Ιόνιο Πανεπιστήμιο – Τμήμα Πληροφορικής

Εισαγωγή στην Επιστήμη των Υπολογιστών

2024-25

Αλγόριθμοι και Δομές Δεδομένων (Ι)

(εισαγωγικές έννοιες)

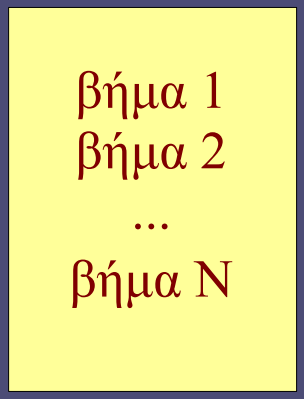
https://mixstef.github.io/courses/csintro/

Μ.Στεφανιδάκης

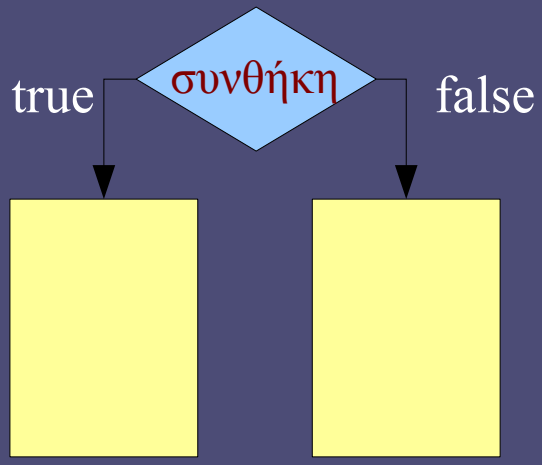
Τι είναι αλγόριθμος;

- «Βήμα προς βήμα μέθοδος για την επίλυση ενός προβλήματος»
  - Ανεξάρτητα από το υπολογιστικό σύστημα εκτέλεσης
- Τυπικός ορισμός:
  - Μια διαδικασία με πεπερασμένο αριθμό διατεταγμένων βημάτων
    - πιθανώς με επαναλήψεις και συνθήκες
  - η οποία επιλύει ένα συγκεκριμένο πρόβλημα
    - μετασχηματισμός εισόδων σε εξόδους
  - σε πεπερασμένο χρόνο

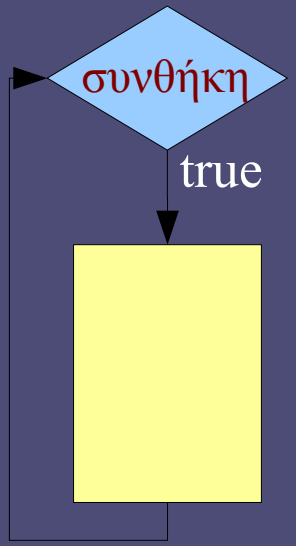
Βασικές αλγοριθμικές δομές



ακολουθία

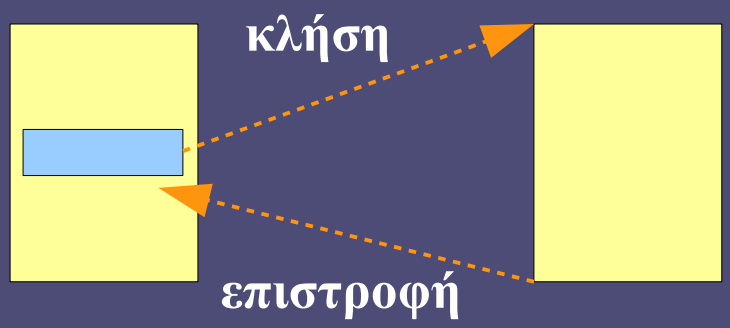


απόφαση υπό συνθήκη



επανάληψη

Υποπρογράμματα (υποαλγόριθμοι)

