Δομές Δεδομένων

Ταξινόμηση

Δημήτρης Μιχαήλ

Τμήμα Πληροφορικής και Τηλεματικής
Χαροκόπειο Πανεπιστήμιο

Το πρόβλημα

Είσοδος

n αντικείμενα $\langle a_1, a_2, \ldots, a_n \rangle$ με κλειδιά (συνήθως σε ένα πίνακα, ή λίστα, κ.τ.λ)

Έξοδος

τα n αντικείμενα ταξινομημένα ως προς τα κλειδιά τους

Παράδειγμα

Είσοδος

100, 4, 200, 8, 11, 12, 20, 100, 4

Έξοδος

200, 100, 100, 20, 12, 11, 8, 4, 4

Χρήση Ταξινόμησης

Η ταξινόμηση είναι από τα πιο θεμελιώδη προβλήματα στην επιστήμη των υπολογιστών.

Άμεση Εφαρμογή

π.χ σε μια εφαρμογή e-banking θέλουμε να δούμε τις συναλλαγές μας ταξινομημένες με την ημερομηνία

Βάση για άλλους αλγορίθμους

Οι τεχνικές που έχουν προκύψει με την μελέτη την ταξινόμησης μας βοηθούν να λύσουμε και άλλα προβλήματα.

Γενικά η ταξινόμηση εμφανίζεται συνέχεια στην επιστήμη των υπολογιστών.

Μοντέλο Υπολογισμού

Μας ενδιαφέρει να μετρήσουμε τον χρόνο που παίρνει να ταξινομήσουμε n αριθμούς.

Πρέπει όμως να καθορίσουμε τι μετράμε ακριβώς...

Μοντέλα Υπολογισμού

Εσωτερική ταξινόμηση

Γίνεται στην μνήμη του υπολογιστή

Μοντέλα Υπολογισμού

Εσωτερική ταξινόμηση

Γίνεται στην μνήμη του υπολογιστή

Εξωτερική ταξινόμηση

Οι αριθμοί δεν χωρούν όλοι στην μνήμη ταυτόχρονα, οπότε χρησιμοποιείται και πιο αργή μνήμη, π.χ SSD ή σκληρός δίσκος

Μοντέλο Σύγκρισης

Θεωρούμε πως ο χρόνος εκτέλεσης είναι ανάλογος με τον αριθμό των συγκρίσεων που θα γίνουν ώστε να ταξινομήσουμε n αριθμούς. Με άλλα λόγια μετράμε **μόνο** τον αριθμό των συγκρίσεων.

Μοντέλα Υπολογισμού

Μοντέλο Σύγκρισης

Θεωρούμε πως ο χρόνος εκτέλεσης είναι ανάλογος με τον αριθμό των συγκρίσεων που θα γίνουν ώστε να ταξινομήσουμε n αριθμούς. Με άλλα λόγια μετράμε **μόνο** τον αριθμό των συγκρίσεων.

Άλλα μοντέλα

Στους σύγχρονους υπολογιστές όμως μπορούμε να ταξινομήσουμε και με άλλους τρόπους, χωρίς καθόλου συγκρίσεις.

Διπλά Κλειδιά και Ευσταθής Ταξινόμηση

Πολλές φορές έχουμε εγγραφές με διπλά κλειδιά, πχ

Όνομα	Αριθμός Μητρώου	Βαθμός
Κώστας	1	8
Ελένη	2	9
Νίκος	3	4.5
Γιάννης	4	8

Διπλά Κλειδιά και Ευσταθής Ταξινόμηση

Πολλές φορές έχουμε εγγραφές με διπλά κλειδιά, πχ

Όνομα	Αριθμός Μητρώου	Βαθμός
Κώστας	1	8
Ελένη	2	9
Νίκος	3	4.5
Γιάννης	4	8

Ορισμός

Μια μέθοδος ταξινόμησης ονομάζεται **ευσταθής (stable)** αν διατηρεί τη σχετική σειρά των στοιχείων ενός αρχείου με διπλά κλειδιά.

Διπλά Κλειδιά και Ευσταθής Ταξινόμηση

Όνομα	Αριθμός Μητρώου	Βαθμός
Κώστας	1	8
Ελένη	2	9
Νίκος	3	4.5
Γιάννης	4	8

Νίκος	3	4.5	
Κώστας	1	8	**ευσταθής**
Γιάννης	4	8	
Ελένη	2	9	

Νίκος	3	4.5	
Γιάννης	4	8	**μη-ευσταθής**
Κώστας	1	8	
Ελένη	2	9	

Ταξινόμηση σε Θέση

In-Place

Μια ταξινόμηση ονομάζεται ταξινόμηση **σε θέση** εάν δεν χρησιμοποιεί βοηθητικό χώρο για την ταξινόμηση, αλλά απλά χρησιμοποιεί τον χώρο που καταλαμβάνει η είσοδος (συν $\mathcal{O}(1)$ θέσης μνήμης).

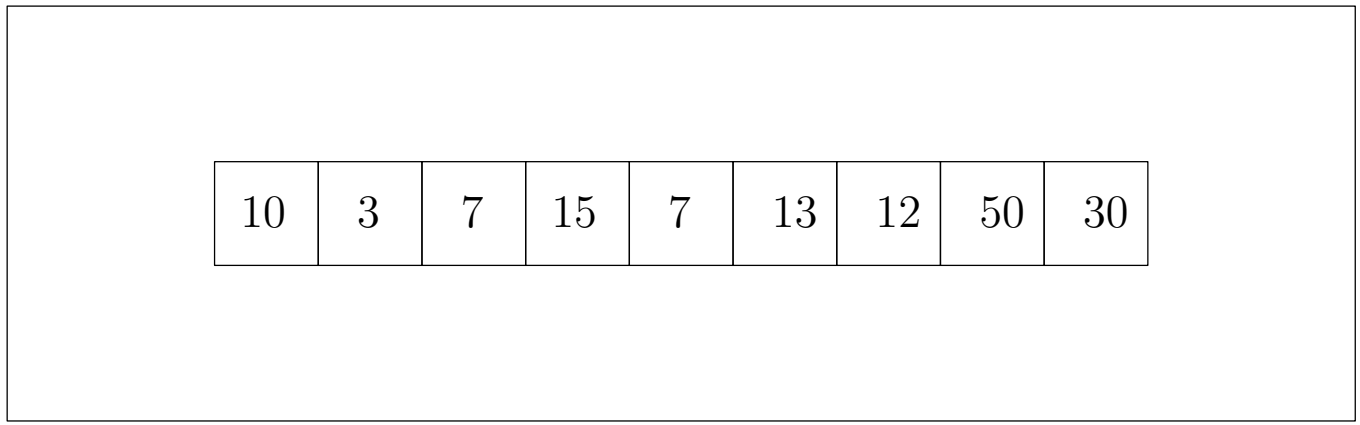
Ταξινόμηση Επιλογής
Selection Sort

Είσοδος
Πίνακας $a[0..n-1]$ με n στοιχεία

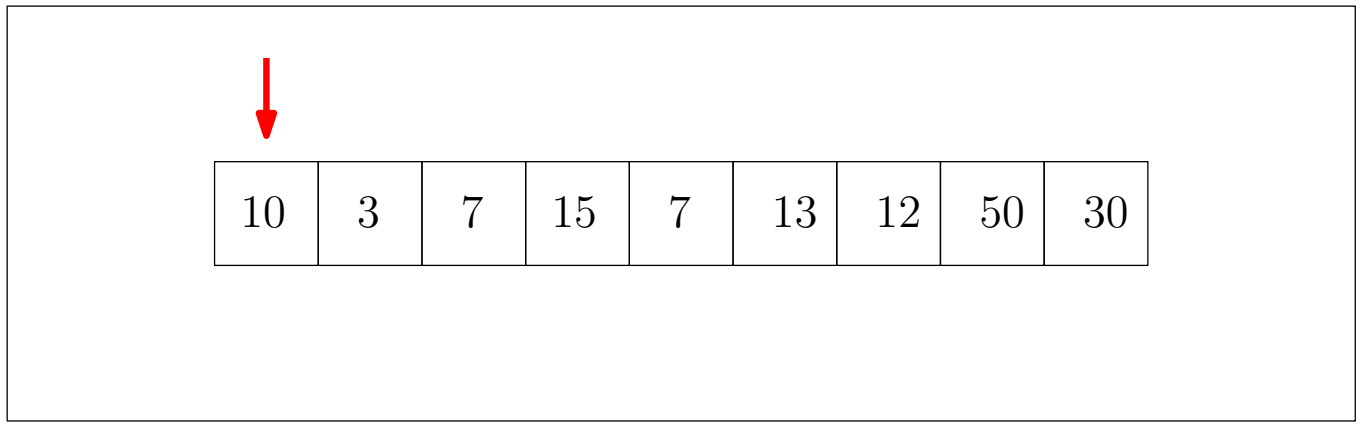
Αλγόριθμος

- βρες το μικρότερο στοιχείο του πίνακα και κάνε αντιμετάθεση με το πρώτο
- βρες το δεύτερο μικρότερο στοιχείο του πίνακα και κάνε αντιμετάθεση με το δεύτερο
- $\vdots$

