχι από Η/Υ, γλώσσα προγραμματισμού κ.τ.λ.)

Πλήθος βασικών πράξεων = *f* (Μέγεθος εισόδου)

ΠΟΛΥΠΛΟΚΟΤΗΤΑ ΑΛΓΟΡΙΘΜΟΥ



ΚΑΛΥΤΕΡΗ/ΧΕΙΡΟΤΕΡΗ ΠΕΡΙΠΤΩΣΗ

- Έστω D_n το σύνολο όλων των εισόδων μεγέθους n.
- Για κάθε είσοδο $I \in D_n$, έστω $t(I)$ το σύνολο των Βασικών Πράξεων που εκτελούνται από τον αλγόριθμο, δηλαδή

$$t: D_n \to \mathbb{N}$$

Πολυπλοκότητα καλύτερης περίπτωσης

$$B(n) = \min\{t(I) | I \in D_n\}$$

Πολυπλοκότητα χειρότερης περίπτωσης

$$W(n) = \max\{t(I) | I \in D_n\}$$

ΜΕΣΗ ΠΕΡΙΠΤΩΣΗ

- Έστω ότι μπορούμε να αντιστοιχίσουμε μια πιθανότητα $p(I)$ για κάθε είσοδο $I \in D_n$, έτσι ώστε

$$p(I) = \text{Πιθανότητα εμφάνισης της εισόδου } I$$

Πολυπλοκότητα μέσης περίπτωσης

$$A(n) = \sum_{I \in D_n} p(I)t(I)$$

ΠΑΡΑΔΕΙΓΜΑ (ΣΕΙΡΙΑΚΗ ΑΝΑΖΗΤΗΣΗ – SEQUENTIAL/LINEAR SEARCH)

```
Algorithm Linear-Search (S,x)
begin
   index ← 1
   while (S[index]≠x and index≤n) do
      index ← index+1
   end
   if (index > n) then
      index ← 0
end
```

ΠΑΡΑΔΕΙΓΜΑ (ΣΕΙΡΙΑΚΗ ΑΝΑΖΗΤΗΣΗ)

```
Algorithm Linear-Search (S,x)
begin
   index ← 1
   while (S[index]≠x and index≤n) do
      index ← index+1
   end
   if (index > n) then
      index ← 0
end
```

x=4

α) S=[1,3,7,4,6,9]

β) S=[4,2,6,8,9,3]

γ) S=[9,6,2,1,3,5]

δ) S=[3,1,5,9,6,4]

ΠΑΡΑΔΕΙΓΜΑ (ΣΕΙΡΙΑΚΗ ΑΝΑΖΗΤΗΣΗ)

```
Algorithm Linear-Search (S,x)
begin
   index ← 1
   while (S[index]≠x and index≤n) do
      index ← index+1
   end
   if (index > n) then
      index ← 0
end
```

x=4

α) S=[1,3,7,4,6,9]

β) S=[4,2,6,8,9,3]

γ) S=[9,6,2,1,3,5]

δ) S=[3,1,5,9,6,4]

$$B(n) = 2 + 1$$

$$W(n) = 2n + 1$$

ΣΕΙΡΙΑΚΗ ΑΝΑΖΗΤΗΣΗ – ΜΕΣΗ ΠΕΡΙΠΤΩΣΗ

- Υποθέτουμε τα ακόλουθα:
  - Όλα τα στοιχεία του S είναι διαφορετικά
  - Το στοιχείο αναζήτησης υπάρχει στο S ($x \in S$)
  - Οι θέσεις εμφάνισης του x έχουν την ίδια πιθανότητα
- Έστω I_i η είσοδος για την οποία ισχύει $x = S[i], 1 \leq i \leq n$. Τότε