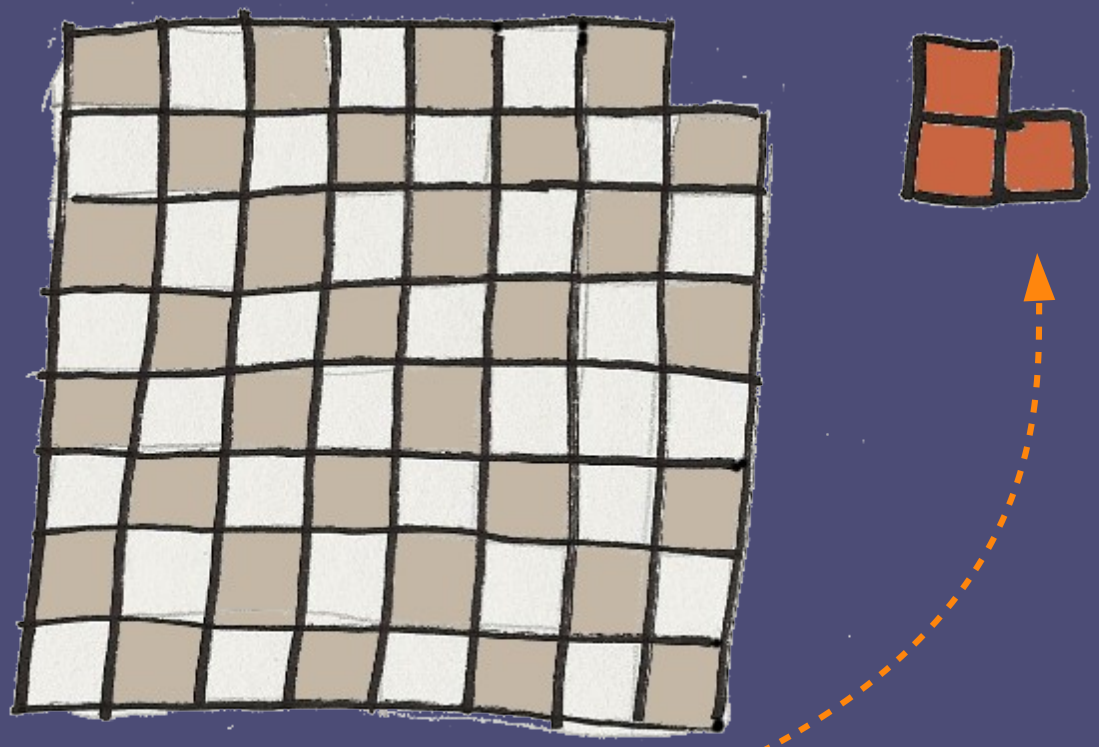


- Επαναχρησιμοποίηση
  - Κλήση συναρτήσεων
- Ευκολότερη κατανόηση
  - Σημαντικό όσο και η απόδοση