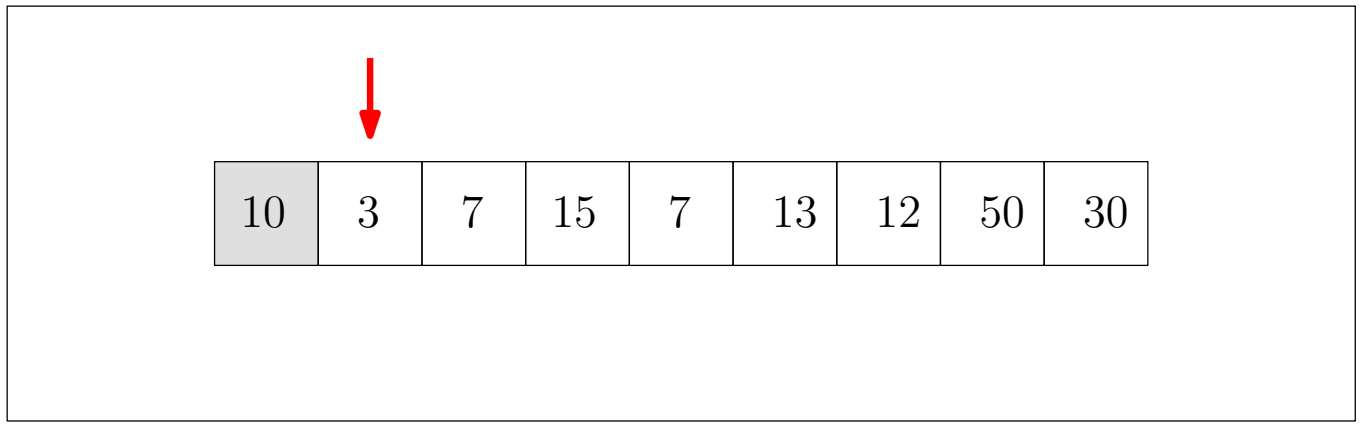
Ταξινόμηση Επιλογής
Selection Sort



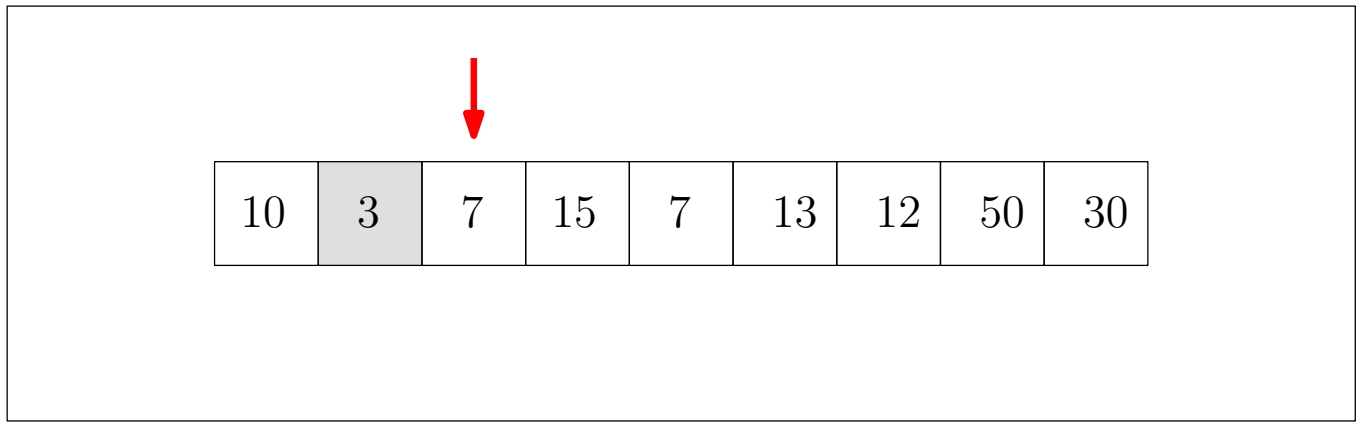
Ταξινόμηση Επιλογής
Selection Sort



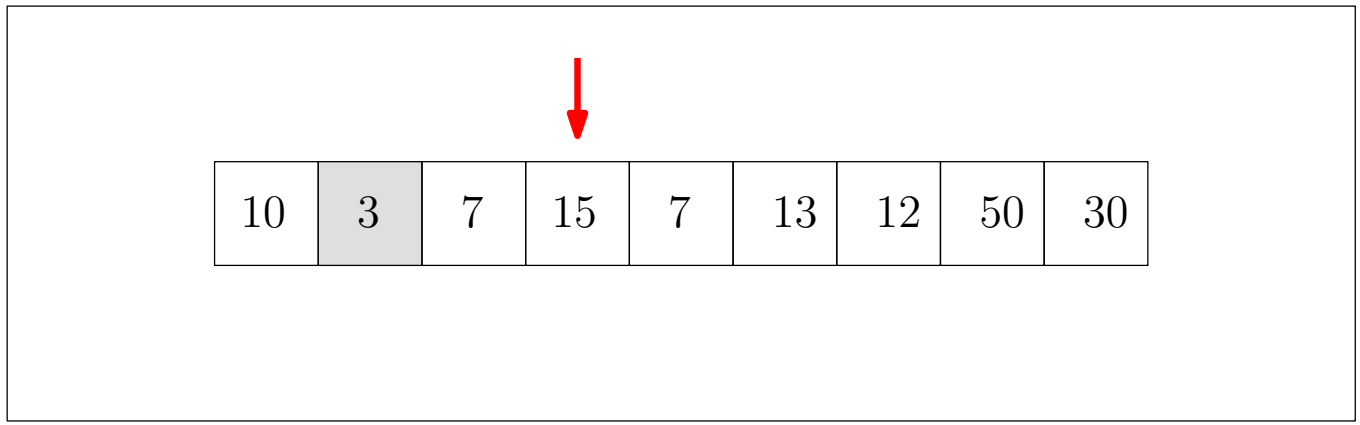
Ταξινόμηση Επιλογής
Selection Sort



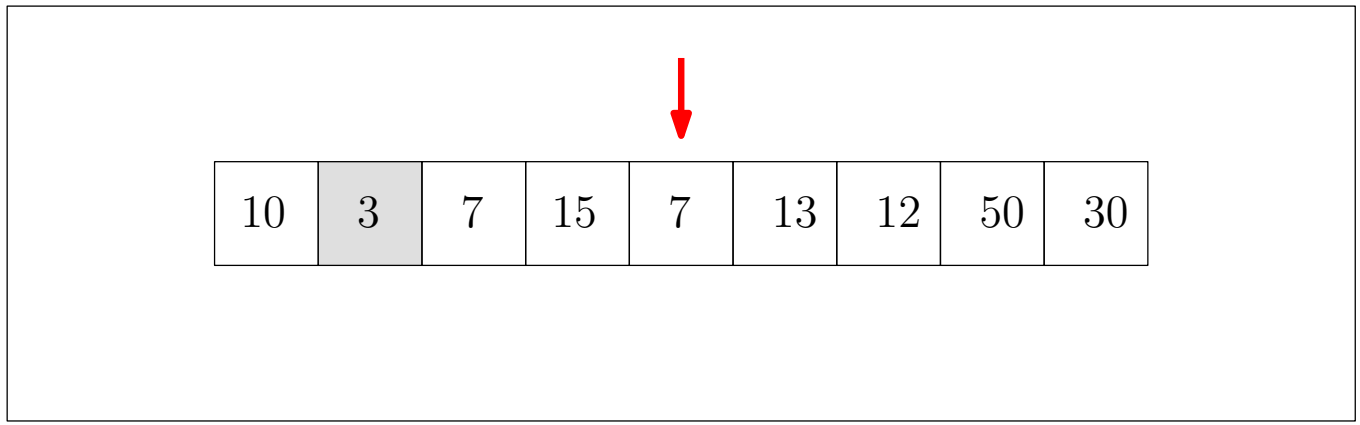
Ταξινόμηση Επιλογής
Selection Sort



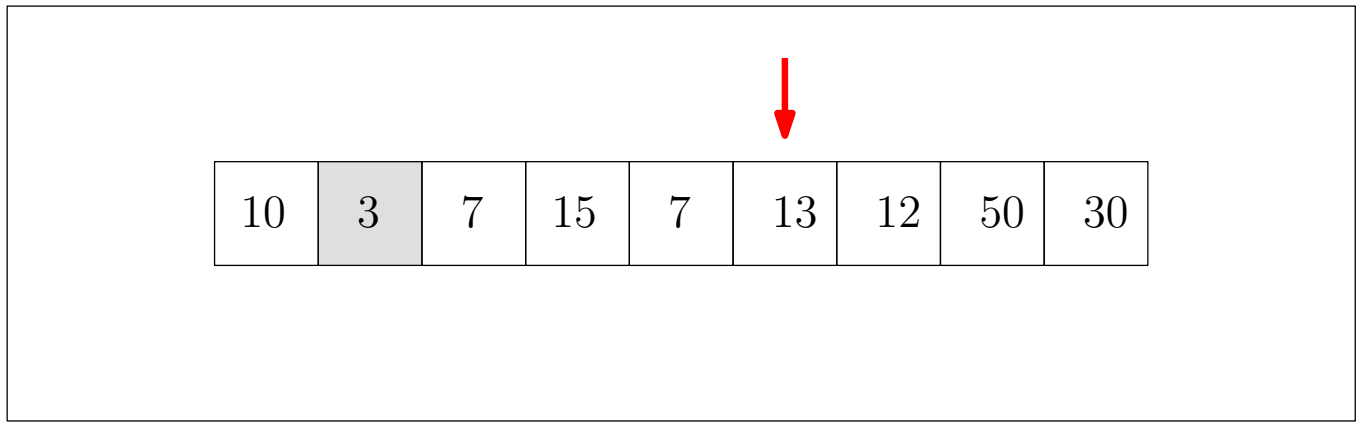
Ταξινόμηση Επιλογής
Selection Sort



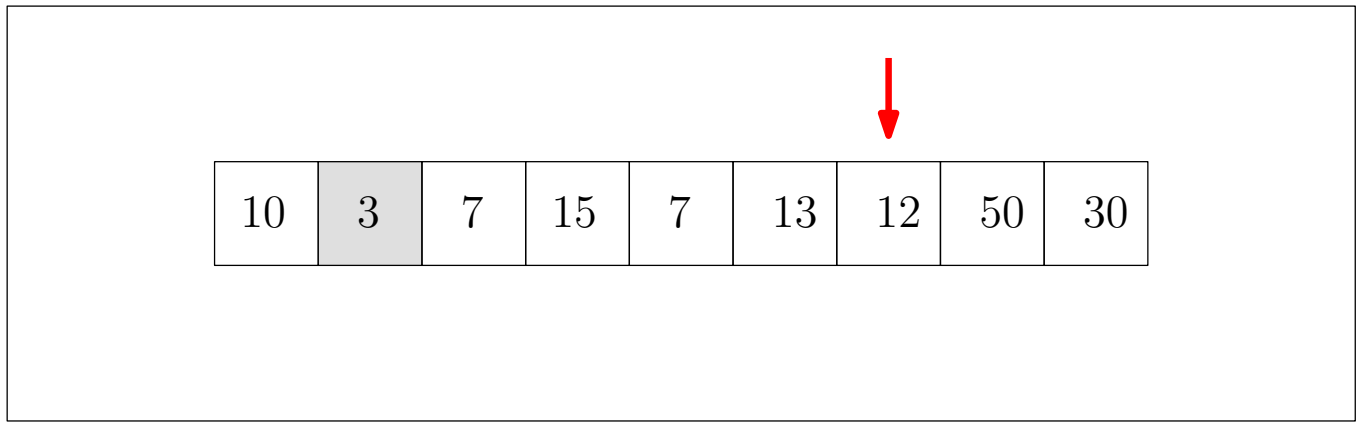
Ταξινόμηση Επιλογής
Selection Sort



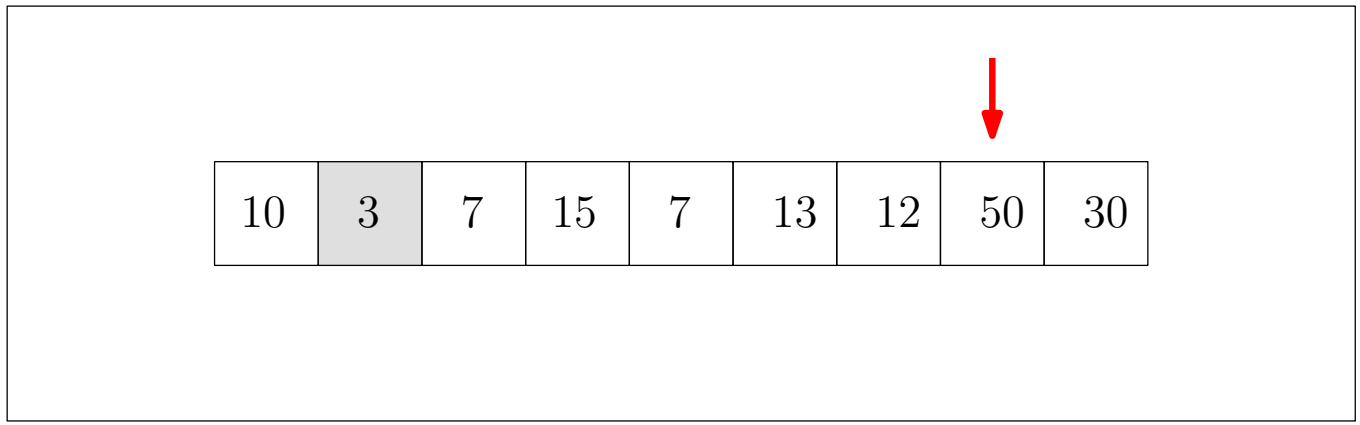
Ταξινόμηση Επιλογής
Selection Sort



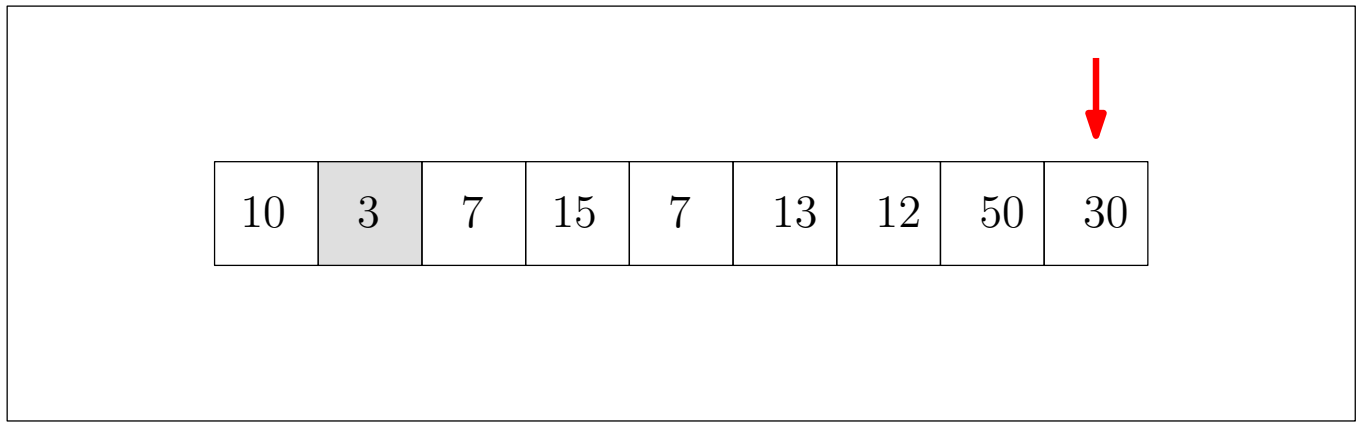
Ταξινόμηση Επιλογής
Selection Sort



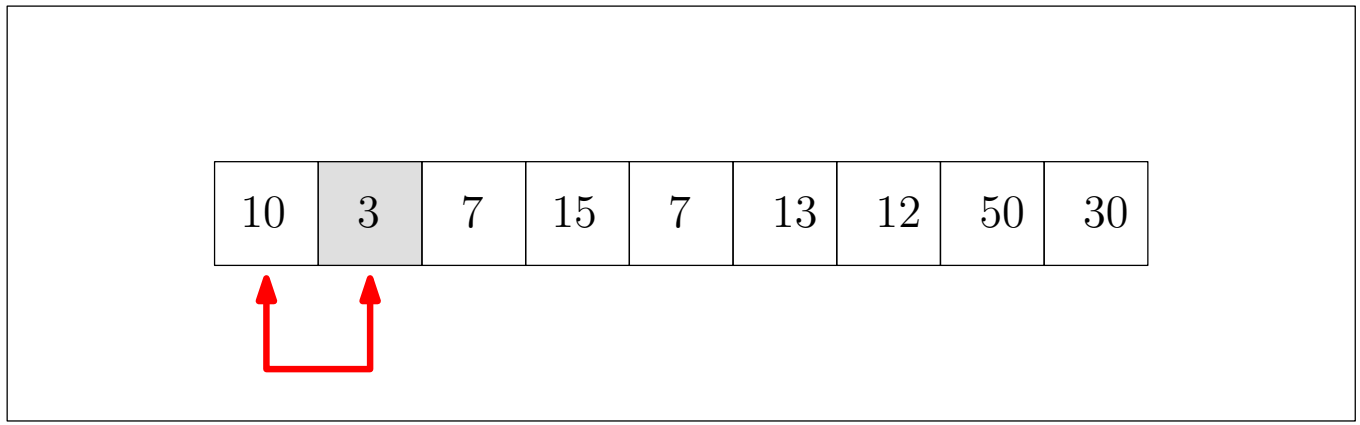
Ταξινόμηση Επιλογής
Selection Sort



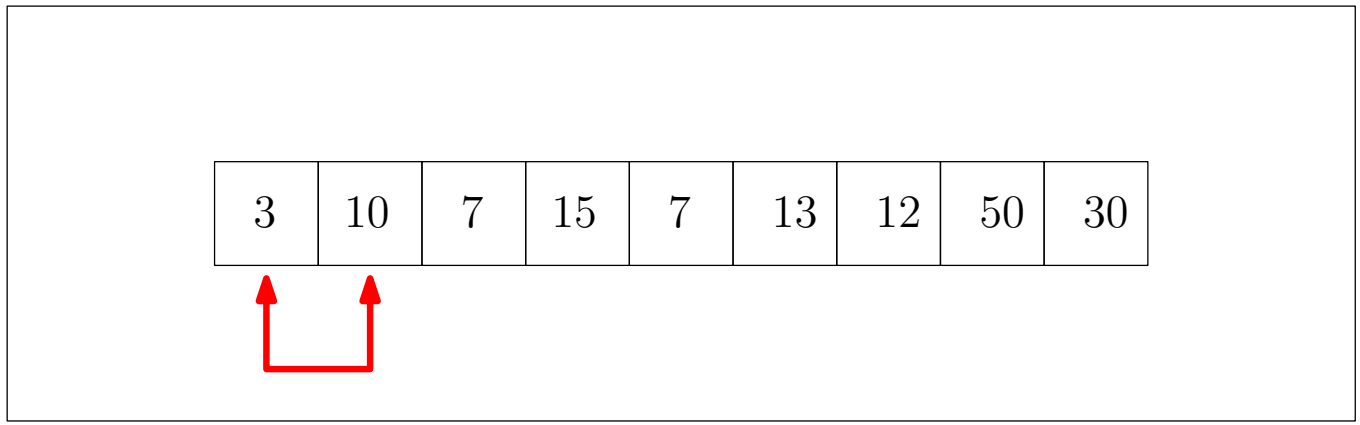
Ταξινόμηση Επιλογής
Selection Sort



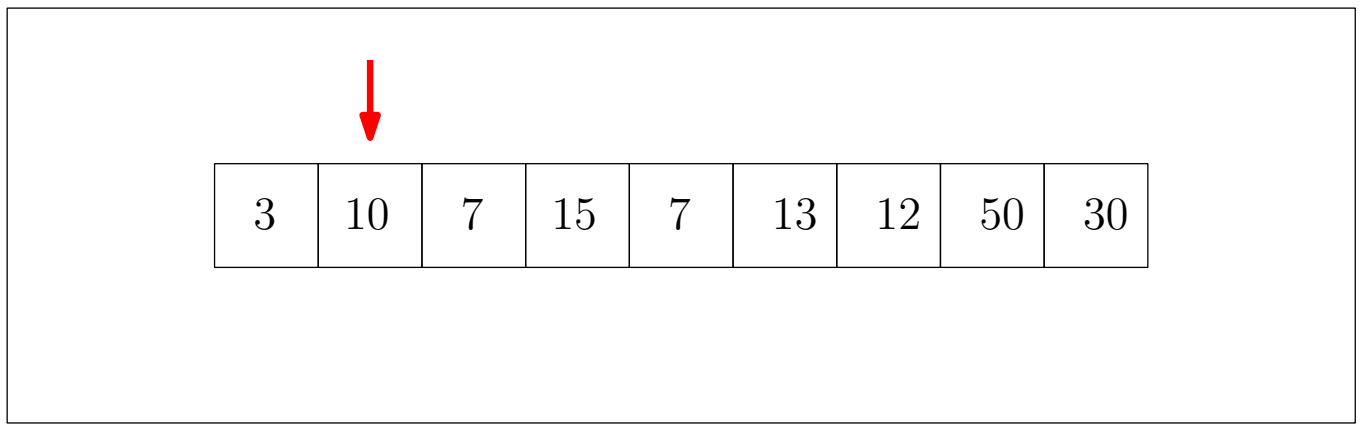
Ταξινόμηση Επιλογής
Selection Sort



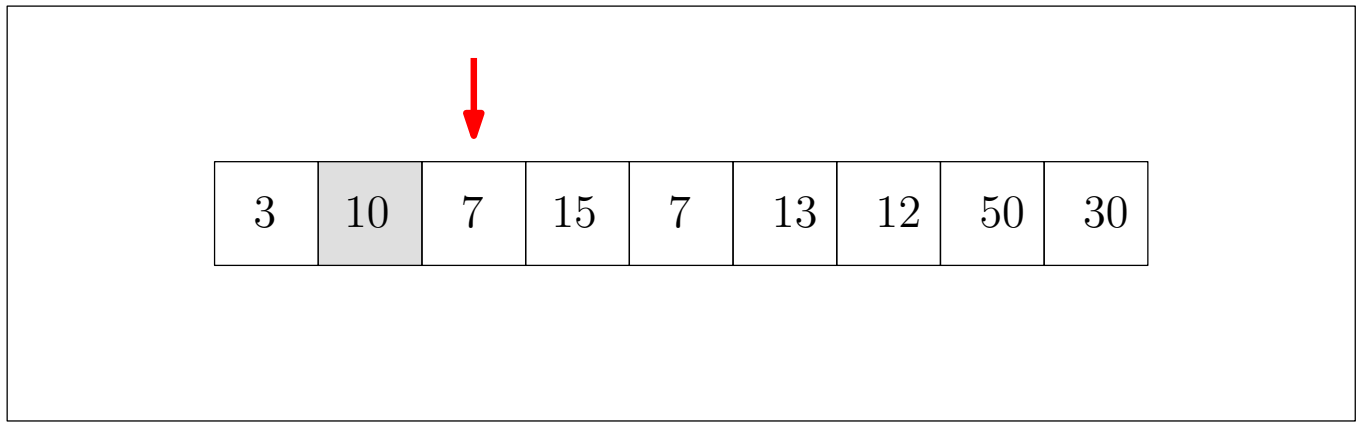
Ταξινόμηση Επιλογής
Selection Sort



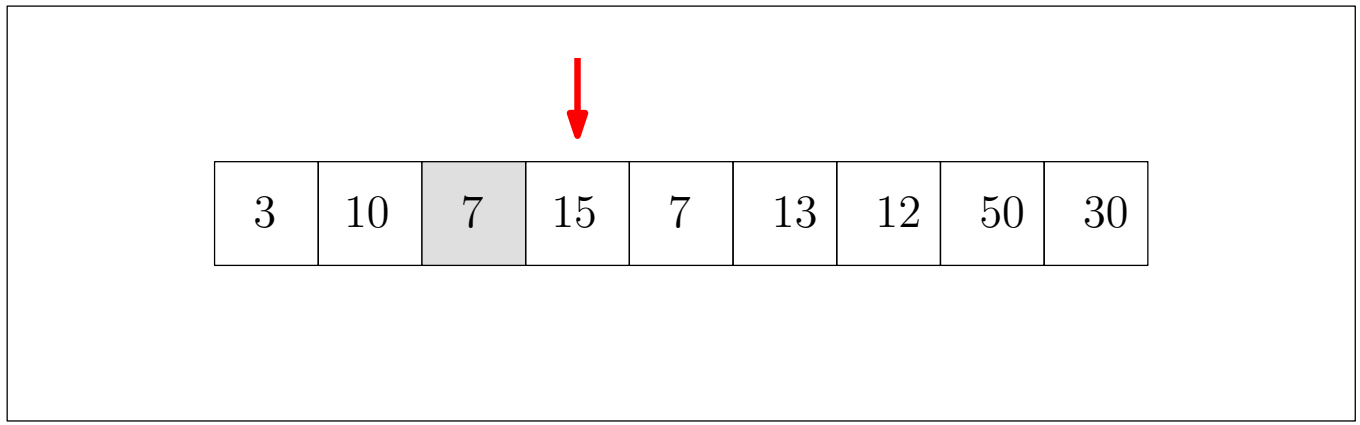
Ταξινόμηση Επιλογής
Selection Sort



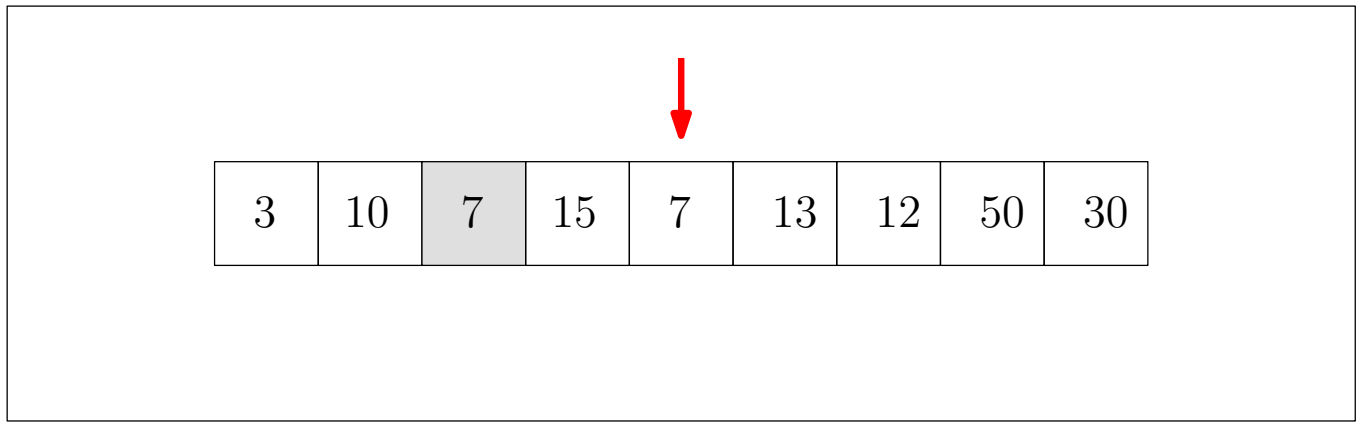
Ταξινόμηση Επιλογής
Selection Sort



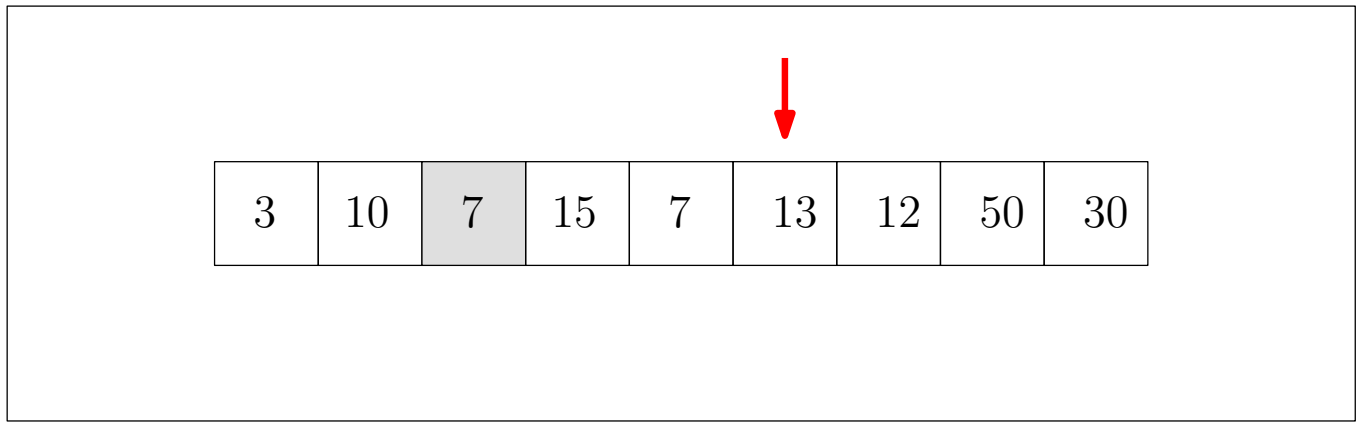
Ταξινόμηση Επιλογής
Selection Sort



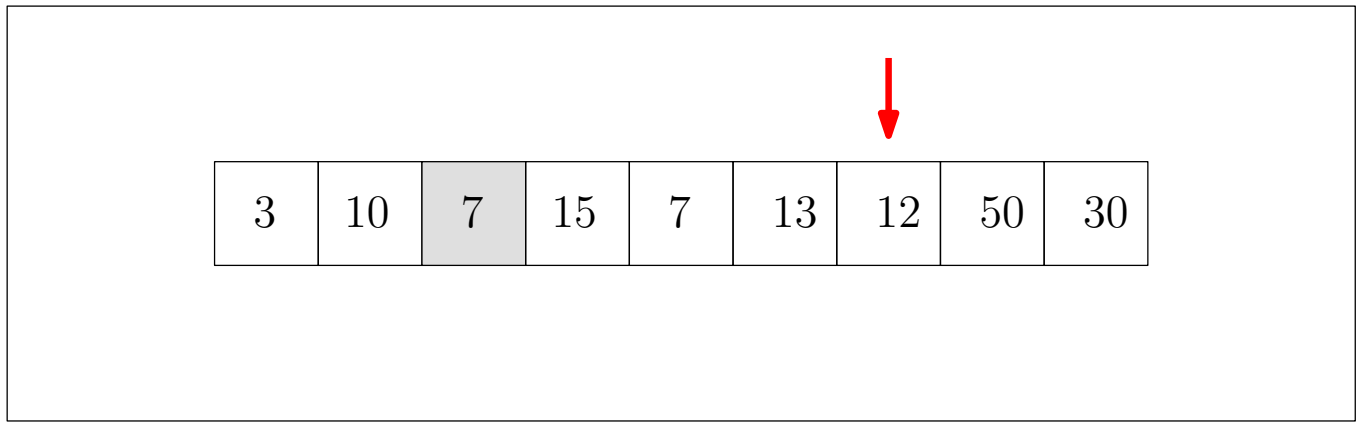
Ταξινόμηση Επιλογής
Selection Sort



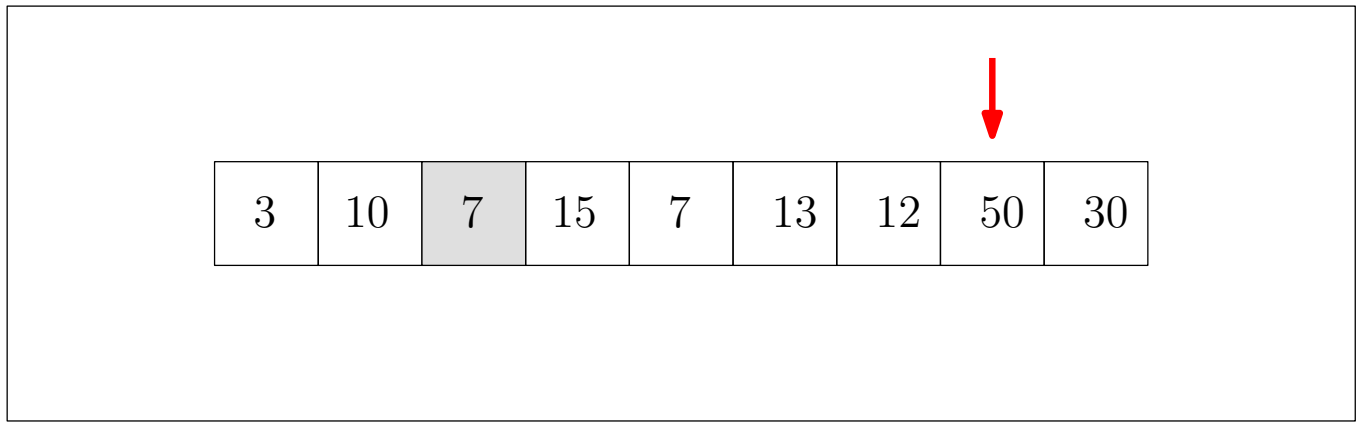
Ταξινόμηση Επιλογής
Selection Sort



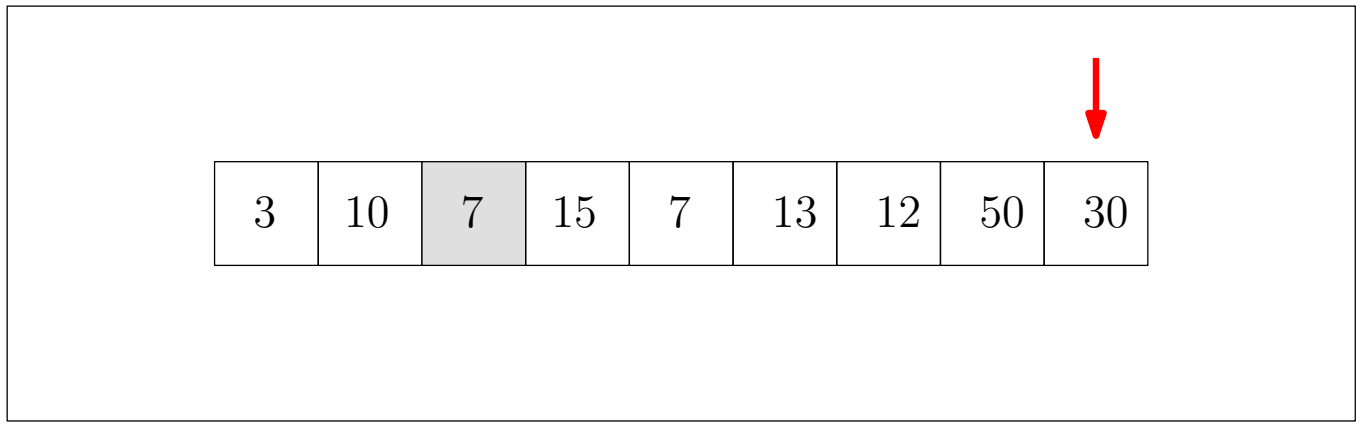
Ταξινόμηση Επιλογής
Selection Sort



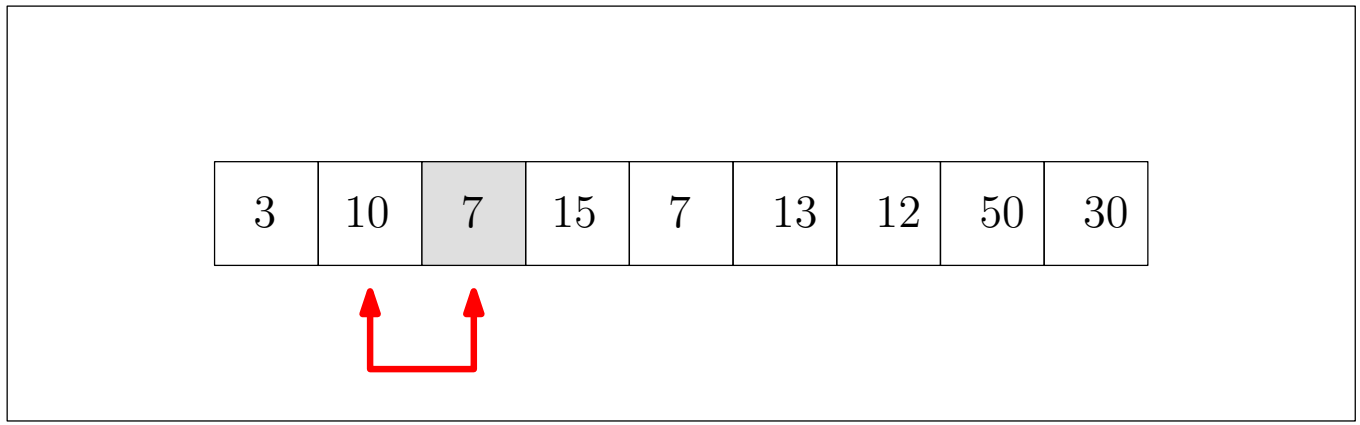
Ταξινόμηση Επιλογής
Selection Sort



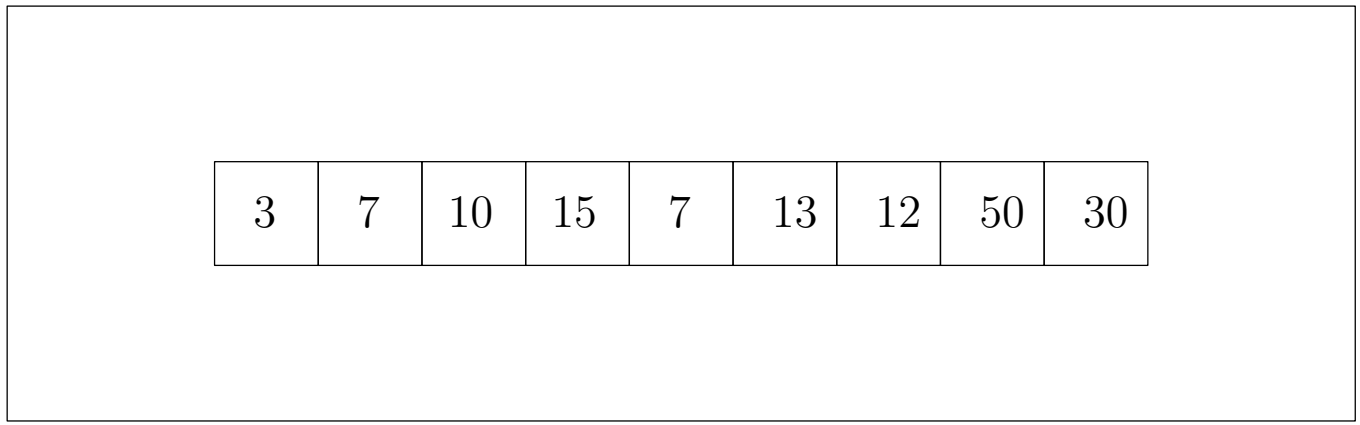
Ταξινόμηση Επιλογής
Selection Sort



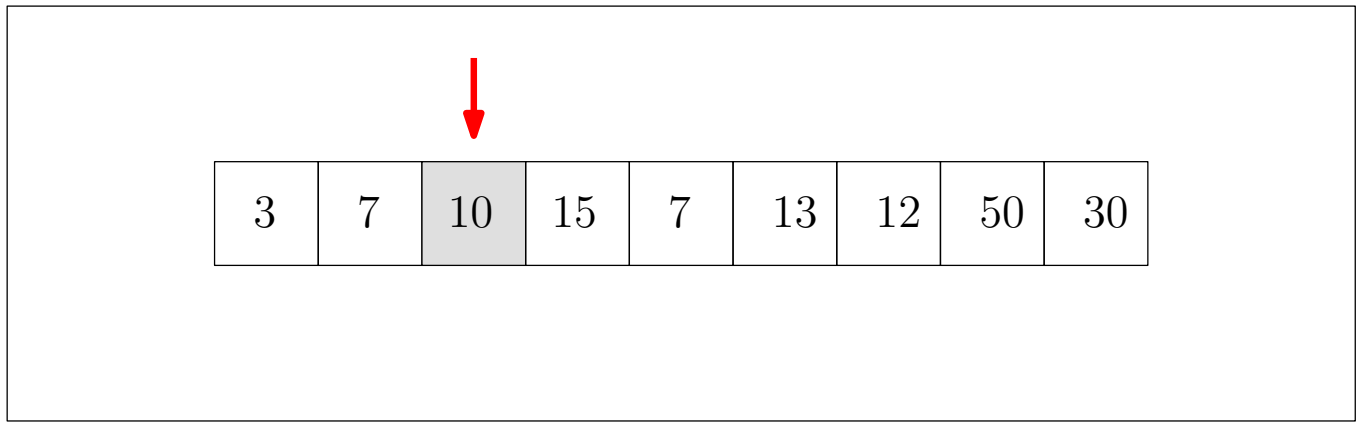
Ταξινόμηση Επιλογής
Selection Sort



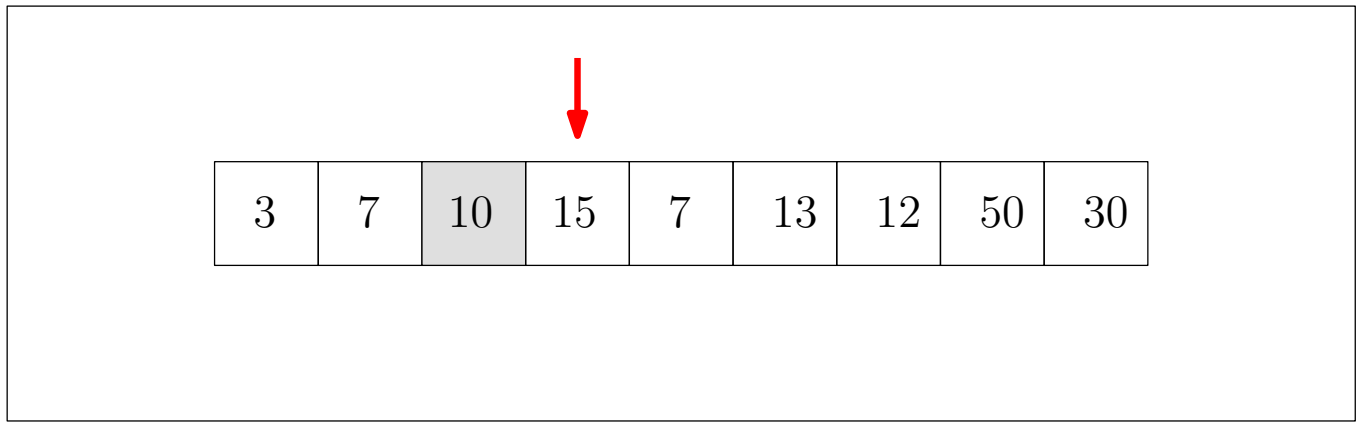
Ταξινόμηση Επιλογής
Selection Sort



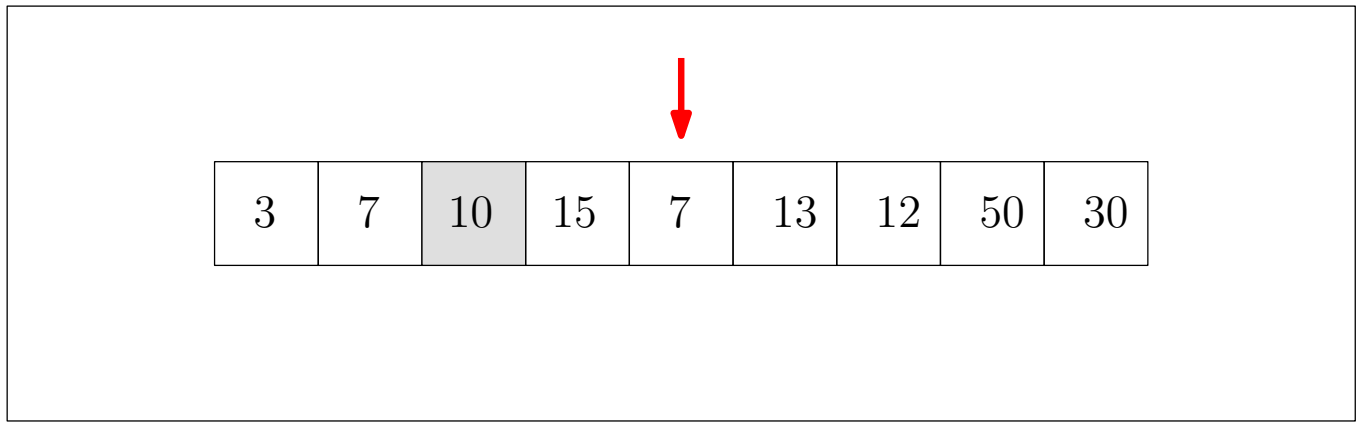
Ταξινόμηση Επιλογής
Selection Sort



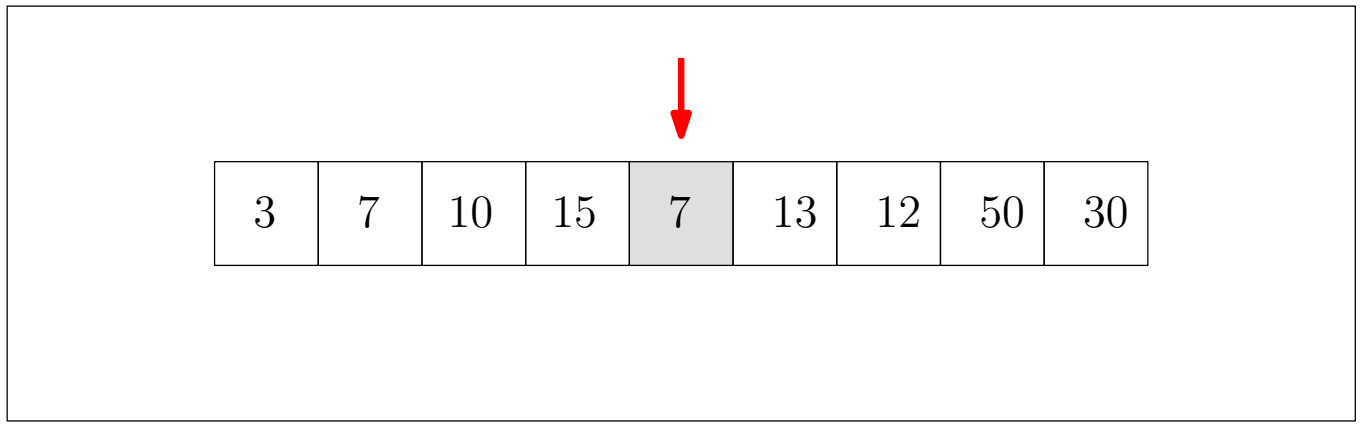
Ταξινόμηση Επιλογής
Selection Sort



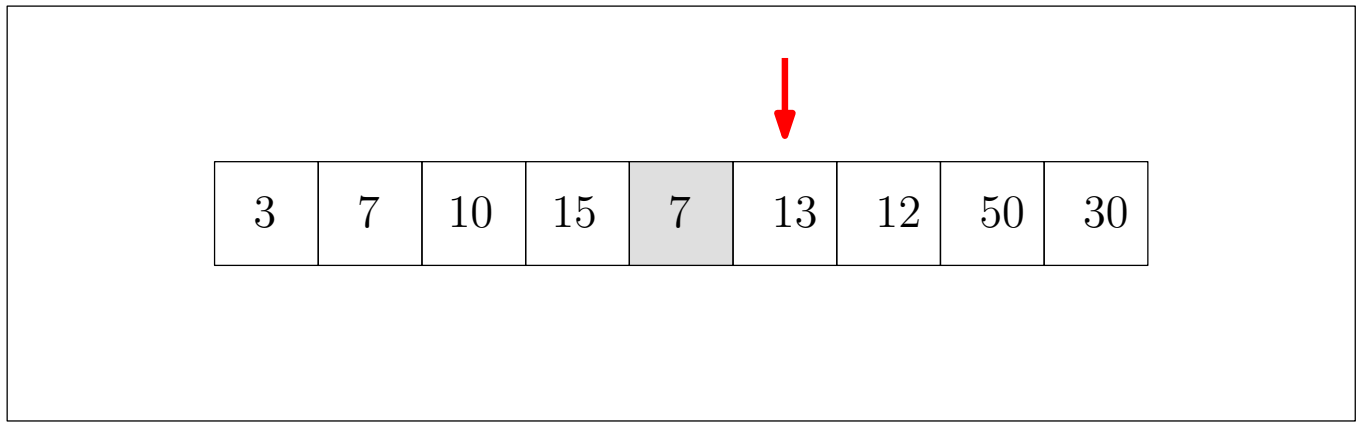
Ταξινόμηση Επιλογής
Selection Sort



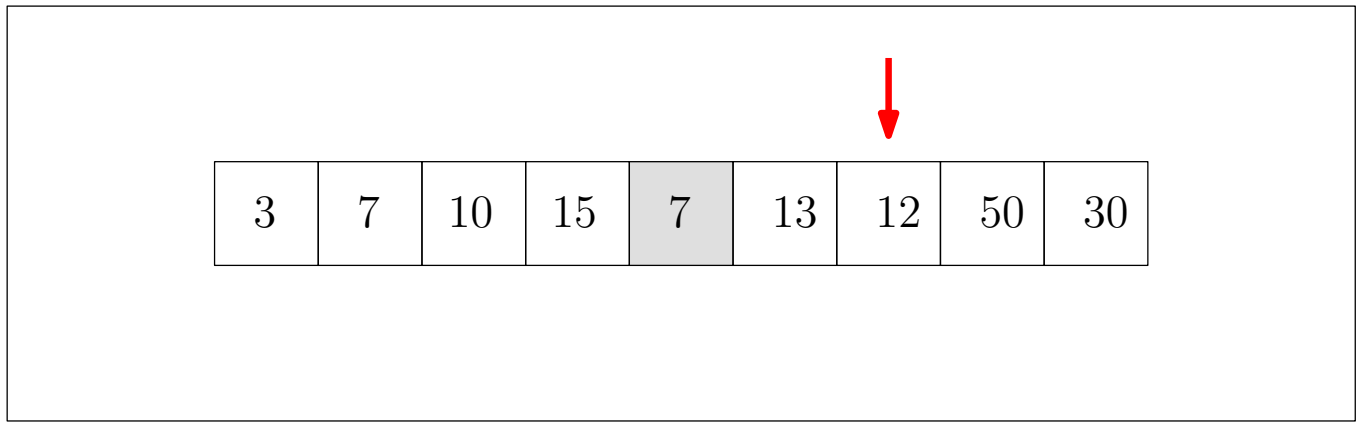
Ταξινόμηση Επιλογής
Selection Sort



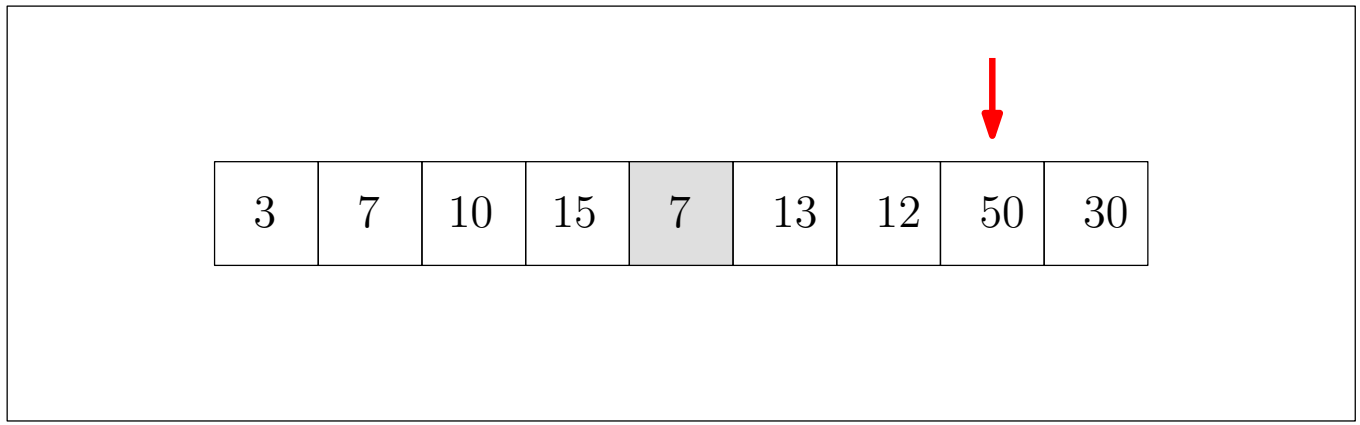
Ταξινόμηση Επιλογής
Selection Sort



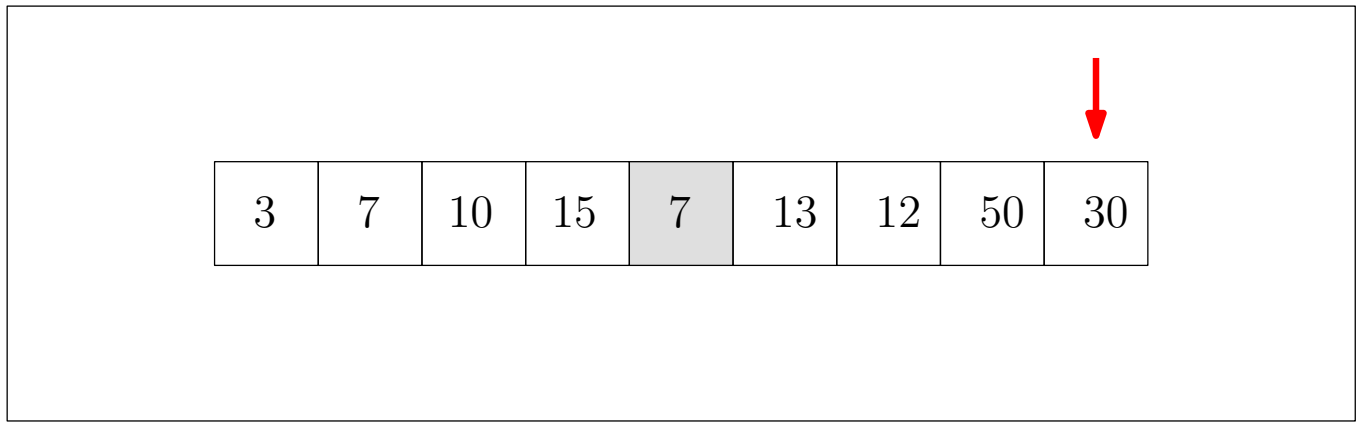
Ταξινόμηση Επιλογής
Selection Sort



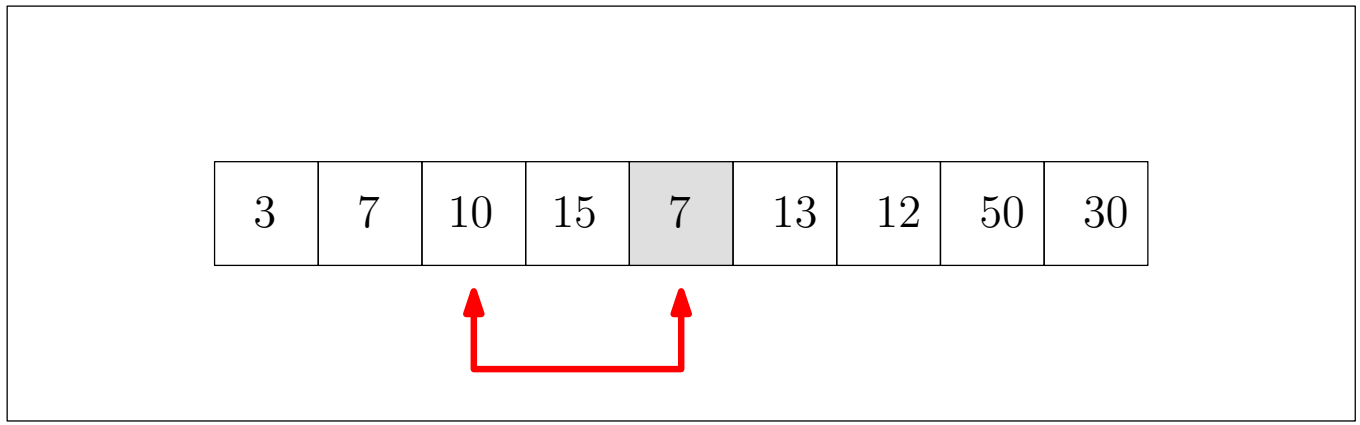
Ταξινόμηση Επιλογής
Selection Sort



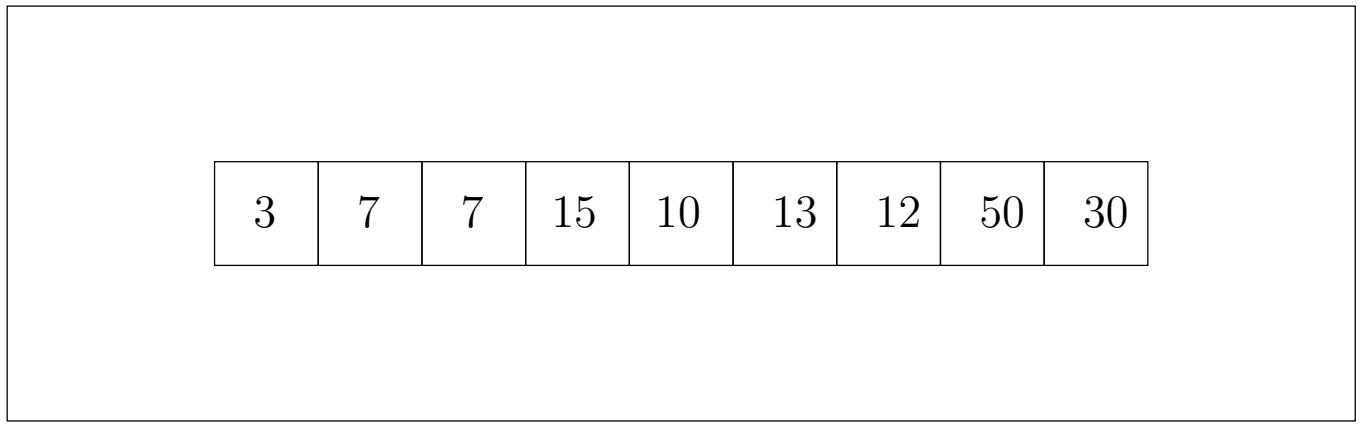
Ταξινόμηση Επιλογής
Selection Sort



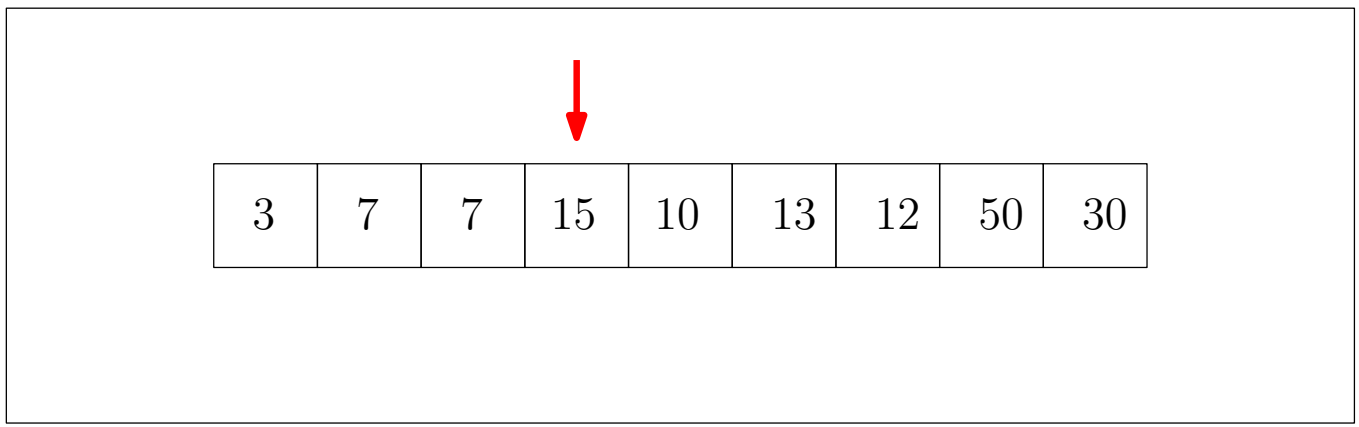
Ταξινόμηση Επιλογής
Selection Sort



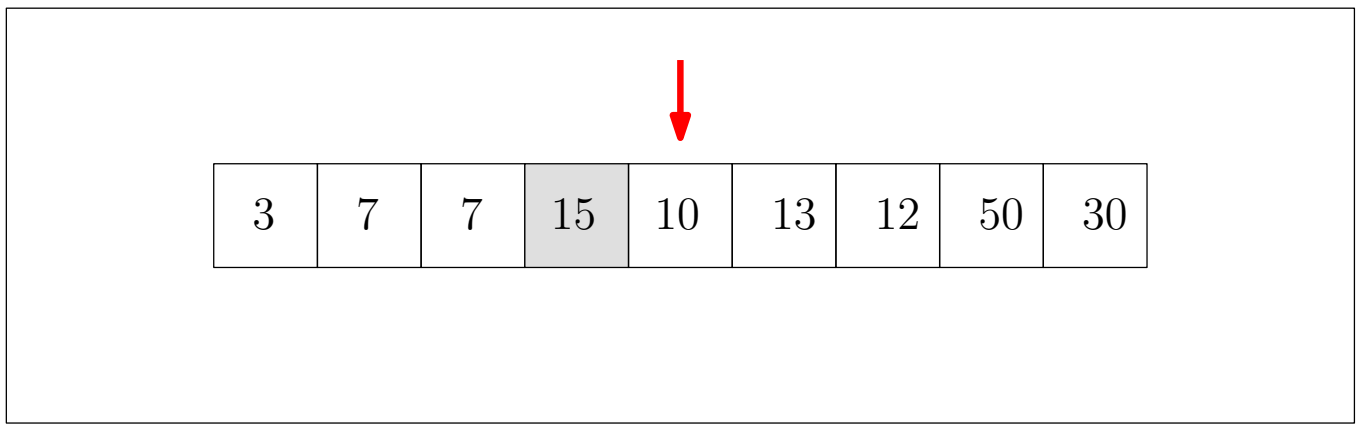
Ταξινόμηση Επιλογής
Selection Sort



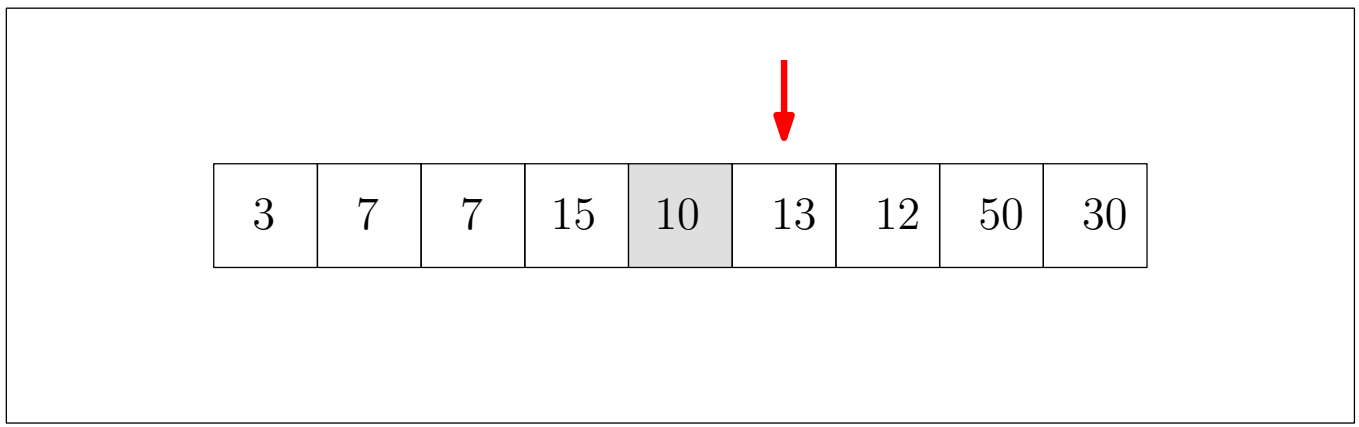
Ταξινόμηση Επιλογής
Selection Sort



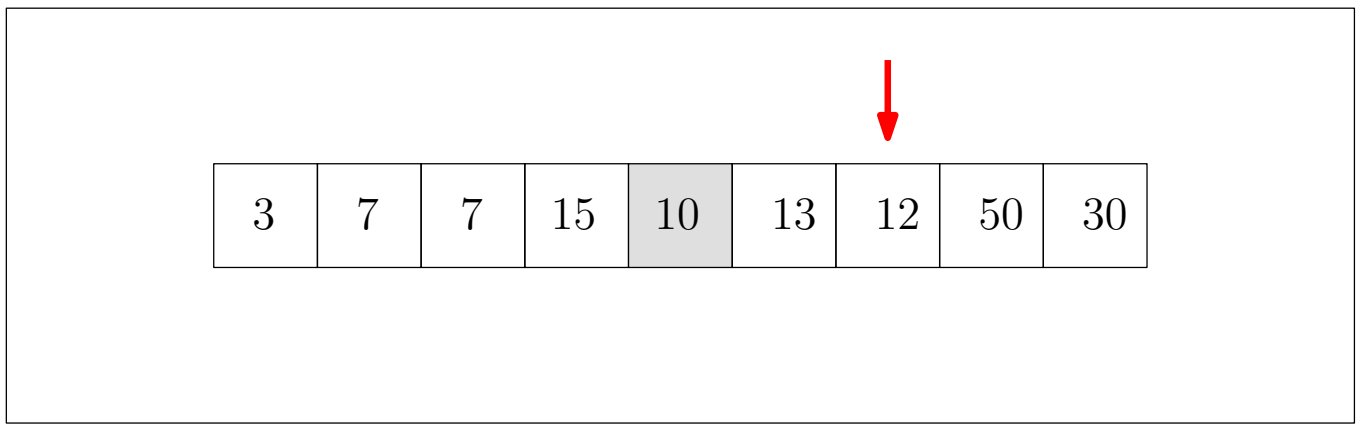
Ταξινόμηση Επιλογής
Selection Sort



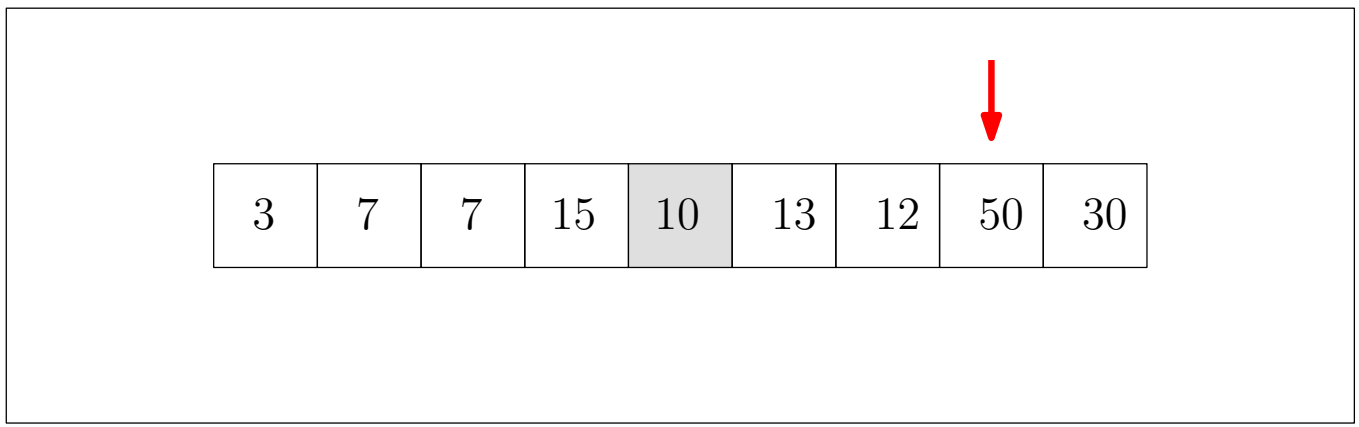
Ταξινόμηση Επιλογής
Selection Sort



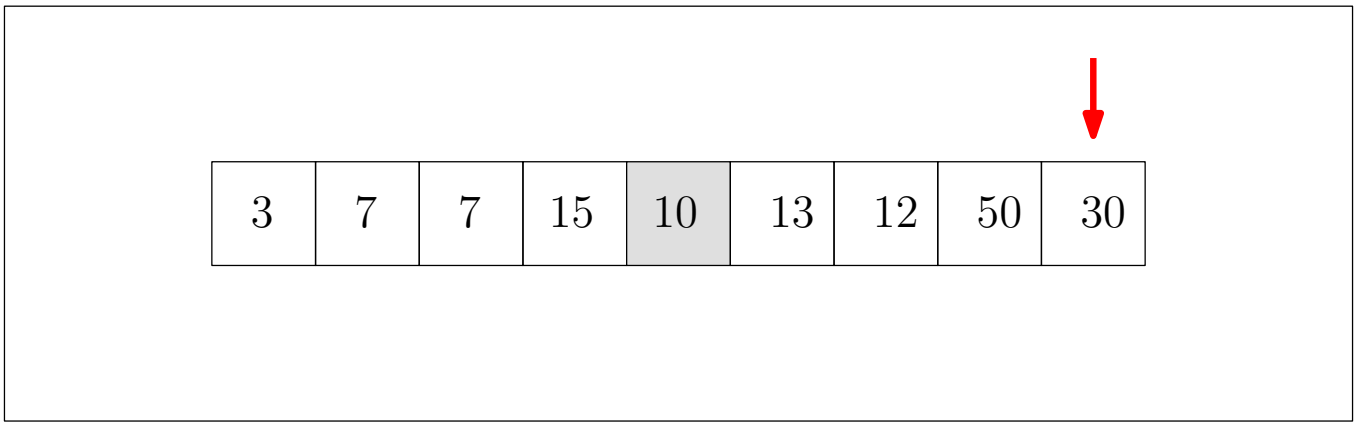
Ταξινόμηση Επιλογής
Selection Sort



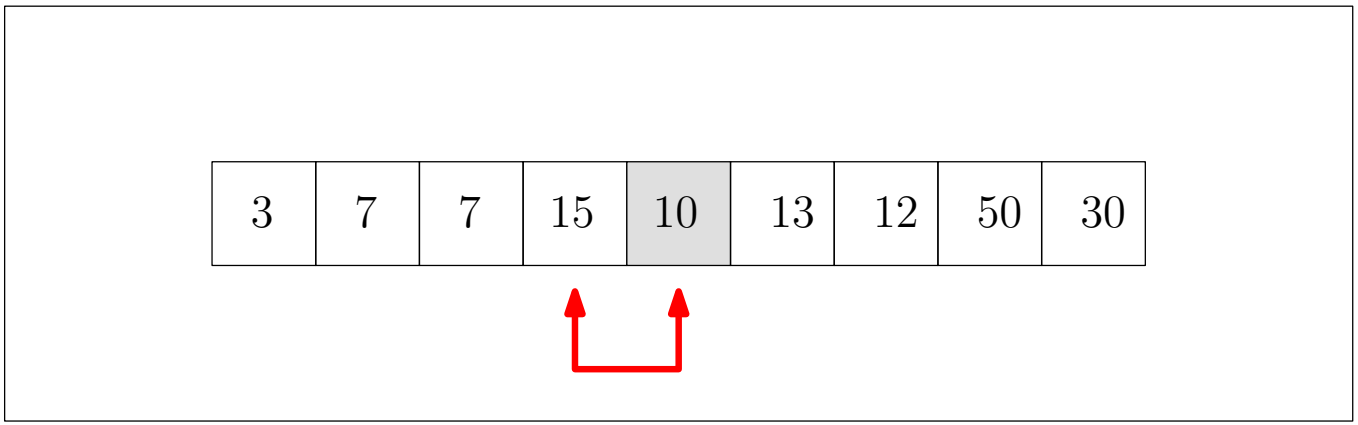
Ταξινόμηση Επιλογής
Selection Sort



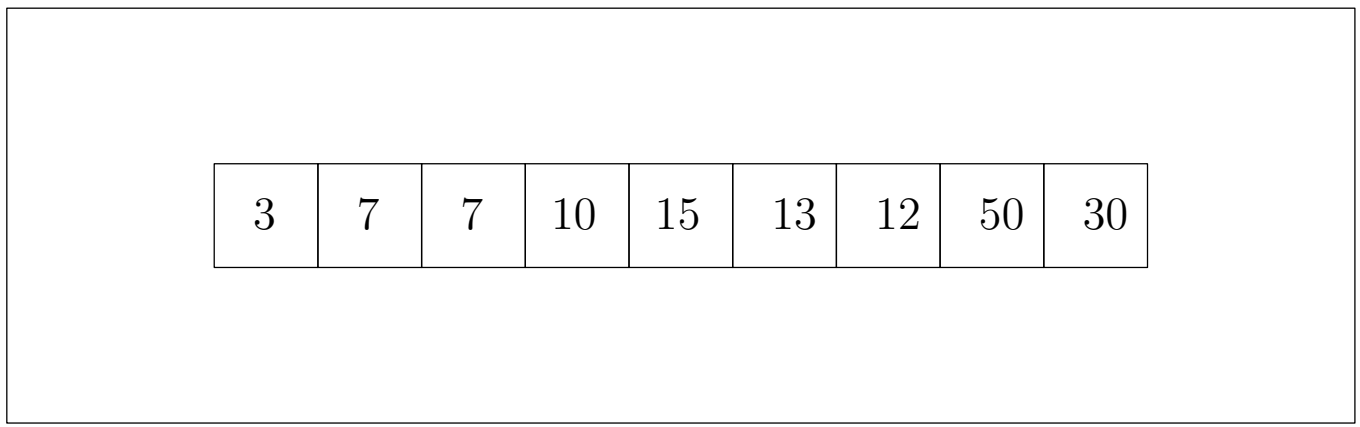
Ταξινόμηση Επιλογής
Selection Sort



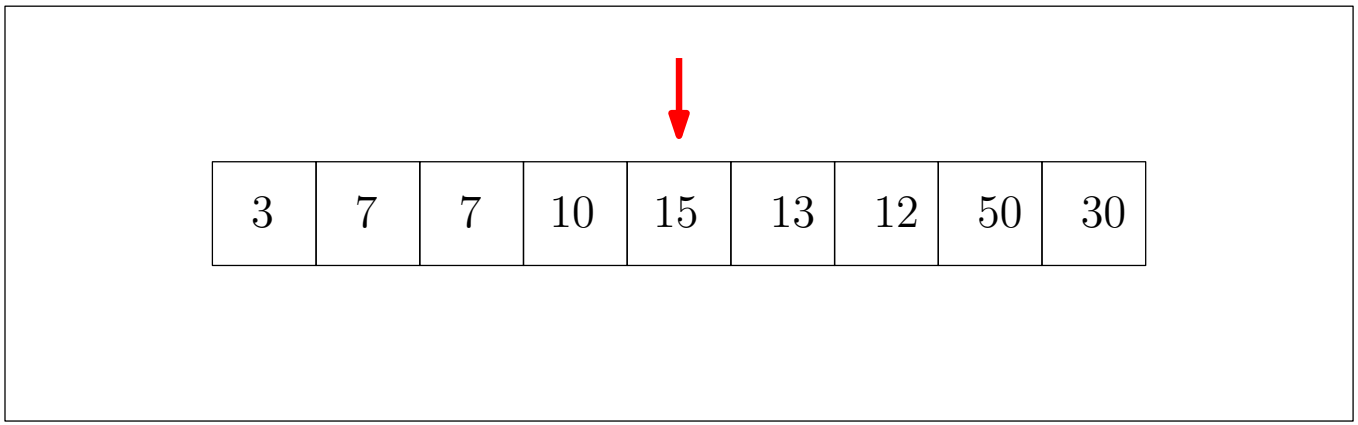