$$p(I_i) = \frac{1}{n} \text{ και } t(I_i) = 2i + 1$$

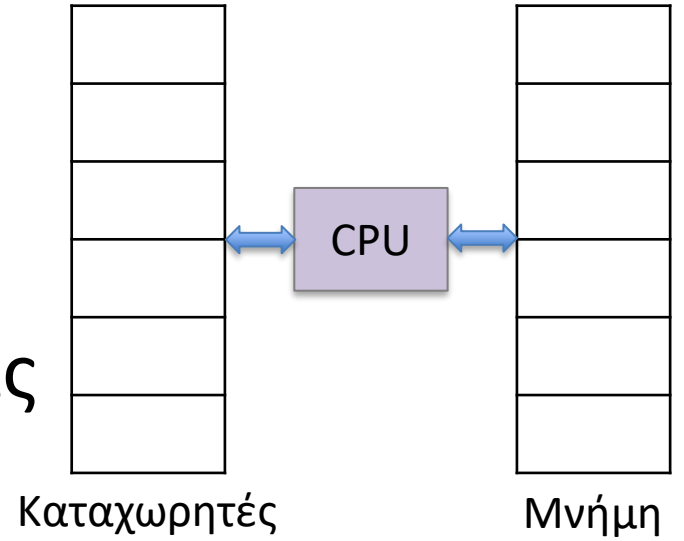
Άρα

$$A(n) = \sum_{i=1}^{n} p(I_i)t(I_i) = \sum_{i=1}^{n} \frac{2i+1}{n} = \frac{2}{n}\sum_{i=1}^{n} i + \sum_{i=1}^{n} \frac{1}{n} = \frac{2}{n}\frac{n(n+1)}{2} + 1 = n + 2$$

ΜΟΝΤΕΛΟ ΜΗΧΑΝΗΣ ΤΥΧΑΙΑΣ ΠΡΟΣΠΕΛΑΣΗΣ (RANDOM ACCESS MACHINE – RAM)

Το μοντέλο περιλαμβάνει:

1. Ένας επεξεργαστής χωρίς παραλληλισμό
2. Καταχωρητές και μία κύρια μνήμη απευθείας προσπέλασης
3. Δυνατότητα εκτέλεσης αριθμητικών πράξεων, λήψης αποφάσεων διακλάδωσης και εγγραφής / διαβάσματος μνήμης
4. Ανάλυση συναρτήσει του μεγέθους της εισόδου



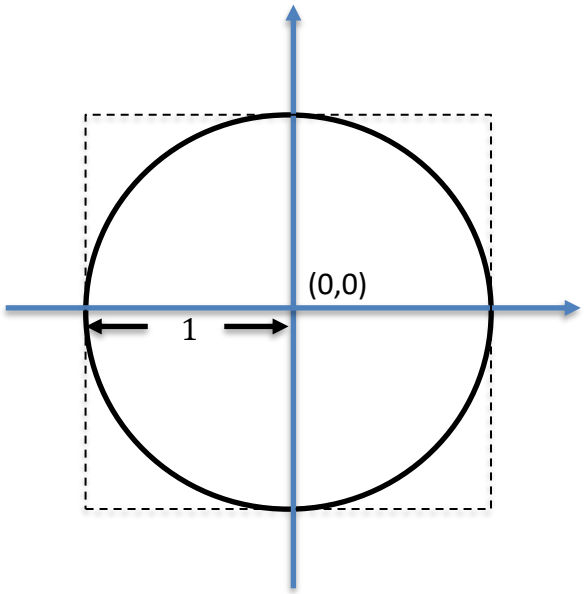
ΜΟΝΤΕΛΟ ΥΠΟΛΟΓΙΣΜΟΥ RAM

Στοιχειώδης εντολές

- Καταχώρηση τιμής σε μεταβλητή
- Εκτέλεση αριθμητικής πράξης
- Σύγκριση δύο αριθμών
- Δεικτοδότηση πίνακα
- Πρόσβαση στη διεύθυνση μνήμης ενός δείκτη
- Κλήση μιας μεθόδου/συνάρτησης
- Επιστροφή από μια μέθοδο/συνάρτηση

ΑΣΚΗΣΗ

- Αλγόριθμος υπολογισμού του π
  - Στοχαστικός αλγόριθμος (Monte Carlo τεχνική)
- Θεωρούμε ένα κύκλο με κέντρο το (0,0) και ακτίνα 1, καθώς επίσης και ένα εφαπτόμενο τετράγωνο
  - Δημιουργούμε <<τυχαία>> σημεία στο χωρίο που προσδιορίζεται από το τετράγωνο
  - Υπολογίζουμε το πλήθος των σημείων που βρίσκονται μέσα στον κύκλο



$$\frac{\text{εμβαδόν κύκλου}}{\text{εμβαδόν τετραγώνου}} = \frac{\pi r^2}{4r^2} = \frac{\pi}{4}$$

$$\pi = 4\frac{\text{εμβαδόν κύκλου}}{\text{εμβαδόν τετραγώνου}} \approx 4\frac{\text{πληθος σημειων στον κύκλο}}{\text{πληθος σημειων στο τετραγωνο}}$$

- Να γραφεί σε γλώσσα προγραμματισμού C, ο κώδικας για τον υπολογισμό του π με την παραπάνω διαδικασία.
  - Δημιουργία <<τυχαίων>> αριθμών με την rand() - **stdlib.h**
    - <<Τυχαίοι>> αριθμοί στο διάστημα [0,RAND_MAX]

Κωνσταντίνος Γιαννουτάκης

Επίκ. Καθηγητής

kgiannou@uom.edu.gr