Αναδρομή: ένα κομψό εργαλείο

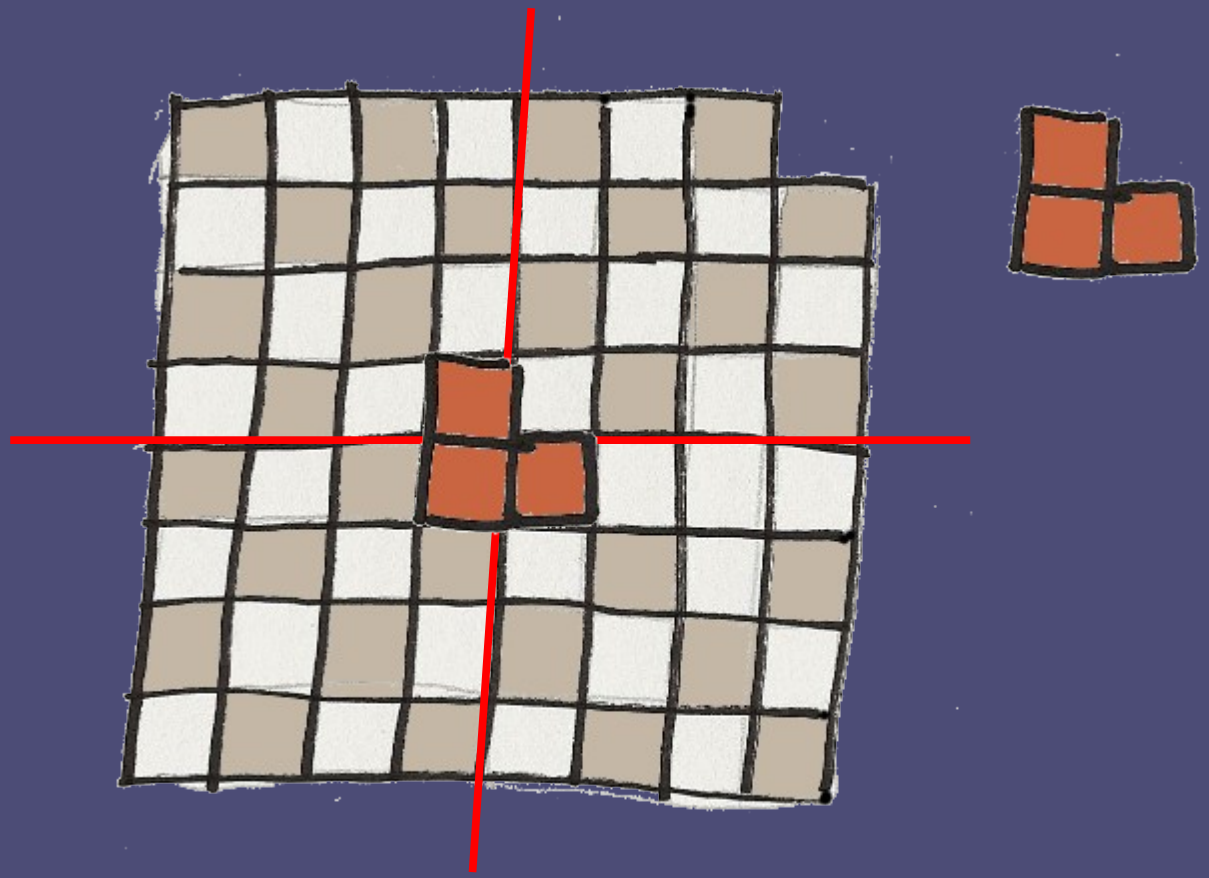
- Το γενικότερο πλαίσιο: μείωση ενός προβλήματος σε πολλά υποπροβλήματα
  - Και στη συνέχεια, συνδυασμός των μερικών λύσεων
- Αναδρομή
  - Μία συνάρτηση καλεί τον εαυτό της
    - δημιουργεί μικρότερα υποπροβλήματα
  - Μέχρι να φτάσει σε μια βασική περίπτωση
    - με άμεσο υπολογισμό του (μερικού) αποτελέσματος
  - Ακολουθούν επιστροφές από τις συναρτήσεις με συνδυασμό των μερικών αποτελεσμάτων
    - μέχρι το τελικό αποτέλεσμα

Αναδρομή: ένα παράδειγμα



- Πώς θα καλύψουμε (αν μπορούμε) τη σκακιέρα με σχήματα τύπου L;

Αναδρομή: ένα παράδειγμα



- Βλέπετε τη λύση; Είναι η αναδρομή!

Αναδρομή

- Αλγοριθμικά «κομψή» λύση αλλά
  - Πιθανή επιβάρυνση κατά την πολλαπλή κλήση των συναρτήσεων
  - Και μεγαλύτερη χρήση πόρων για διατήρηση προηγούμενων καταστάσεων
- Συχνά πρέπει να μετατρέψουμε την αναδρομή σε επανάληψη
  - Ευτυχώς αυτό είναι πάντα δυνατό
    - Ακόμα κι αν οδηγεί σε λιγότερο κομψή διατύπωση του αλγορίθμου
    - Με τη βοήθεια δομών δεδομένων που «μιμούνται» τη λειτουργία των αναδρομικών κλήσεων της συνάρτησης