Ταξινόμηση Επιλογής
Selection Sort



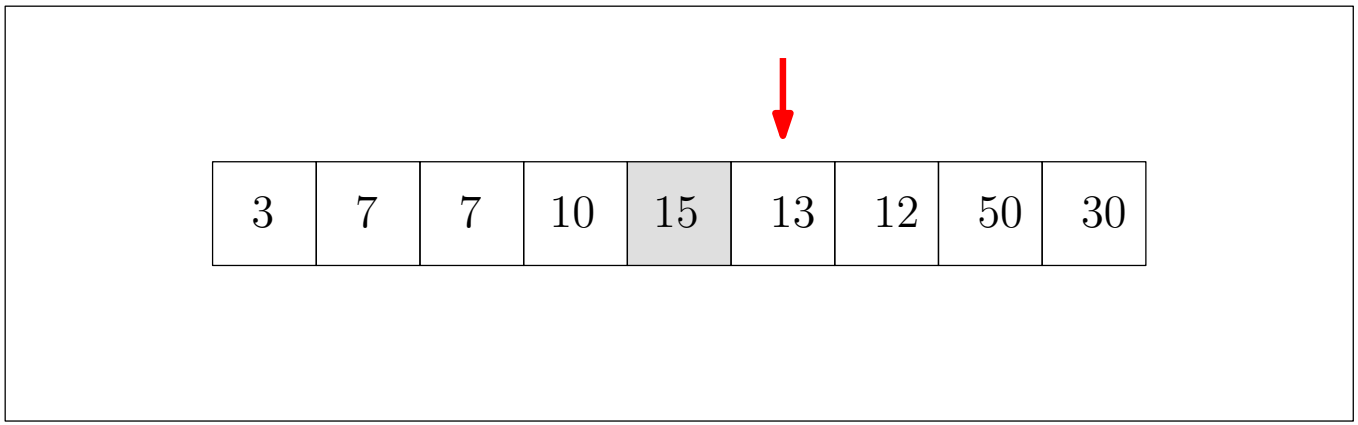
Ταξινόμηση Επιλογής
Selection Sort



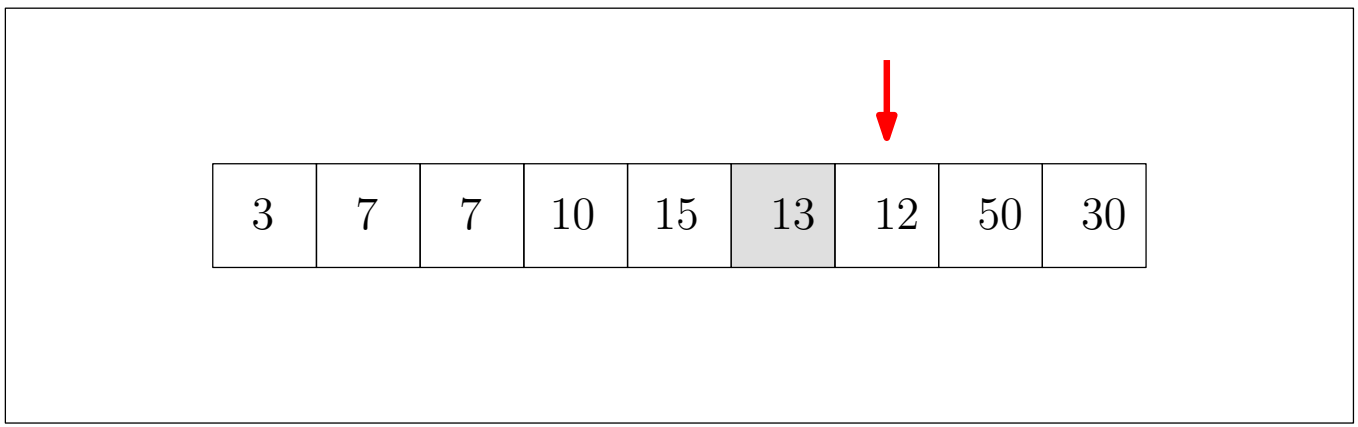
Ταξινόμηση Επιλογής
Selection Sort



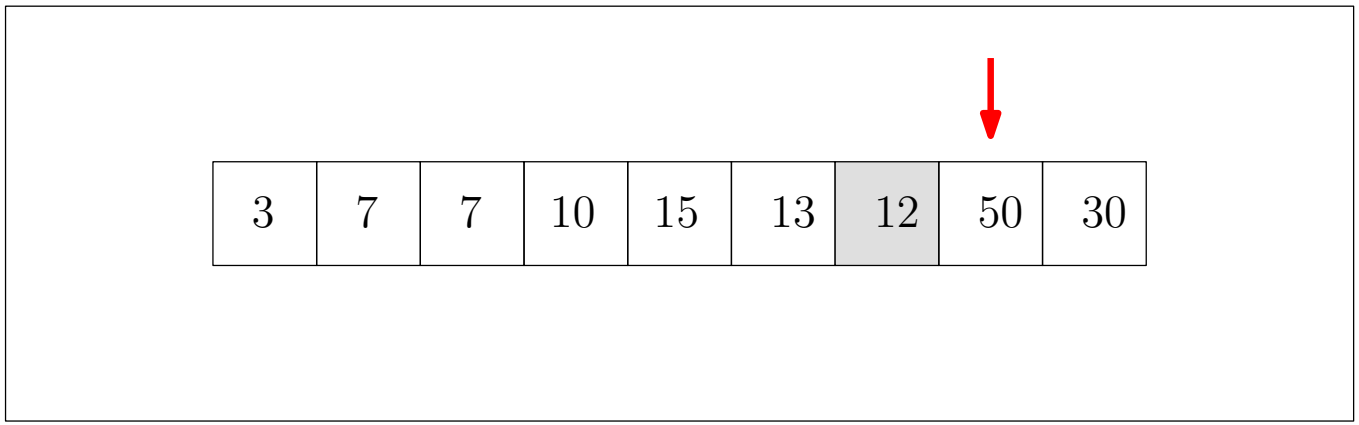
Ταξινόμηση Επιλογής
Selection Sort



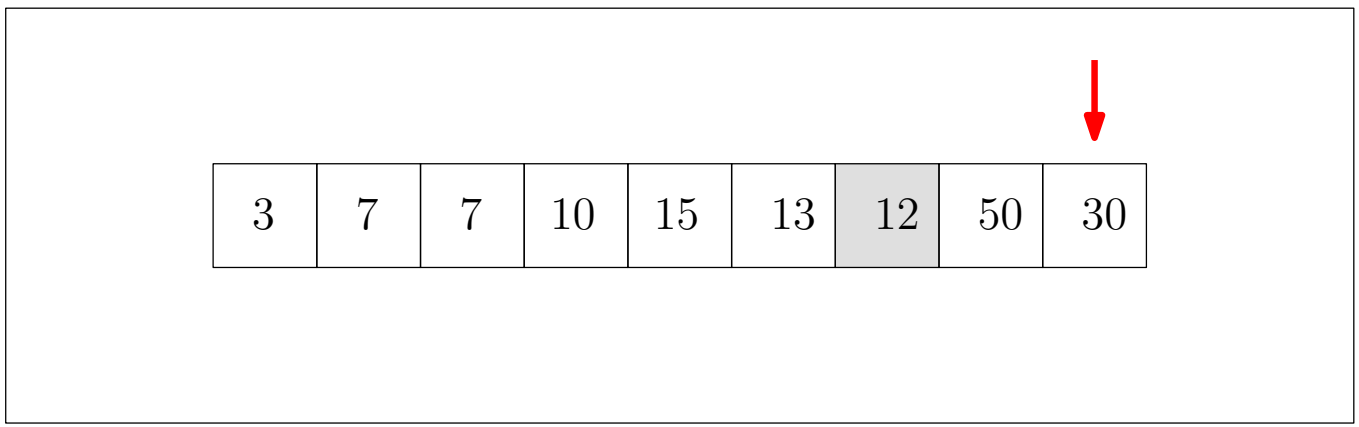
Ταξινόμηση Επιλογής
Selection Sort



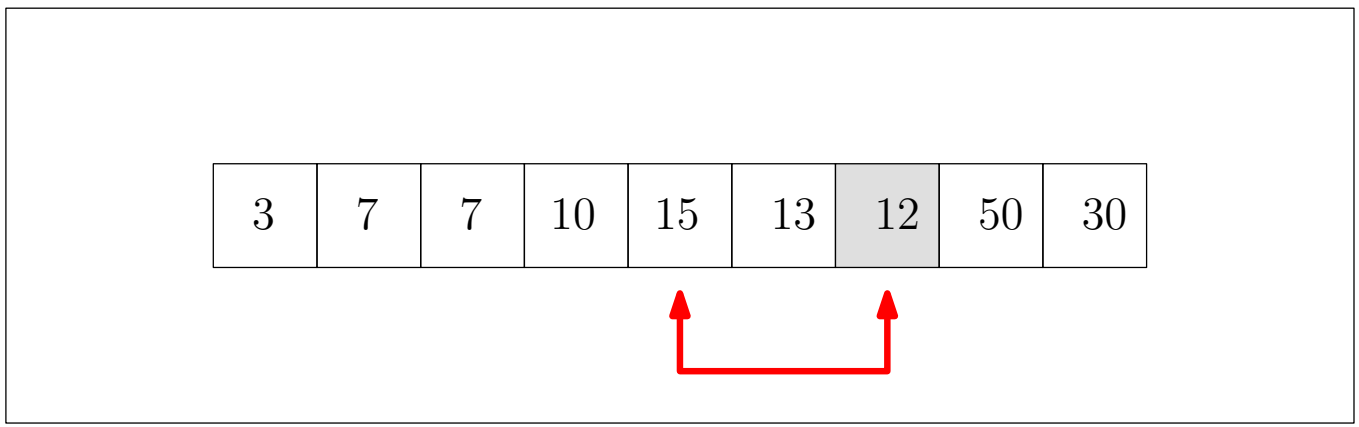
Ταξινόμηση Επιλογής
Selection Sort



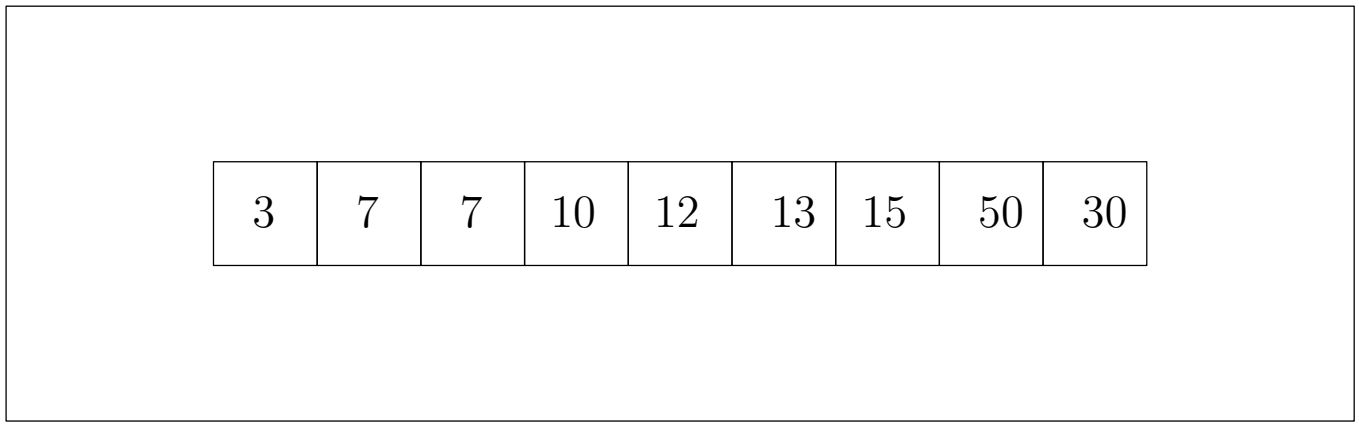
Ταξινόμηση Επιλογής
Selection Sort



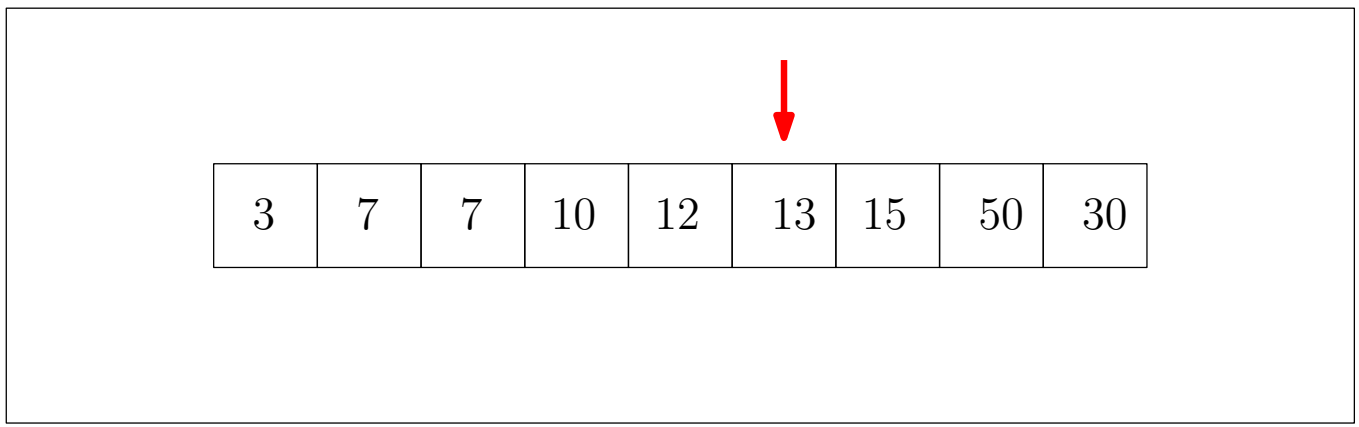
Ταξινόμηση Επιλογής
Selection Sort



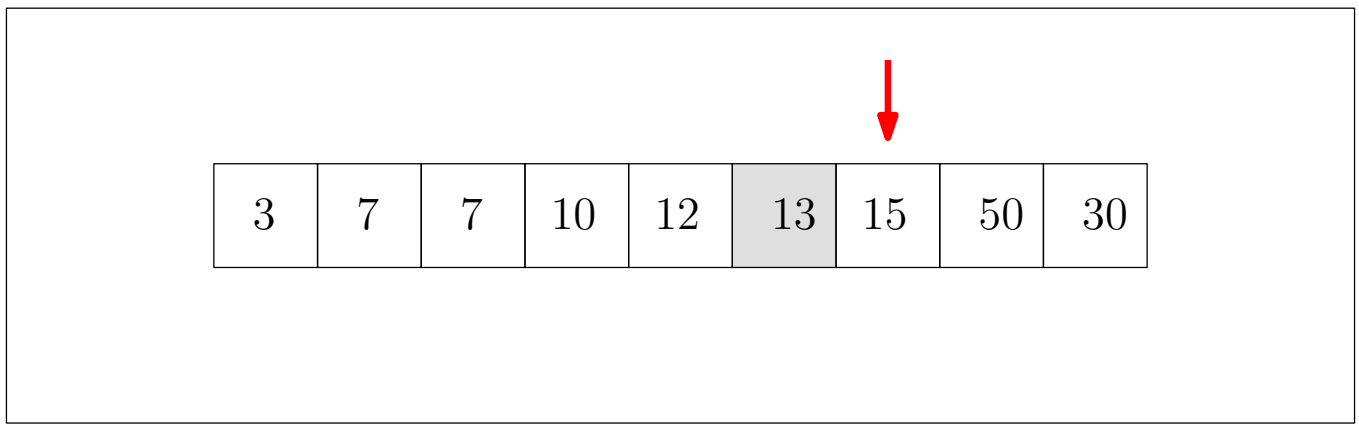
Ταξινόμηση Επιλογής
Selection Sort



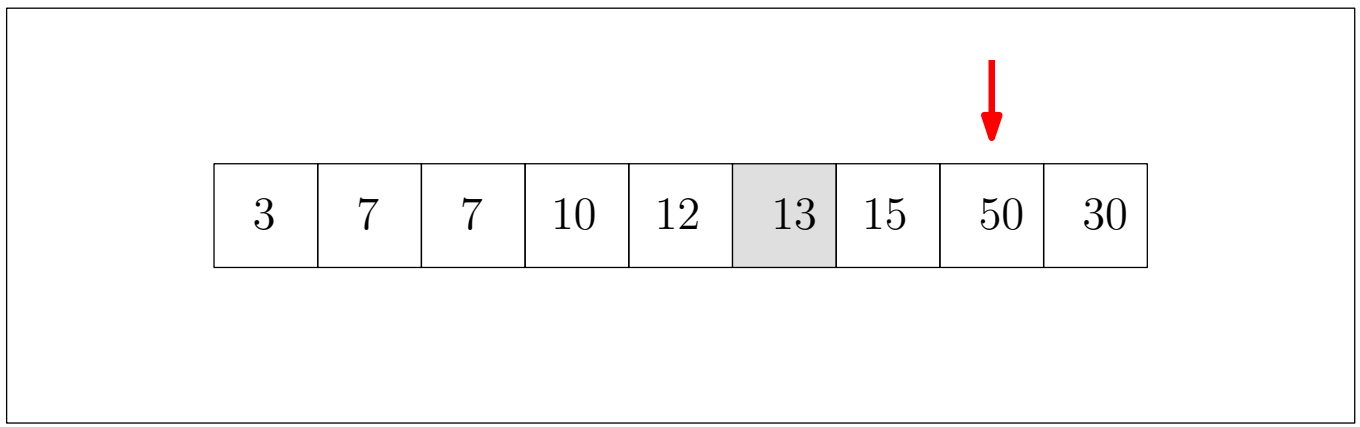
Ταξινόμηση Επιλογής
Selection Sort



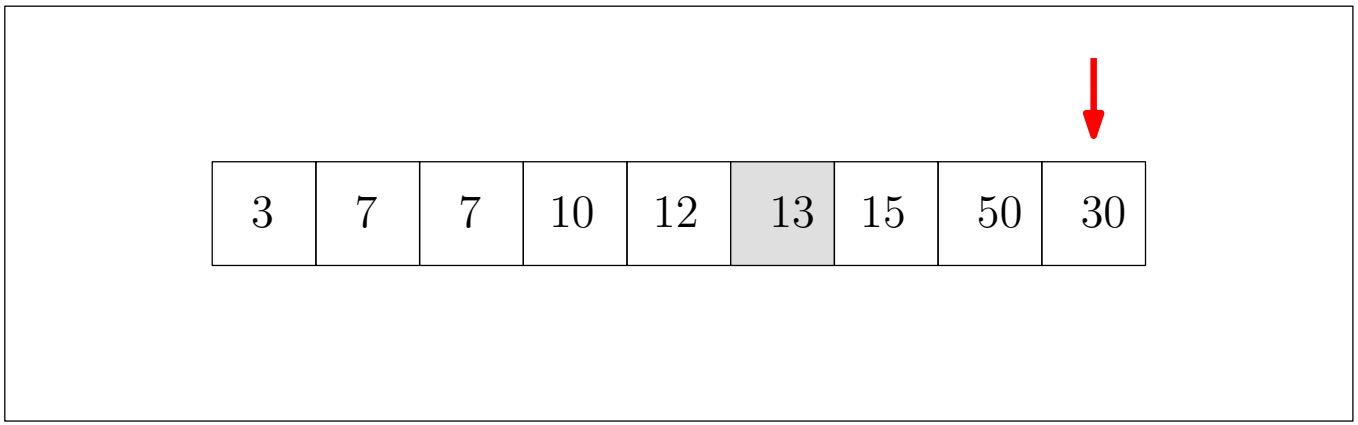
Ταξινόμηση Επιλογής
Selection Sort



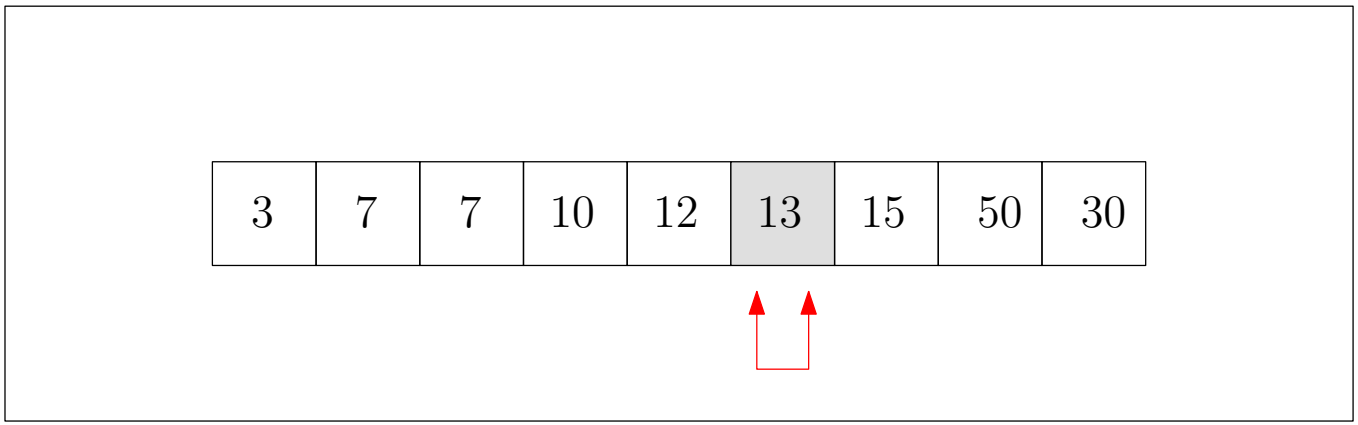
Ταξινόμηση Επιλογής
Selection Sort



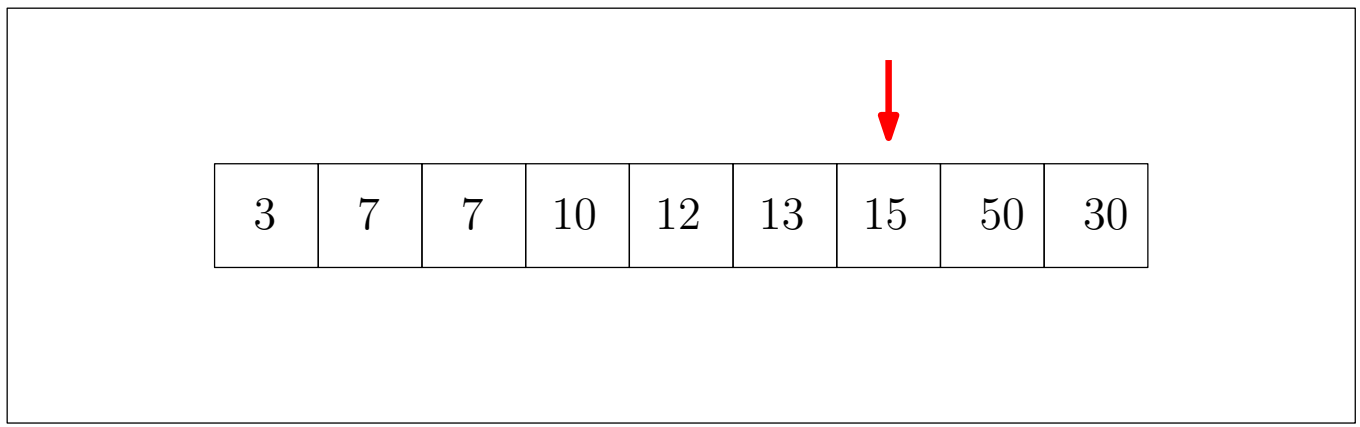
Ταξινόμηση Επιλογής
Selection Sort



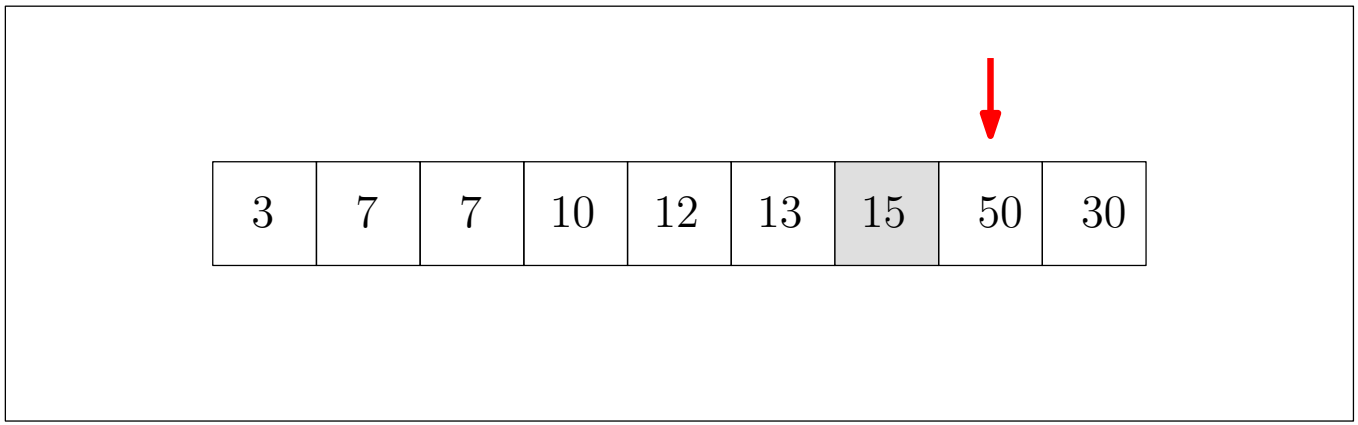
Ταξινόμηση Επιλογής
Selection Sort



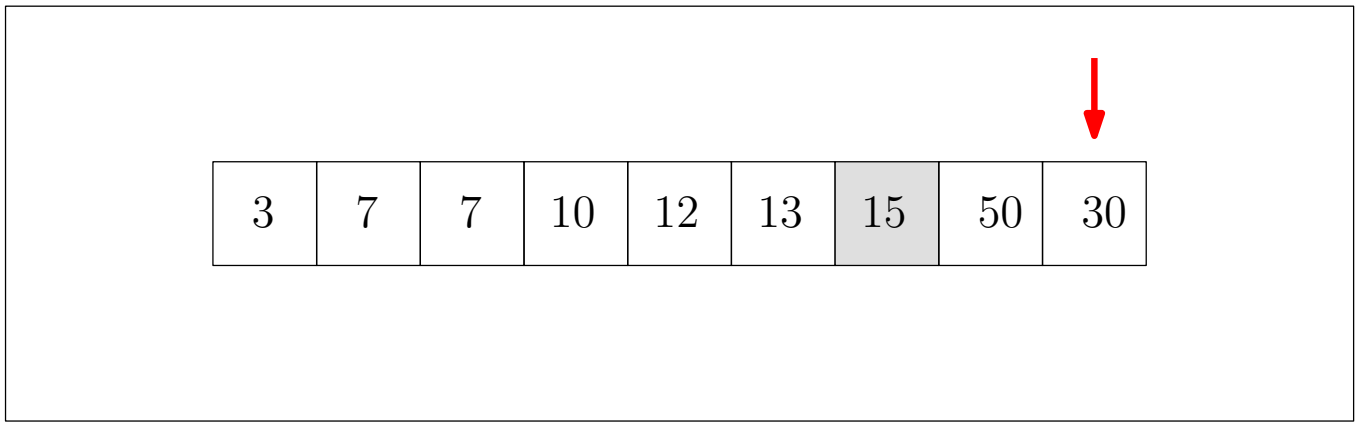
Ταξινόμηση Επιλογής
Selection Sort



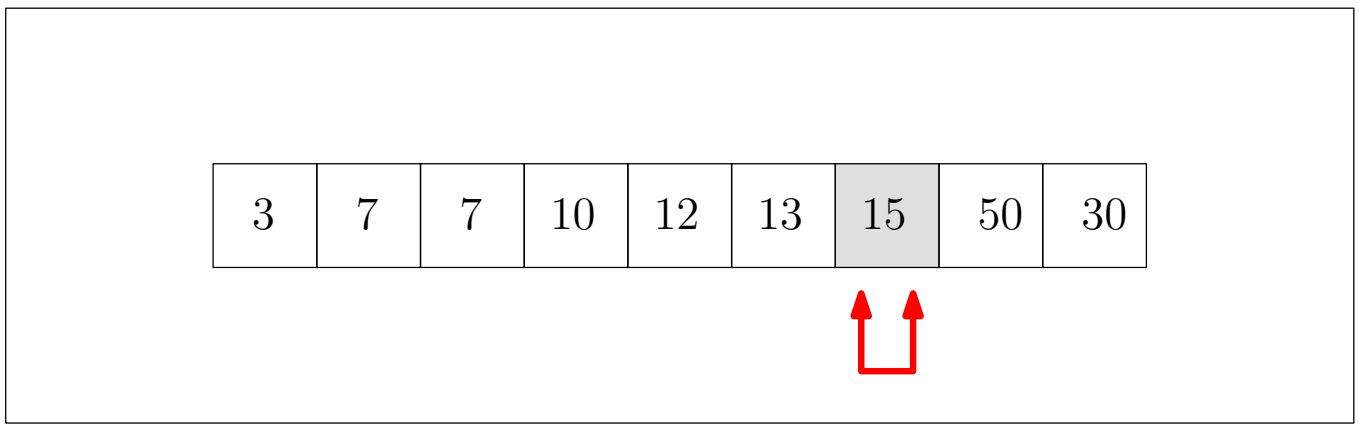
Ταξινόμηση Επιλογής
Selection Sort



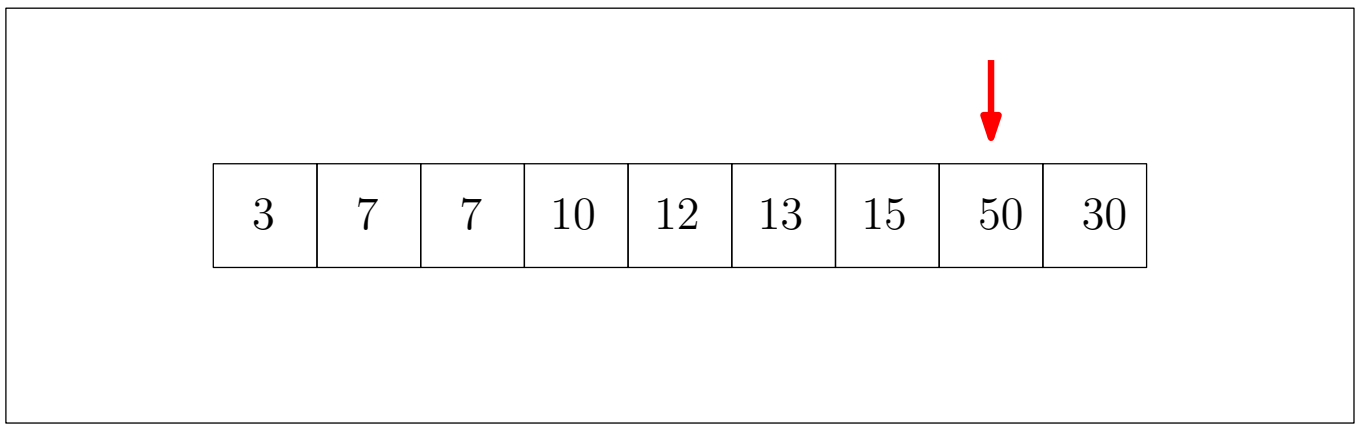
Ταξινόμηση Επιλογής
Selection Sort



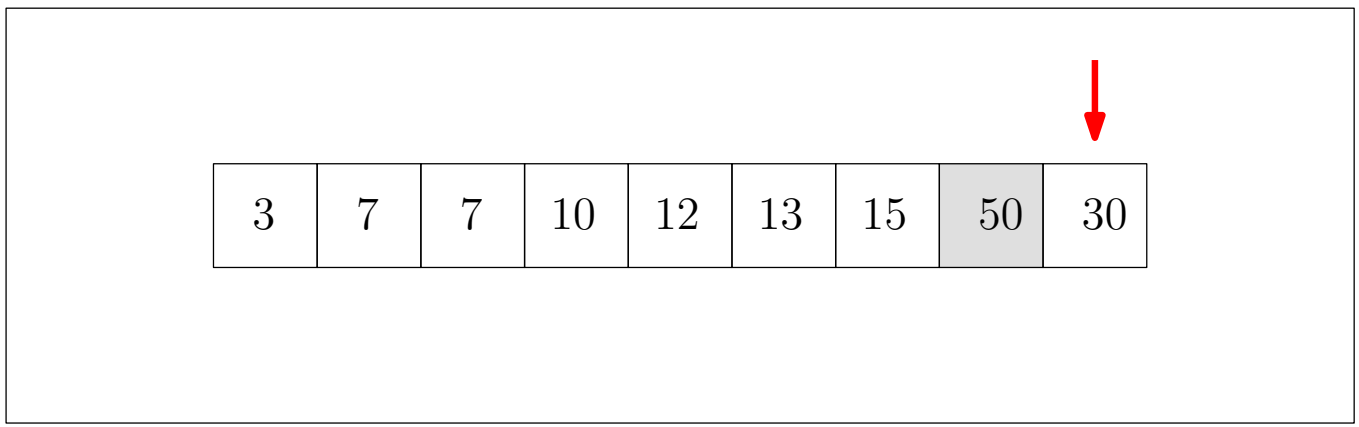
Ταξινόμηση Επιλογής
Selection Sort



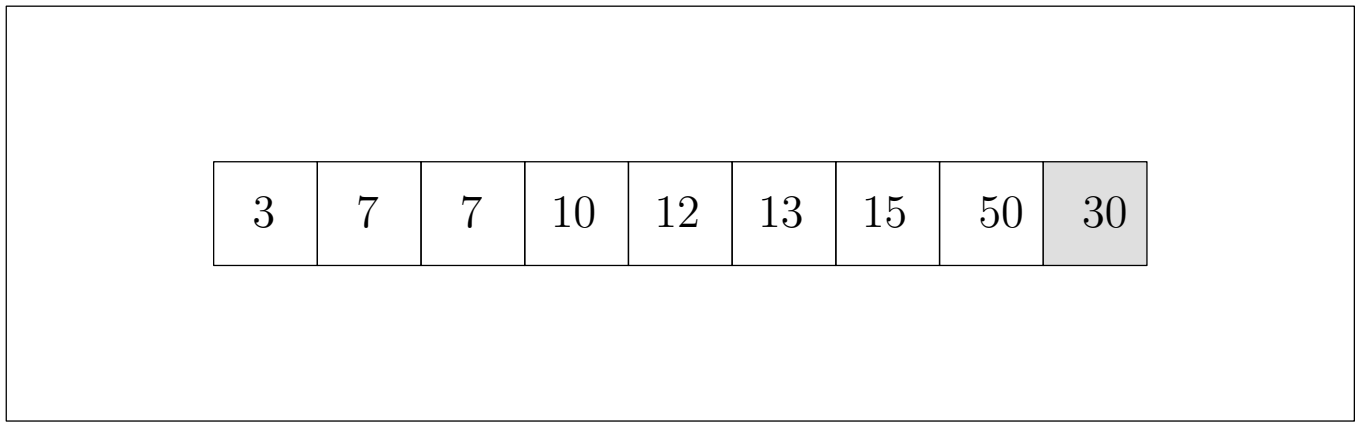
Ταξινόμηση Επιλογής
Selection Sort



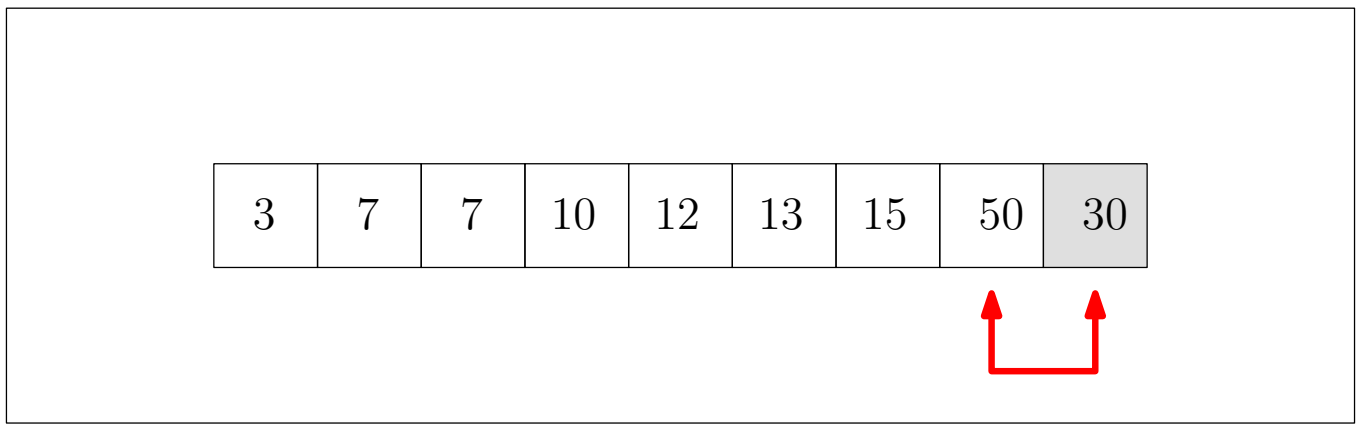
Ταξινόμηση Επιλογής
Selection Sort



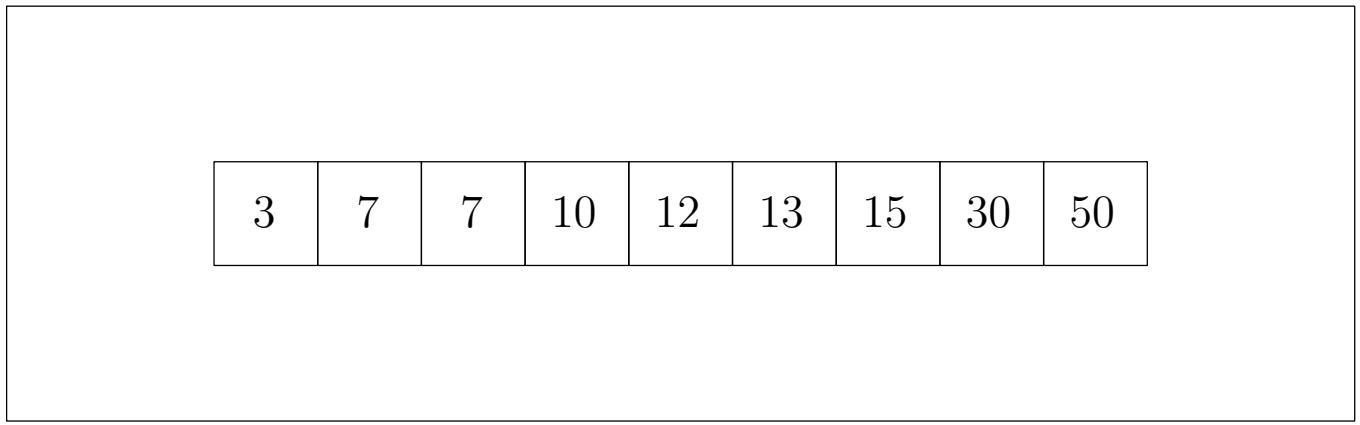
Ταξινόμηση Επιλογής
Selection Sort



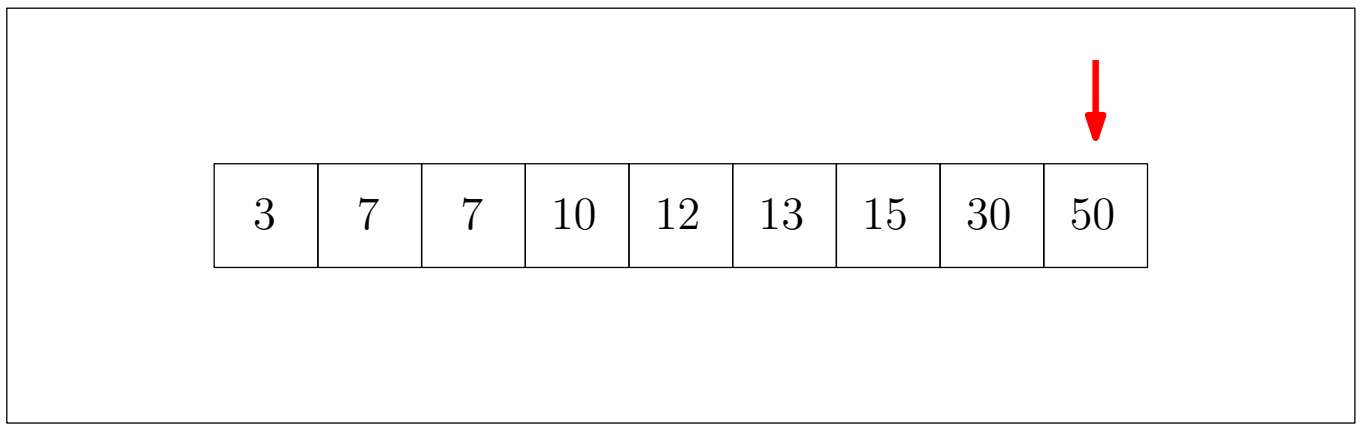
Ταξινόμηση Επιλογής
Selection Sort



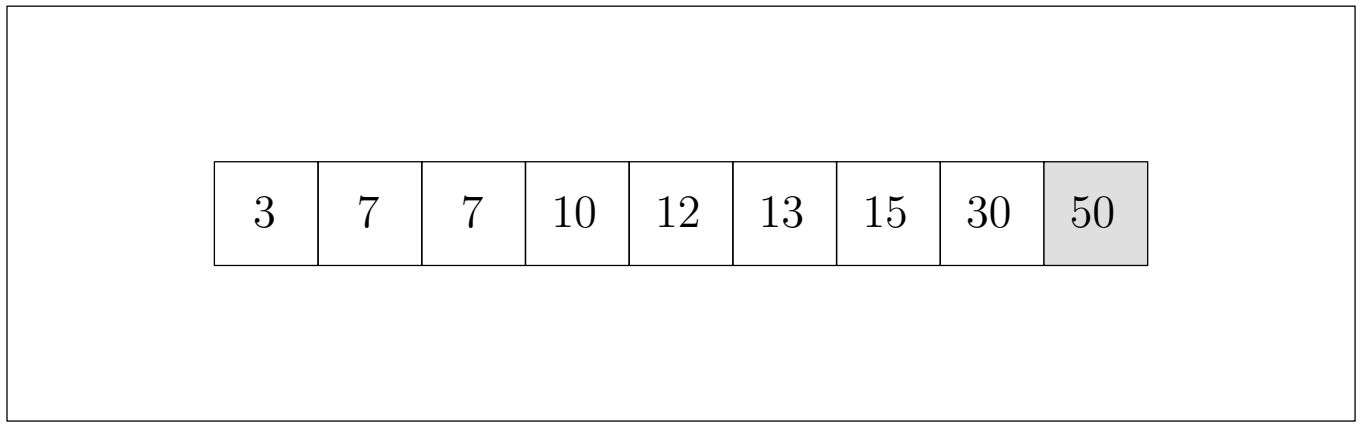
Ταξινόμηση Επιλογής
Selection Sort



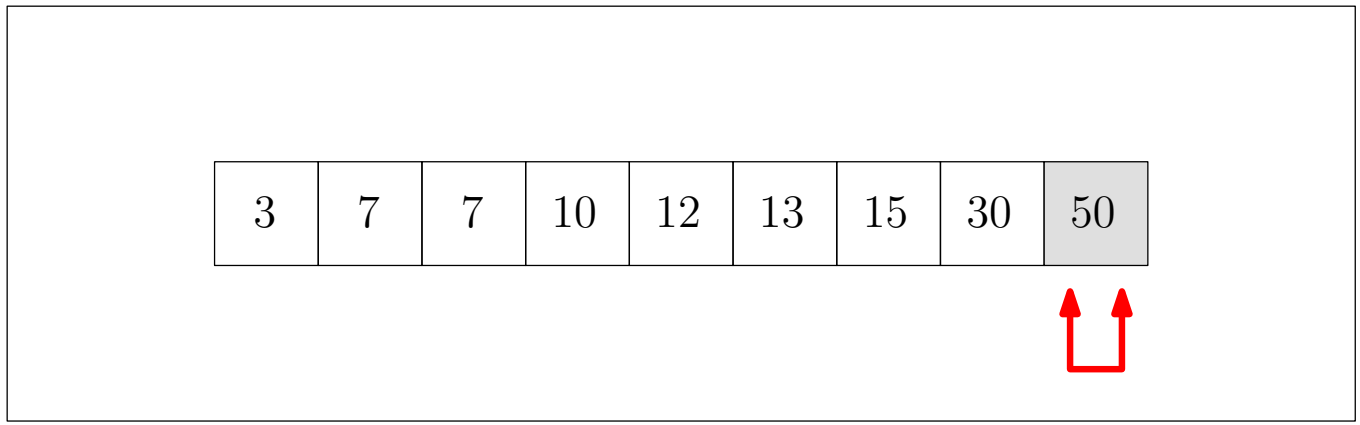
Ταξινόμηση Επιλογής
Selection Sort



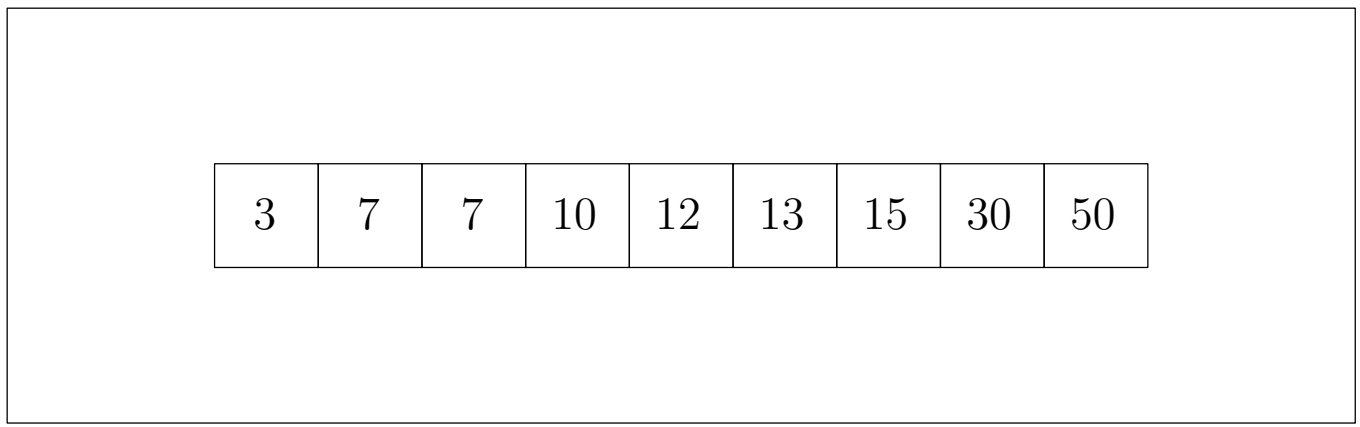
Ταξινόμηση Επιλογής
Selection Sort



Ταξινόμηση Επιλογής
Selection Sort



Ταξινόμηση Επιλογής
Selection Sort



Χρόνος Εκτέλεσης Ταξινόμησης Επιλογής

- για να βρούμε το πρώτο στοιχείο κάνουμε $n-1$ συγκρίσεις
- για το δεύτερο $n-2$ συγκρίσεις
- κ.τ.λ

Χρόνος Εκτέλεσης Ταξινόμησης Επιλογής

- για να βρούμε το πρώτο στοιχείο κάνουμε $n-1$ συγκρίσεις
- για το δεύτερο $n-2$ συγκρίσεις
- κ.τ.λ

Συνολικός αριθμός συγκρίσεων:

$$(n-1)+(n-2)+\ldots+2+1 = n(n-1)/2 = O(n^2)$$

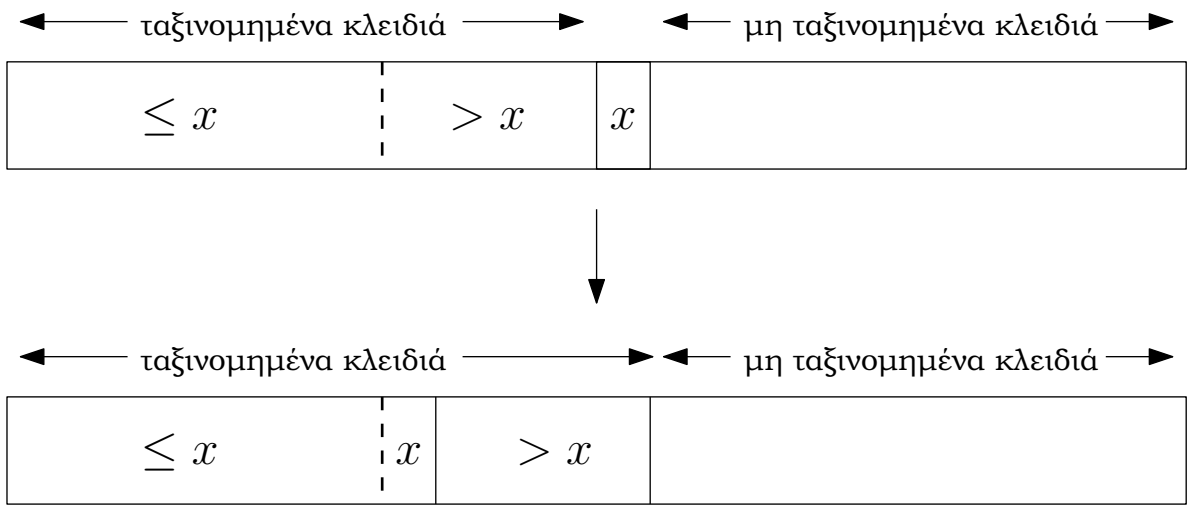
Ταξινόμηση Επιλογής

Δείξε μου τον κώδικα!

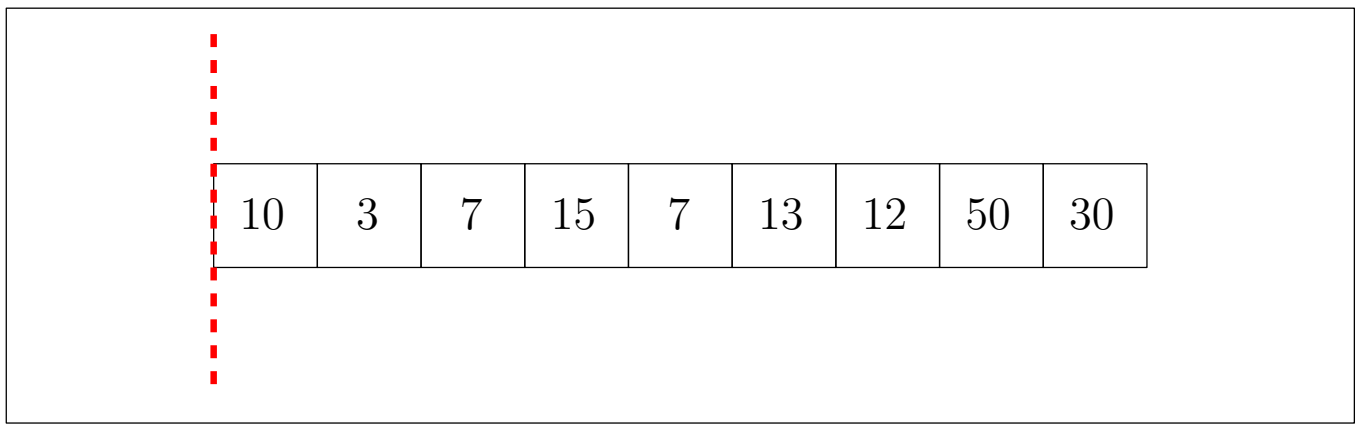
```
void selection_sort(int *a, int n) {
    int i, j, t;

    for(i=0; i<n-1; i++) {
        int min_pos = i;
        for(j=i+1; j<n; j++) {
            if (a[j]<a[min_pos]) {
                min_pos = j;
            }
        }

        if (i != min_pos) {
            t = a[i];
            a[i] = a[min_pos];
            a[min_pos] = t;
        }
    }
}
```

Ταξινόμηση με εισαγωγή
Insertion Sort

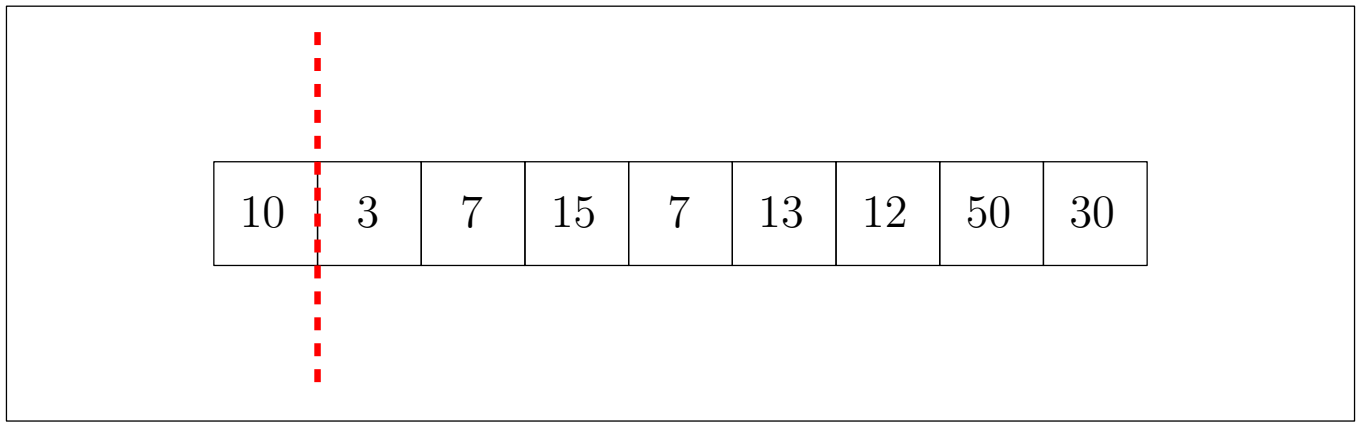
σε κάθε βήμα παίρνουμε ένα κλειδί και το βάζουμε στην σωστή σειρά μέχρι να μας τελειώσουν τα κλειδιά



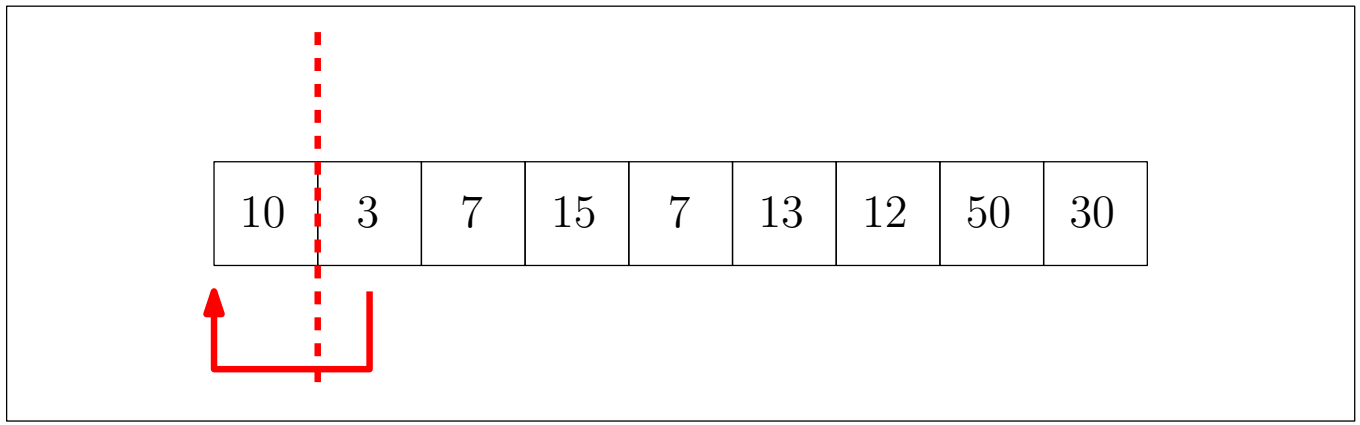
Ταξινόμηση με εισαγωγή
Insertion Sort



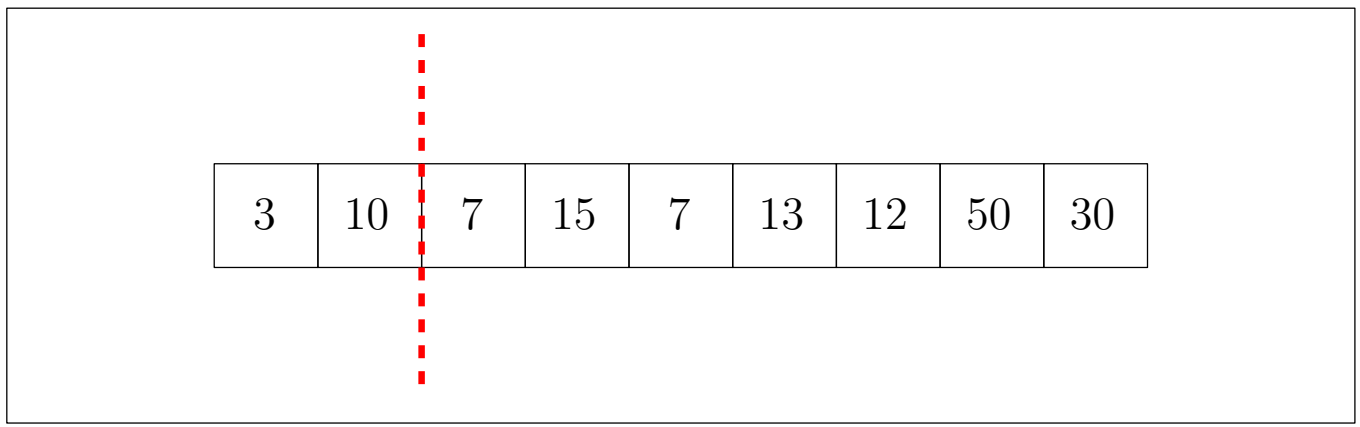
Ταξινόμηση με εισαγωγή
Insertion Sort



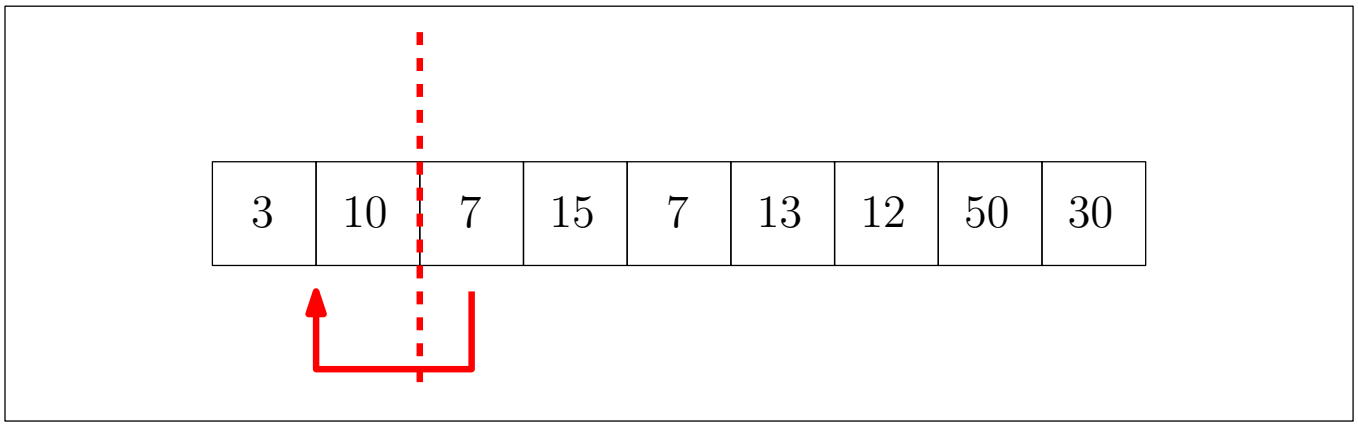
Ταξινόμηση με εισαγωγή
Insertion Sort



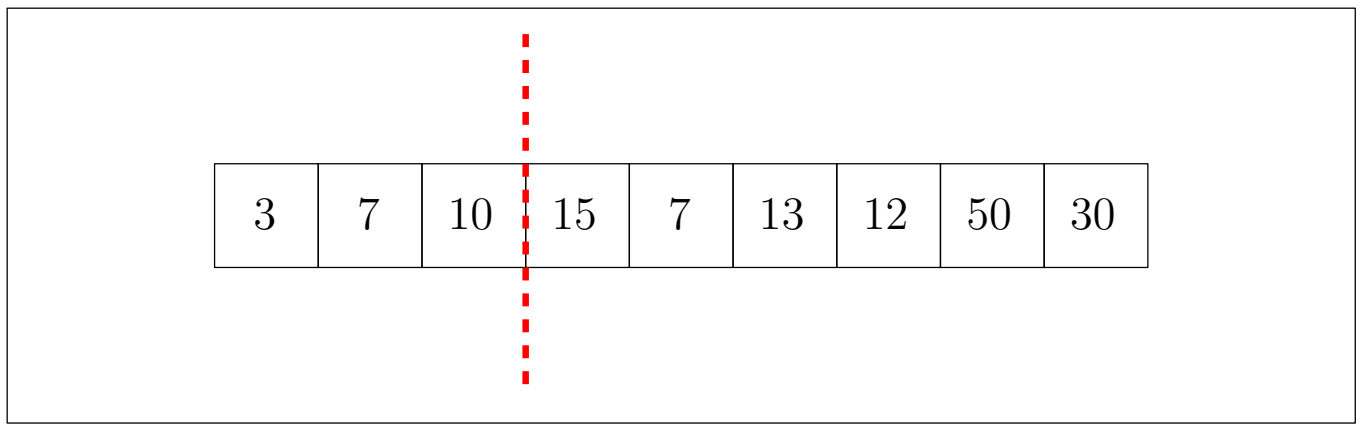
Ταξινόμηση με εισαγωγή
Insertion Sort



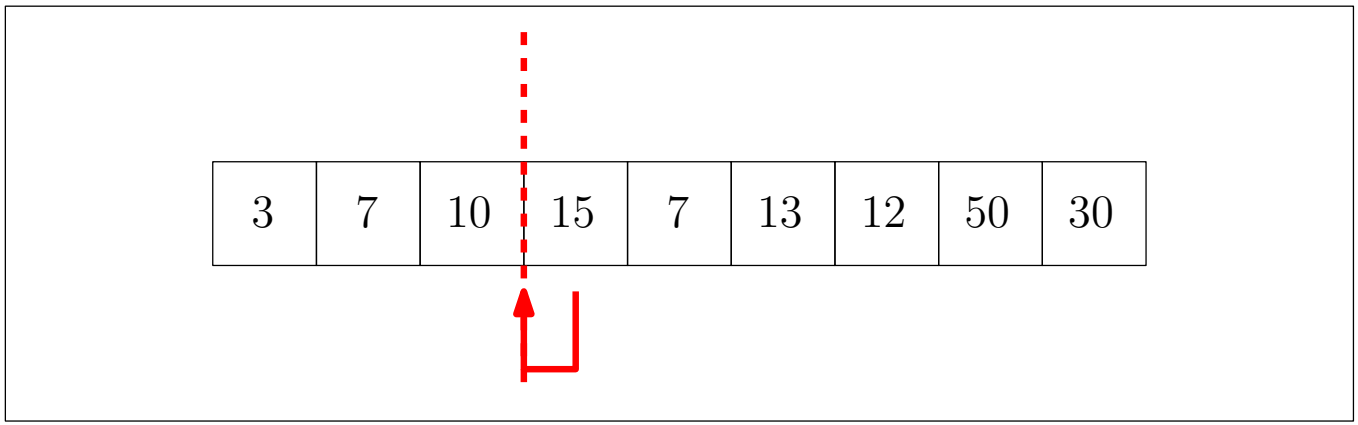
Ταξινόμηση με εισαγωγή
Insertion Sort



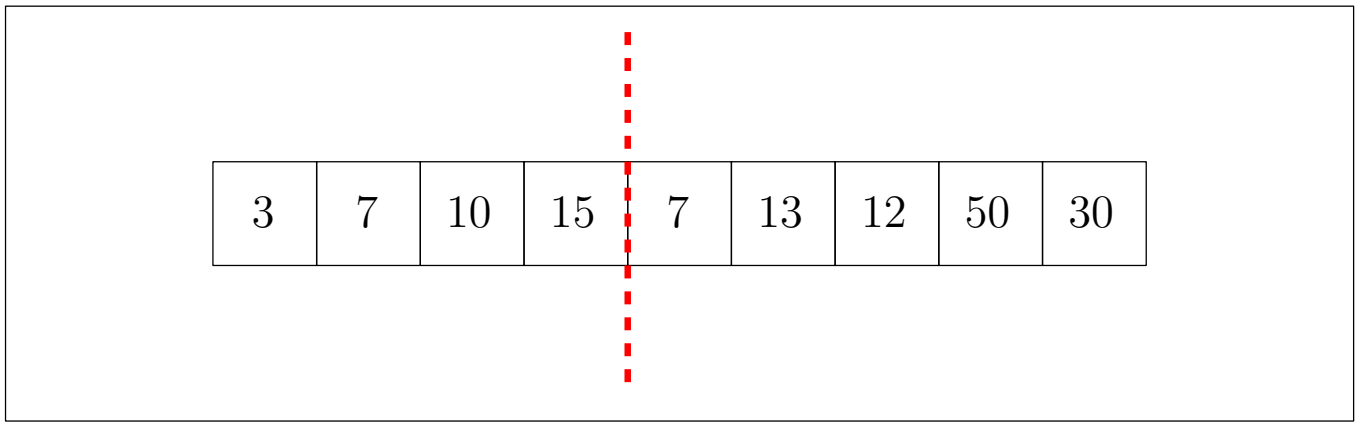
Ταξινόμηση με εισαγωγή
Insertion Sort



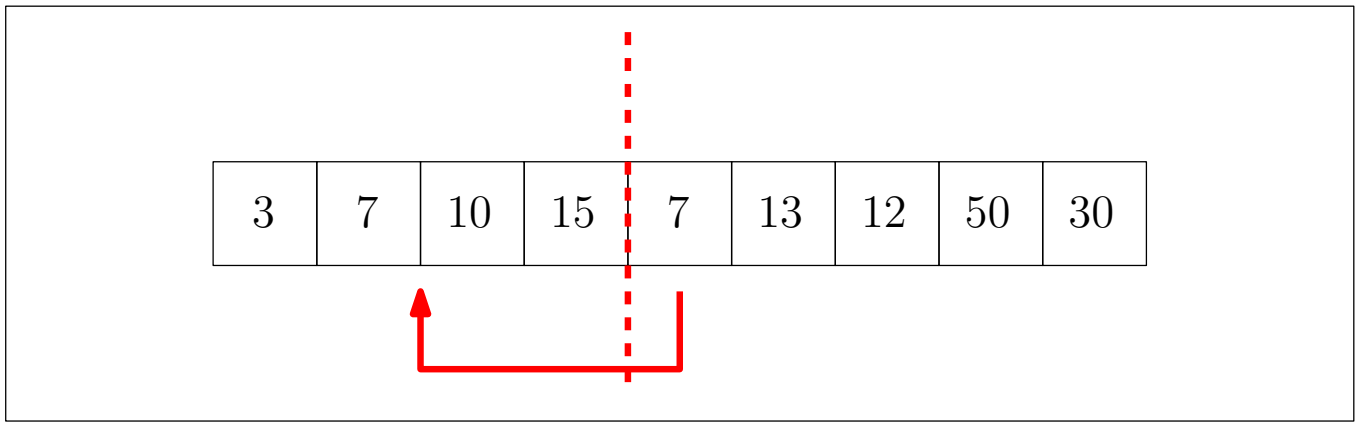
Ταξινόμηση με εισαγωγή
Insertion Sort



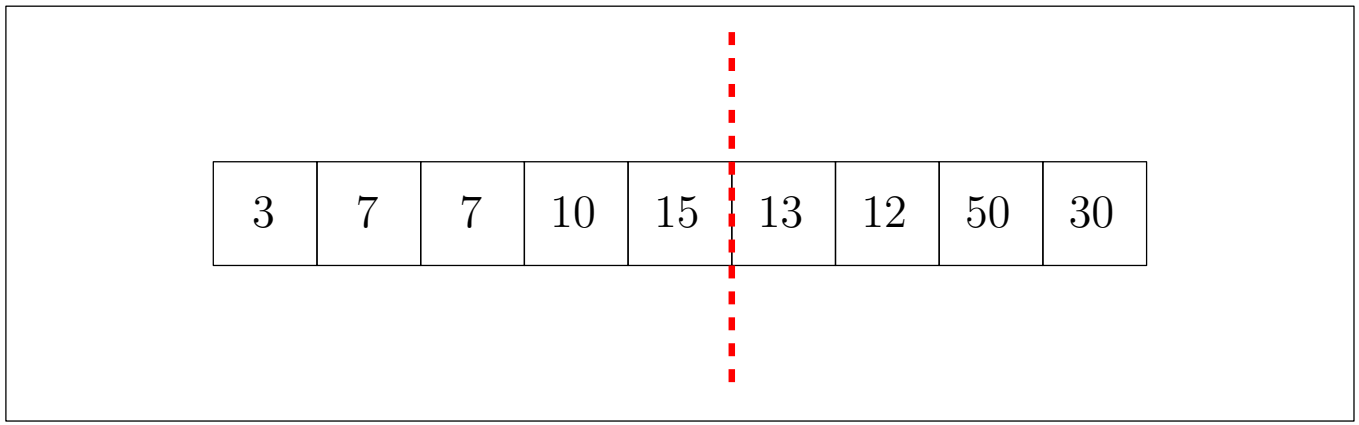
Ταξινόμηση με εισαγωγή
Insertion Sort



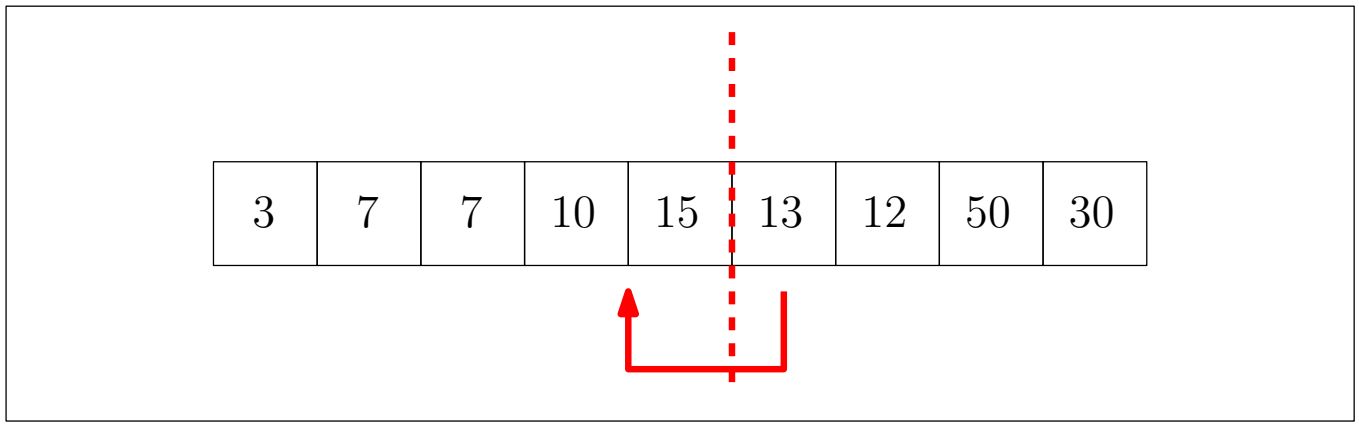
Ταξινόμηση με εισαγωγή
Insertion Sort



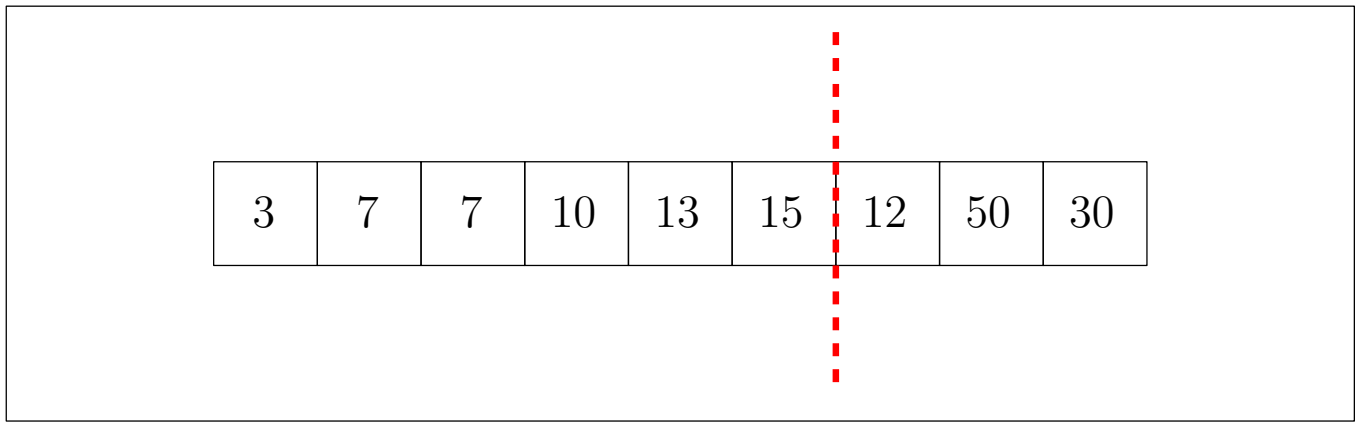
Ταξινόμηση με εισαγωγή
Insertion Sort



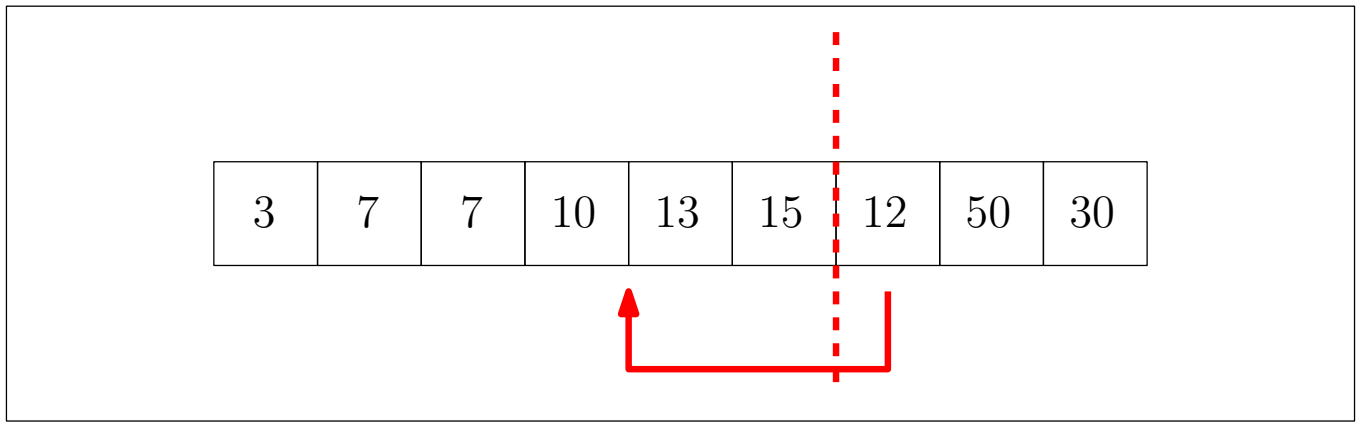
Ταξινόμηση με εισαγωγή
Insertion Sort



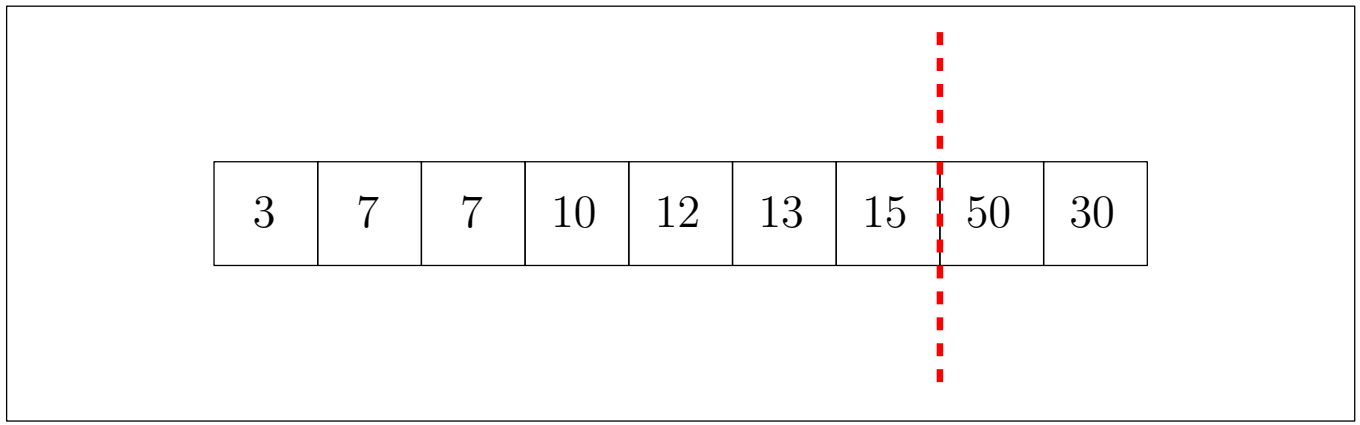
Ταξινόμηση με εισαγωγή
Insertion Sort



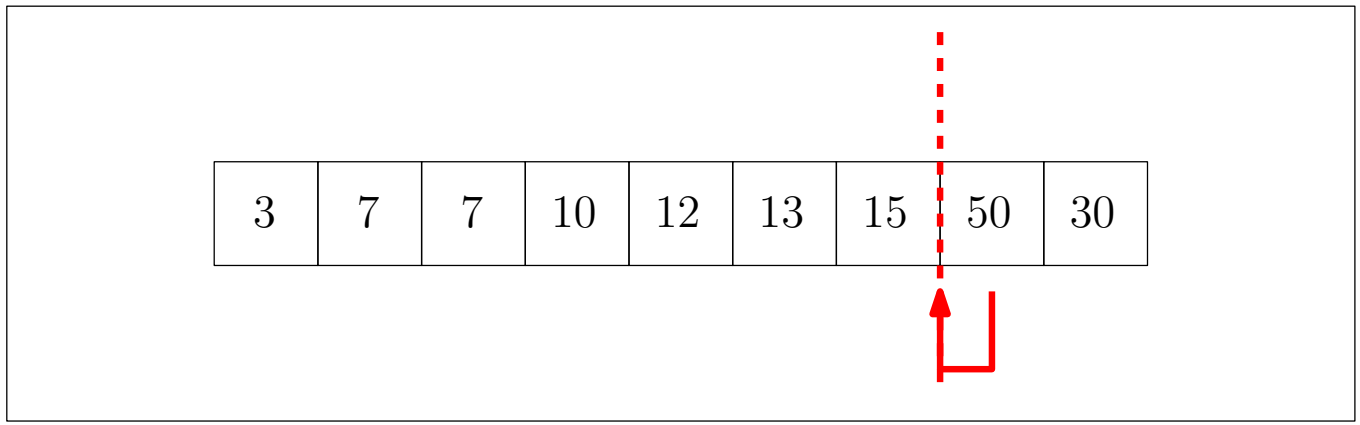
Ταξινόμηση με εισαγωγή
Insertion Sort



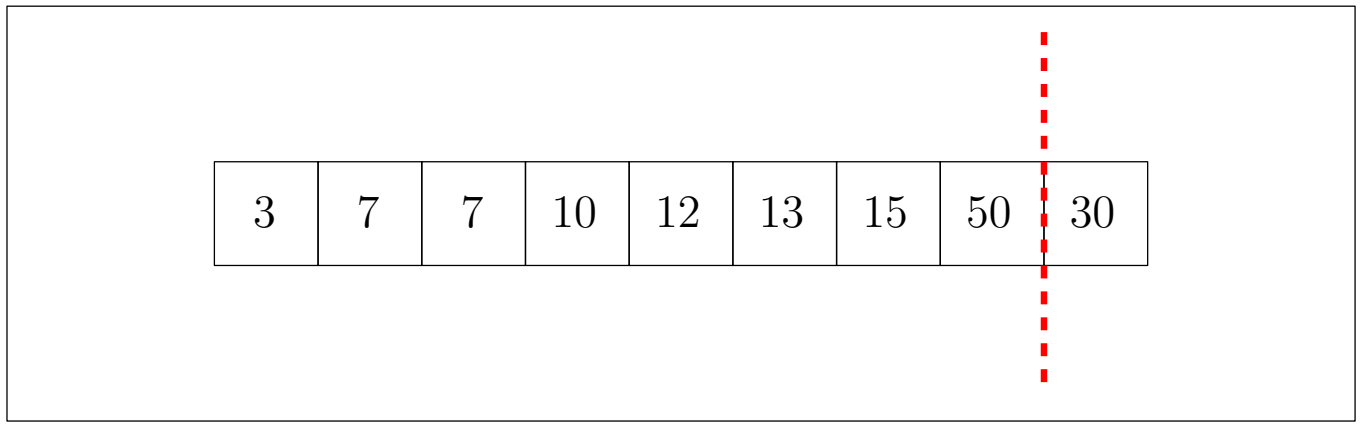
Ταξινόμηση με εισαγωγή
Insertion Sort



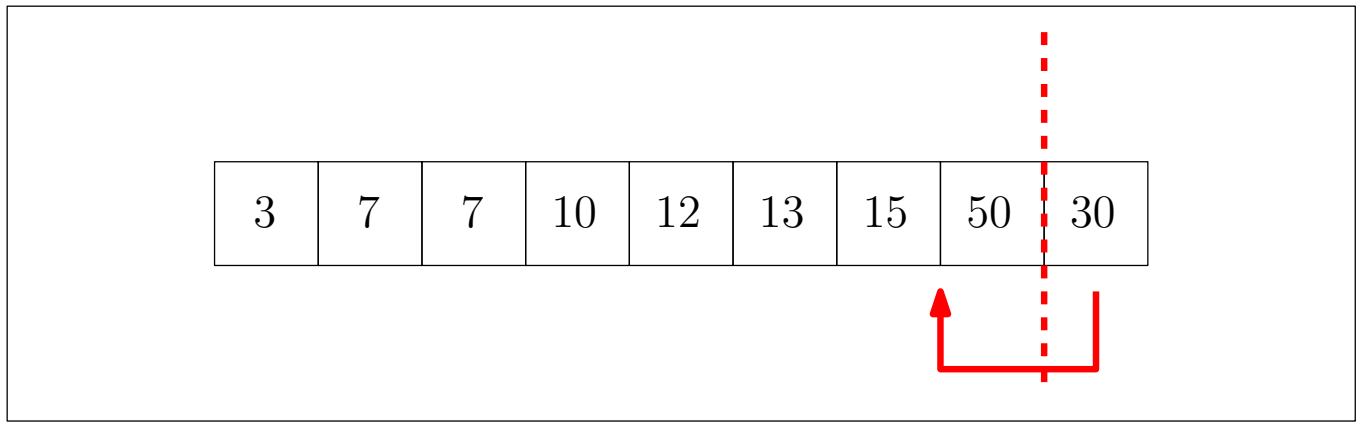
Ταξινόμηση με εισαγωγή
Insertion Sort



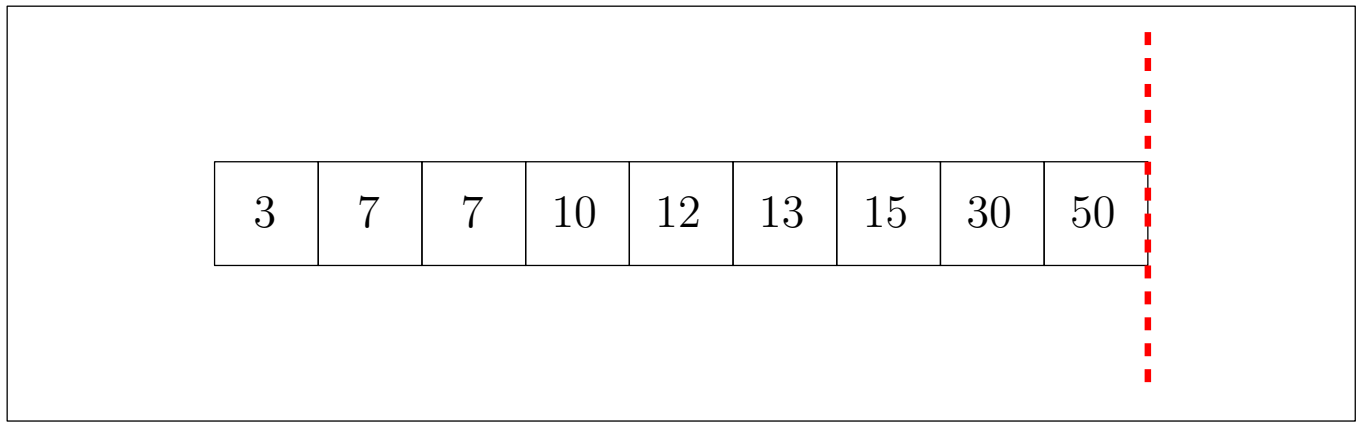
Ταξινόμηση με εισαγωγή
Insertion Sort



Ταξινόμηση με εισαγωγή
Insertion Sort



Ταξινόμηση με εισαγωγή
Insertion Sort



Χρόνος Εκτέλεσης Ταξινόμησης με Εισαγωγή

για το i στοιχείο κάνουμε στην χειρότερη περίπτωση $i - 1$ συγκρίσεις

Χρόνος Εκτέλεσης Ταξινόμησης με Εισαγωγή

για το i στοιχείο κάνουμε στην χειρότερη περίπτωση $i-1$ συγκρίσεις

Συνολικός αριθμός συγκρίσεων:

$$1+2+\ldots+(n-1) = n(n-1)/2 = O(n^2)$$

Ταξινόμηση με εισαγωγή

- Απλή στην υλοποίηση
- Πολύ γρήγορη για πολύ μικρές εισόδους (καλή επιλογή για βάση αναδρομής σε αναδρομικούς αλγορίθμους)
- Πιο αποδοτική από άλλους τετραγωνικούς αλγορίθμους
- Καλή για εισόδους που είναι ήδη μερικώς ταξινομημένες
- Ευσταθής, σε-θέση και καλή για λίστες

Ταξινόμηση με εισαγωγή

Δείξε μου τον κώδικα!

```
void insertion_sort(int *a, int n) {
    int i, j, t;
    for(i=1; i<n; i++) {
        t = a[i];
        for (j=i; j>0 && a[j-1] > t; j--) {
            a[j] = a[j-1];
        }
        a[j] = t;
    }
}
```

Πολύ αποδοτικός κώδικας από το βιβλίο:

- Bentley, Jon (2000), Programming Pearls, ACM Press/Addison–Wesley, pp. 116, 121.

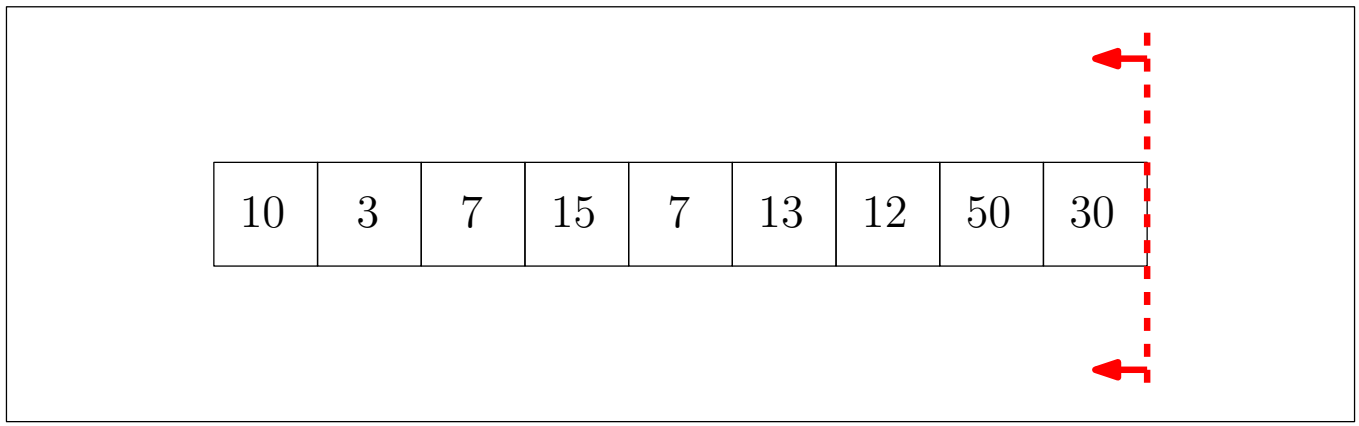
Ταξινόμησης Φυσαλίδας

Bubble Sort

Σαρώνουμε τον πίνακα αντιμεταθέτοντας γειτονικά στοιχεία που βρίσκονται σε λάθος σειρά, μέχρι να μη μπορούμε να αντιμεταθέσουμε τίποτα.