Πολυπλοκότητα ενός αλγορίθμου

- Για την κατανόηση της απόδοσής του
  - Χρειαζόμαστε ένα βασικό μέγεθος
  - Που θα εστιάζει στη μεγάλη εικόνα
    - Την αύξηση του χρόνου εκτέλεσης του αλγορίθμου ανάλογα με την αύξηση του μεγέθους του προβλήματος (δηλ. των δεδομένων εισόδου)
    - Ανεξάρτητα από σταθερούς παράγοντες όπως η ταχύτητα της γλώσσας προγραμματισμού ή η ταχύτητα του υλικού
  - Προσοχή: δεν ενδιαφερόμαστε για απόλυτους χρόνους
    - Αντιθέτως, υπολογίζουμε πόσες φορές εκτελείται μια βασική λειτουργία
    - Σε σχέση με το μέγεθος των δεδομένων εισόδου

Ο ασυμπτωτικός συμβολισμός O(...)

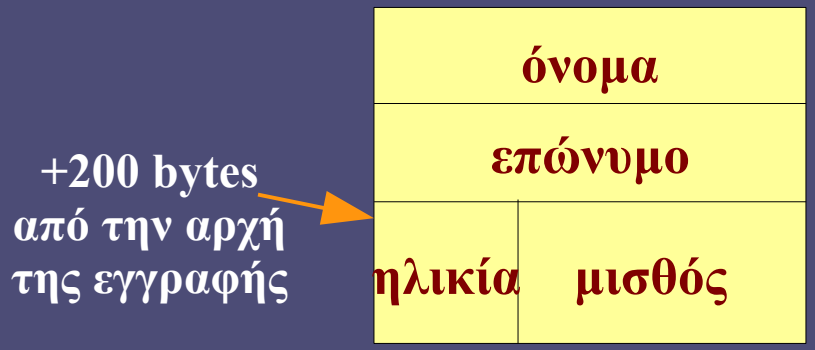
- Big O notation
  - Έκφραση της κλάσης πολυπλοκότητας του χρόνου εκτέλεσης ενός αλγορίθμου σε σχέση με το μέγεθος *n* των δεδομένων εισόδου
    - Αφαιρώντας σταθερούς παράγοντες
  - Τυπικά
    - O(g(n)), είναι ένα σύνολο συναρτήσεων
    - f(n) ανήκει στο O(g(n)) εάν υπάρχει n_0 και θετική σταθερά c έτσι ώστε $f(n) \leq cg(n)$ για κάθε $n \geq n_0$
  - Πρακτικά
    - O(g(n)), είναι οι συναρτήσεις που αυξάνονται με αργότερο ρυθμό από το g(n) – άρα είναι αποδοτικότερες
  - Παράδειγμα
    - Τα n^2 , $3n^2$, $85.8n^2+3.44$ ανήκουν όλα στο $O(n^2)$

Ο ασυμπτωτικός συμβολισμός O(...)

- Αν για παράδειγμα η πολυπλοκότητα ενός αλγορίθμου είναι $O(n^2)$
  - Αυτό σημαίνει ότι για μέγεθος δεδομένων n θα εκτελεστεί αριθμός λειτουργιών της τάξης του n^2
  - Και ότι ένας αλγόριθμος O(n) είναι αποδοτικότερος
  - Απόδοση ανεξάρτητη από μέγεθος δεδομένων
    - O(1) – η ιδανική περίπτωση!
  - Πολυωνυμικά προβλήματα
    - O(logn), O(n), $O(n^2)$, $O(n^k)$
    - Συνήθως επιλύσιμα
  - Μη πολυωνυμικά προβλήματα
    - $O(k^n)$ ή O(n!)
    - Γενικά, μη επιλύσιμα

Πίνακες (arrays)

- Ακολουθία όμοιων εγγραφών
  - Σε συνεχόμενες θέσεις μνήμης
- Η "εγγραφή" (record ή structure)
  - Μία αυτοτελής ομάδα δεδομένων
  - Με συγκεκριμένη μορφή αποθήκευσης
    - Κάθε μέλος της ομάδας βρίσκεται σε καθορισμένη θέση (διεύθυνση) μέσα στην εγγραφή
  - Ως "κόμβος" δεδομένων σε πιο σύνθετες δομές δεδομένων