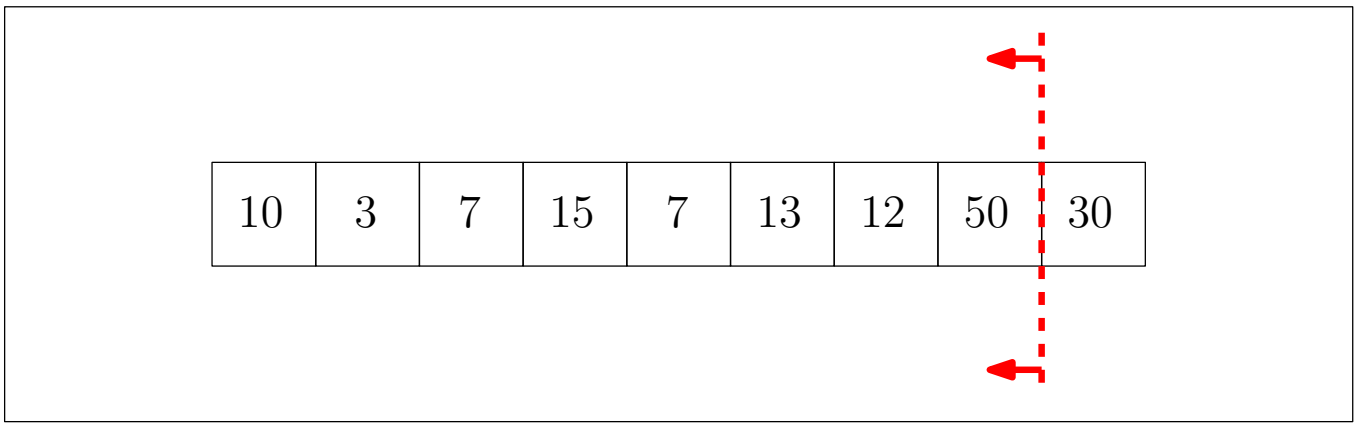
Ταξινόμησης Φυσαλίδας

Bubble Sort



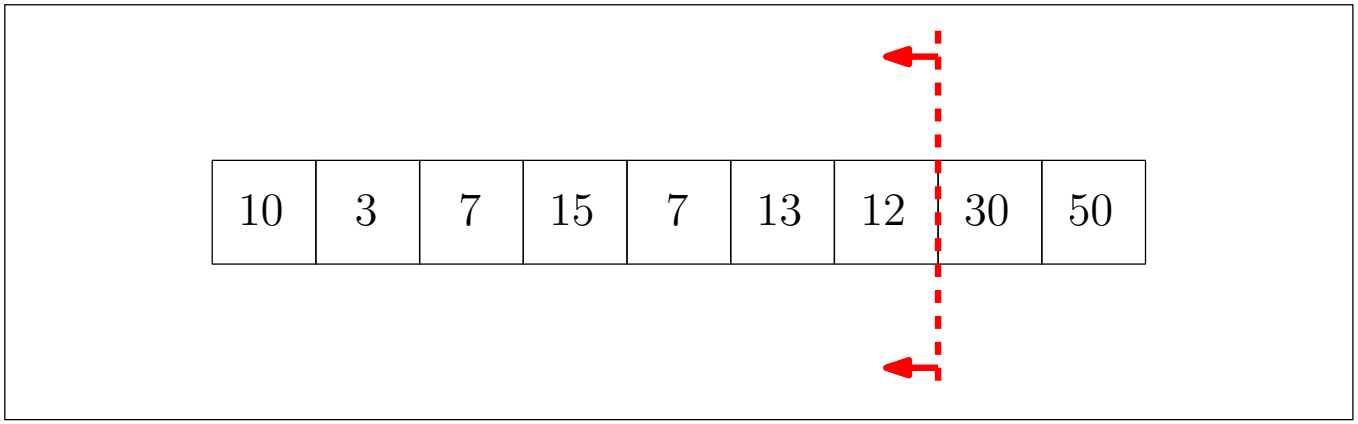
Ταξινόμησης Φυσαλίδας

Bubble Sort



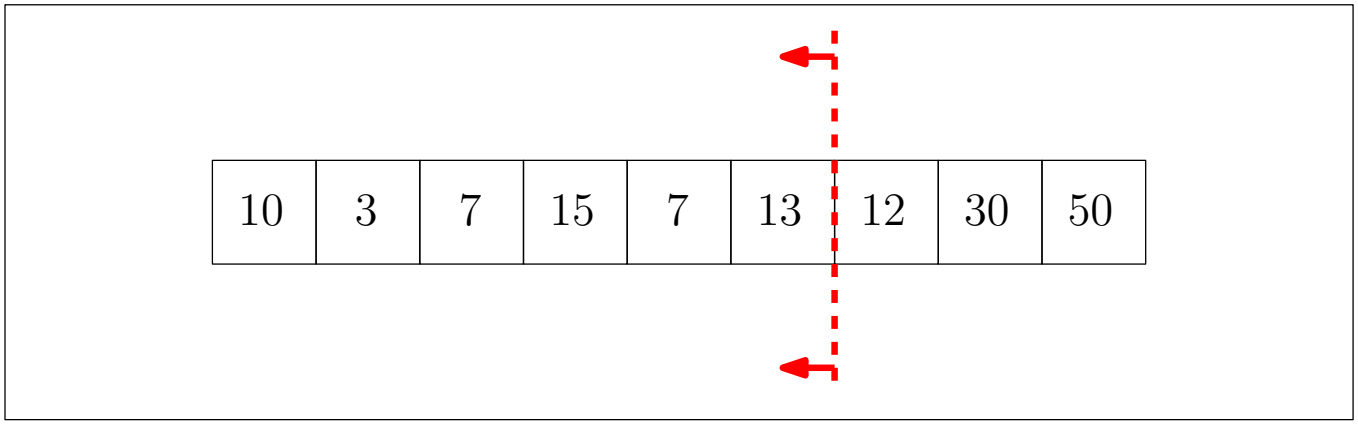
Ταξινόμησης Φυσαλίδας

Bubble Sort



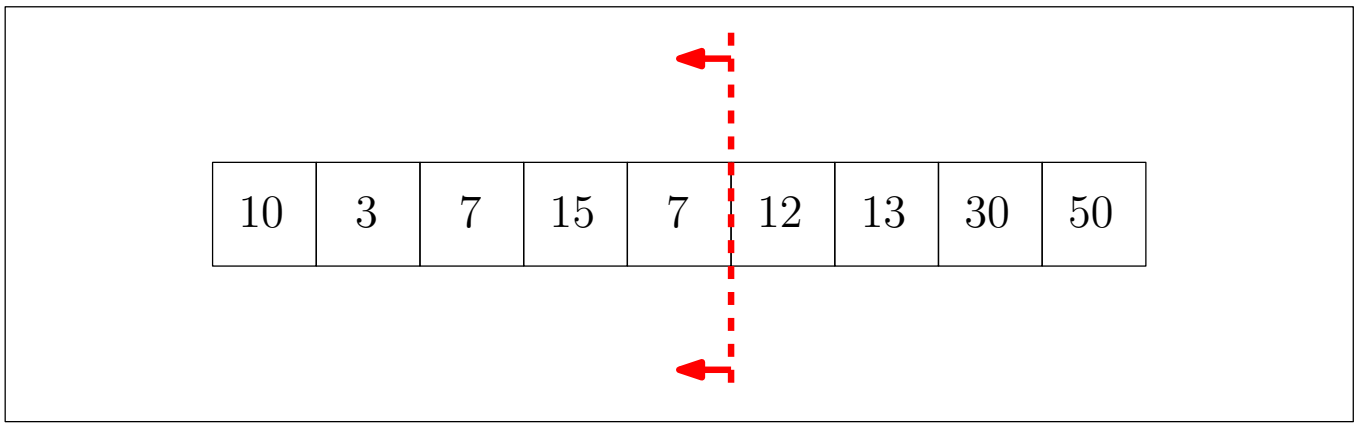
Ταξινόμησης Φυσαλίδας

Bubble Sort



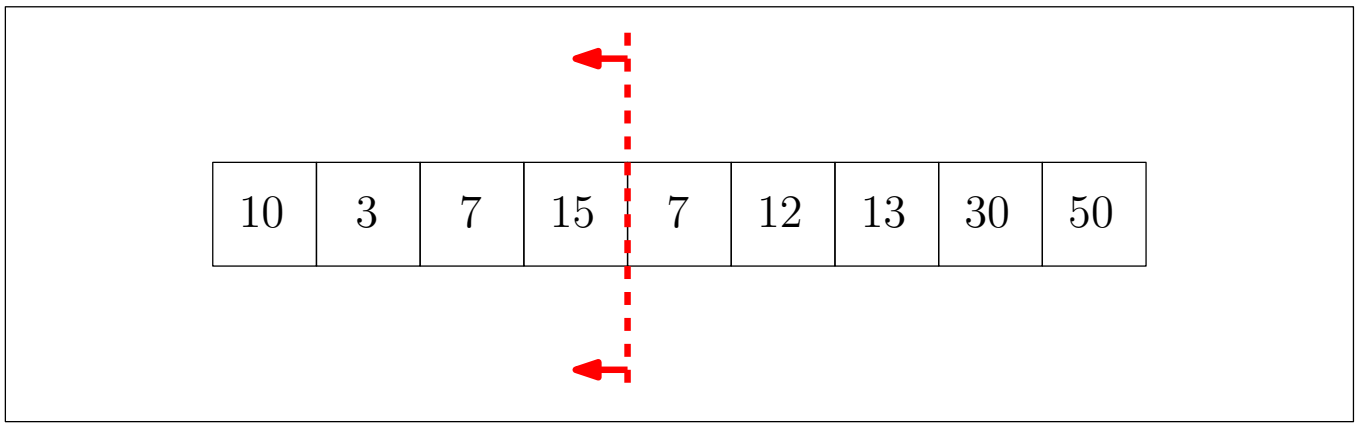
Ταξινόμησης Φυσαλίδας

Bubble Sort



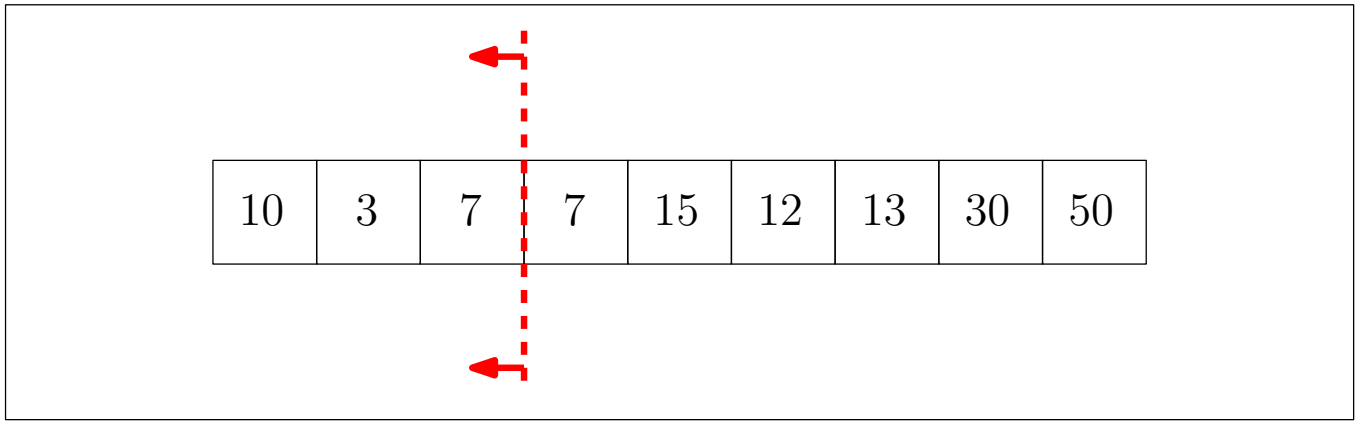
Ταξινόμησης Φυσαλίδας

Bubble Sort



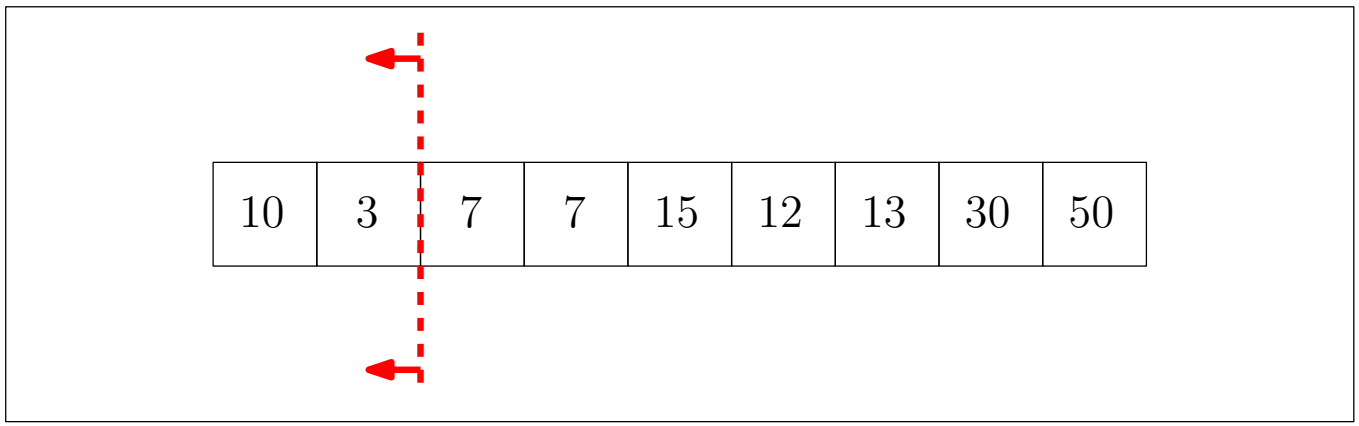
Ταξινόμησης Φυσαλίδας

Bubble Sort



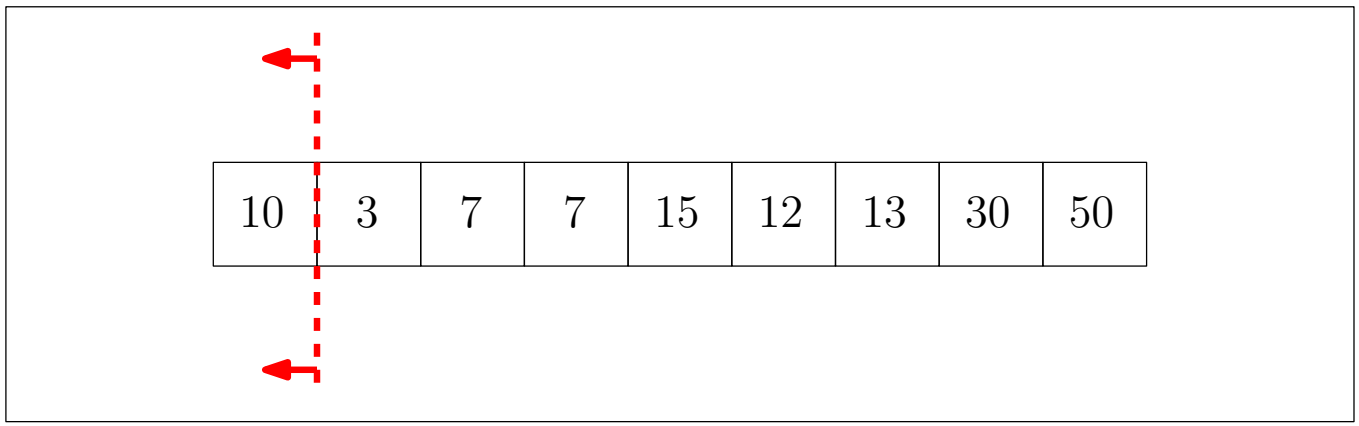
Ταξινόμησης Φυσαλίδας

Bubble Sort



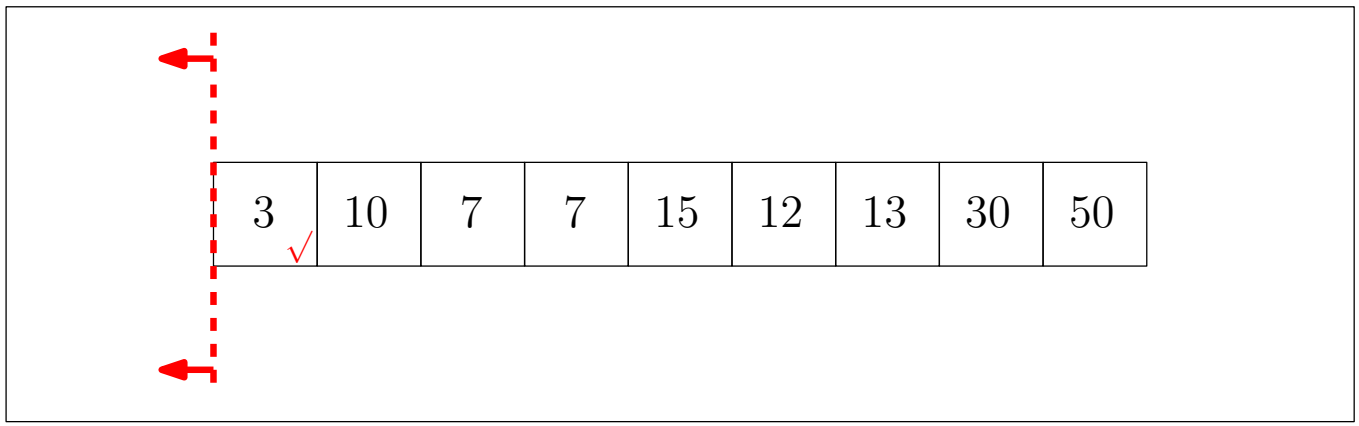
Ταξινόμησης Φυσαλίδας

Bubble Sort



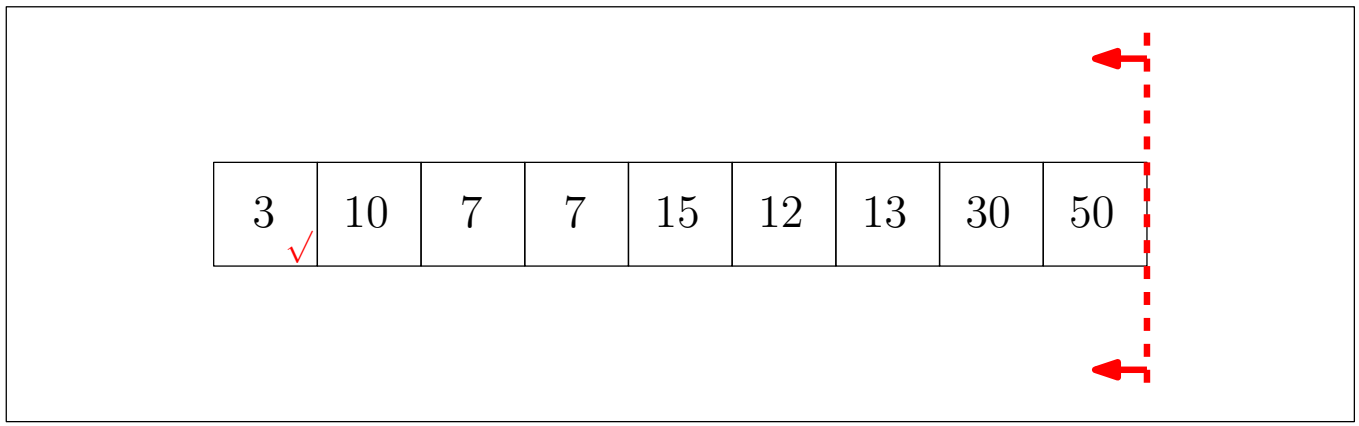
Ταξινόμησης Φυσαλίδας

Bubble Sort



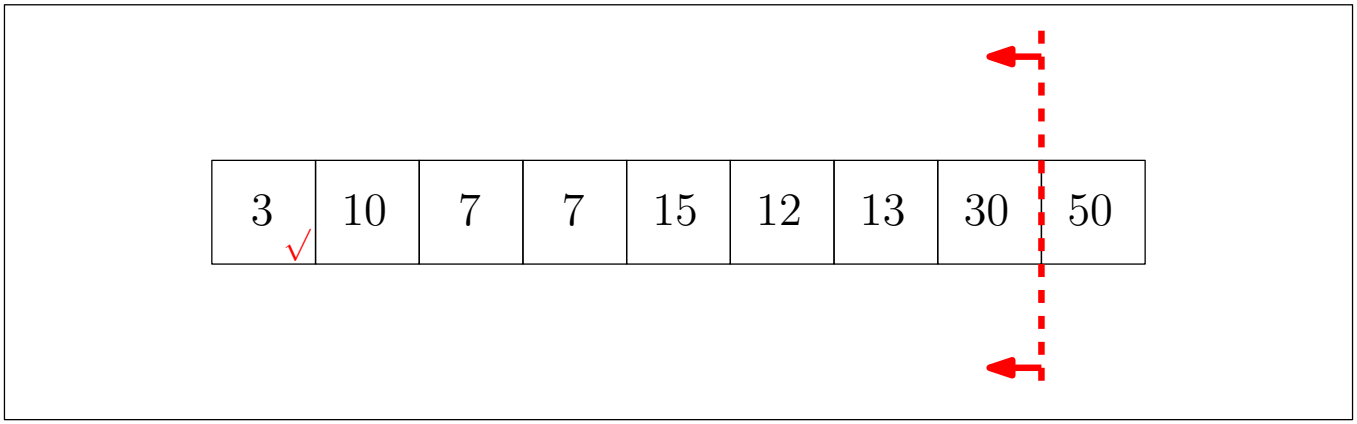
Ταξινόμησης Φυσαλίδας

Bubble Sort



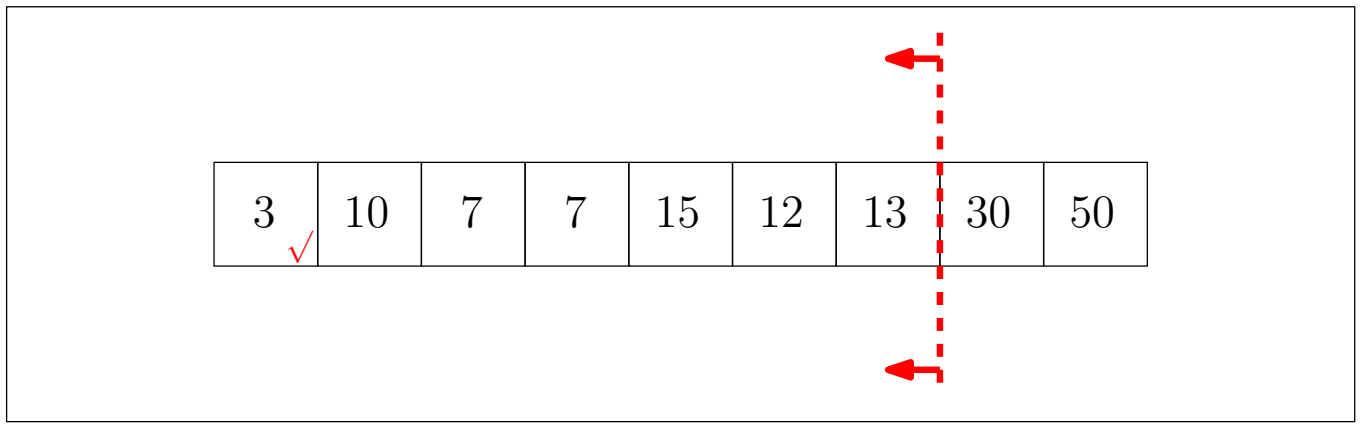
Ταξινόμησης Φυσαλίδας

Bubble Sort



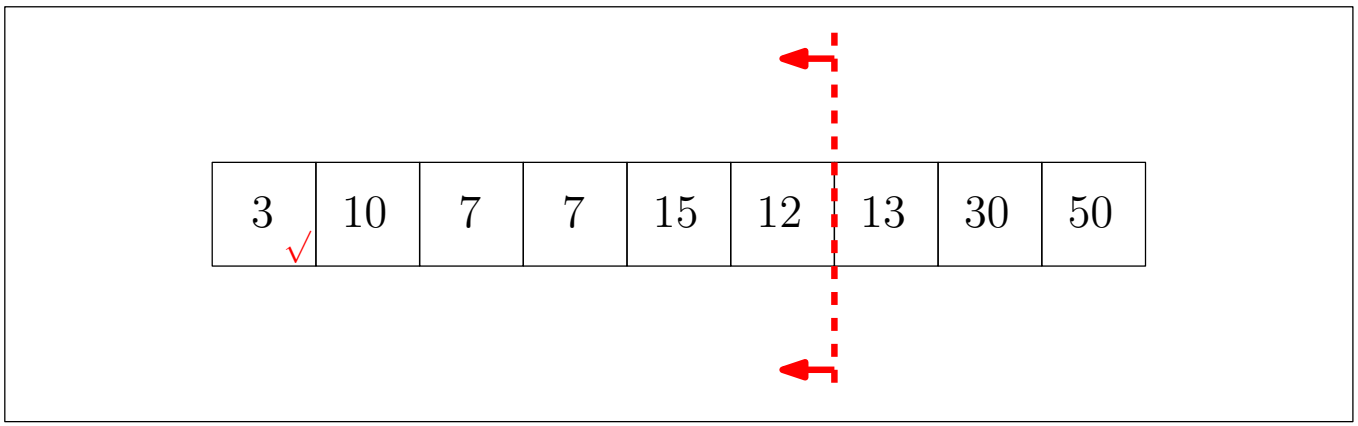
Ταξινόμησης Φυσαλίδας

Bubble Sort



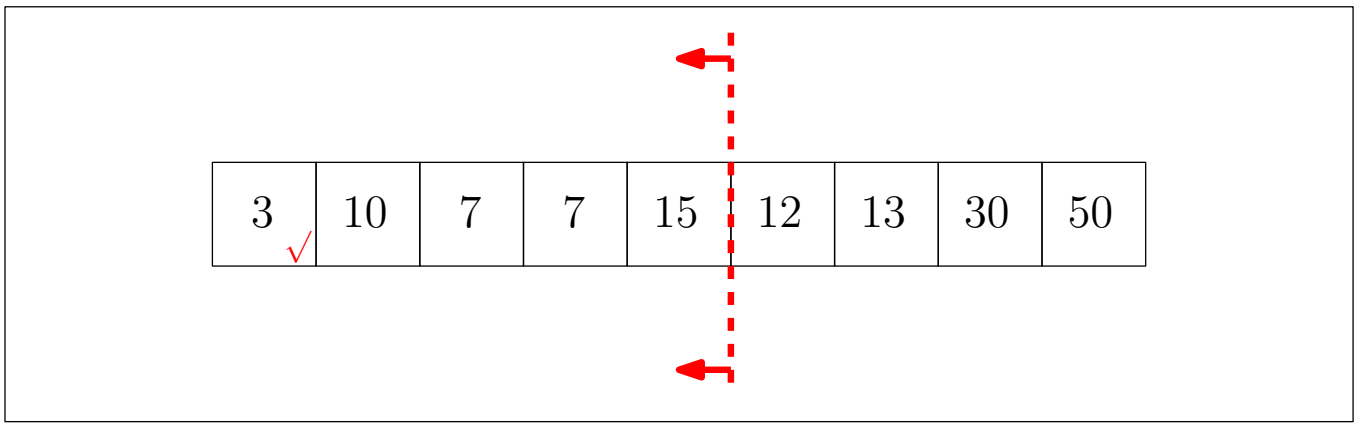
Ταξινόμησης Φυσαλίδας

Bubble Sort



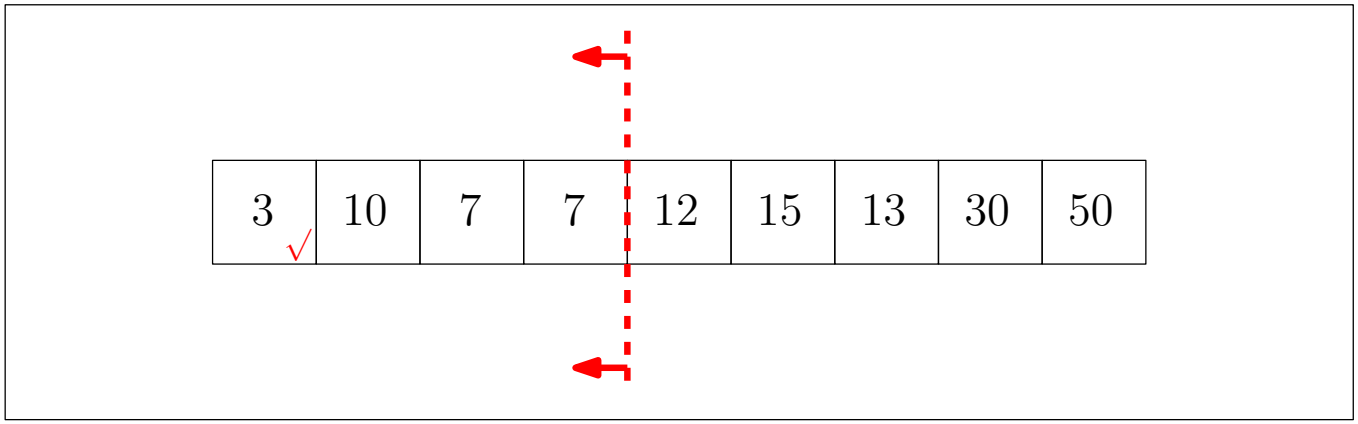
Ταξινόμησης Φυσαλίδας

Bubble Sort



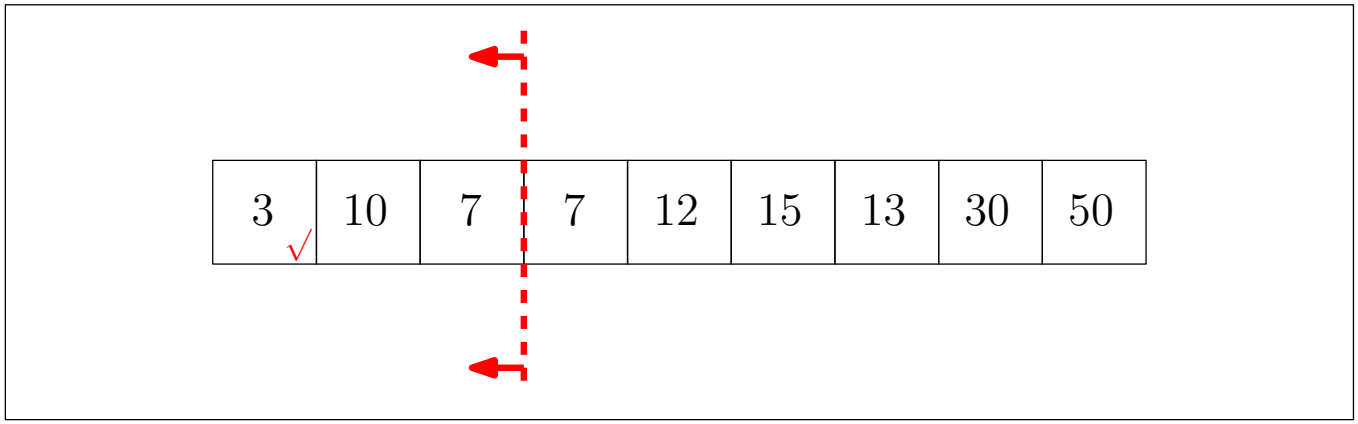
Ταξινόμησης Φυσαλίδας

Bubble Sort



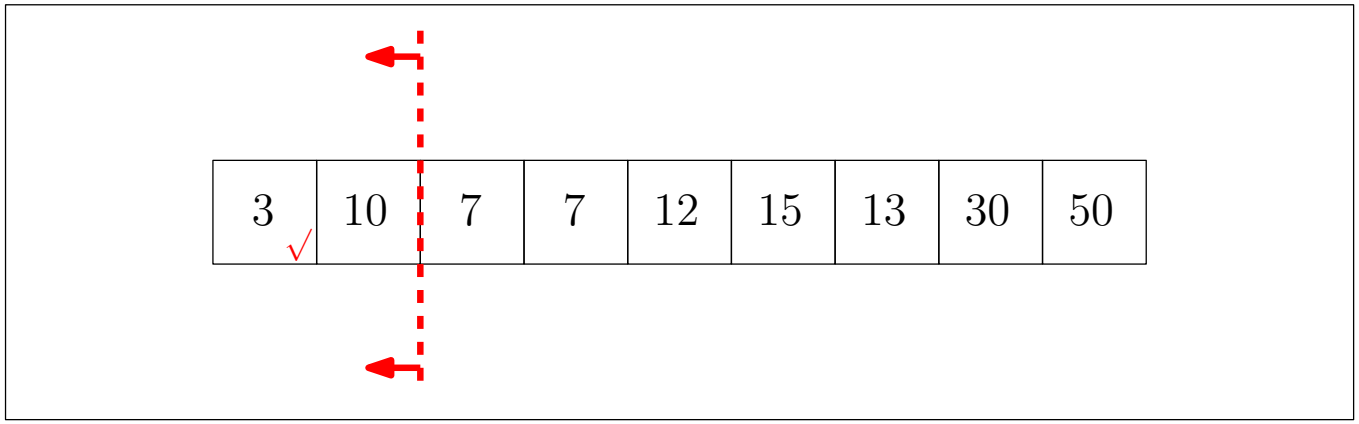
Ταξινόμησης Φυσαλίδας

Bubble Sort



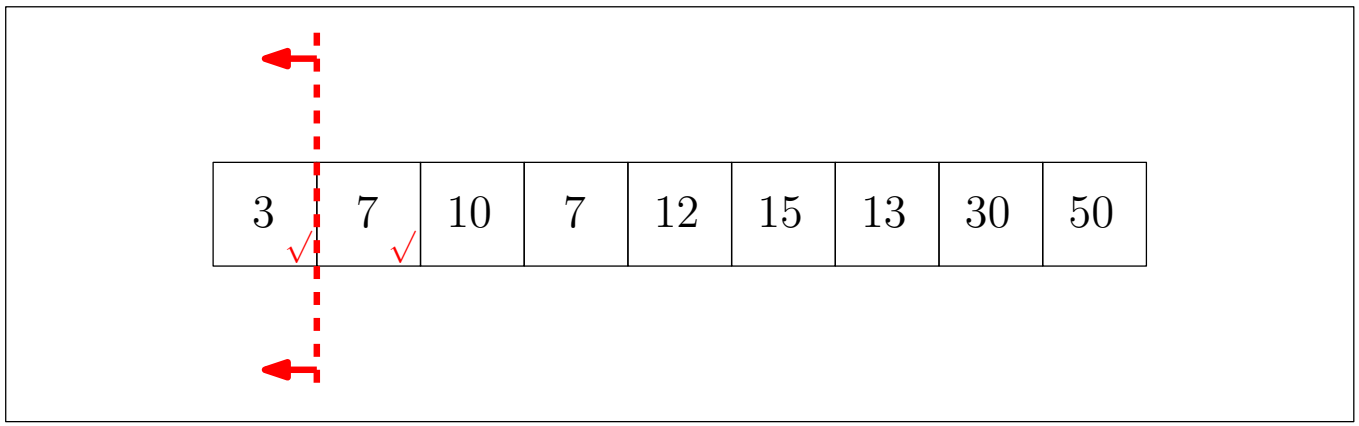
Ταξινόμησης Φυσαλίδας

Bubble Sort



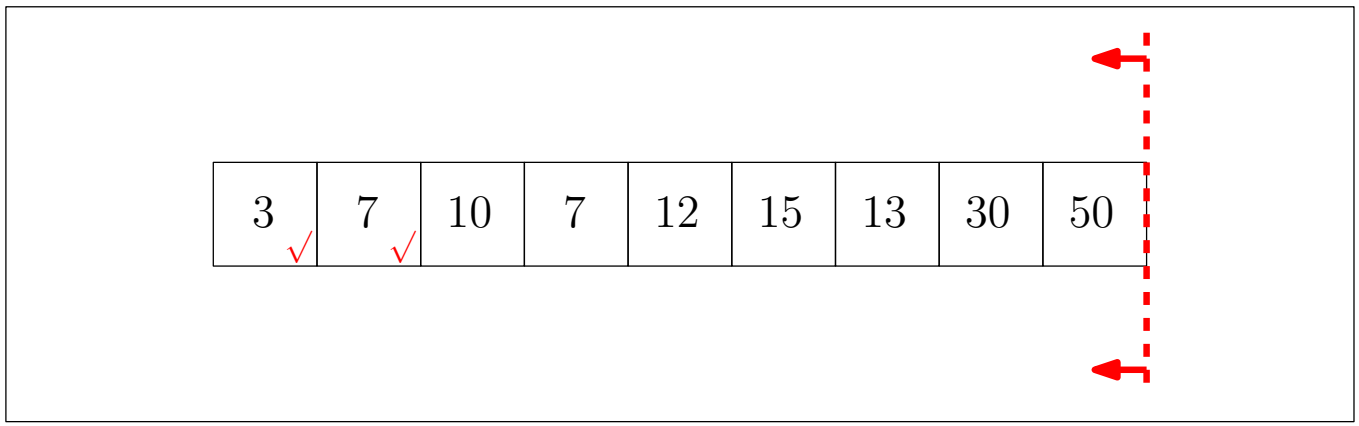
Ταξινόμησης Φυσαλίδας

Bubble Sort



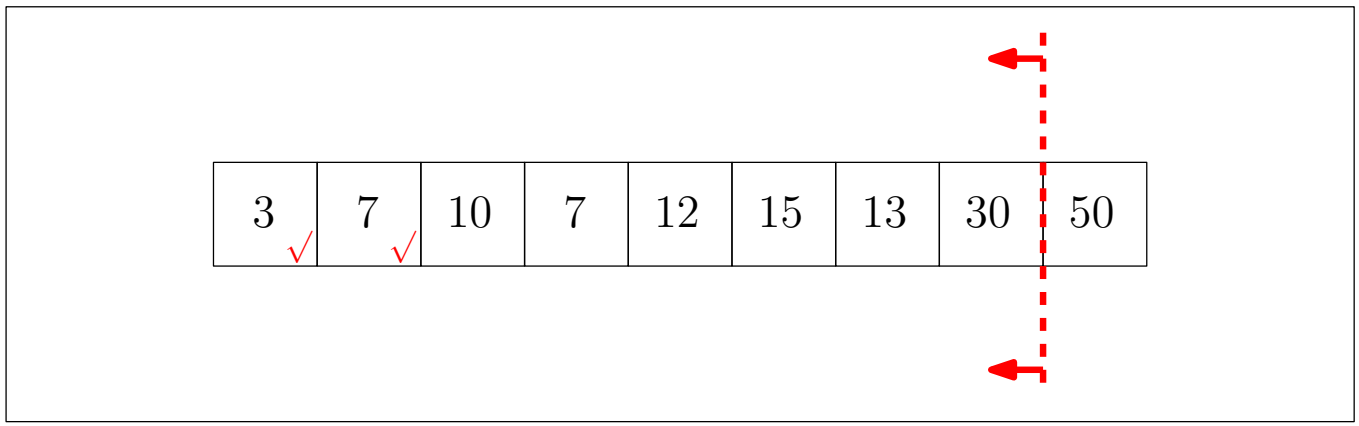
Ταξινόμησης Φυσαλίδας

Bubble Sort



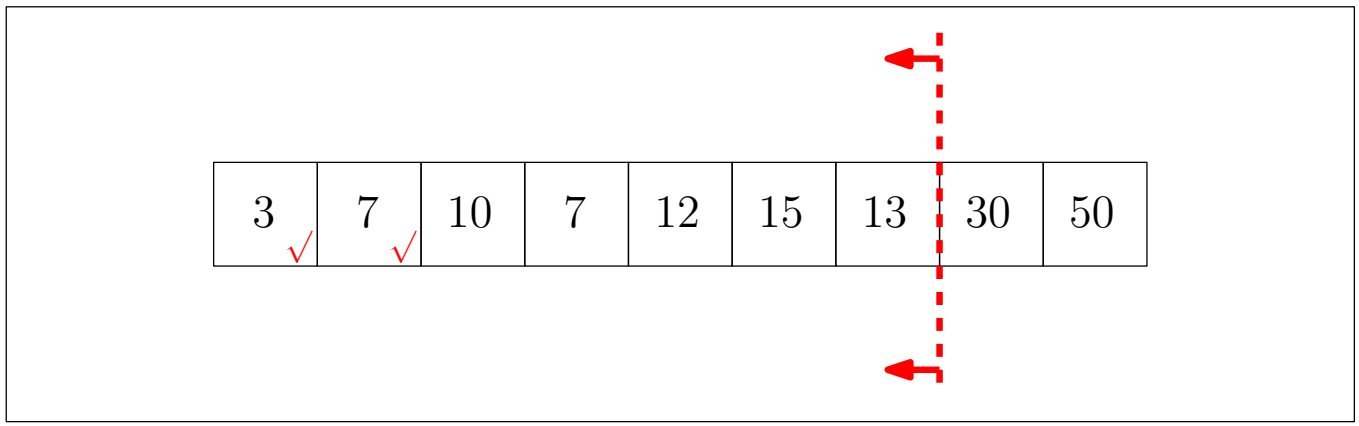
Ταξινόμησης Φυσαλίδας

Bubble Sort



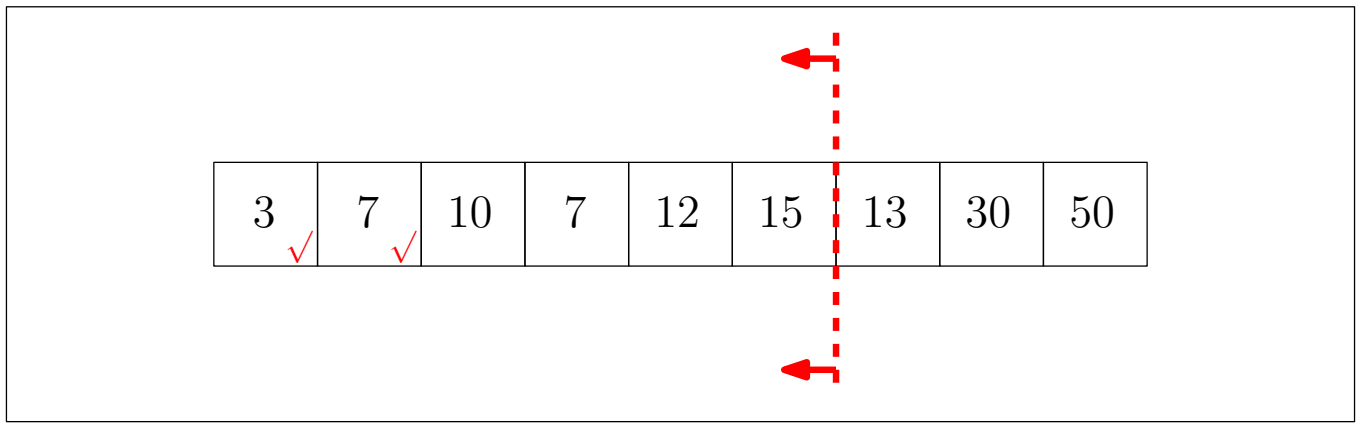
Ταξινόμησης Φυσαλίδας

Bubble Sort



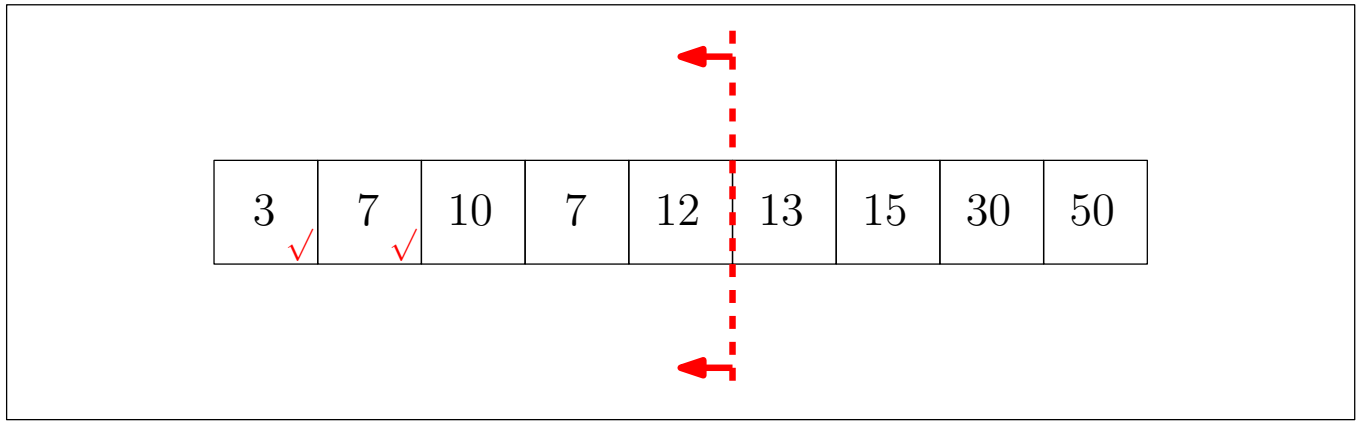
Ταξινόμησης Φυσαλίδας

Bubble Sort



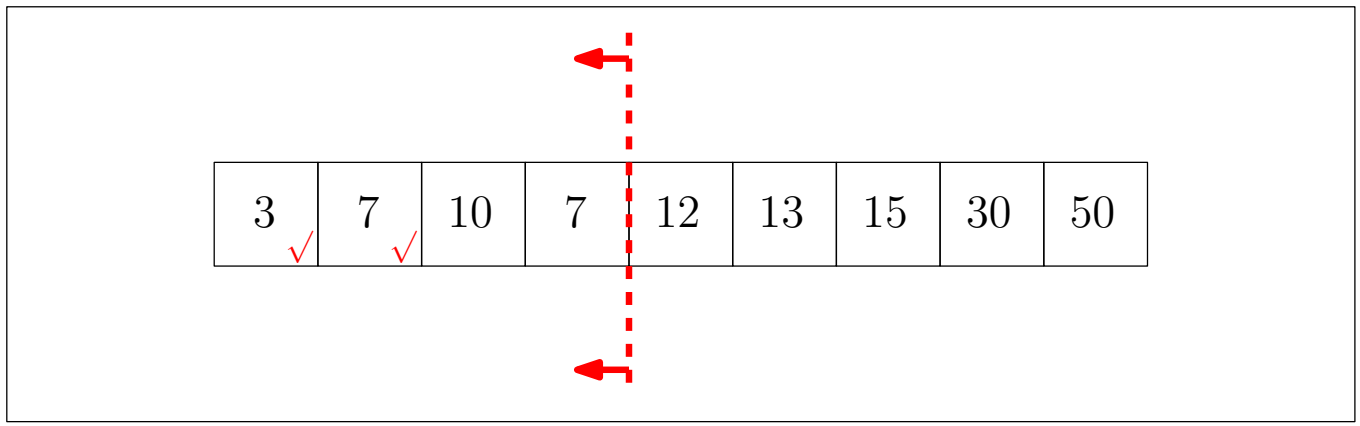
Ταξινόμησης Φυσαλίδας

Bubble Sort



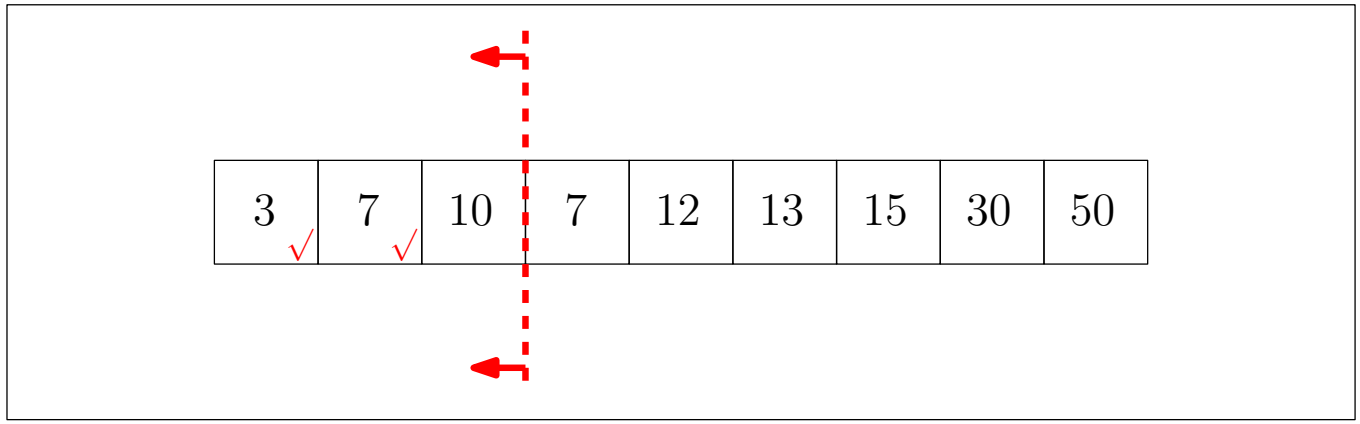
Ταξινόμησης Φυσαλίδας

Bubble Sort



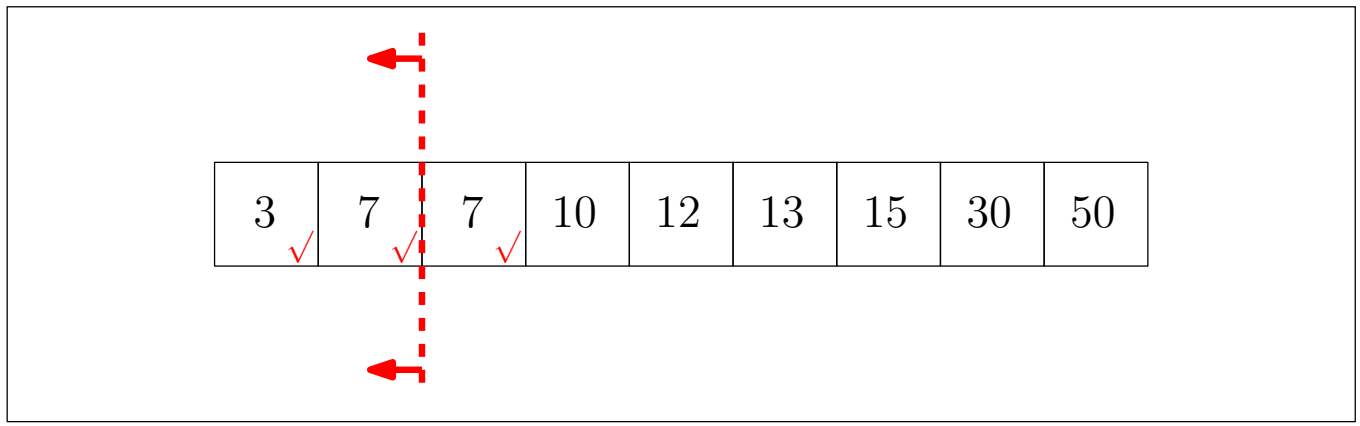
Ταξινόμησης Φυσαλίδας

Bubble Sort



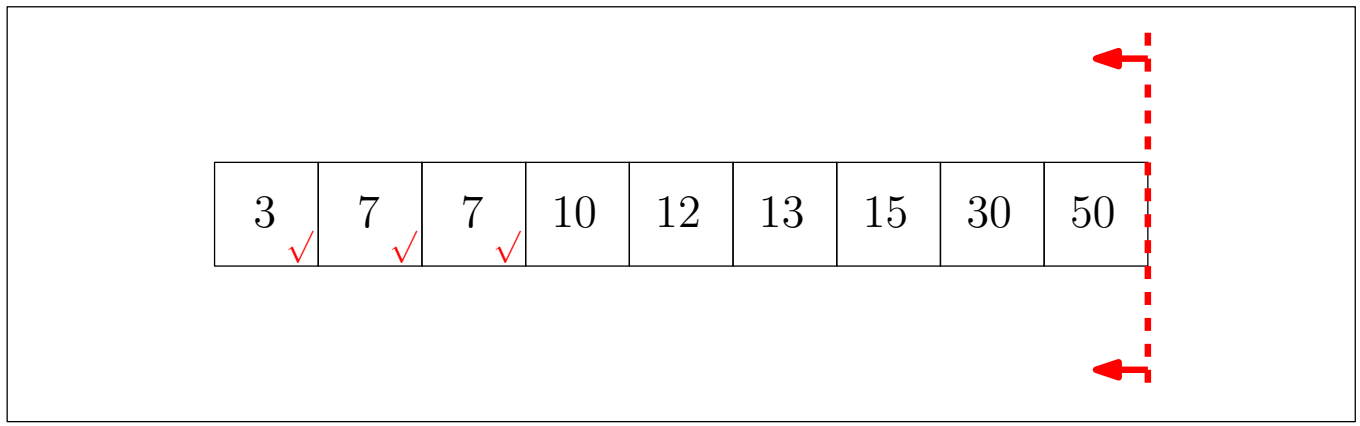
Ταξινόμησης Φυσαλίδας

Bubble Sort



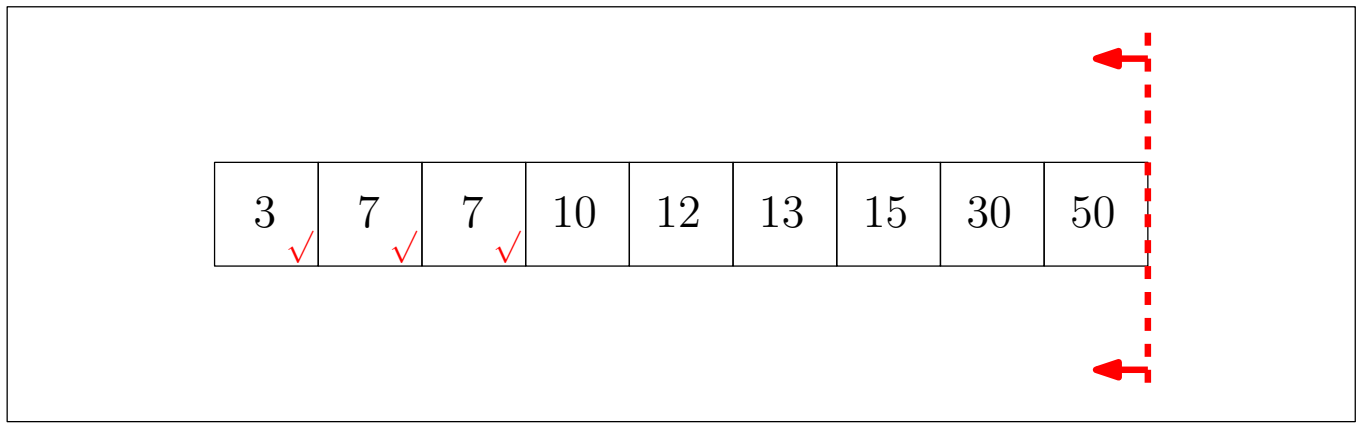
Ταξινόμησης Φυσαλίδας

Bubble Sort



Ταξινόμησης Φυσαλίδας

Bubble Sort



Στο πέρασμα *i* έχουμε βάλει στην σωστή θέση τουλάχιστον τα *i* μικρότερα κλειδιά. Στην χειρότερη περίπτωση πρέπει για να κάνουμε *n* περάσματα (εάν η ακολουθία εισόδου είναι φθίνουσα).

Χρόνος Εκτέλεσης Ταξινόμησης Φυσαλίδας

στο i πέρασμα κάνουμε $n - i$ συγκρίσεις και μπορεί να πρέπει να κάνουμε n περάσματα

Χρόνος Εκτέλεσης Ταξινόμησης Φυσαλίδας

στο i πέρασμα κάνουμε $n - i$ συγκρίσεις και μπορεί να πρέπει να κάνουμε n περάσματα

Συνολικός αριθμός συγκρίσεων:

$$1 + 2 + \ldots + (n-1) = n(n-1)/2 = O(n^2)$$

Ταξινόμηση Φυσσαλίδας

Δείξε μου τον κώδικα!

```
void bubble_sort(int *a, int n) {
    int i, t, reached;

    do{
        reached = 0;
        for(i=1;i<n;i++) {
            if(a[i-1]>a[i]) {
                t = a[i];
                a[i] = a[i-1];
                a[i-1] = t;
                reached = i;
            }
        }
        n = reached;
    }
    while(n>1);
}
```

Στοιχειώδεις Μέθοδοι Ταξινόμησης

$O(n^2)$

- ταξινόμηση επιλογής (selection sort)
- ταξινόμηση εισαγωγής (insertion sort)
- ταξινόμηση φυσαλίδας (bubble sort)

Μειονέκτημα

Για 10^6 κλειδιά χρειάζονται περίπου 10^{12} συγκρίσεις

Θα δούμε πως μπορούμε και καλύτερα..

Γρήγορη Ταξινόμηση

Quicksort

Η γρήγορη ταξινόμηση είναι η πιο συνήθης επιλογής για όλα τα σύγχρονα συστήματα.

Ανήκει σε μια κατηγορία αλγορίθμων που λέγονται ”Διαίρει και Βασίλευε”, μιας και μοιράζει το πρόβλημα σε μικρότερα ανεξάρτητα υποπροβλήματα και τα λύνει αναδρομικά.

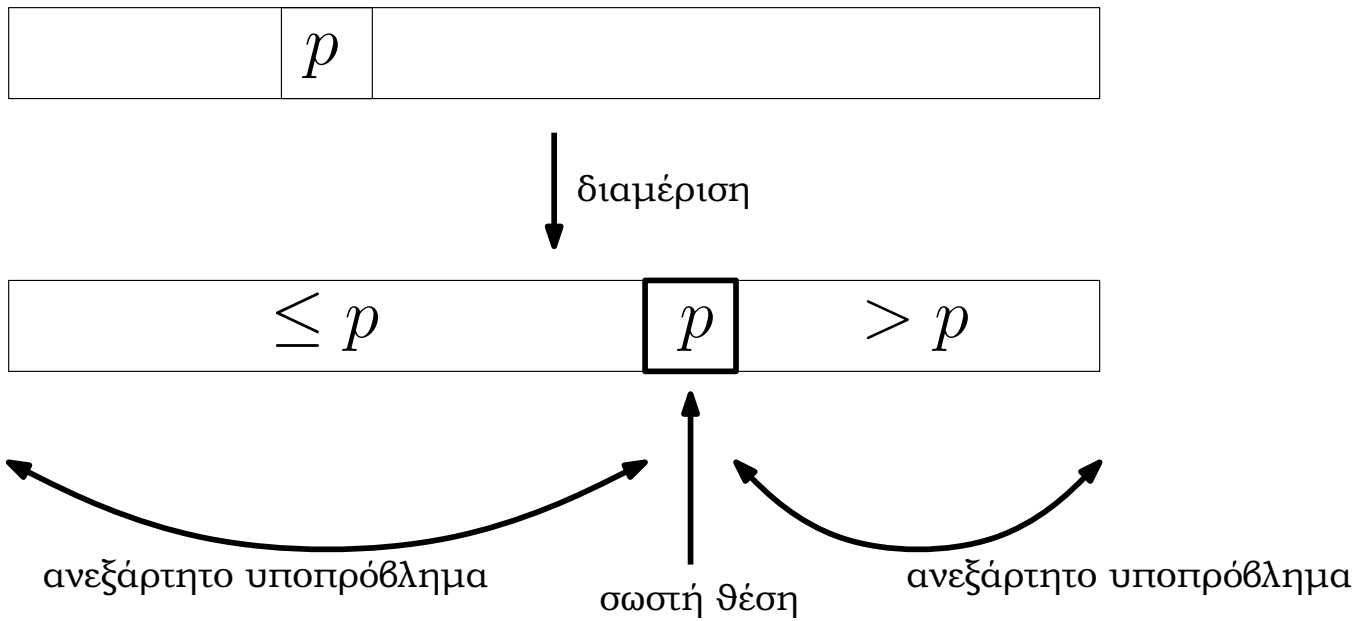
- στην χειρότερη περίπτωση τρέχει σε $O(n^2)$ χρόνο
- στην μέση περίπτωση όμως τρέχει σε χρόνο $O(n \log n)$
- είναι εύκολο να αποφύγουμε την χειρότερη περίπτωση με μεγάλη πιθανότητα

Είναι πάρα πολύ γρήγορη στην πράξη...

- διάλεξε ένα κλειδί του πίνακα $a[0..n-1]$ το οποίο θα είναι το στοιχείο διαμέρισης
- έπειτα αναδιατάσσουμε τα στοιχεία του πίνακα a έτσι ώστε όλα τα κλειδιά που είναι μικρότερα του στοιχείου διαμέρισης να είναι στα αριστερά του και όλα τα μεγαλύτερα κλειδιά στα δεξιά του
- έπειτα ταξινομούμε αναδρομικά το αριστερό και το δεξιό μέρος του πίνακα

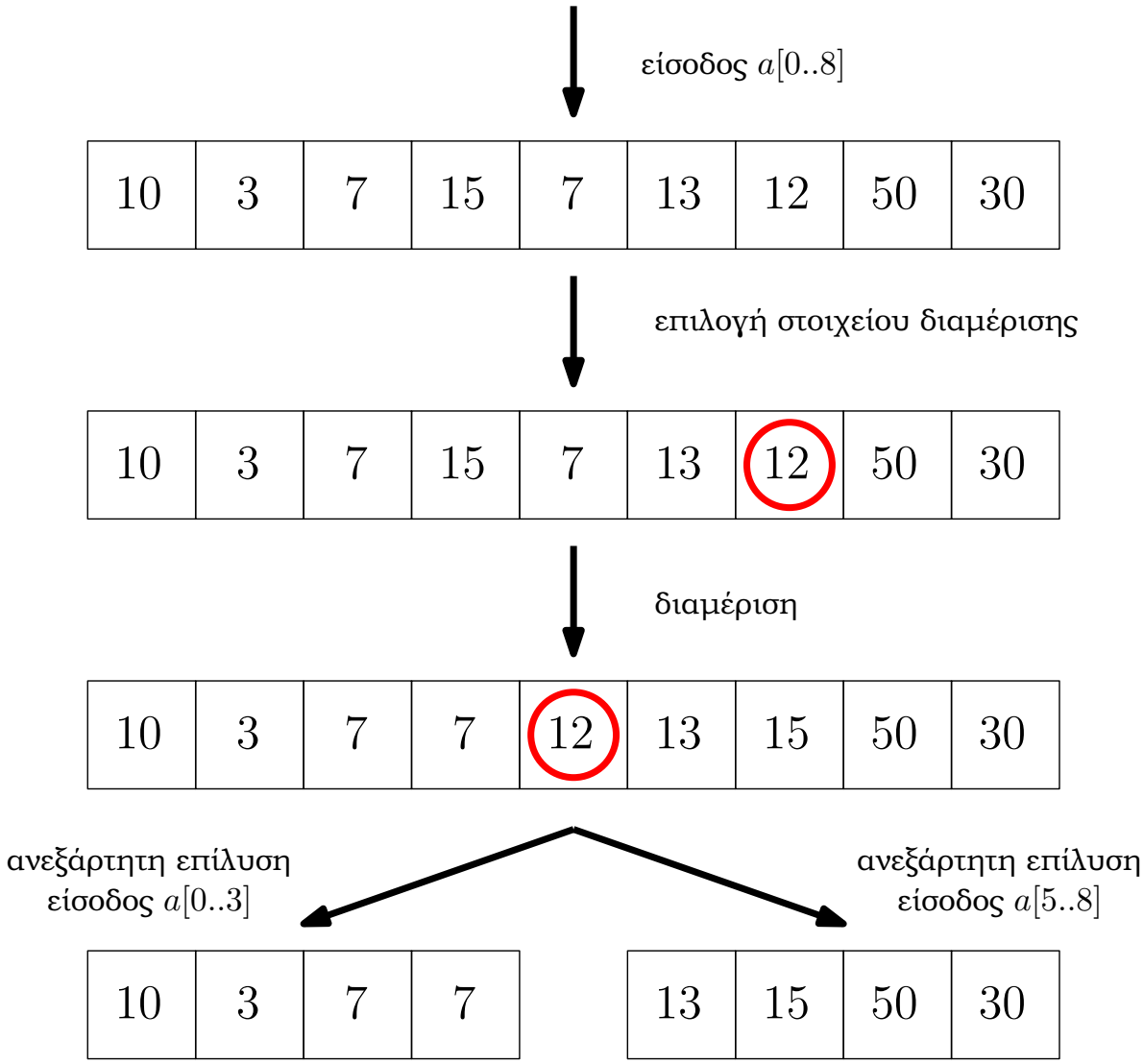
Γρήγορη Ταξινόμηση

Quicksort

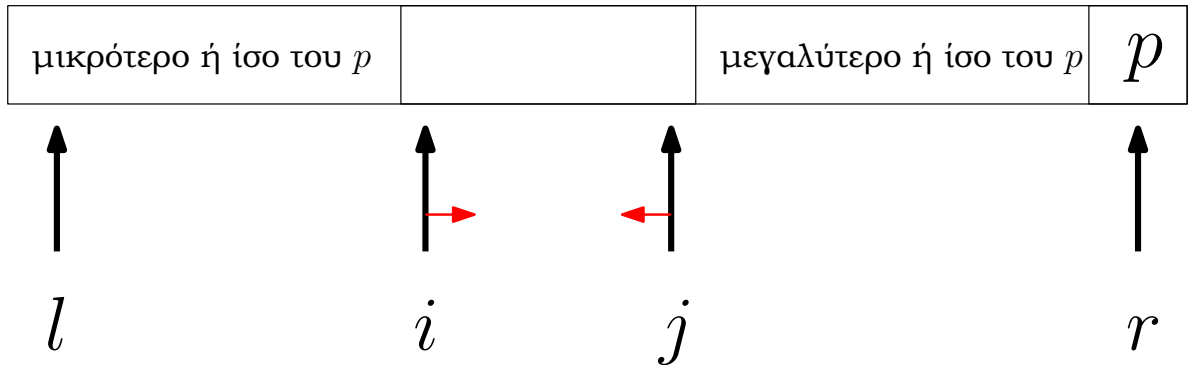


Γρήγορη Ταξινόμηση

Quicksort



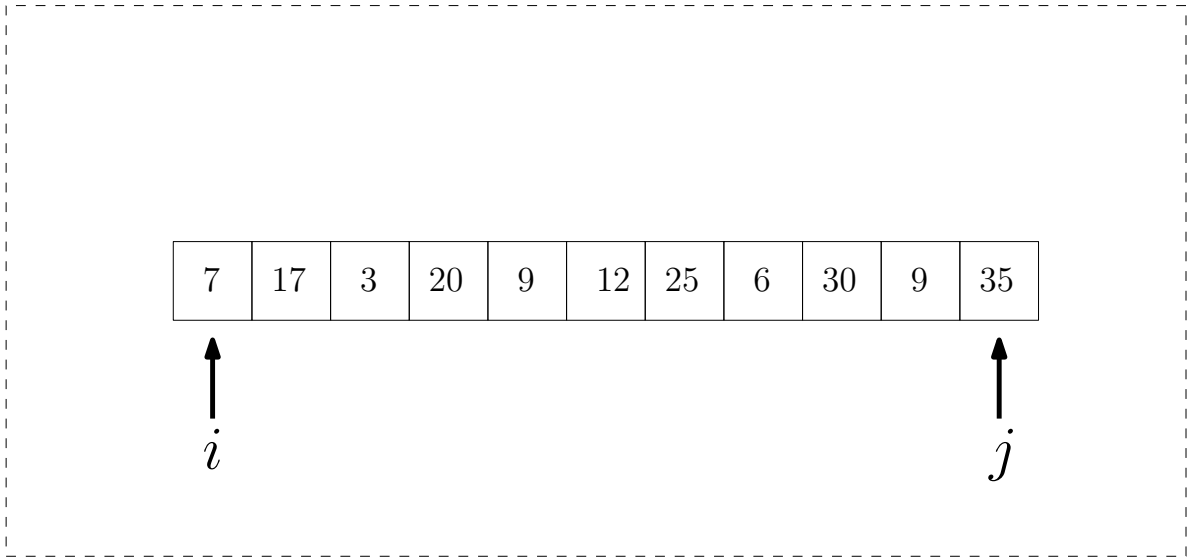
Διαμέριση (partition) σε θέση



Μπορούμε να κάνουμε διαμέριση σε θέση (in place) χρησιμοποιώντας δύο δείκτες *i* και *j*, ένα από αριστερά και ένα από δεξιά.

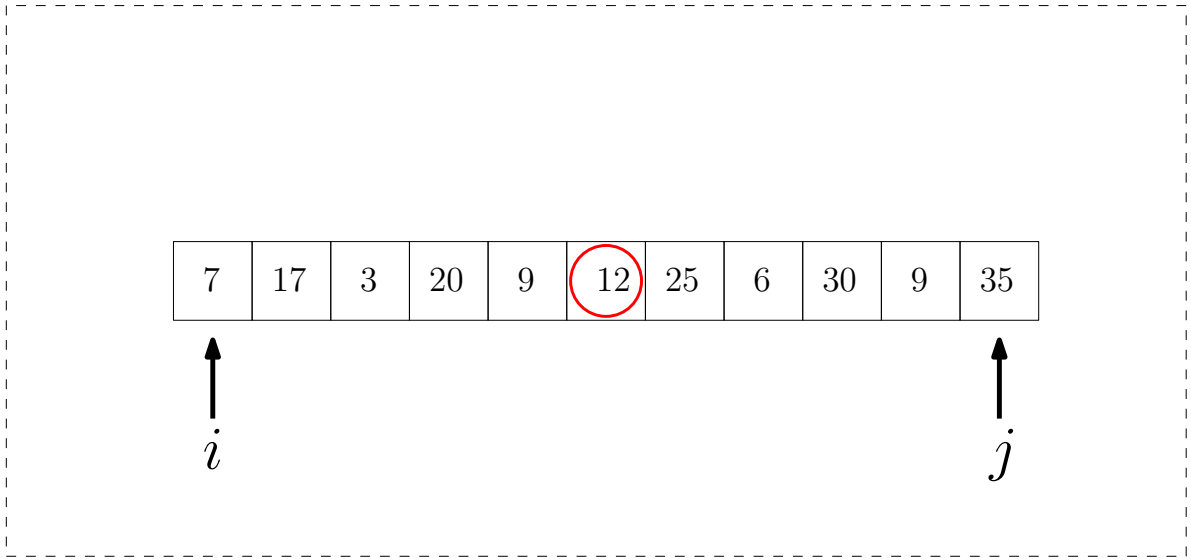
- Κινούνται προς το κέντρο μέχρι ο αριστερός να βρει ένα κλειδί μεγαλύτερο από το κλειδί διαμέρισης και ο δεξιός ένα κλειδί μικρότερο από το κλειδί διαμέρισης.
- Τα κλειδιά αυτά αντιμετατίθενται.
- Η διαδικασία συνεχίζει μέχρι οι δείκτες να συναντηθούν.
- Το στοιχείο διαμέρισης μπαίνει στη θέση του δεξιού δείκτη.

Παράδειγμα Διαμέρισης



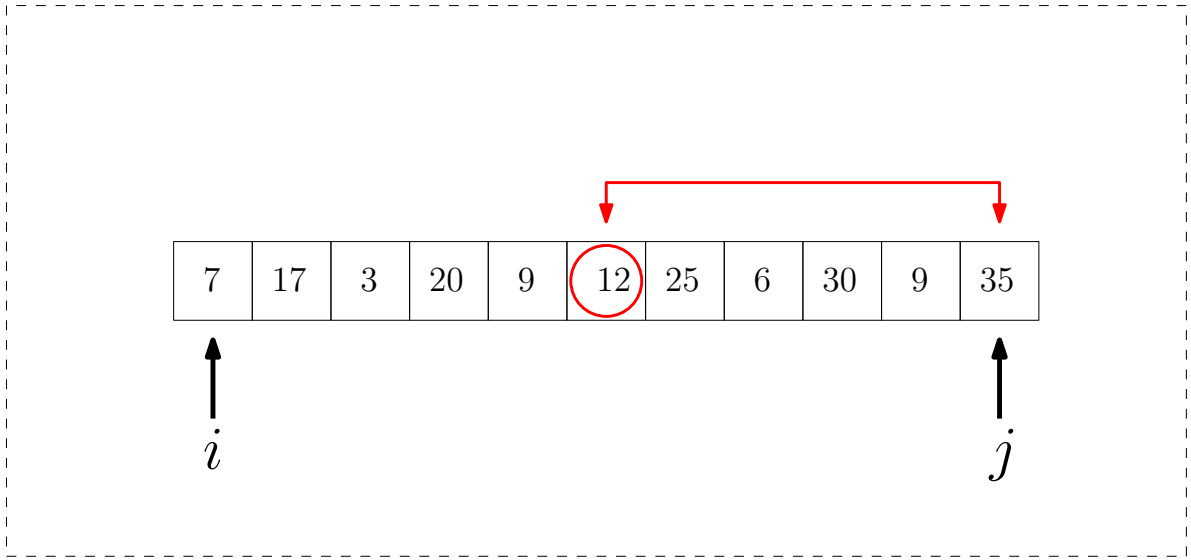
Επιλογή στοιχείου διαμέρισης

Παράδειγμα Διαμέρισης



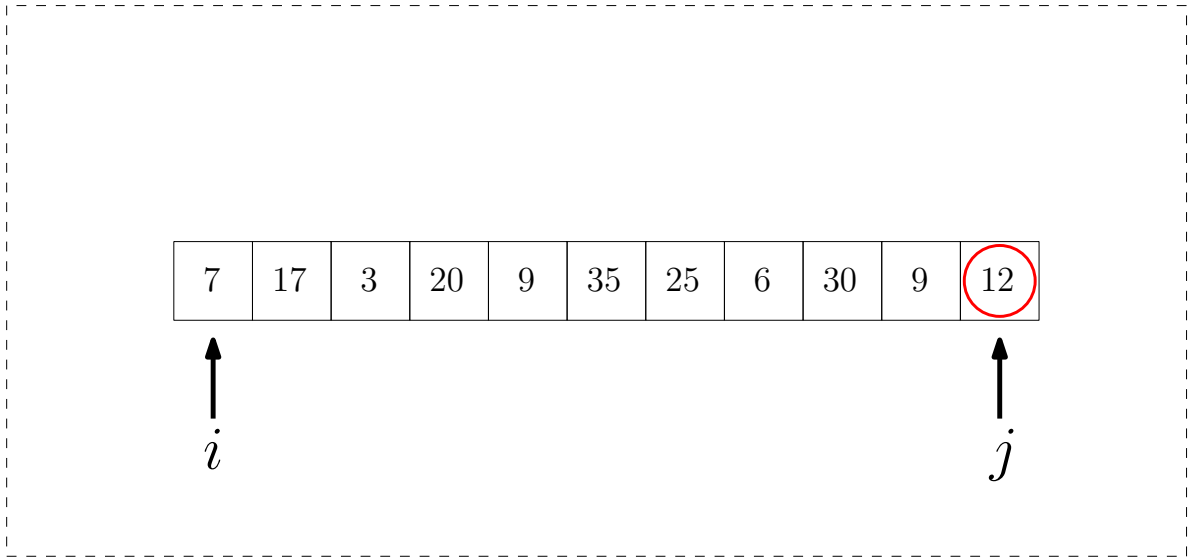
Στοιχείο διαμέρισης το 12

Παράδειγμα Διαμέρισης



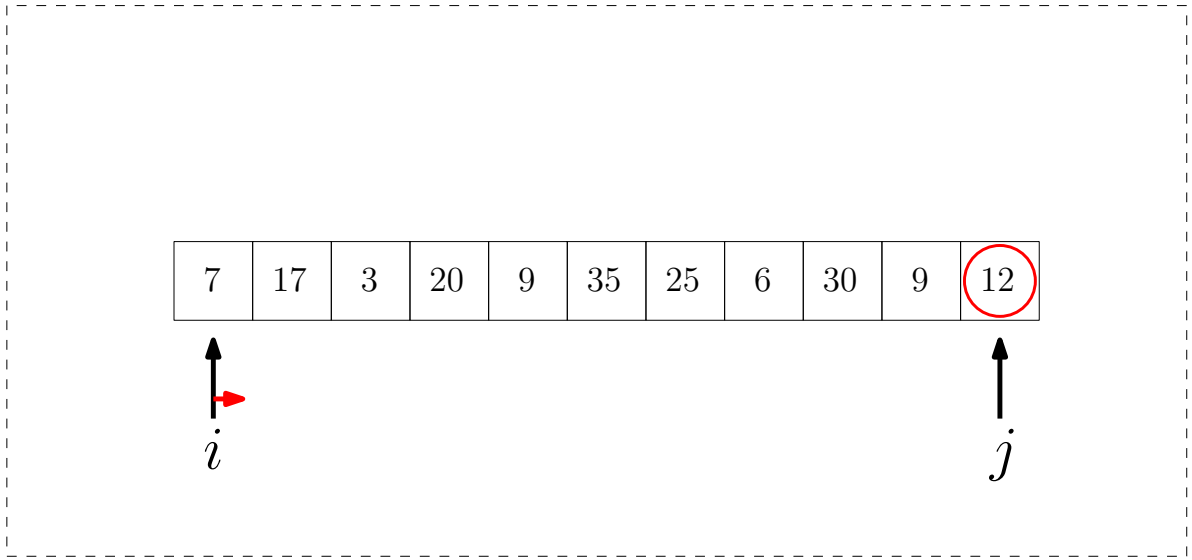
Μετακινούμε το στοιχείο διαμέρισης στο τέλος

Παράδειγμα Διαμέρισης



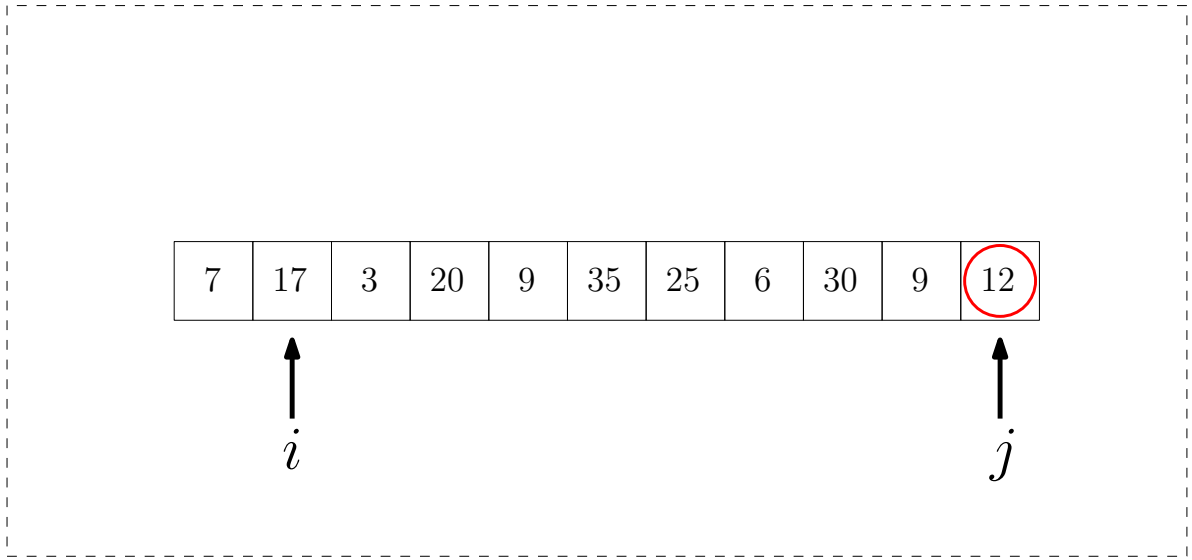
Μετακινούμε το στοιχείο διαμέρισης στο τέλος

Παράδειγμα Διαμέρισης



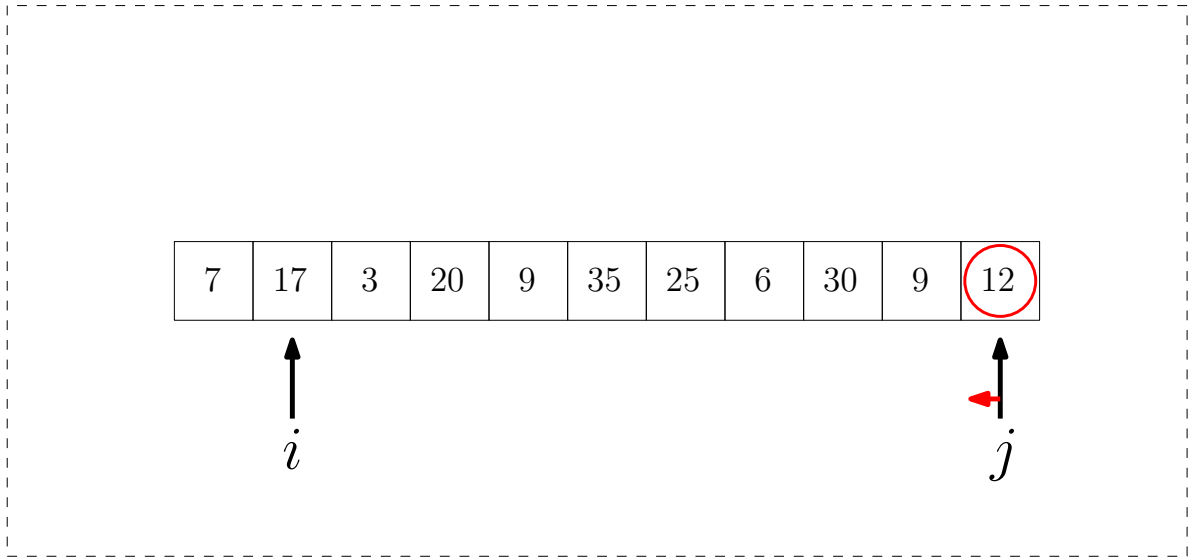
Αναζήτηση στοιχείου μεγαλύτερου του 12 από αριστερά

Παράδειγμα Διαμέρισης



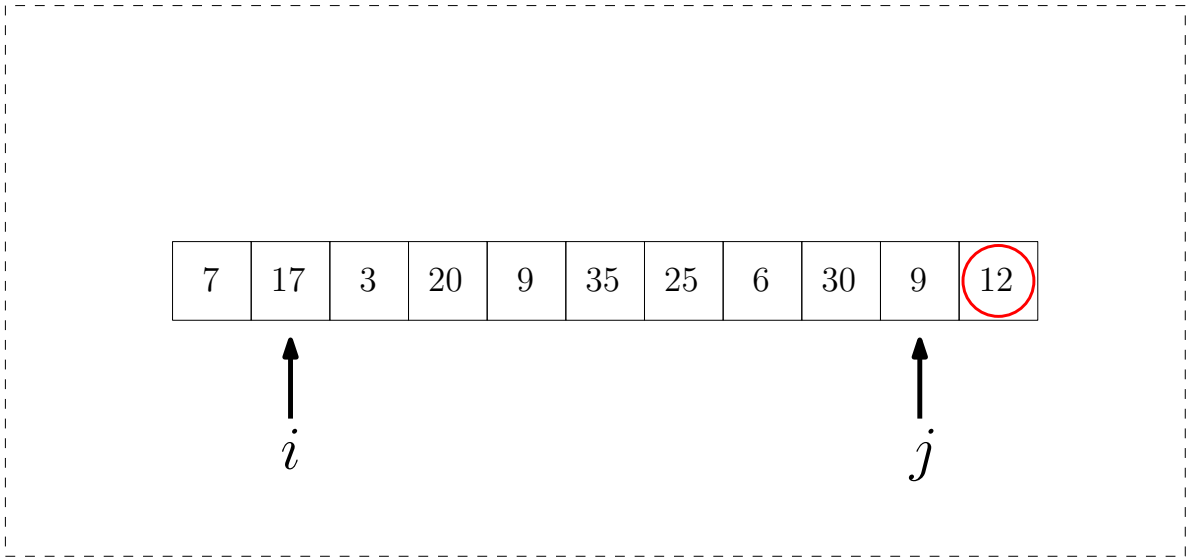
Αναζήτηση στοιχείου μεγαλύτερου του 12 από αριστερά

Παράδειγμα Διαμέρισης



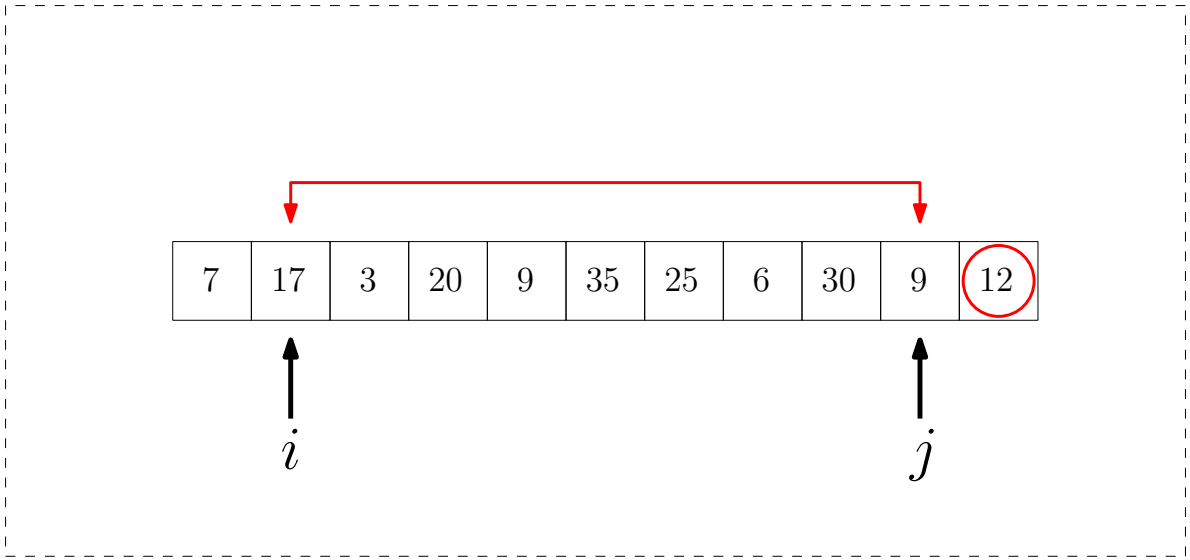
Αναζήτηση στοιχείου μικρότερου του 12 από δεξιά

Παράδειγμα Διαμέρισης



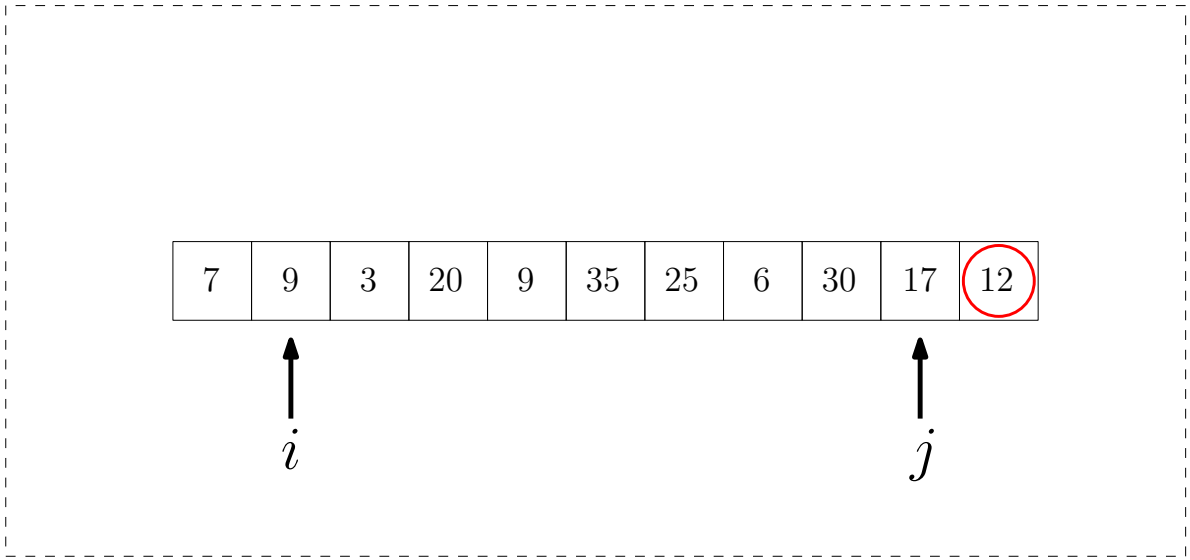
Αναζήτηση στοιχείου μικρότερου του 12 από δεξιά

Παράδειγμα Διαμέρισης



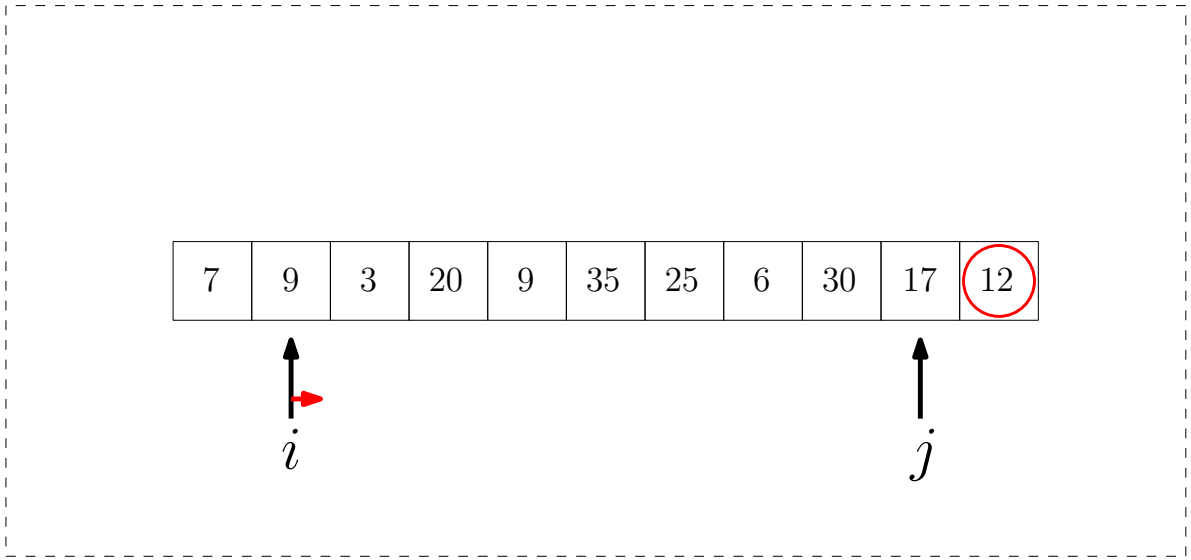
Αντιμετάθεση στοιχείων

Παράδειγμα Διαμέρισης



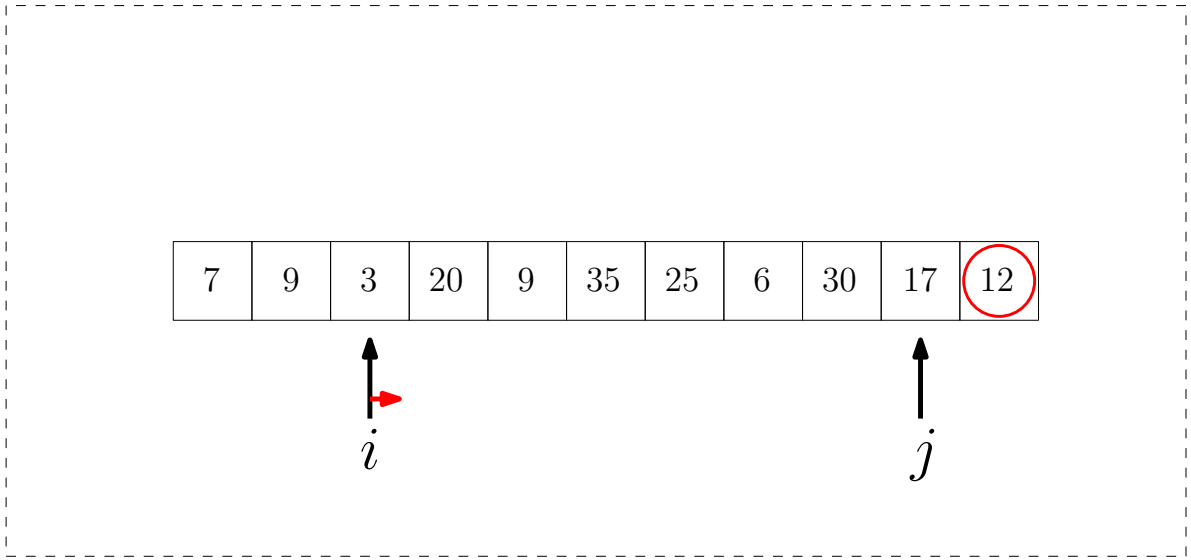
Αντιμετάθεση στοιχείων

Παράδειγμα Διαμέρισης



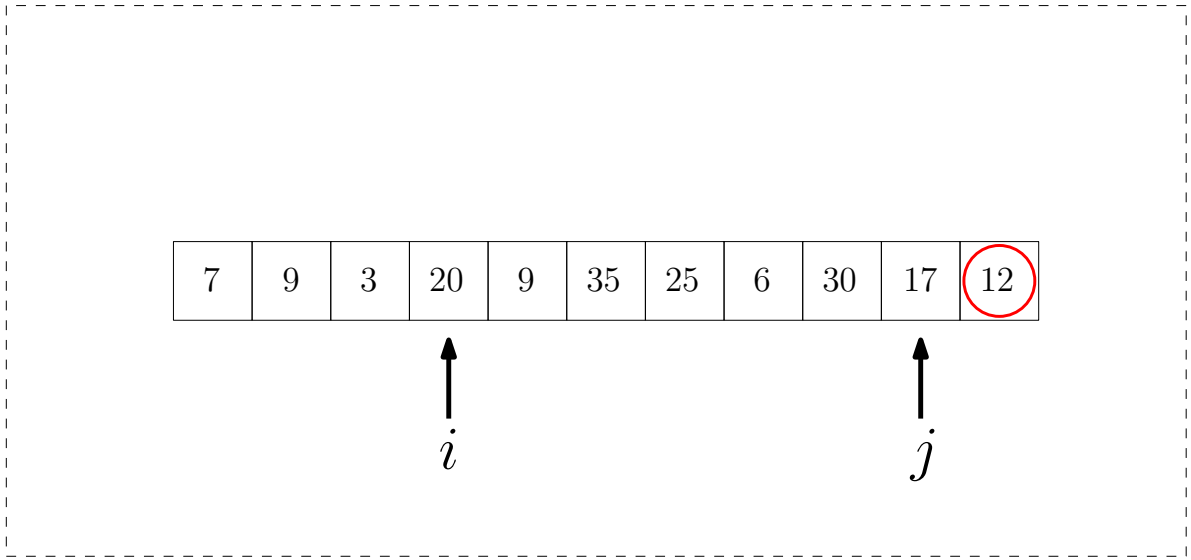
Αναζήτηση στοιχείου μεγαλύτερου του 12 από αριστερά

Παράδειγμα Διαμέρισης



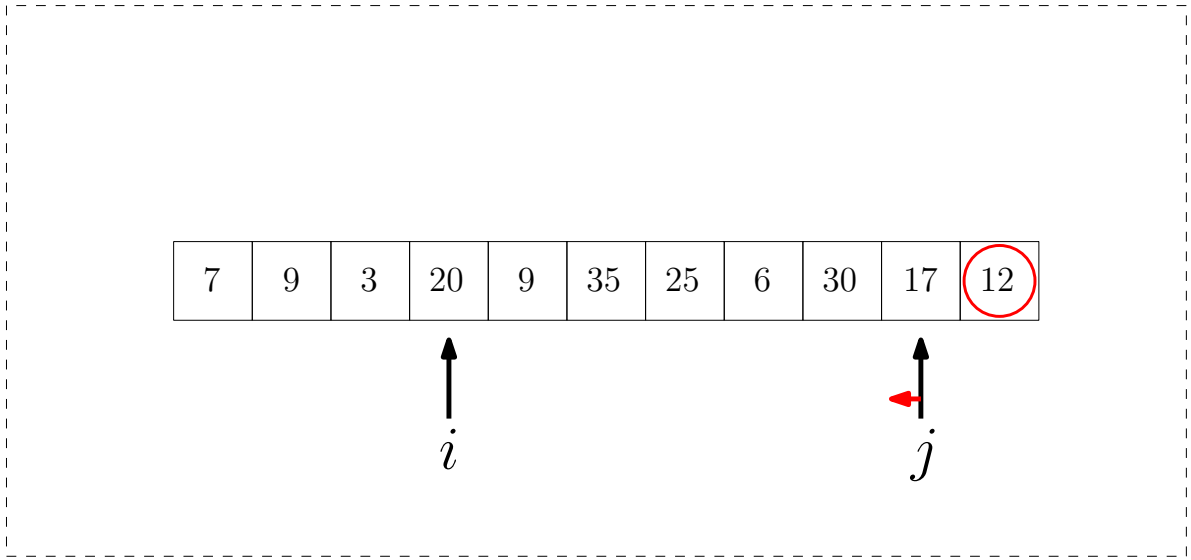
Αναζήτηση στοιχείου μεγαλύτερου του 12 από αριστερά

Παράδειγμα Διαμέρισης



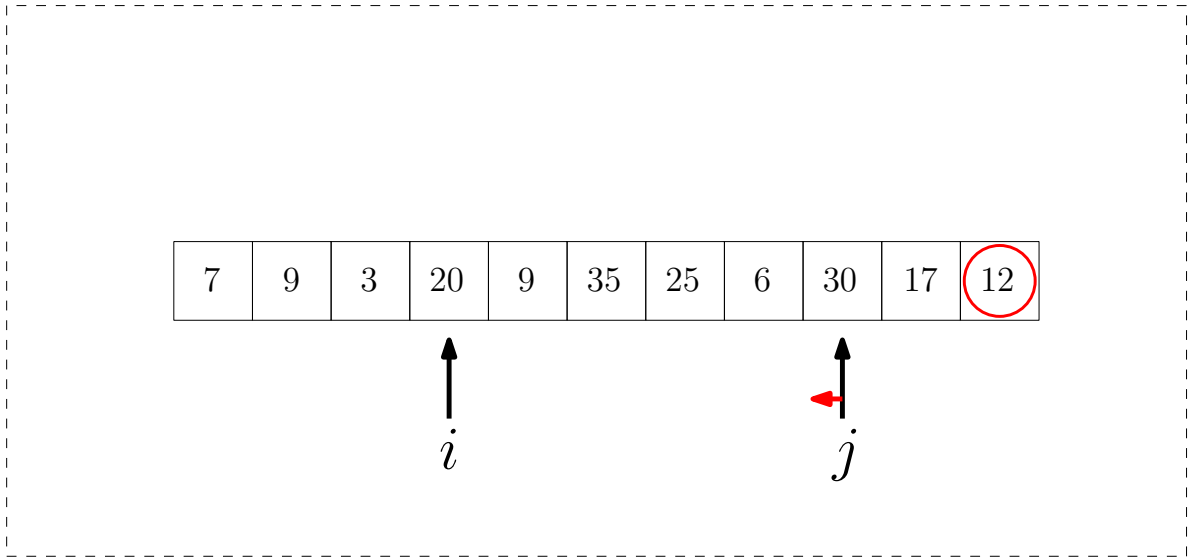
Αναζήτηση στοιχείου μεγαλύτερου του 12 από αριστερά

Παράδειγμα Διαμέρισης



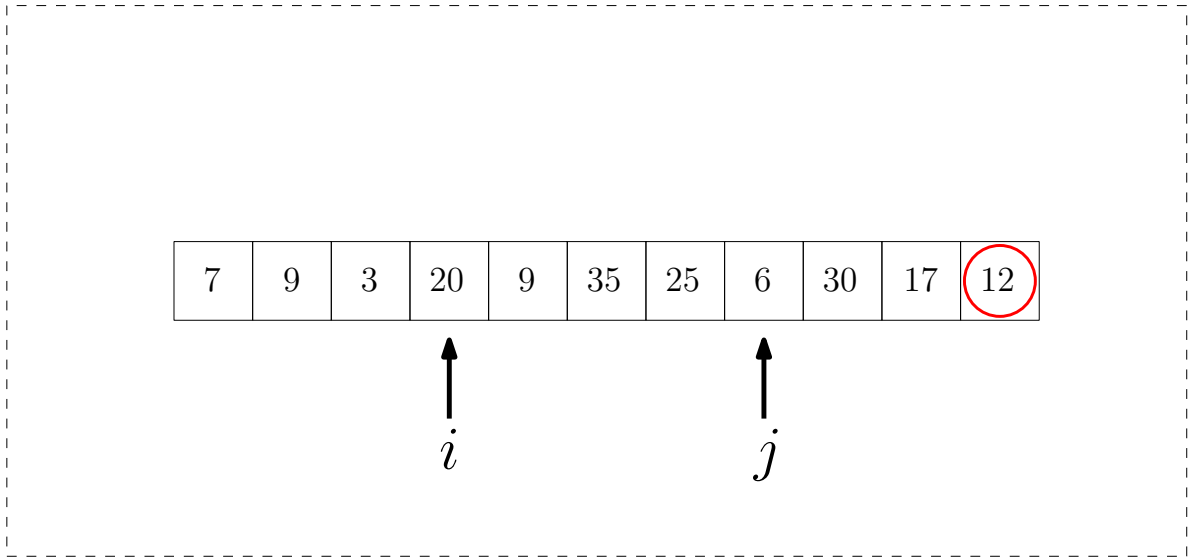
Αναζήτηση στοιχείου μικρότερου του 12 από δεξιά

Παράδειγμα Διαμέρισης



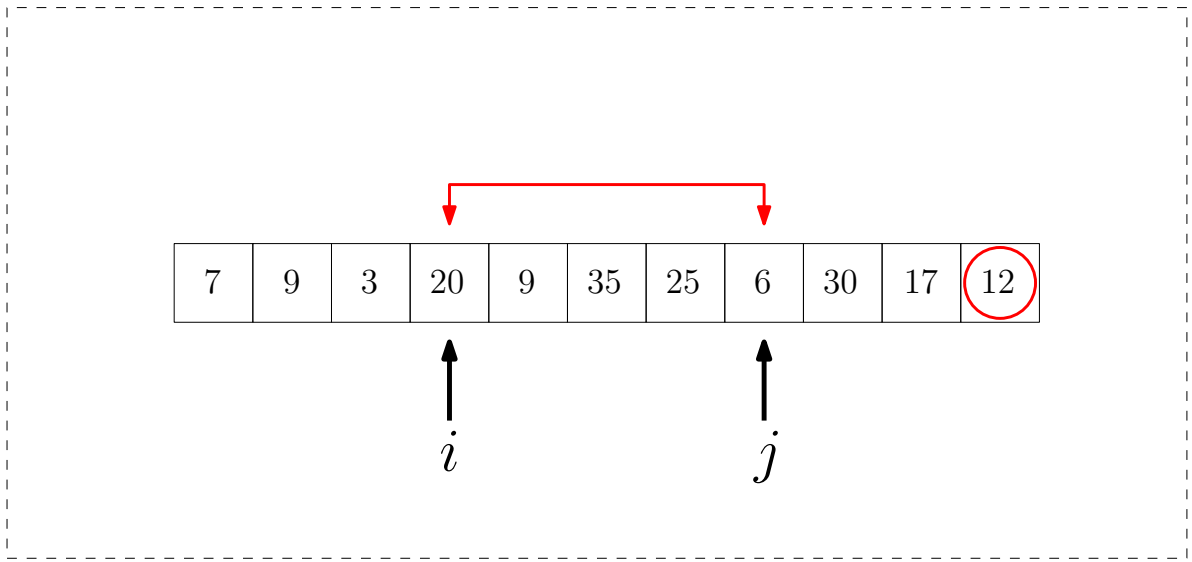
Αναζήτηση στοιχείου μικρότερου του 12 από δεξιά

Παράδειγμα Διαμέρισης



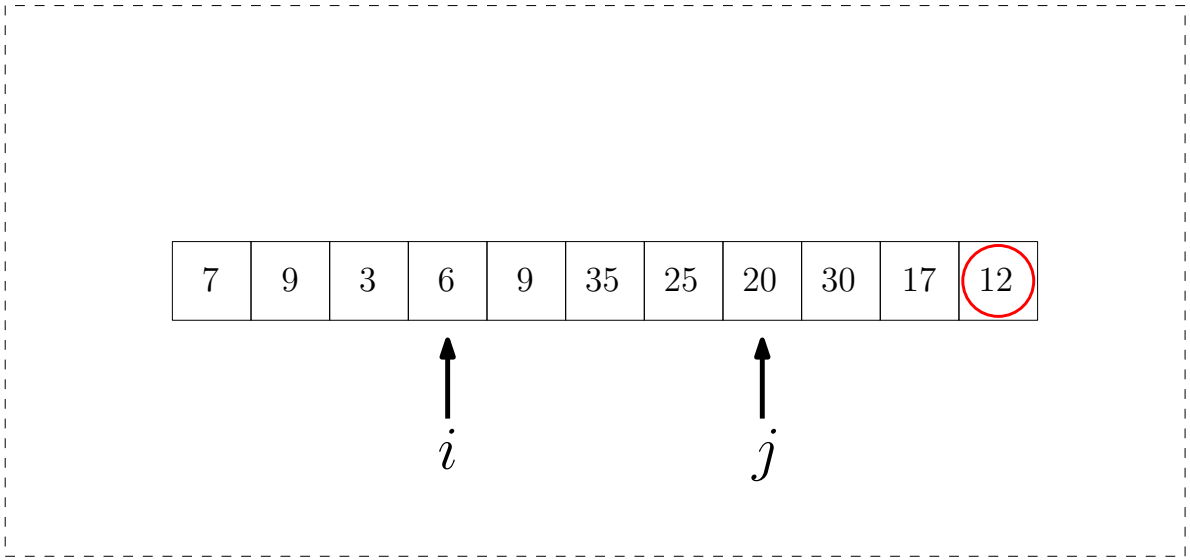
Αναζήτηση στοιχείου μικρότερου του 12 από δεξιά

Παράδειγμα Διαμέρισης



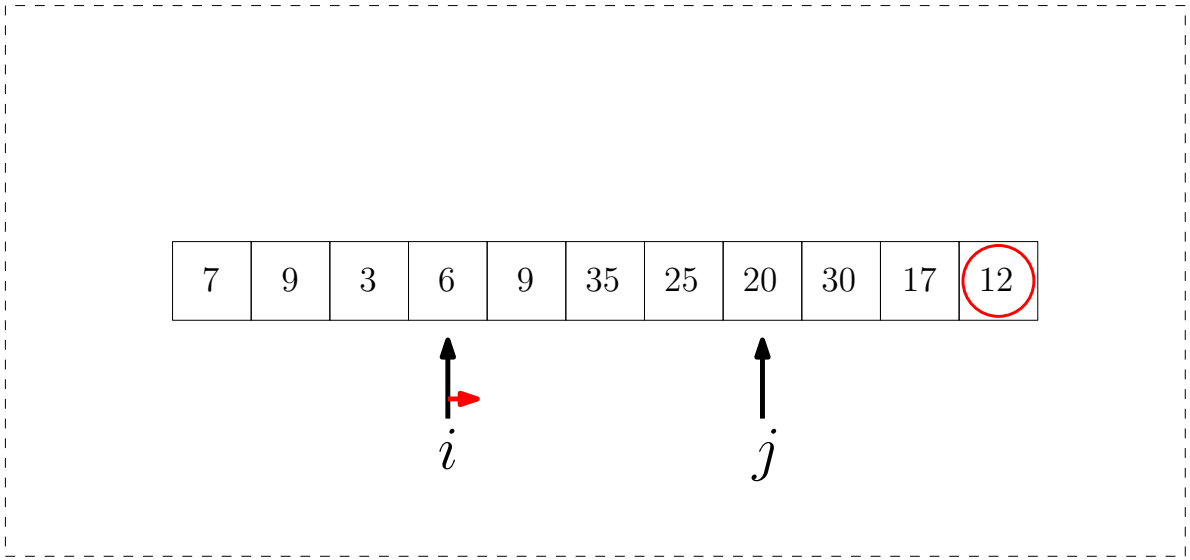
Αντιμετάθεση στοιχείων

Παράδειγμα Διαμέρισης



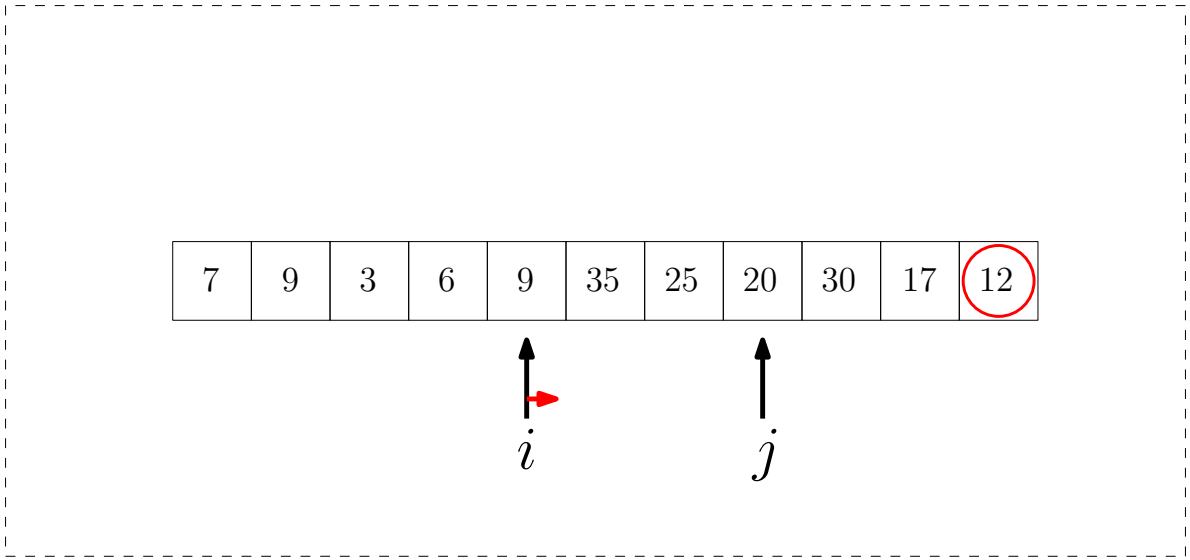
Αντιμετάθεση στοιχείων

Παράδειγμα Διαμέρισης



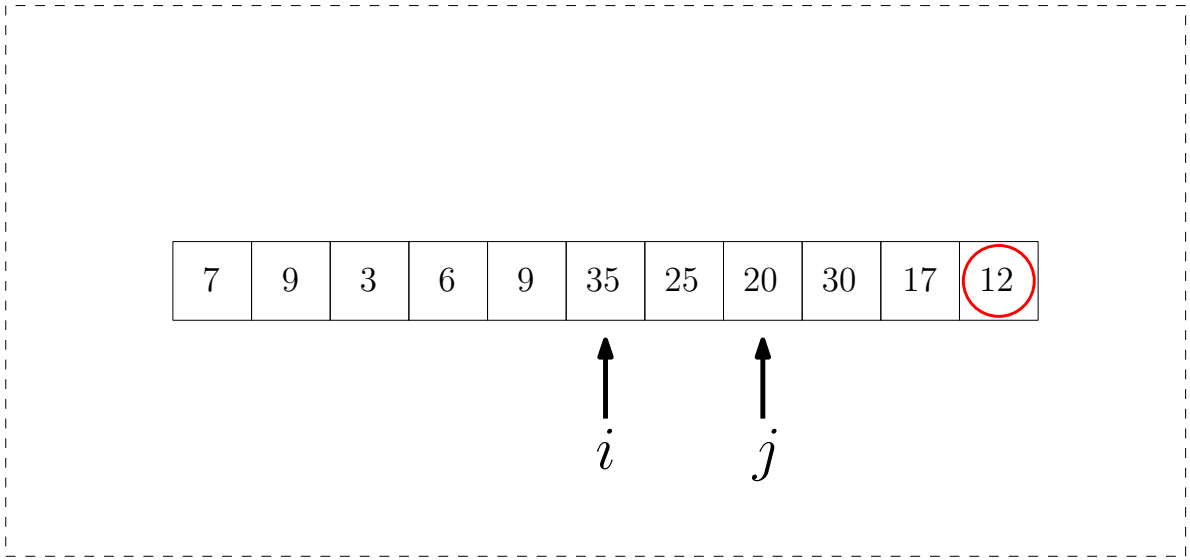
Αναζήτηση στοιχείου μεγαλύτερου του 12 από αριστερά

Παράδειγμα Διαμέρισης



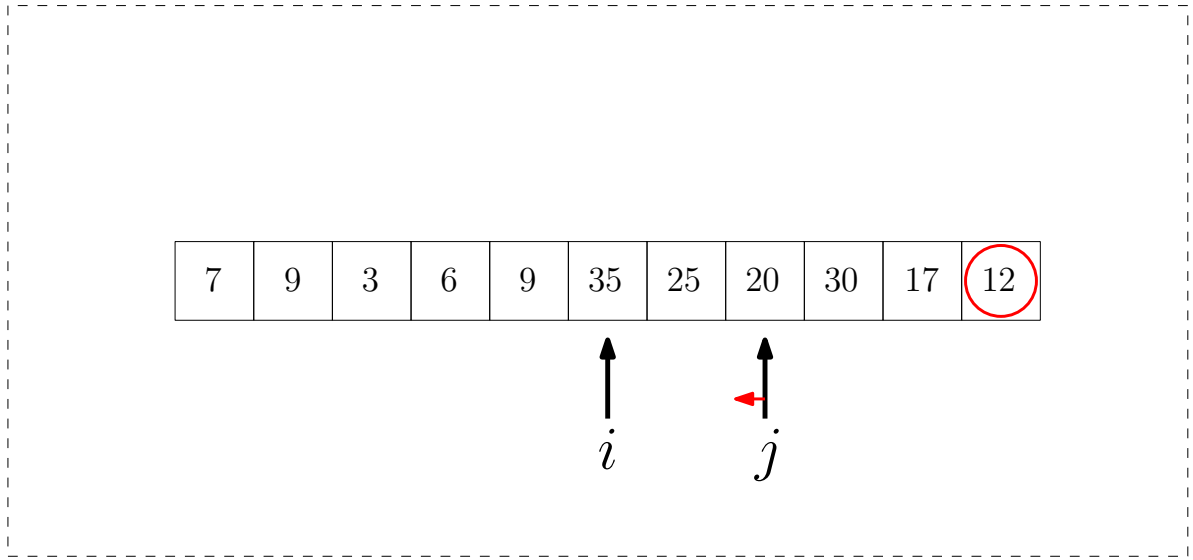
Αναζήτηση στοιχείου μεγαλύτερου του 12 από αριστερά