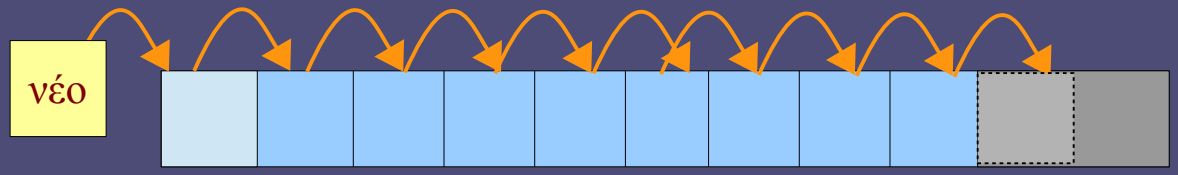


Πίνακες (arrays)

- Πίνακας μιας διάστασης
  - Ο υπολογισμός της θέσης (διεύθυνσης) του i−οστού στοιχείου είναι άμεσος
    - Αρχή πίνακα + i * μέγεθος εγγραφής
    - Θυμηθείτε: το i ξεκινά από το 0!
- Υλοποίηση πινάκων πιο πολλών διαστάσεων
  - Π.χ. για πίνακα δύο διαστάσεων, ως συνεχόμενες γραμμές στη μνήμη
    - Πώς υπολογίζεται στην περίπτωση αυτή η διεύθυνση του στοιχείου $_{i, j}$;

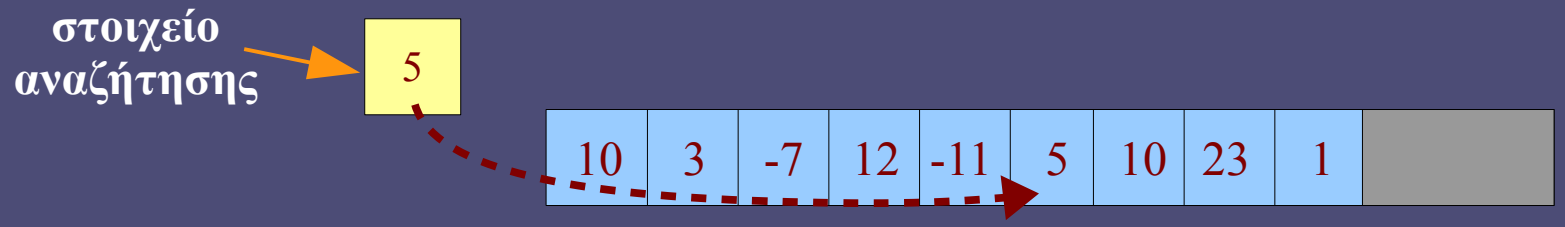
Λειτουργίες σε πίνακες

- Λήψη στοιχείου i
  - Σταθερή πολυπλοκότητα O(1)
    - Εφόσον ο υπολογισμός της θέσης του κάθε στοιχείου είναι άμεσος (αρχή πίνακα + i * μέγεθος εγγραφής)
- Προσθήκη στοιχείου (στο τέλος)
  - O(1) – θεωρώντας ότι υπάρχει χώρος
    - Άμεσος υπολογισμός θέσης νέου στοιχείου
- Εισαγωγή στοιχείου (π.χ. στην αρχή)
  - O(n) – θα πρέπει να μετατοπιστούν τα ήδη υπάρχοντα στοιχεία