Παράδειγμα Διαμέρισης



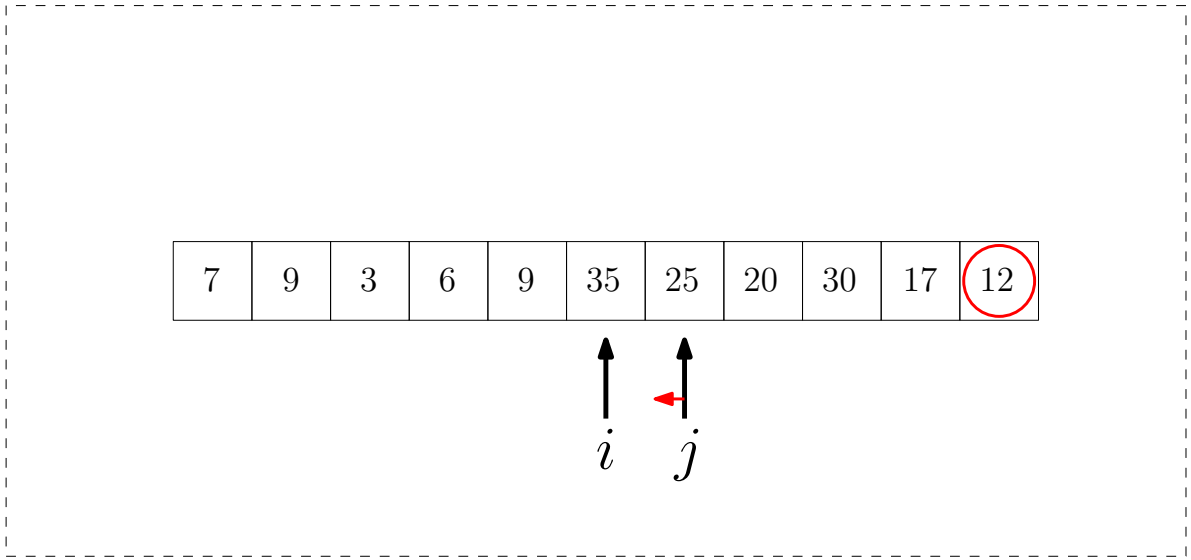
Αναζήτηση στοιχείου μεγαλύτερου του 12 από αριστερά

Παράδειγμα Διαμέρισης



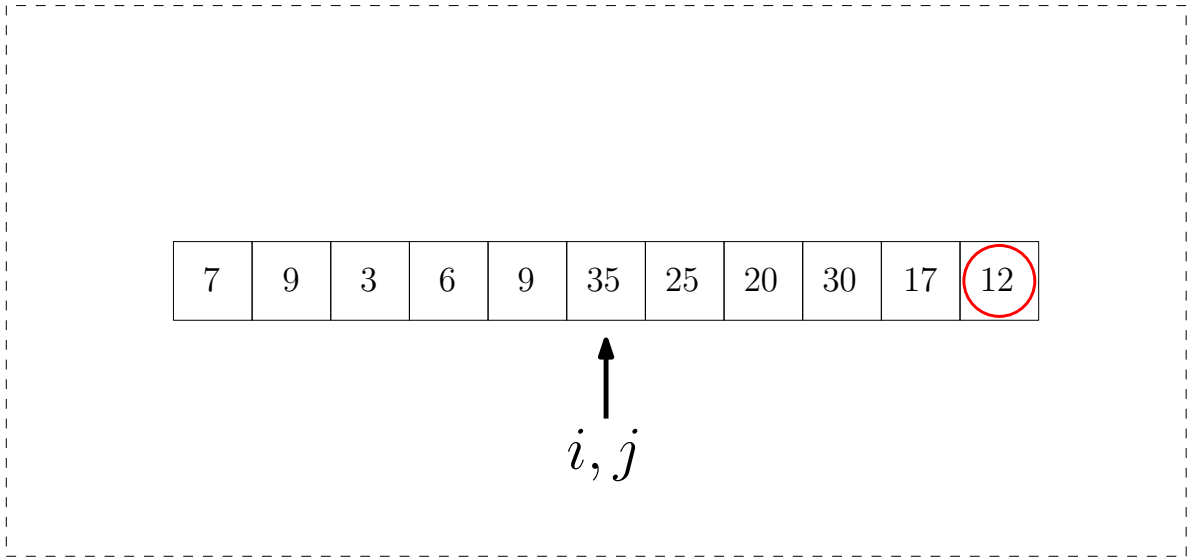
Αναζήτηση στοιχείου μικρότερου του 12 από δεξιά

Παράδειγμα Διαμέρισης



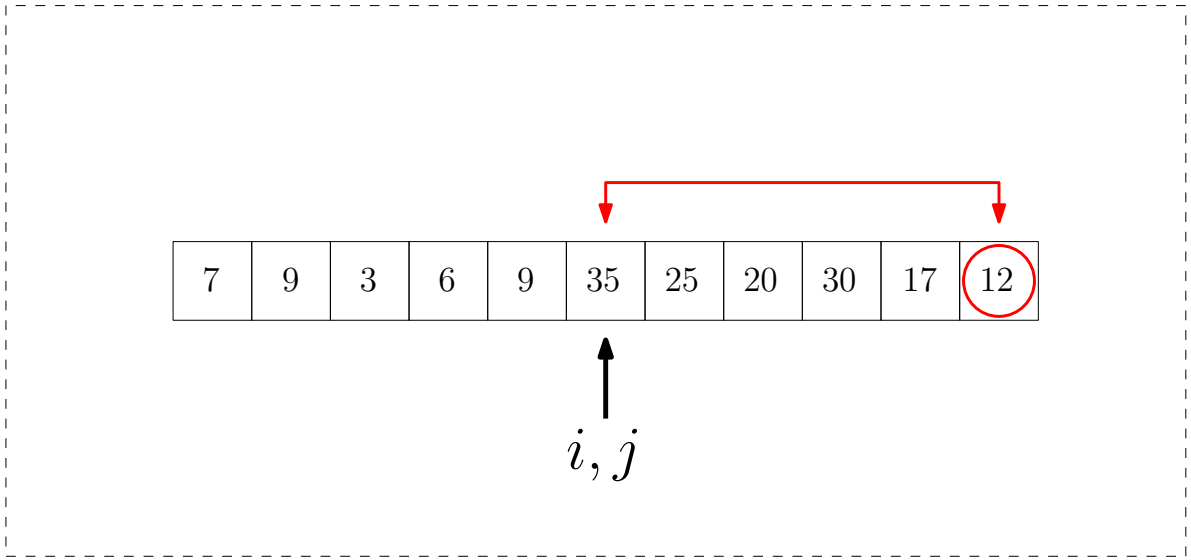
Αναζήτηση στοιχείου μικρότερου του 12 από δεξιά

Παράδειγμα Διαμέρισης



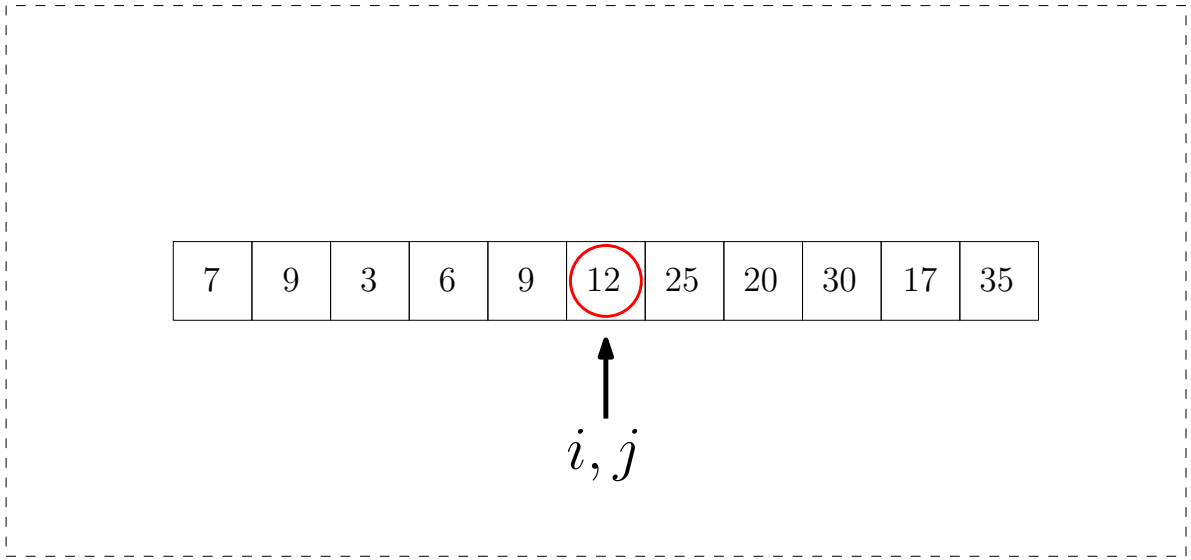
Αναζήτηση στοιχείου μικρότερου του 12 από δεξιά

Παράδειγμα Διαμέρισης



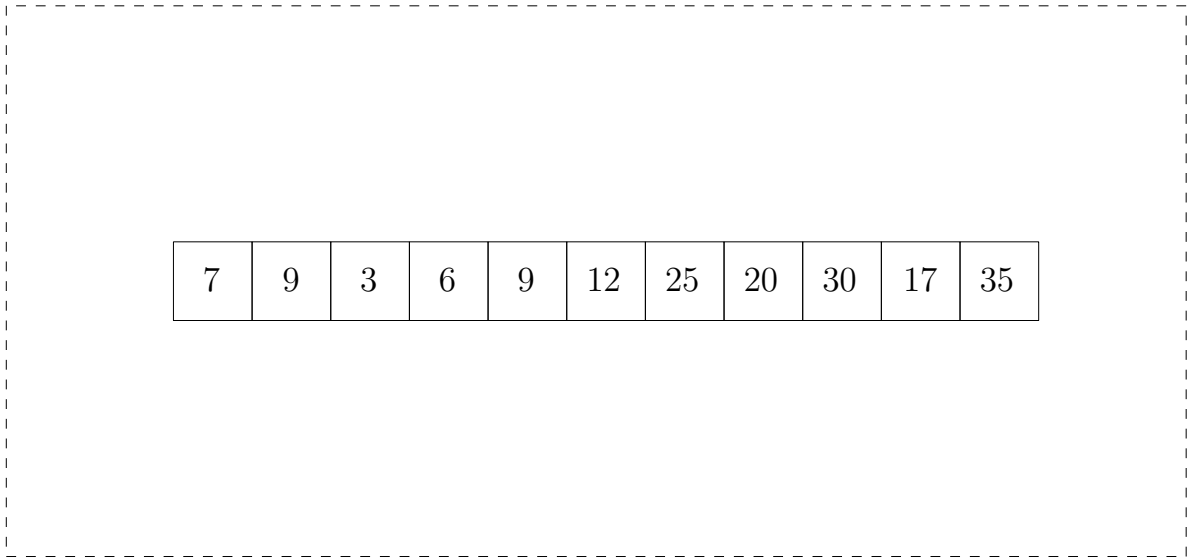
Ανταλλαγή στοιχείου διαμέρισης με δεξί δείκτη

Παράδειγμα Διαμέρισης



Ανταλλαγή στοιχείου διαμέρισης με δεξί δείκτη

Παράδειγμα Διαμέρισης



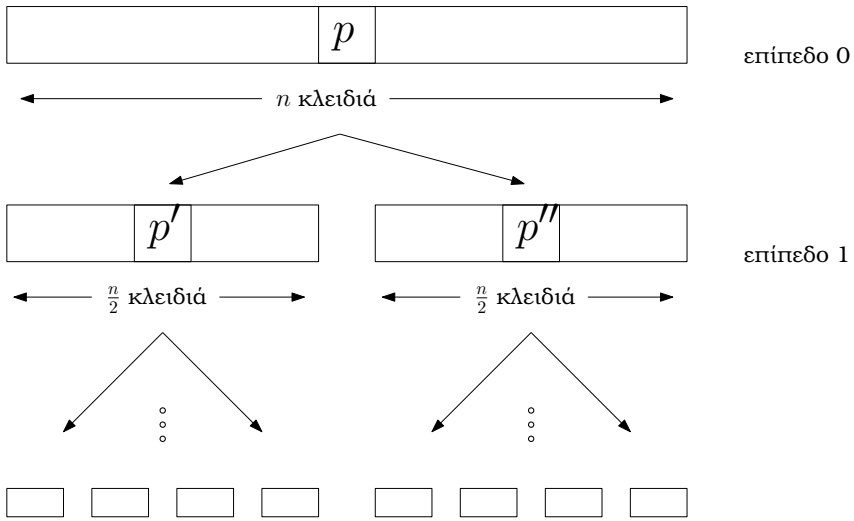
Τέλος διαδικασίας διαμέρισης

Επιδόσεις Διαμέρισης

Συγκρίνουμε το κάθε κλειδί ένα σταθερό αριθμό φορών.

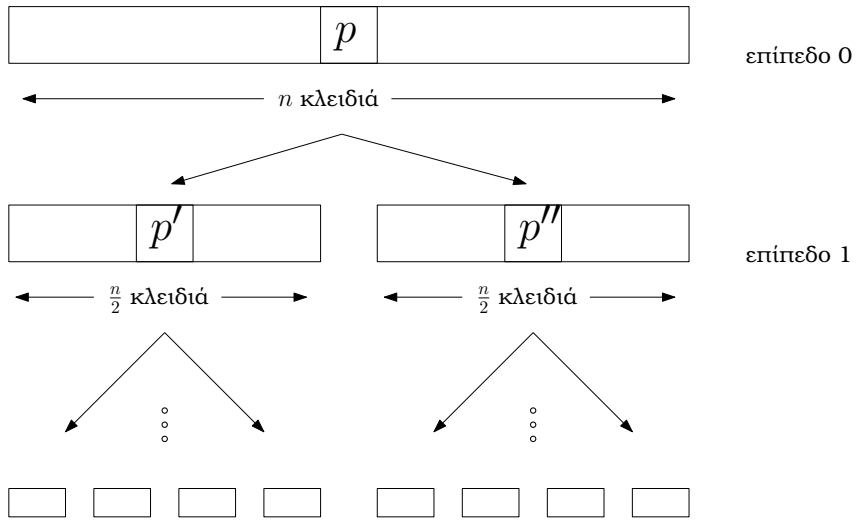
Για να διαμερίσουμε ένα πίνακα με n κλειδιά θέλουμε χρόνο $O(n)$.

Η **καλή** περίπτωση



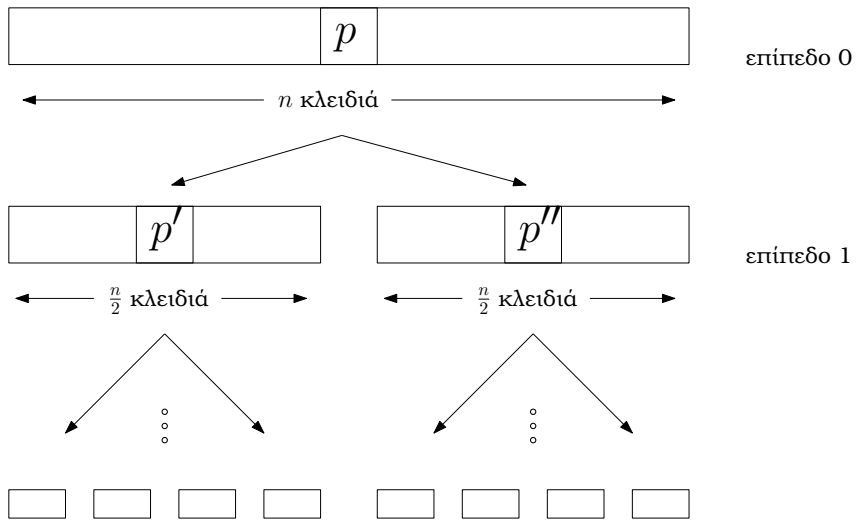
Σκοπός της επιλογής του στοιχείου διαμέρισης είναι να πετύχουμε ένα κλειδί το οποίο να μοιράζει τα υπόλοιπα στοιχεία στα δύο.

Η **καλή** περίπτωση



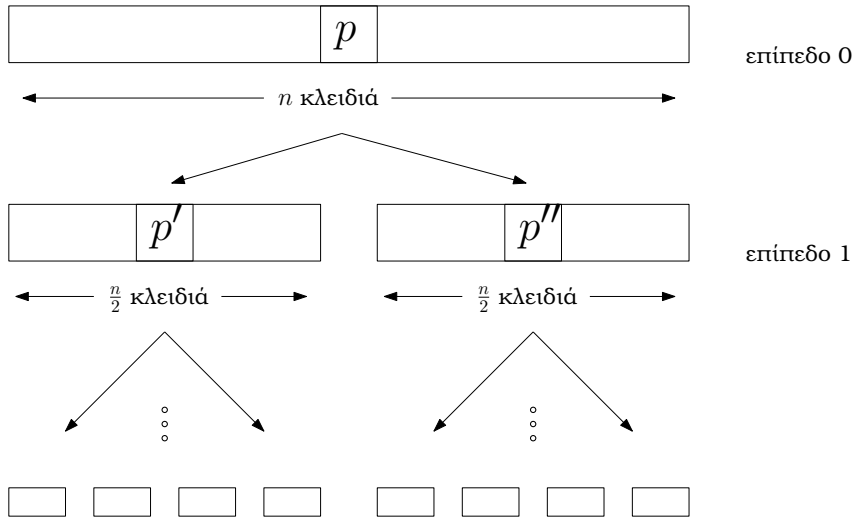
Πόσα επίπεδα αναδρομής έχουμε εάν είμαστε πάντα τυχεροί;

Η **καλή** περίπτωση



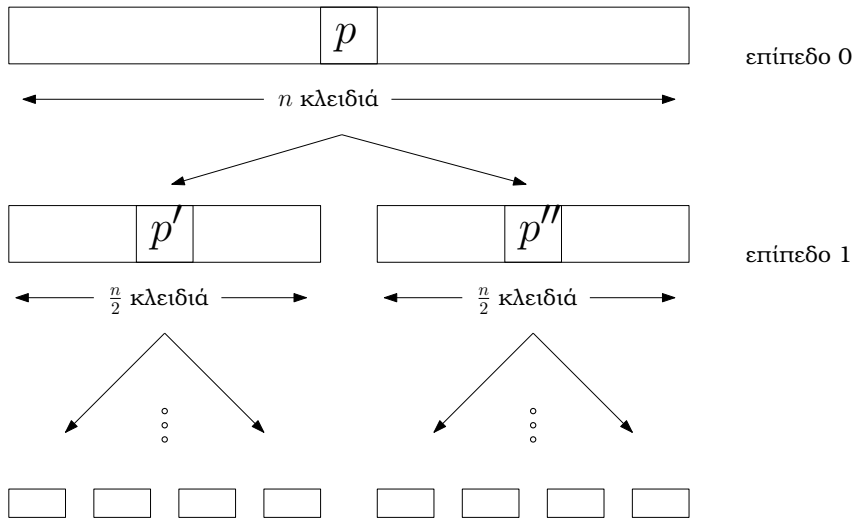
Όσες φορές μπορούμε να μοιράσουμε το n στα δύο δηλαδή $O(\log n)$.

Η **καλή** περίπτωση



- Το μοναδικό κόστος που έχουμε είναι η διαμέριση, το οποίο έχει γραμμικό κόστος.
- Στο επίπεδο *i* της αναδρομής έχουμε κόστος διαμέρισης 2^i υποπροβλημάτων το οποίο το καθένα χρειάζεται $O(\frac{n}{2^i})$ χρόνο.

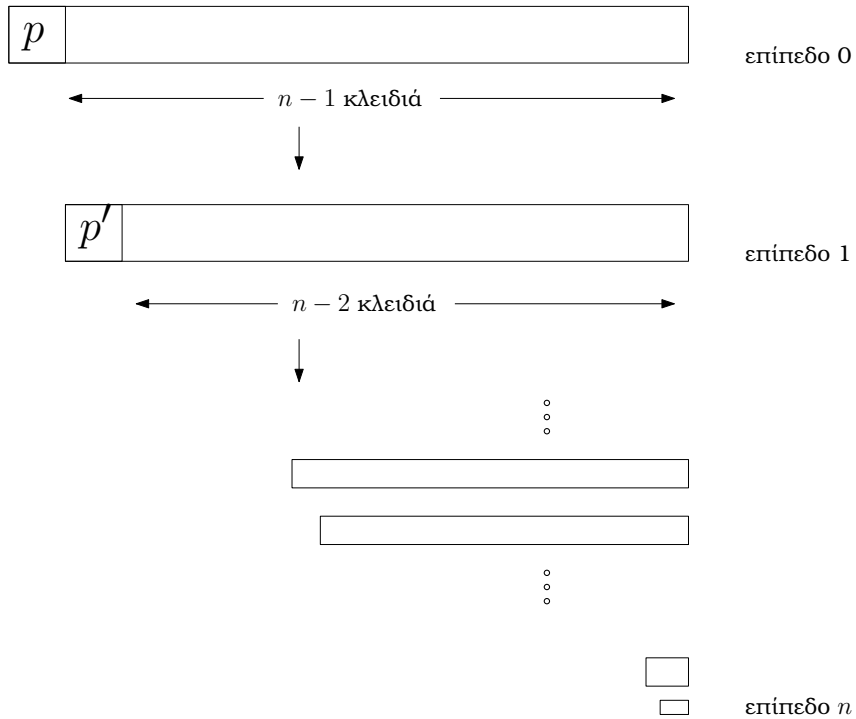
Η **καλή** περίπτωση



Σύνολο

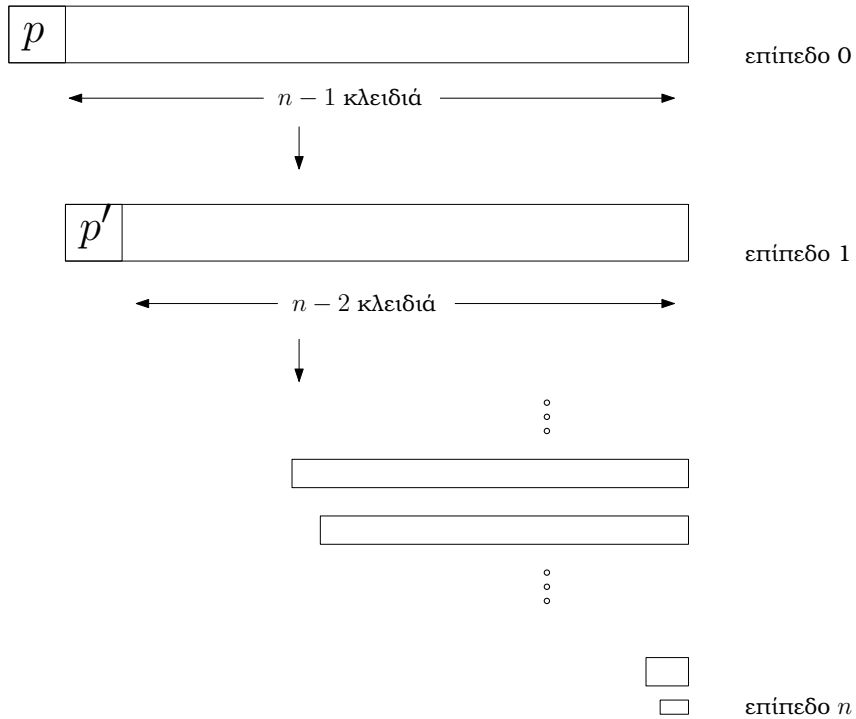
$$n + 2 \cdot \frac{n}{2} + 4 \cdot \frac{n}{4} \cdot + \ldots + 2^i \cdot \frac{n}{2^i} + \ldots \approx n \log n \in O(n \log n)$$

Η **χειρότερη** περίπτωση



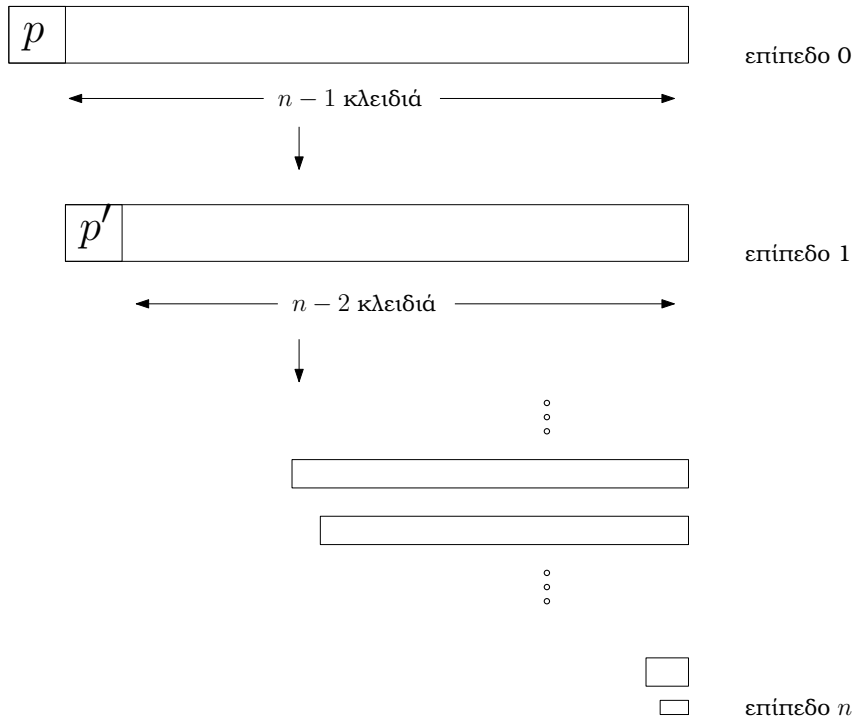
Εάν είμαστε πραγματικά άτυχοι μπορούμε να διαλέγουμε για κλειδί διαμέρισης πάντα το μικρότερο κλειδί.

Η **χειρότερη** περίπτωση



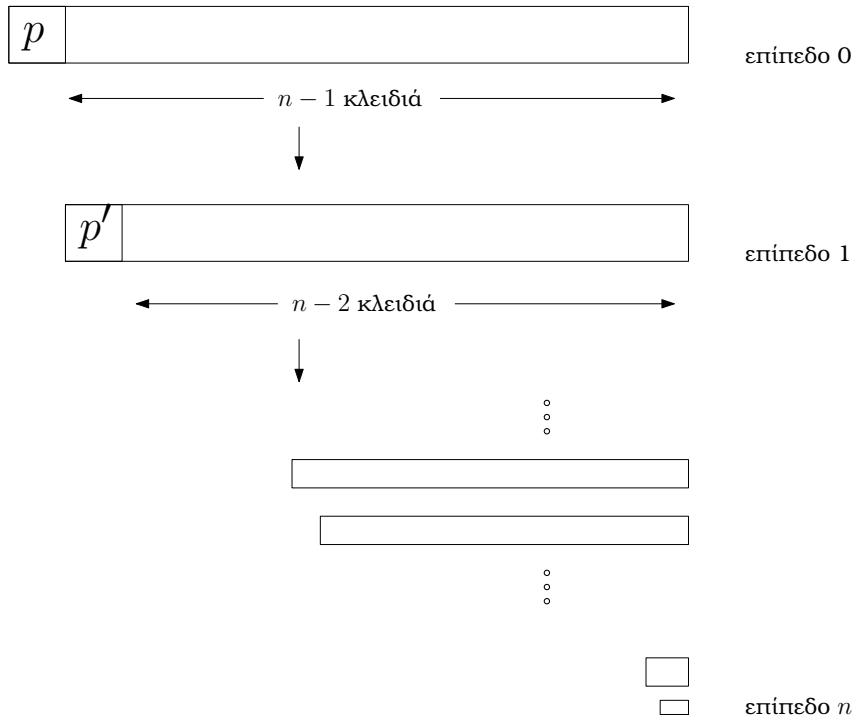
Τότε ο αλγόριθμος θα τρέξει αναδρομικά μόνο στο ένα υποπρόβλημα και θα έχουμε n επίπεδα αναδρομής

Η **χειρότερη** περίπτωση



- Το μοναδικό κόστος που έχουμε είναι η διαμέριση, το οποίο έχει γραμμικό κόστος.
- Στο επίπεδο i της αναδρομής έχουμε κόστος διαμέρισης 1 υποπροβλήματος το οποίο έχει μέγεθος $n - i$.

Η **χειρότερη** περίπτωση



Σύνολο

$$n + (n-1) + (n-2) + \ldots + 2 + 1 = O(n^2)$$

Μέση Περίπτωση

Μπορούμε να διαλέγουμε το στοιχείο διαμέρισης στην τύχη, ώστε να προσπαθήσουμε να ελαχιστοποιήσουμε την πιθανότητα να πετυχαίνουμε συνέχεια το μικρότερο κλειδί.

Έστω ο αλγόριθμος quicksort όπου επιλέγει το στοιχείο διαμέρισης στην τύχη, δηλαδή κάθε στοιχείο μπορεί να είναι στοιχείο διαμέρισης με ίση πιθανότητα.

Μέση Περίπτωση

Θεώρημα

Ο αναμενόμενος αριθμός συγκρίσεων τις οποίες πραγματοποιεί ο αλγόριθμος quicksort είναι $\leq 2n \ln n \leq 1.39 n \log n$.

Μέση Περίπτωση

Θεώρημα

Ο αναμενόμενος αριθμός συγκρίσεων τις οποίες πραγματοποιεί ο αλγόριθμος quicksort είναι $\leq 2n \ln n \leq 1.39 n \log n$.

Έστω ότι η

$$s = \langle e_1, e_2, \ldots, e_i, \ldots, e_j, \ldots, e_n \rangle$$

συμβολίζει τα στοιχεία της εισόδου σε ταξινομημένη σειρά.

Μέση Περίπτωση

Θεώρημα

Ο αναμενόμενος αριθμός συγκρίσεων τις οποίες πραγματοποιεί ο αλγόριθμος quicksort είναι $\leq 2n \ln n \leq 1.39 n \log n$.

Έστω ότι η

$$s = \langle e_1, e_2, \ldots, e_i, \ldots, e_j, \ldots, e_n \rangle$$

συμβολίζει τα στοιχεία της εισόδου σε ταξινομημένη σειρά.

Τα στοιχεία e_i και e_j συγκρίνονται το πολύ μία φορά, και μόνο εαν ένα από τα δύο επιλεγεί ως στοιχείο διαμέρισης.

Μέση Περίπτωση

Θεώρημα

Ο αναμενόμενος αριθμός συγκρίσεων τις οποίες πραγματοποιεί ο αλγόριθμος quicksort είναι $\leq 2n \ln n \leq 1.39 n \log n$.

Έστω ότι η

$$s = \langle e_1, e_2, \ldots, e_i, \ldots, e_j, \ldots, e_n \rangle$$

συμβολίζει τα στοιχεία της εισόδου σε ταξινομημένη σειρά.

Τα στοιχεία e_i και e_j συγκρίνονται το πολύ μία φορά, και μόνο εαν ένα από τα δύο επιλεγεί ως στοιχείο διαμέρισης.

Έστω η τυχαία μεταβλητή X_{ij}, $i < j$ όπου $X_{ij} = 1$ αν συγκρίνονται τα e_i και e_j, και $X_{ij} = 0$ διαφορετικά.

Μέση Περίπτωση

Θεώρημα

Ο αναμενόμενος αριθμός συγκρίσεων τις οποίες πραγματοποιεί ο αλγόριθμος quicksort είναι $\leq 2n \ln n \leq 1.39 n \log n$.

Έστω ότι η

$$s = \langle e_1, e_2, \ldots, e_i, \ldots, e_j, \ldots, e_n \rangle$$

συμβολίζει τα στοιχεία της εισόδου σε ταξινομημένη σειρά.

Τα στοιχεία e_i και e_j συγκρίνονται το πολύ μία φορά, και μόνο εαν ένα από τα δύο επιλεγεί ως στοιχείο διαμέρισης.

Έστω η τυχαία μεταβλητή X_{ij}, $i < j$ όπου $X_{ij} = 1$ αν συγκρίνονται τα e_i και e_j, και $X_{ij} = 0$ διαφορετικά.

Ο μέσος αριθμός συγκρίσεων είναι

$$C(n) = E\left[\sum_{i=1}^{n}\sum_{j=i+1}^{n} X_{ij}\right] = \sum_{i=1}^{n}\sum_{j=i+1}^{n} E[X_{ij}] = \sum_{i=1}^{n}\sum_{j=i+1}^{n} prob(X_{ij} = 1).$$

Μέση Περίπτωση

Θεώρημα

Ο αναμενόμενος αριθμός συγκρίσεων τις οποίες πραγματοποιεί ο αλγόριθμος quicksort είναι $\leq 2n \ln n \leq 1.39 n \log n$.

Έστω ότι η

$$s = \langle e_1, e_2, \ldots, e_i, \ldots, e_j, \ldots, e_n \rangle$$

συμβολίζει τα στοιχεία της εισόδου σε ταξινομημένη σειρά.

Τα στοιχεία e_i και e_j συγκρίνονται το πολύ μία φορά, και μόνο εαν ένα από τα δύο επιλεγεί ως στοιχείο διαμέρισης.

Έστω η τυχαία μεταβλητή X_{ij}, $i < j$ όπου $X_{ij} = 1$ αν συγκρίνονται τα e_i και e_j, και $X_{ij} = 0$ διαφορετικά.

Ο μέσος αριθμός συγκρίσεων είναι

$$C(n) = E\left[\sum_{i=1}^{n}\sum_{j=i+1}^{n} X_{ij}\right] = \sum_{i=1}^{n}\sum_{j=i+1}^{n} E[X_{ij}] = \sum_{i=1}^{n}\sum_{j=i+1}^{n} prob(X_{ij} = 1).$$

Για περαιτέρω απλοποίηση πρέπει να υπολογίσουμε την πιθανότητα.

Μέση Περίπτωση

Λήμμα

Για οποιοδήποτε $i < j$, $prob(X_{ij} = 1) = \frac{2}{j-i+1}$.

Απόδειξη

Θεωρήστε το σύνολο των $j - i + 1$ στοιχείων $M = \{e_i, \ldots, e_j\}$.

Μέση Περίπτωση

Λήμμα

Για οποιοδήποτε $i < j$, $prob(X_{ij} = 1) = \frac{2}{j-i+1}$.

Απόδειξη

Θεωρήστε το σύνολο των $j - i + 1$ στοιχείων $M = \{e_i, \ldots, e_j\}$.

Όσο δεν επιλέγεται κάποιο στοιχείο διαμέρισης από το M, τα e_i και e_j δεν συγκρίνονται, αλλά όλα τα στοιχεία του M μεταβιβάζονται στις ίδιες αναδρομικές κλήσεις.

Μέση Περίπτωση

Λήμμα

Για οποιοδήποτε $i < j$, $prob(X_{ij} = 1) = \frac{2}{j-i+1}$.

Απόδειξη

Θεωρήστε το σύνολο των $j - i + 1$ στοιχείων $M = \{e_i, \ldots, e_j\}$.

Όσο δεν επιλέγεται κάποιο στοιχείο διαμέρισης από το M, τα e_i και e_j δεν συγκρίνονται, αλλά όλα τα στοιχεία του M μεταβιβάζονται στις ίδιες αναδρομικές κλήσεις.

Τελικά, θα επιλεγεί ένα στοιχείο διαμέρισης από το M. Κάθε στοιχείο του M έχει την ίδια πιθανότητα $\frac{1}{|M|} = \frac{1}{j-i+1}$.

Μέση Περίπτωση

Λήμμα

Για οποιοδήποτε $i < j$, $prob(X_{ij} = 1) = \frac{2}{j-i+1}$.

Απόδειξη

Θεωρήστε το σύνολο των $j - i + 1$ στοιχείων $M = \{e_i, \ldots, e_j\}$.

Όσο δεν επιλέγεται κάποιο στοιχείο διαμέρισης από το M, τα e_i και e_j δεν συγκρίνονται, αλλά όλα τα στοιχεία του M μεταβιβάζονται στις ίδιες αναδρομικές κλήσεις.

Τελικά, θα επιλεγεί ένα στοιχείο διαμέρισης από το M. Κάθε στοιχείο του M έχει την ίδια πιθανότητα $\frac{1}{|M|} = \frac{1}{j-i+1}$.

Αν επιλεγεί το e_i ή το e_j έχουμε $X_{ij} = 1$. Η πιθανότητα για αυτό το συμβάν είναι $\frac{2}{j-i+1}$. Διαφορετικά, τα e_i και e_j μεταβιβάζονται σε διαφορετικές αναδρομικές κλήσεις, οπότε δεν θα συγκριθούν ποτέ. □

Θεώρημα

Ο αναμενόμενος αριθμός συγκρίσεων τις οποίες πραγματοποιεί ο αλγόριθμος quicksort είναι $\leq 2n \ln n \leq 1.39 n \log n$.

Ο μέσος αριθμός συγκρίσεων είναι

$$C(n) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} prob(X_{ij} = 1) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \frac{2}{j-i+1} = \sum_{i=1}^{n} \sum_{k=2}^{n-i+1} \frac{2}{k}$$

$$\leq \sum_{i=1}^{n} \sum_{k=2}^{n} \frac{2}{k} = 2n \sum_{k=2}^{n} \frac{1}{k} = 2n(H_n - 1) \leq 2n(1 + \ln n - 1) = 2n \ln n$$

Οι τελευταίες ισότητες ισχύουν αφού ισχύει για τους αρμονικούς αριθμούς:

$$\ln n \leq H_n = \sum_{i=1}^{n} \frac{1}{i} \leq \ln n + 1.$$

□

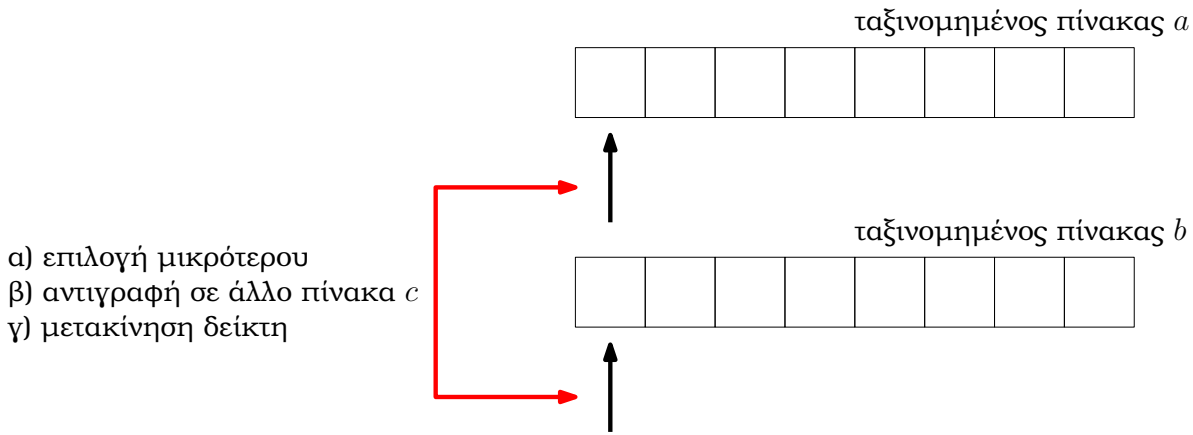
Χρόνος εκτέλεσης Quicksort

- στην χειρότερη περίπτωση $O(n^2)$
- στην καλύτερη περίπτωση $O(n \log n)$
- μέση περίπτωση $O(n \log n)$

Συγχώνευση και Ταξινόμηση Συγχώνευσης

Merging + Mergesort

Συγχώνευση είναι η διαδικασία συγχώνευσης δύο ταξινομημένων αρχείων σε ένα ταξινομημένο αρχείο.



Γίνεται σε γραμμικό χρόνο.

Ταξινόμηση Συγχώνευσης

```
void mergesort(int a[], int l, int r) {
    if (r <= l)
        return;

    int m = (r+l)/2;

    mergesort(a, l, m);
    mergesort(a, m+1, r);

    merge(a, l, m, r);
}
```

Η συνάρτηση merge χρησιμοποιεί έναν επιπλέον πίνακα μεγέθους n.

Επιδόσεις Ταξινόμησης Συγχώνευσης

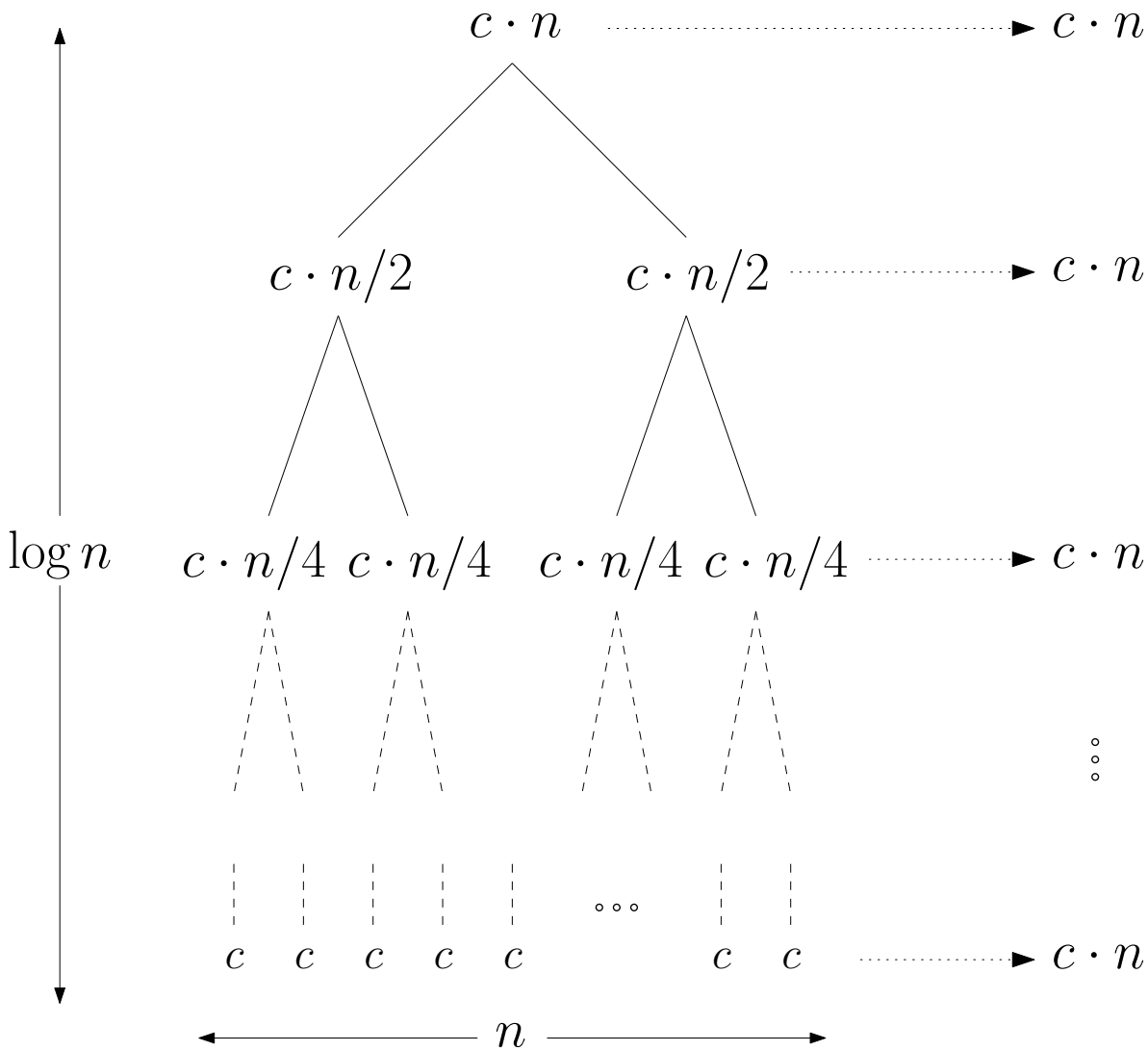
Έστω $T(n)$ ο χρόνος ταξινόμησης n αριθμών.

Τότε:

$$T(n) = \begin{cases} c & \text{εάν } n = 1, \\ 2 \cdot T(n/2) + c \cdot n & \text{εάν} n > 1. \end{cases}$$

όπου c μια σταθερά και $c \cdot n$ ο χρόνος που χρειάζεται για να κάνουμε merge δύο πίνακες με n στοιχεία (ή λιγότερα) ο καθένας.

Αναδρομή


$c \cdot n$
$c \cdot n$
$c \cdot n/2$
$c \cdot n/2$
$c \cdot n$
$\log n$
$c \cdot n/4$ $c \cdot n/4$ $c \cdot n/4$ $c \cdot n/4$
$c \cdot n$
c c c c c c c
$c \cdot n$
n

Βασικά Χαρακτηριστικά Ταξινόμησης Συγχώνευσης

- πάντα $O(n \log n)$, και στην καλύτερη και στην χειρότερη περίπτωση
- για πίνακες θέλει επιπλέον χώρο $O(n)$
- ιδανικός για λίστες
- ευσταθής

Μοντέλο δεντρικών αποφάσεων

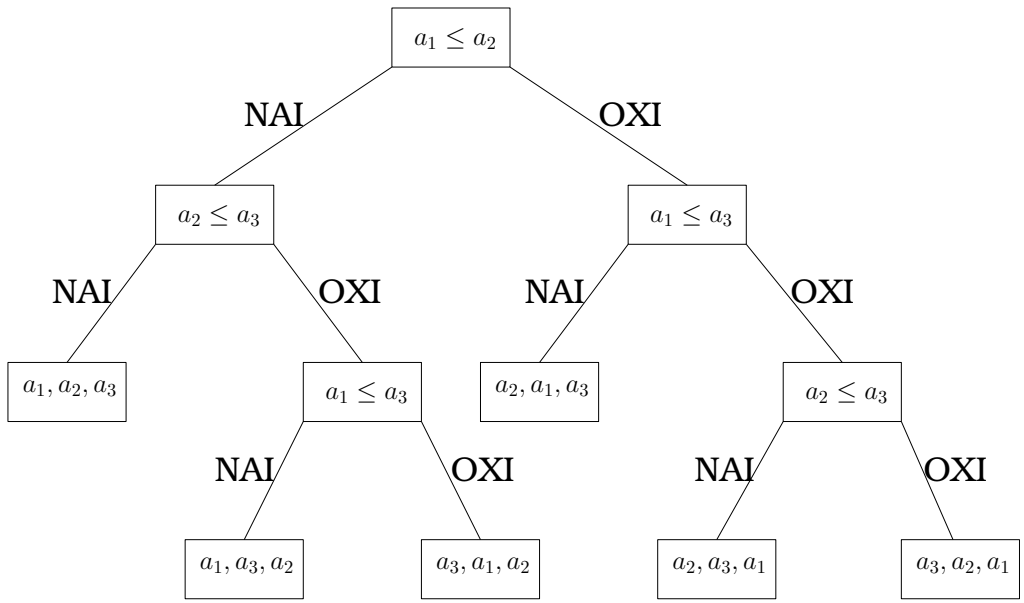
Το μοντέλο

- έστω μια ακολουθία εισόδου $\langle a_1, a_2, \ldots, a_n \rangle$
- άντληση πληροφορίας μπορεί να γίνει μόνο με συγκρίσεις
- θεωρούμε πως όλα τα στοιχεία εισόδου είναι διαφορετικά οπότε οι μόνες συγκρίσεις που χρειάζονται είναι της μορφής $a_i \leq a_j$.

Δέντρα Αποφάσεων

Decision Trees

Πλήρες δυαδικό δέντρο που αντιπροσωπεύει τις συγκρίσεις που πραγματοποιούνται μεταξύ στοιχείων από έναν συγκεκριμένο αλγόριθμο ταξινόμησης.



- Ένας ορθός αλγόριθμος πρέπει να μπορεί να παράγει και τις $n!$ μεταθέσεις.
- Το μακρύτερο μονοπάτι προς κάποιο φύλλο είναι η χειρότερη περίπτωση αυτού του αλγορίθμου.

Θεώρημα

Έστω h το ύψος ενός δέντρου αποφάσεων για είσοδο μεγέθους n. Τότε $h \in \Omega(n \log n)$.

Θεώρημα

Έστω h το ύψος ενός δέντρου αποφάσεων για είσοδο μεγέθους n. Τότε $h \in \Omega(n \log n)$.

Απόδειξη

Ένα πλήρες δυαδικό δέντρο ύψους h έχει το πολύ 2^h φύλλα.

Θεώρημα

Έστω h το ύψος ενός δέντρου αποφάσεων για είσοδο μεγέθους n. Τότε $h \in \Omega(n \log n)$.