Αναζήτηση σε πίνακα

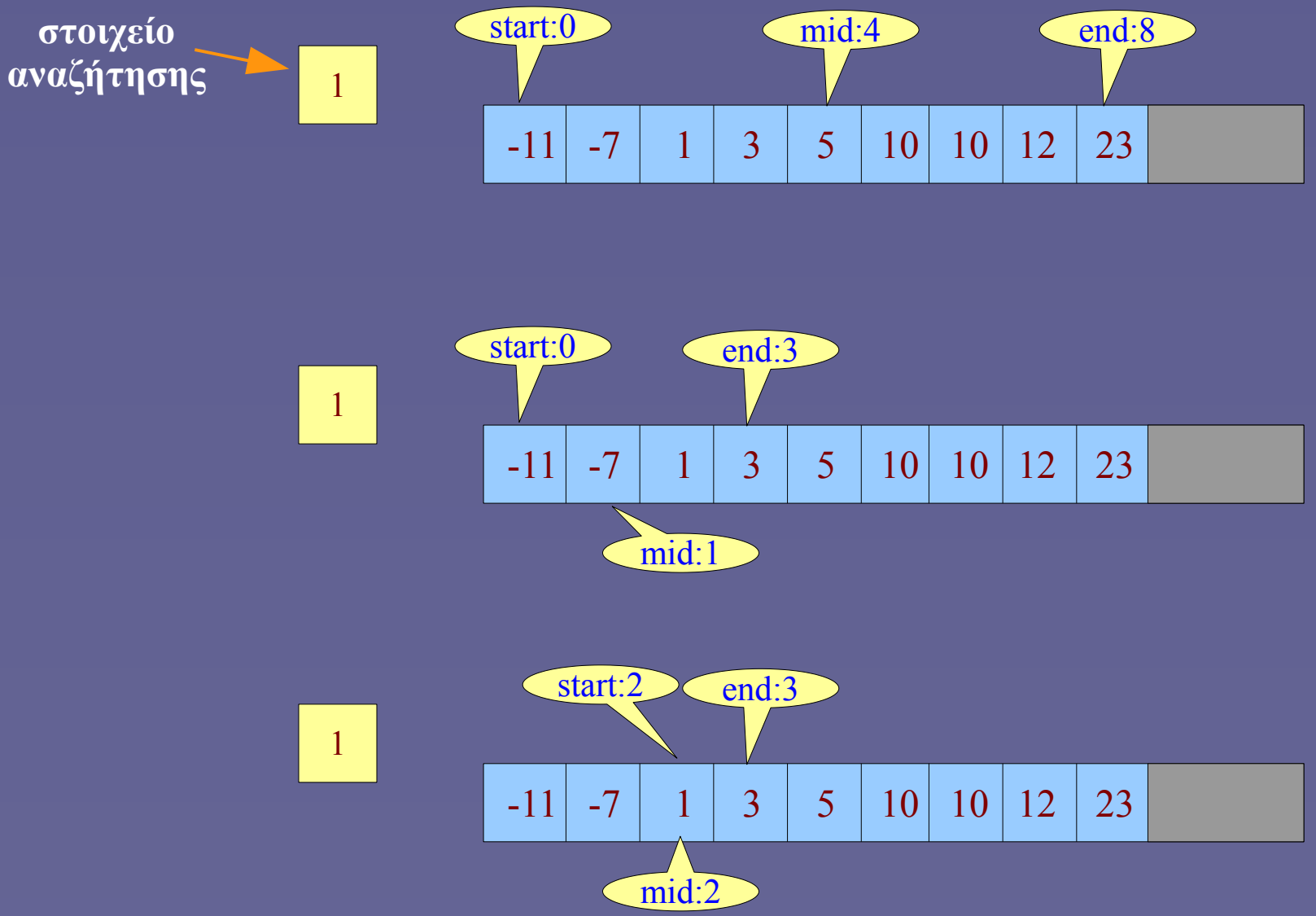
- Αναζήτηση στοιχείου με κάποιες ιδιότητες
  - Τιμές-κλειδιά
  - Εύρεση θέσης (i) στοιχείου – αν υπάρχει
- Εάν τα στοιχεία δεν είναι ταξινομημένα...
  - Με βάση τις αναζητούμενες τιμές-κλειδιά
- ...τότε πρέπει να διασχίσουμε τον πίνακα σειριακά
  - από τη μία άκρη προς την άλλη (διάσχιση πίνακα)
  - συγκρίνοντας κάθε στοιχείο που συναντάμε
  - O(n) – OK για μικρούς πίνακες (και λίγες αναζητήσεις)



Δυαδική Αναζήτηση (binary search)

- Για να εφαρμόσουμε δυαδική αναζήτηση τα στοιχεία του πίνακα πρέπει να είναι ταξινομημένα
  - Με βάση τις αναζητούμενες τιμές-κλειδιά
- Η αναζήτηση γίνεται πολύ πιο αποδοτική από τη σειριακή
  - Χωρίζοντας τον πίνακα στη μέση
  - Και ψάχνοντας μόνο ένα από τα δύο μέρη
    - Μικρότερο υποπρόβλημα
  - Επαναλαμβάνουμε τη διαδικασία μέχρι την εύρεση
    - ή μέχρι να φανεί ότι δεν υπάρχει αυτό που ψάχνουμε
  - $O(\log_2 n)$– η μόνη βιώσιμη λύση για μεγάλους πίνακες

Δυαδική Αναζήτηση (binary search)



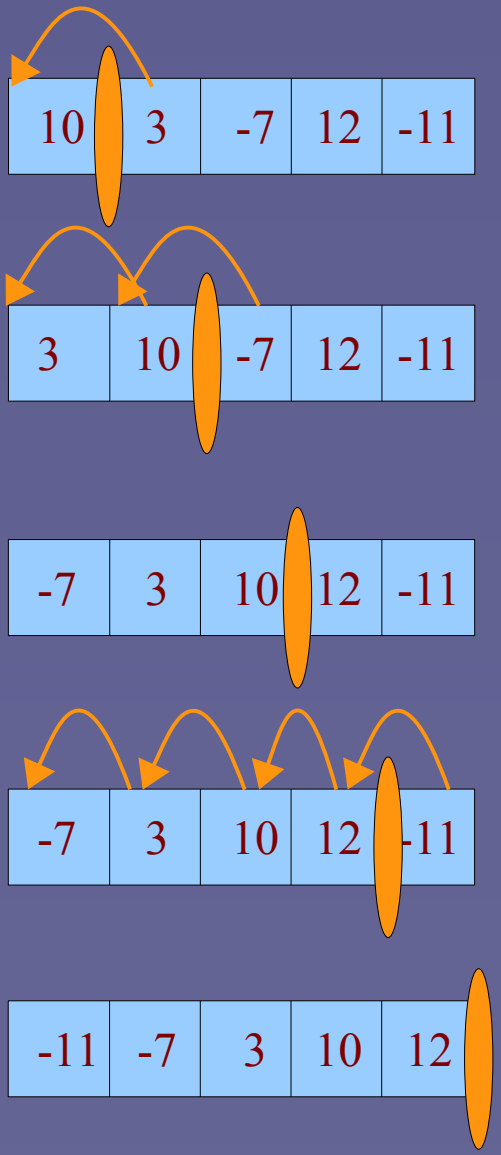
Ταξινόμηση

- Συνήθης λειτουργία σε πίνακες
  - Με βάση τιμές κλειδιών κάθε στοιχείου του πίνακα
  - Ως βάση για την εφαρμογή άλλων αλγορίθμων
    - Όπως η δυαδική αναζήτηση που είδαμε προηγουμένως
  - Δεν υπάρχει μοναδικός αλγόριθμος ταξινόμησης με την βέλτιστη πολυπλοκότητα χρόνου εκτέλεσης
    - Εξαρτάται από την αρχική διάταξη των δεδομένων στον πίνακα
    - Συνήθως αναφέρεται η χειρότερη, η καλύτερη και η μέση (κατά μέσο όρο) πολυπλοκότητα κάθε αλγορίθμου
- Στη συνέχεια θα δούμε ορισμένους μόνο από το σύνολο των αλγορίθμων ταξινόμησης

Ταξινόμηση παρεμβολής (insertion sort)

- Ο πίνακας χωρίζεται σε δύο μέρη
  - Το ταξινομημένο μέρος
    - Αρχικά περιέχει το πρώτο στοιχείο του πίνακα
  - Και το αταξινόμητο
    - Τα υπόλοιπα στοιχεία
- Κάθε στοιχείο του αταξινόμητου
  - Προωθείται στη σωστή θέση μέσα στο ταξινομημένο μέρος
    - Το ταξινομημένο μέρος σταδιακά μεγαλώνει
- Πολυπλοκότητα
  - $O(n^2)$ στη χειρότερη και τη μέση περίπτωση
  - $O(n)$ στην καλύτερη (ήδη ταξινομημένα δεδομένα)

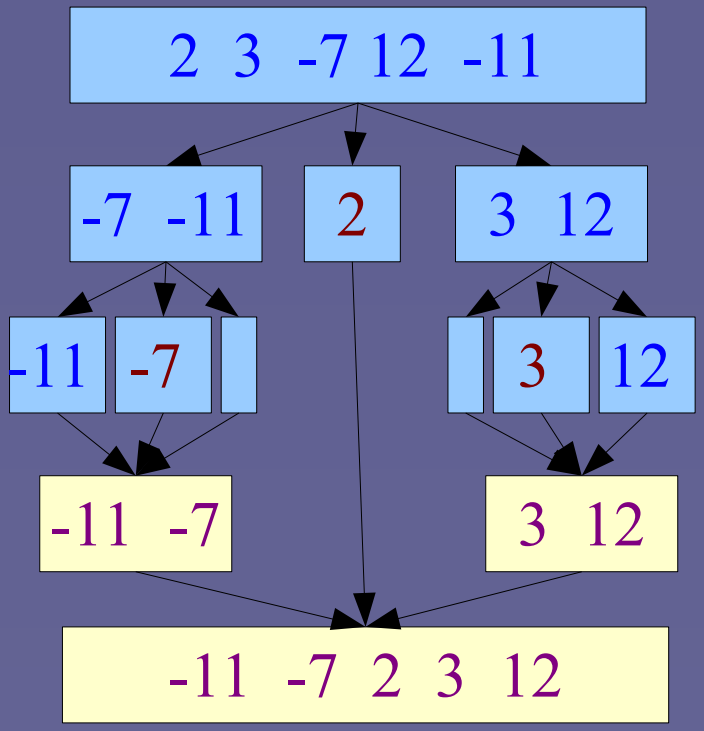
Ταξινόμηση παρεμβολής (insertion sort)



Quicksort

- Ένας από τους γνωστότερους αλγορίθμους ταξινόμησης
  - Ακολουθεί την τακτική της διαμέρισης του πίνακα σε δύο μέρη
    - Μικρότερα και μεγαλύτερα στοιχεία από (τυπικά) το πρώτο στοιχείο του πίνακα
  - Και στη συνέχεια ταξινομεί αναδρομικά τους δύο υποπίνακες
  - Στο τέλος συνενώνει τους πίνακες
    - Μικρότερα + στοιχείο διαμέρισης + μεγαλύτερα
- Πολυπλοκότητα
  - $O(n^2)$ στη χειρότερη περίπτωση (ήδη ταξινομημένα)
  - $O(n\log_2 n)$ στην καλύτερη και μέση περίπτωση

Quicksort



Διασυνδεδεμένες λίστες

- Για αποθήκευση ακολουθίας στοιχείων
  - Εναλλακτική λύση αντί της χρήσης πινάκων
  - Αποτελείται από κόμβους
    - Στοιχεία (εγγραφές δεδομένων)
  - Κάθε κόμβος διασυνδέεται με τον επόμενο
    - Μονή ή διπλή φορά διασύνδεσης
  - Η εισαγωγή είναι O(1)
    - Αλλαγή μόνο της διασύνδεσης των κόμβων
    - Θα πρέπει όμως να ξέρουμε το σημείο εισαγωγής
  - Αλλά η προσπέλαση τυχαίου στοιχείου γίνεται τώρα O(n)
    - Θα πρέπει να διασχίσουμε τη λίστα από κάποια άκρη της μέχρι να φτάσουμε στο στοιχείο που θέλουμε