Απόδειξη

Ένα πλήρες δυαδικό δέντρο ύψους h έχει το πολύ 2^h φύλλα.
Ένα δέντρο αποφάσεων έχει $n!$ φύλλα.

Θεώρημα

Έστω h το ύψος ενός δέντρου αποφάσεων για είσοδο μεγέθους n. Τότε $h \in \Omega(n \log n)$.

Απόδειξη

Ένα πλήρες δυαδικό δέντρο ύψους h έχει το πολύ 2^h φύλλα.
Ένα δέντρο αποφάσεων έχει $n!$ φύλλα.
Άρα

$$n! \leq 2^h.$$

Θεώρημα

Έστω h το ύψος ενός δέντρου αποφάσεων για είσοδο μεγέθους n. Τότε $h \in \Omega(n \log n)$.

Απόδειξη

Ένα πλήρες δυαδικό δέντρο ύψους h έχει το πολύ 2^h φύλλα.
Ένα δέντρο αποφάσεων έχει $n!$ φύλλα.
Άρα

$$n! \leq 2^h.$$

Λογαριθμίζοντας και αφού η συνάρτηση $\log_2$ είναι μονότονα αύξουσα έχουμε πως:

$$h \geq \log_2(n!).$$

Θεώρημα

Έστω h το ύψος ενός δέντρου αποφάσεων για είσοδο μεγέθους n. Τότε $h \in \Omega(n \log n)$.

Απόδειξη

Ένα πλήρες δυαδικό δέντρο ύψους h έχει το πολύ 2^h φύλλα.
Ένα δέντρο αποφάσεων έχει $n!$ φύλλα.
Άρα

$$n! \leq 2^h.$$

Λογαριθμίζοντας και αφού η συνάρτηση $\log_2$ είναι μονότονα αύξουσα έχουμε πως:

$$h \geq \log_2(n!).$$

Χρησιμοποιώντας την προσέγγιση του Stirling, $n! \approx \sqrt{2\pi n}(\frac{n}{e})^n$ προκύπτει πως $\log(n!) = \Omega(n \log n)$. □

Κάτω Φράγμα Συγκρίσεων Ταξινόμησης

Οποιοσδήποτε αλγόριθμος χρησιμοποιεί μόνο συγκρίσεις και αντιμεταθέσεις για την ταξινόμηση n αριθμών χρειάζεται $\Omega(n \log n)$ συγκρίσεις.

Επιλογή

Selection ή Order Statistics

Ένα πρόβλημα που προκύπτει πολλές φορές είναι η επιλογή (εύρεση) του k-οστού στοιχείου ενός συνόλου αριθμών.

Έστω για παράδειγμα ένα σύνολο αριθμών:

$$5, 7, 1, 2, 6, 9, 32, 100, 5$$

- Θέλουμε να μπορούμε να απαντήσουμε ερώτηματα όπως βρείτε το 6ο στοιχείο εαν το σύνολο ήταν ταξινομημένο.
- Είναι προφανές πως μπορούμε πρώτα να ταξινομήσουμε και μετά να βρούμε το 6ο στοιχείο. Άρα το πρόβλημα λύνεται σε χρόνο $O(n \log n)$.

Το 6ο στοιχείο του παραπάνω συνόλου είναι το 7 αφού η είσοδος ταξινομημένη είναι: $1, 2, 5, 5, 6, 7, 9, 32, 100$.

Μπορούμε να λύσουμε το πρόβλημα με την ιδέα του ”διαίρει και βασίλευε” όπως ακριβώς κάναμε στον αλγόριθμο **quicksort**.

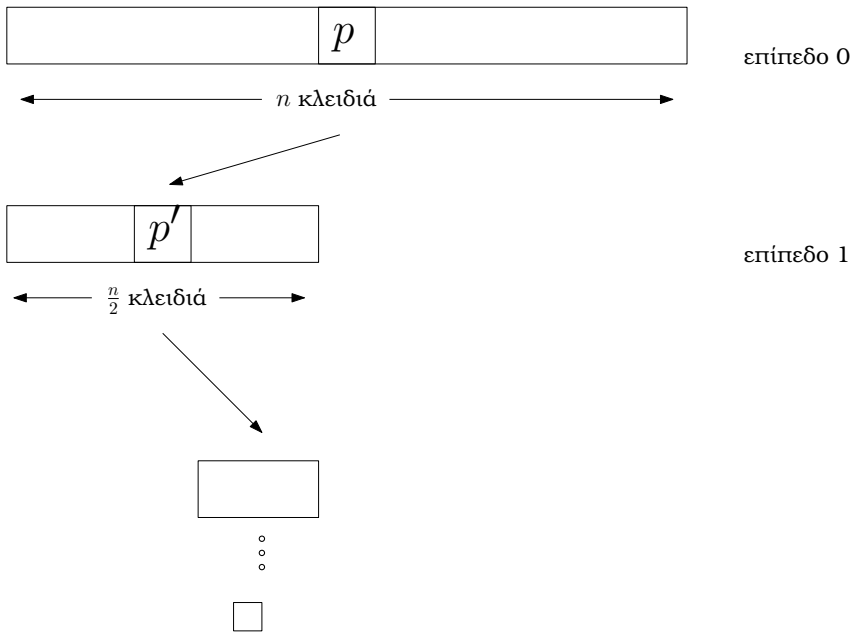
Για αυτό τον λόγο θα ονομάσουμε τον καινούριο αλγόριθμο **quickselect**.

Επιλογή
Quickselect

Έστω πως έχουμε ως είσοδο ένα πίνακα `a[1...n]` με n στοιχεία και έναν αριθμό $1 \leq k \leq n$. Θέλουμε να επιστρέψουμε το στοιχείο του πίνακα `a` που είναι το k-οστό μικρότερο.

1. διάλεξε ένα στοιχείο διαμέρισης `p`
2. κάνε διαμέριση του πίνακα `a[1...n]` με βάση το `p` και έστω πως το `p` καταλήγει στην θέση i
3. εαν $k = i$ τότε βρήκαμε το στοιχείο μας, και το επιστρέφουμε
4. αλλιώς
   - εαν $k < i$ τρέξε αναδρομικά με πίνακα `a[1..i-1]` ψάχνοντας το k στοιχείο
   - εαν $k > i$ τρέξε αναδρομικά με πίνακα `a[i+1..n]` ψάχνοντας το $k - i$ στοιχείο

Αναδρομή του Quickselect



Ο αλγόριθμος δεν κάνει αναδρομικές κλήσεις και προς τις δύο πλευρές όπως ο quicksort αλλά μόνο προς την μια πλευρά.

Αναδρομή του Quickselect

Η απόδοση και αυτού του αλγορίθμου εξαρτάται από την επιλογή του κλειδιού διαμέρισης.

Στην χειρότερη περίπτωση μπορεί να είναι $O(n^2)$, όπως και του quicksort.

Εαν το κλειδί διαμέρισης επιλέγεται κάθε φορά στην τύχη, μπορούμε να δείξουμε με προσεκτική ανάλυση πως ο αλγόριθμος έχει αναμενόμενο χρόνο εκτέλεσης $O(n)$.

Γραμμικός Αλγόριθμος

Θα δείξουμε τώρα έναν αλγόριθμο που τρέχει σε γραμμικό χρόνο $O(n)$ στην χειρότερη περίπτωση.

- ο αλγόριθμος διαλέγει ένα κλειδί διαμέρισης που μοιράζει τον πίνακα σχετικά ισομερώς
- κατά τα άλλα είναι ίδιος με τον quickselect

Η δυσκολία είναι στην σχεδίαση ενός αλγορίθμου επιλογής του στοιχείου διαμέρισης (χωρίς τυχαιότητα).

Έξυπνη Επιλογή Στοιχείου Διαμέρισης

Επιλογή του k-οστού ενός πίνακα $a[1 \ldots n]$

1. Μοίρασε τα n κλειδιά εισόδου σε $\lfloor n/5 \rfloor$ ομάδες των 5 κλειδιών και το πολύ μια ομάδα με τα υπόλοιπα $n\%5$ κλειδιά.

Έξυπνη Επιλογή Στοιχείου Διαμέρισης

Επιλογή του k-οστού ενός πίνακα $a[1 \ldots n]$

1. Μοίρασε τα n κλειδιά εισόδου σε $\lfloor n/5 \rfloor$ ομάδες των 5 κλειδιών και το πολύ μια ομάδα με τα υπόλοιπα $n\%5$ κλειδιά.
2. Βρες τον μέσο κάθε μιας από τις $\lceil n/5 \rceil$ ομάδες.
   Επειδή οι ομάδες έχουν το πολύ 5 κλειδιά μπορούμε να χρησιμοποιήσουμε για παράδειγμα insertion sort.
   Εαν μια ομάδα έχει ζυγό αριθμό κλειδιών, πάρε τον μεγαλύτερο από τους δύο μέσους.

Έξυπνη Επιλογή Στοιχείου Διαμέρισης

Επιλογή του k-οστού ενός πίνακα $a[1 \ldots n]$

1. Μοίρασε τα n κλειδιά εισόδου σε $\lfloor n/5 \rfloor$ ομάδες των 5 κλειδιών και το πολύ μια ομάδα με τα υπόλοιπα $n\%5$ κλειδιά.
2. Βρες τον μέσο κάθε μιας από τις $\lceil n/5 \rceil$ ομάδες.
   Επειδή οι ομάδες έχουν το πολύ 5 κλειδιά μπορούμε να χρησιμοποιήσουμε για παράδειγμα insertion sort.
   Εαν μια ομάδα έχει ζυγό αριθμό κλειδιών, πάρε τον μεγαλύτερο από τους δύο μέσους.
3. Βρες αναδρομικά τον μέσο p των $\lceil n/5 \rceil$ μέσων

Έξυπνη Επιλογή Στοιχείου Διαμέρισης

Επιλογή του k-οστού ενός πίνακα $a[1 \ldots n]$

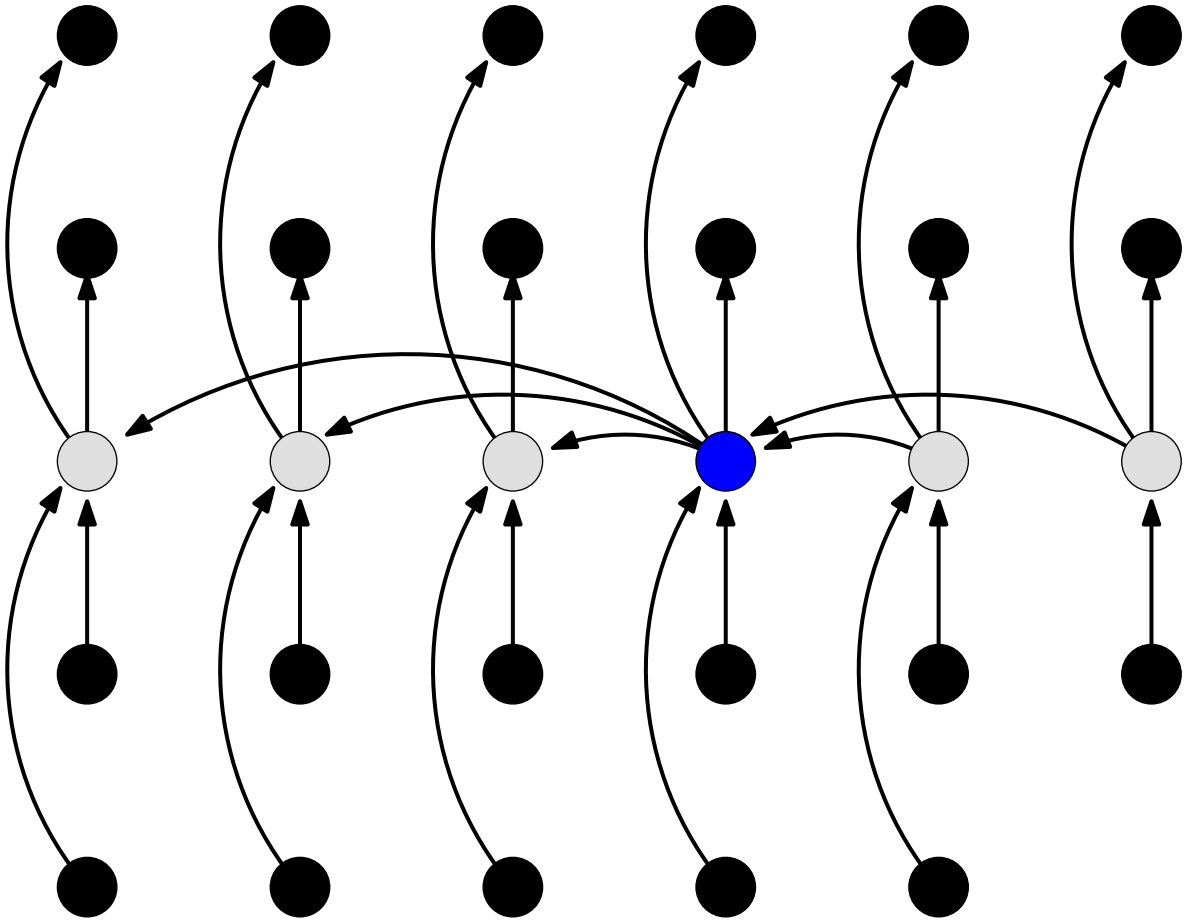
1. Μοίρασε τα n κλειδιά εισόδου σε $\lfloor n/5 \rfloor$ ομάδες των 5 κλειδιών και το πολύ μια ομάδα με τα υπόλοιπα $n\%5$ κλειδιά.
2. Βρες τον μέσο κάθε μιας από τις $\lceil n/5 \rceil$ ομάδες.
   Επειδή οι ομάδες έχουν το πολύ 5 κλειδιά μπορούμε να χρησιμοποιήσουμε για παράδειγμα insertion sort.
   Εαν μια ομάδα έχει ζυγό αριθμό κλειδιών, πάρε τον μεγαλύτερο από τους δύο μέσους.
3. Βρες αναδρομικά τον μέσο p των $\lceil n/5 \rceil$ μέσων
4. Κάνε διαμέριση γύρω από τον μέσο-των-μέσων p

Έξυπνη Επιλογή Στοιχείου Διαμέρισης

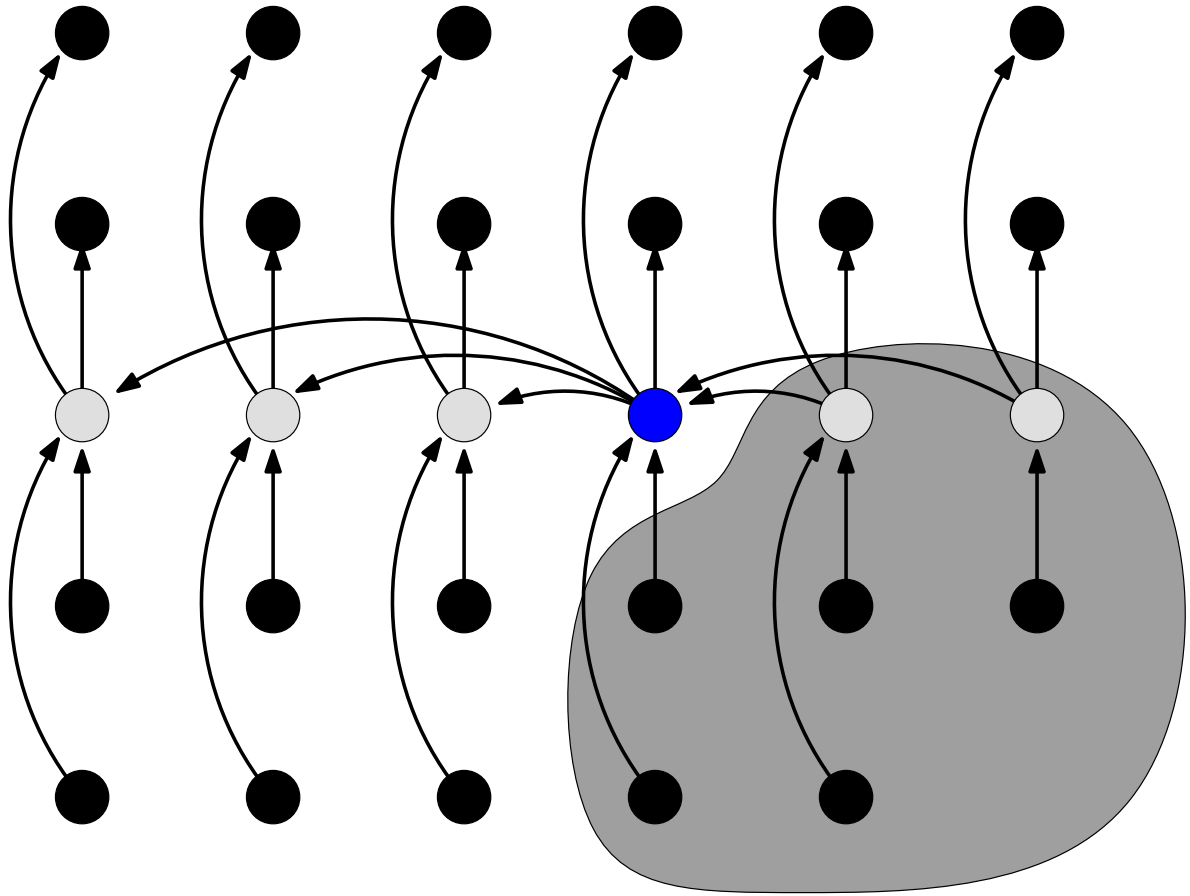
Επιλογή του k-οστού ενός πίνακα $a[1 \ldots n]$

1. Μοίρασε τα n κλειδιά εισόδου σε $\lfloor n/5 \rfloor$ ομάδες των 5 κλειδιών και το πολύ μια ομάδα με τα υπόλοιπα $n\%5$ κλειδιά.
2. Βρες τον μέσο κάθε μιας από τις $\lceil n/5 \rceil$ ομάδες.
   Επειδή οι ομάδες έχουν το πολύ 5 κλειδιά μπορούμε να χρησιμοποιήσουμε για παράδειγμα insertion sort.
   Εαν μια ομάδα έχει ζυγό αριθμό κλειδιών, πάρε τον μεγαλύτερο από τους δύο μέσους.
3. Βρες αναδρομικά τον μέσο p των $\lceil n/5 \rceil$ μέσων
4. Κάνε διαμέριση γύρω από τον μέσο-των-μέσων p
5. Ανάλογα με την θέση του p σε σχέση με το k, επέστρεψε το p ή τρέξε αναδρομικά τον αλγόριθμο για ένα από τα δύο υποπροβλήματα όπως ακριβώς και στον quickselect.

Ιδιότητες Μέσου-των-Μέσων



Ιδιότητες Μέσου-των-Μέσων



Ιδιότητες Μέσου-των-Μέσων

Θεώρημα

Ο αριθμός των κλειδιών που είναι μεγαλύτερα από τον μέσο-των-μέσων p είναι τουλάχιστον $\frac{3n}{10} - 6$.

Ιδιότητες Μέσου-των-Μέσων

Θεώρημα

Ο αριθμός των κλειδιών που είναι μεγαλύτερα από τον μέσο-των-μέσων p είναι τουλάχιστον $\frac{3n}{10} - 6$.

Απόδειξη

Τουλάχιστον μισοί από τους μέσους των $\lceil n/5 \rceil$ ομάδων είναι μεγαλύτεροι ή ίσοι του p.

Ιδιότητες Μέσου-των-Μέσων

Θεώρημα

Ο αριθμός των κλειδιών που είναι μεγαλύτερα από τον μέσο-των-μέσων p είναι τουλάχιστον $\frac{3n}{10} - 6$.

Απόδειξη

Τουλάχιστον μισοί από τους μέσους των $\lceil n/5 \rceil$ ομάδων είναι μεγαλύτεροι ή ίσοι του p.

Άρα τουλάχιστον οι μισές από τις $\lceil n/5 \rceil$ ομάδες παρέχουν 3 κλειδιά μεγαλύτερα του p, εκτός από μια ομάδα που έχει λιγότερα από 5 κλειδιά (εάν το 5 δεν διαιρεί το n) και μία ομάδα ακόμη που περιέχει το p.

Ιδιότητες Μέσου-των-Μέσων

Θεώρημα

Ο αριθμός των κλειδιών που είναι μεγαλύτερα από τον μέσο-των-μέσων p είναι τουλάχιστον $\frac{3n}{10} - 6$.

Απόδειξη

Τουλάχιστον μισοί από τους μέσους των $\lceil n/5 \rceil$ ομάδων είναι μεγαλύτεροι ή ίσοι του p.

Άρα τουλάχιστον οι μισές από τις $\lceil n/5 \rceil$ ομάδες παρέχουν 3 κλειδιά μεγαλύτερα του p, εκτός από μια ομάδα που έχει λιγότερα από 5 κλειδιά (εάν το 5 δεν διαιρεί το n) και μία ομάδα ακόμη που περιέχει το p.

Μη μετρώντας αυτές τις δύο ομάδες προκύπτει πως τα κλειδιά που μας ενδιαφέρουν είναι τουλάχιστον $3(\lceil \frac{1}{2} \lceil \frac{n}{5} \rceil \rceil - 2) \geq \frac{3n}{10} - 6$. □

Ιδιότητες Μέσου-των-Μέσων

Παρόμοια μπορούμε να δείξουμε πως τα κλειδιά που είναι μικρότερα του p είναι τουλάχιστον $\frac{3n}{10} - 6$.

Μπορούμε λοιπόν να πούμε πως η αναδρομή που κάνουμε στον αλγόριθμο θα έχει ένα υποπρόβλημα με το πολύ

$$n - (\frac{3n}{10} - 6) = \frac{7n}{10} + 6$$

κλειδιά.

Ανάλυση Χρόνου

Έστω $T(n)$ ο χειρότερος χρόνος του αλγορίθμου μας σε ένα πίνακα μεγέθους n.

1. η δημιουργία των $\lceil \frac{n}{5} \rceil$ ομάδων πέρνει $O(n)$ χρόνο.

Ανάλυση Χρόνου

Έστω $T(n)$ ο χειρότερος χρόνος του αλγορίθμου μας σε ένα πίνακα μεγέθους n.

1. η δημιουργία των $\lceil \frac{n}{5} \rceil$ ομάδων πέρνει $O(n)$ χρόνο.
2. για να βρούμε τους μέσους των $\lceil \frac{n}{5} \rceil$ ομάδων χρειαζόμαστε $O(n)$ χρόνο (καλούμε την insertion sort $O(n)$ φορές σε είσοδο μεγέθους $O(1)$.

Ανάλυση Χρόνου

Έστω $T(n)$ ο χειρότερος χρόνος του αλγορίθμου μας σε ένα πίνακα μεγέθους n.

1. η δημιουργία των $\lceil \frac{n}{5} \rceil$ ομάδων πέρνει $O(n)$ χρόνο.
2. για να βρούμε τους μέσους των $\lceil \frac{n}{5} \rceil$ ομάδων χρειαζόμαστε $O(n)$ χρόνο (καλούμε την insertion sort $O(n)$ φορές σε είσοδο μεγέθους $O(1)$.
3. για να βρούμε τον μέσο-των-μέσων καλούμε την συνάρτηση αναδρομικά και άρα χρειάζεται χρόνο $T(\lceil \frac{n}{5} \rceil)$.

Ανάλυση Χρόνου

Έστω $T(n)$ ο χειρότερος χρόνος του αλγορίθμου μας σε ένα πίνακα μεγέθους n.

1. η δημιουργία των $\lceil \frac{n}{5} \rceil$ ομάδων πέρνει $O(n)$ χρόνο.
2. για να βρούμε τους μέσους των $\lceil \frac{n}{5} \rceil$ ομάδων χρειαζόμαστε $O(n)$ χρόνο (καλούμε την insertion sort $O(n)$ φορές σε είσοδο μεγέθους $O(1)$.
3. για να βρούμε τον μέσο-των-μέσων καλούμε την συνάρτηση αναδρομικά και άρα χρειάζεται χρόνο $T(\lceil \frac{n}{5} \rceil)$.
4. η διαμέριση του πίνακα σε σχέση με τον μέσο-των-μέσων θέλει χρόνο $O(n)$.

Ανάλυση Χρόνου

Έστω $T(n)$ ο χειρότερος χρόνος του αλγορίθμου μας σε ένα πίνακα μεγέθους n.

1. η δημιουργία των $\lceil \frac{n}{5} \rceil$ ομάδων πέρνει $O(n)$ χρόνο.
2. για να βρούμε τους μέσους των $\lceil \frac{n}{5} \rceil$ ομάδων χρειαζόμαστε $O(n)$ χρόνο (καλούμε την insertion sort $O(n)$ φορές σε είσοδο μεγέθους $O(1)$.
3. για να βρούμε τον μέσο-των-μέσων καλούμε την συνάρτηση αναδρομικά και άρα χρειάζεται χρόνο $T(\lceil \frac{n}{5} \rceil)$.
4. η διαμέριση του πίνακα σε σχέση με τον μέσο-των-μέσων θέλει χρόνο $O(n)$.
5. η αναδρομική λύση σε ένα μικρότερο υποπρόβλημα πέρνει το πολύ χρόνο $T(\frac{7n}{10} + 6)$ υποθέτοντας πως η συνάρτηση T είναι αύξουσα.

Ανάλυση Χρόνου

Έστω $T(n)$ ο χειρότερος χρόνος του αλγορίθμου μας σε ένα πίνακα μεγέθους n.

Με βάση τα προηγούμενα:

$$T(n) \leq T(\lceil n/5 \rceil) + T(7n/10 + 6) + c' \cdot n$$

Η παραπάνω αναδρομή λύνει σε $T(n) \leq c \cdot n$ για κάποια σταθερά c.

Ειδικές Ιδιότητες Κλειδιών

Ως τώρα είδαμε αλγορίθμους ταξινόμησης που έκαναν συγκρίσεις και αντιμεταθέσεις.

Τώρα θα δούμε τι γίνεται εαν χρησιμοποιήσουμε περισσότερη πληροφορία για την αναπαράσταση των κλειδιών, π.χ την γνώση πως οι αριθμοί είναι όλοι ακέραιοι, κ.τ.λ.

Η ταξινόμηση μέτρησης θεωρεί πως η είσοδος είναι ακέραιοι στο διάστημα μεταξύ 0 και k, όπου k ακέραιος.

Για $k = O(n)$ η ταξινόμηση εκτελείται σε χρόνο $O(n)$.

Βασική Ιδέα

Πρέπει να προσδιορίσουμε για κάθε στοιχείο x της εισόδου, πόσα στοιχεία της εισόδου είναι μικρότερα από το x. Έπειτα μπορούμε να βρούμε απευθείας την θέση του x στην έξοδο.

Ταξινόμηση Δοχείων

Παράδειγμα

Έστω ο παρακάτω πίνακας που θέλουμε να ταξινομήσουμε

3	1	4	1	5	9	2	6	5	4

Φτιάχνουμε ένα βοηθητικό πίνακα μεγέθους k, όπου k είναι ο μεγαλύτερος ακέραιος που υπάρχει στην είσοδο μας.

Ο πίνακας αυτός θα περιέχει στην θέση i τον αριθμό των φορών που ο αριθμός i υπάρχει στην είσοδο μας.

0	2	1	1	2	2	1	0	0	1

Μπορούμε να υπολογίσουμε αυτές τις τιμές με μια διάσχιση του αρχικού μας πίνακα.

Με μια ακόμη διάσχιση του καινούριου πίνακα

0	2	1	1	2	2	1	0	0	1

μπορούμε πλέον να ταξινομήσουμε τους αριθμούς μας.

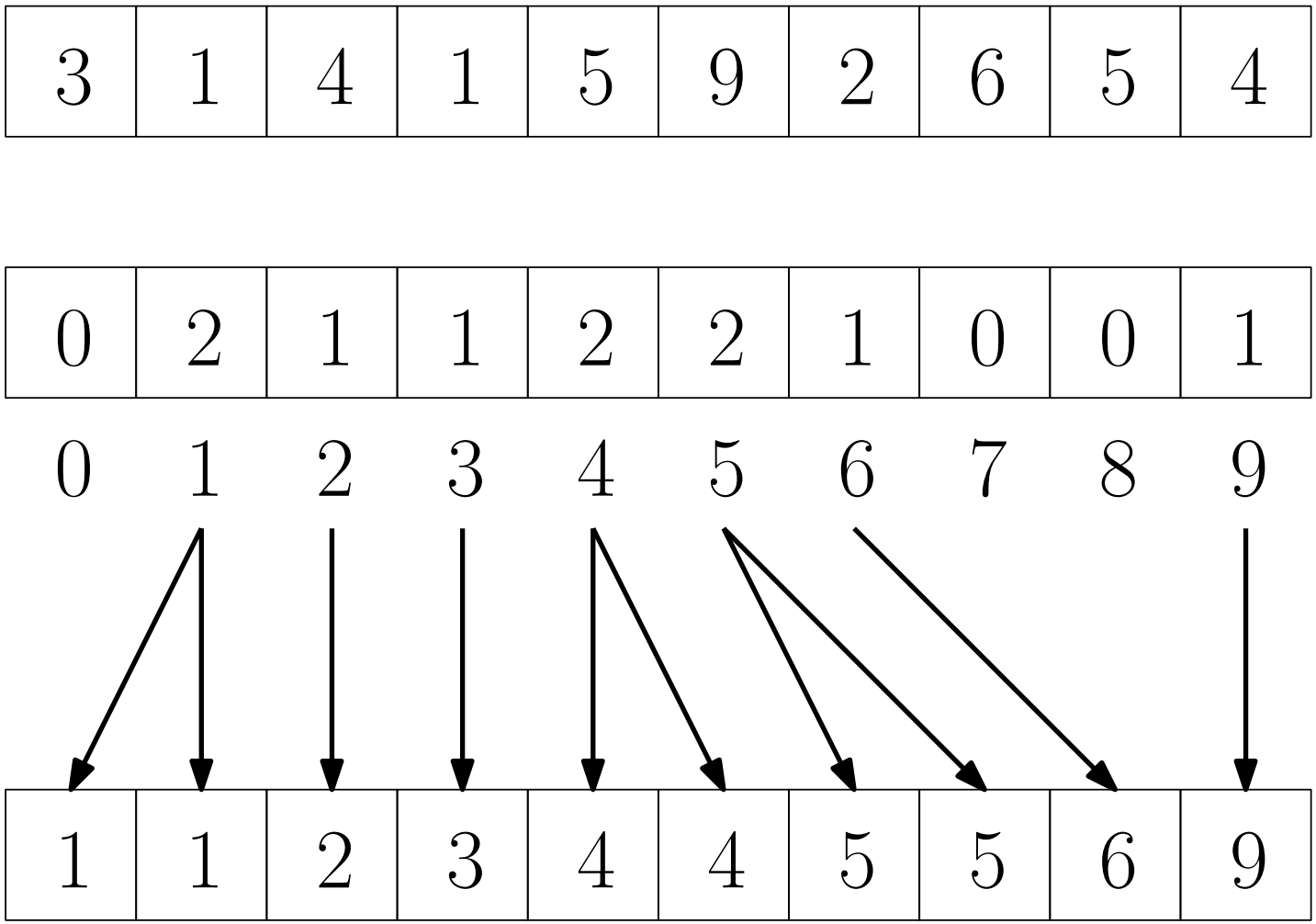
Ο αλγόριθμος είναι πολύ απλός.

Για κάθε *i* από 0 έως *k* τύπωσε στην έξοδο τον αριθμό *i* όσες φορές λέει η θέση *i* του πίνακα.

1	1	2	3	4	4	5	5	6	9

Ταξινόμηση Δοχείων

Παράδειγμα



Ταξινόμηση Βάσης

Radix Sort

Μέθοδος ταξινόμησης που κοιτάει τα κλειδιά ανά τμήματα:

- δυαδικοί αριθμοί είναι ακολουθίες από bit,
- αλφαριθμητικά είναι ακολουθίες χαρακτήρων,
- δεκαδικοί αριθμοί είναι ακολουθίες ψηφίων.

Η μέθοδος αυτή θεωρεί πως τα κλειδιά είναι αριθμοί που αναπαρίστανται σε ένα σύστημα αρίθμησης με βάση R, για διάφορες τιμές του R.

Ταξινόμηση Βάσης (lsd)

Ας υποθέσουμε πως $R = 10$, άρα βλέπουμε την είσοδο ως δεκαδικούς αριθμούς: 34, 12, 42, 32, 44, 41, 34, 11, 32 και 23. Κάθε αριθμός έχει 2 ψηφία.

Η μέθοδος που θα παρουσιάσουμε ταξινομεί τους αριθμούς με βάση τα ψηφία τους.

- πρώτα ταξινομεί τους αριθμούς με βάση το 1ο ψηφίο από δεξιά (least significant digit),
- και έπειτα ταξινομεί τους αριθμούς με βάση το 2ο ψηφίο

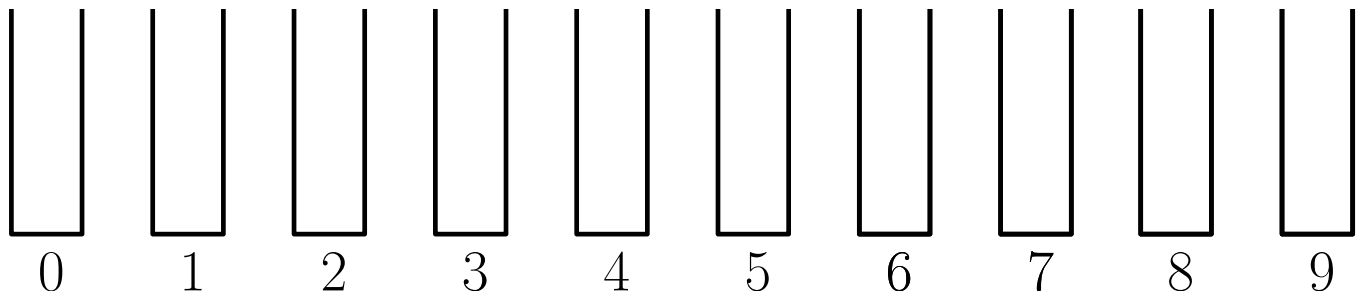
Για να λειτουργήσει σωστά πρέπει η μέθοδος ταξινόμησης που θα χρησιμοποιήσουμε για κάθε ψηφίο να είναι **ευσταθής**. Θα δούμε π.χ πως δουλεύει με την ταξινόμηση δοχείων.

Ταξινόμηση Βάσης (lsd)

Με χρήση δοχείων

Ας υποθέσουμε πως $R = 10$, άρα βλέπουμε την είσοδο ως δεκαδικούς αριθμούς: 34, 12, 42, 32, 44, 41, 34, 11, 32 και 23. Κάθε αριθμός έχει 2 ψηφία.

Επειδή $R = 10$ έχουμε 10 δοχεία που αντιστοιχούν στα 0 έως και 9. Για κάθε ένα από τα ψηφία, θα πετάξουμε τους αριθμούς με την σειρά στα δοχεία και θα τους βγάλουμε με την σειρά των δοχείων.

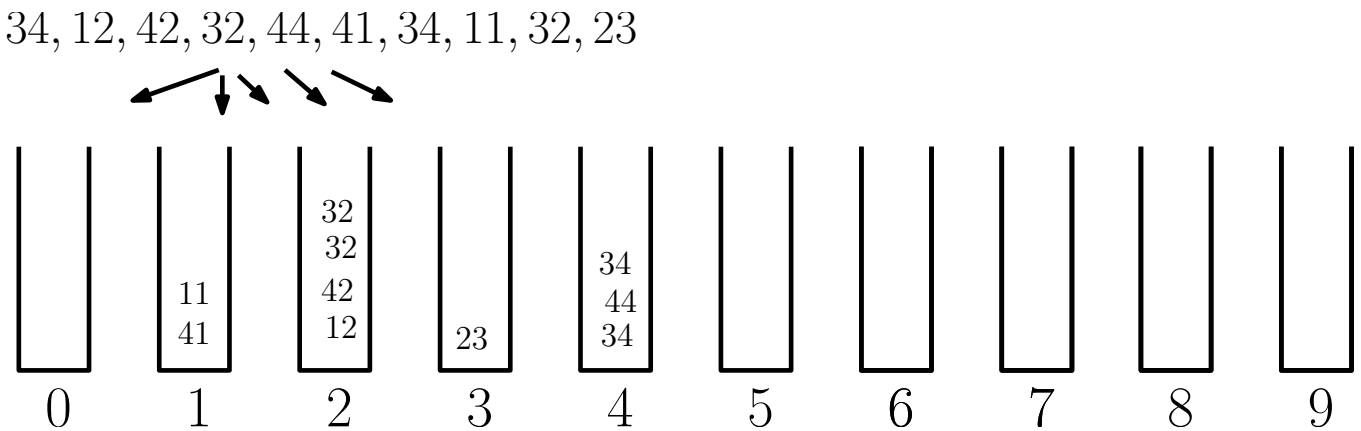


Ταξινόμηση Βάσης (lsd)

Με χρήση δοχείων

Ας υποθέσουμε πως $R = 10$, άρα βλέπουμε την είσοδο ως δεκαδικούς αριθμούς: 34, 12, 42, 32, 44, 41, 34, 11, 32 και 23. Κάθε αριθμός έχει 2 ψηφία.

Επειδή $R = 10$ έχουμε 10 δοχεία που αντιστοιχούν στα 0 έως και 9. Για κάθε ένα από τα ψηφία, θα πετάξουμε τους αριθμούς με την σειρά στα δοχεία και θα τους βγάλουμε με την σειρά των δοχείων.

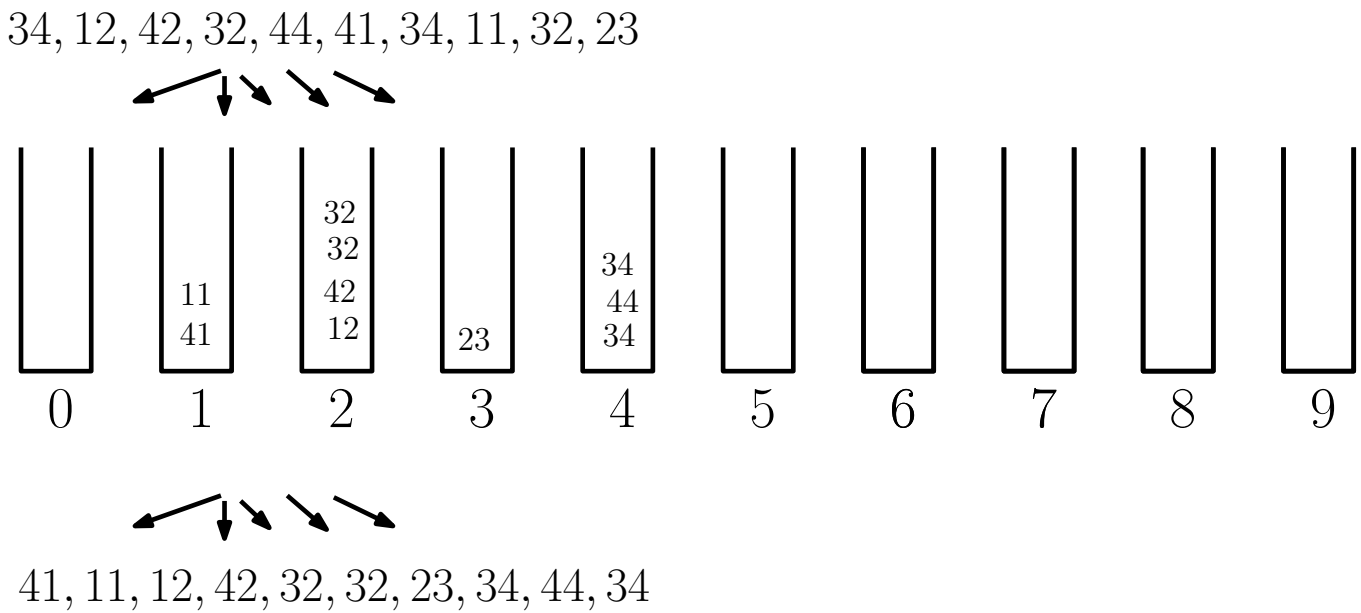


Ταξινόμηση Βάσης (lsd)

Με χρήση δοχείων

Ας υποθέσουμε πως $R = 10$, άρα βλέπουμε την είσοδο ως δεκαδικούς αριθμούς: 34, 12, 42, 32, 44, 41, 34, 11, 32 και 23. Κάθε αριθμός έχει 2 ψηφία.

Επειδή $R = 10$ έχουμε 10 δοχεία που αντιστοιχούν στα 0 έως και 9. Για κάθε ένα από τα ψηφία, θα πετάξουμε τους αριθμούς με την σειρά στα δοχεία και θα τους βγάλουμε με την σειρά των δοχείων.

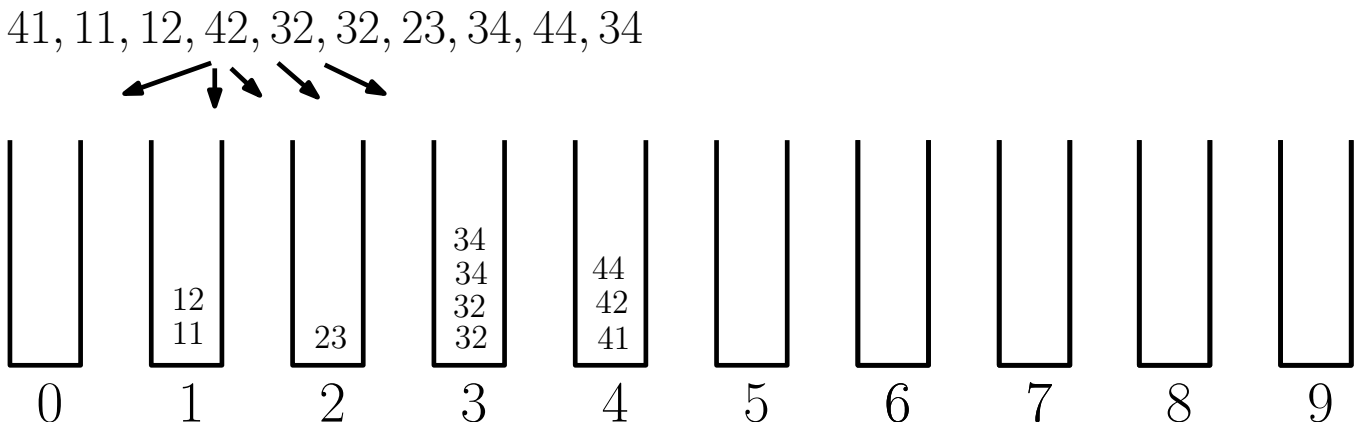


Ταξινόμηση Βάσης (lsd)

Με χρήση δοχείων

Ας υποθέσουμε πως $R = 10$, άρα βλέπουμε την είσοδο ως δεκαδικούς αριθμούς: 34, 12, 42, 32, 44, 41, 34, 11, 32 και 23. Κάθε αριθμός έχει 2 ψηφία.

Επειδή $R = 10$ έχουμε 10 δοχεία που αντιστοιχούν στα 0 έως και 9. Για κάθε ένα από τα ψηφία, θα πετάξουμε τους αριθμούς με την σειρά στα δοχεία και θα τους βγάλουμε με την σειρά των δοχείων.

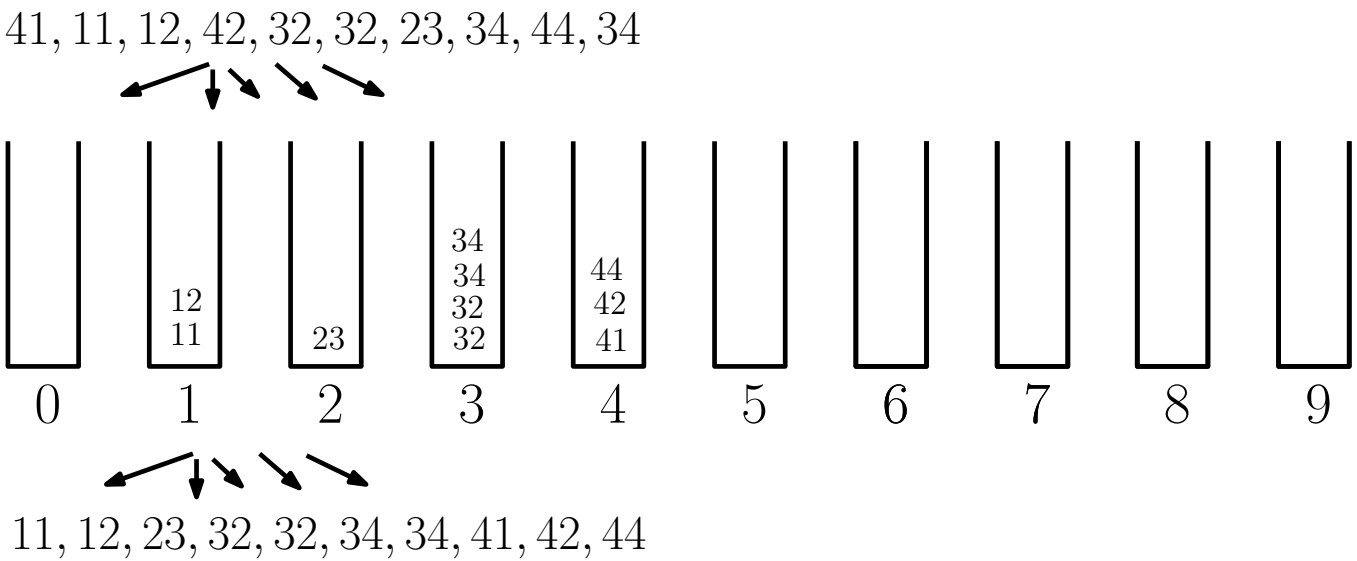


Ταξινόμηση Βάσης (lsd)

Με χρήση δοχείων

Ας υποθέσουμε πως $R = 10$, άρα βλέπουμε την είσοδο ως δεκαδικούς αριθμούς: 34, 12, 42, 32, 44, 41, 34, 11, 32 και 23. Κάθε αριθμός έχει 2 ψηφία.

Επειδή $R = 10$ έχουμε 10 δοχεία που αντιστοιχούν στα 0 έως και 9. Για κάθε ένα από τα ψηφία, θα πετάξουμε τους αριθμούς με την σειρά στα δοχεία και θα τους βγάλουμε με την σειρά των δοχείων.



Ταξινόμηση Βάσης (lsd)

Παράδειγμα

Έστω οι αριθμοί:

$$701, 132, 120, 720, 121, 035, 001, 699, 792$$

Ταξινόμηση Βάσης (lsd)

Παράδειγμα

Έστω οι αριθμοί:

$$701, 132, 120, 720, 121, 035, 001, 699, 792$$

Ταξινομώντας πρώτα με το 1ο ψηφίο (από δεξιά):

$$120, 720, 701, 121, 001, 132, 792, 035, 699$$

Ταξινόμηση Βάσης (lsd)

Παράδειγμα

Έστω οι αριθμοί:

$$701, 132, 120, 720, 121, 035, 001, 699, 792$$

Ταξινομώντας πρώτα με το 1ο ψηφίο (από δεξιά):

$$120, 720, 701, 121, 001, 132, 792, 035, 699$$

Ταξινομώντας με το 2ο ψηφίο (από δεξιά):

$$701, 001, 120, 720, 121, 132, 035, 792, 699$$

Ταξινόμηση Βάσης (lsd)

Παράδειγμα

Έστω οι αριθμοί:

$$701, 132, 120, 720, 121, 035, 001, 699, 792$$

Ταξινομώντας πρώτα με το 1ο ψηφίο (από δεξιά):

$$120, 720, 701, 121, 001, 132, 792, 035, 699$$

Ταξινομώντας με το 2ο ψηφίο (από δεξιά):

$$701, 001, 120, 720, 121, 132, 035, 792, 699$$

Ταξινομώντας με το 3ο ψηφίο (από δεξιά):

$$001, 035, 120, 121, 132, 699, 701, 720, 792$$

Ταξινόμηση Βάσης (lsd)

Εαν η μέθοδος ταξινόμησης που χρησιμοποιούμε δεν είναι ευσταθής, το αποτέλεσμα δεν είναι πάντα σωστό.

π.χ με ευσταθή ταξινόμηση

$$\begin{matrix}01\\11\\10\\00\end{matrix} \rightarrow \begin{matrix}1\mathbf{0}\\0\mathbf{0}\\0\mathbf{1}\\1\mathbf{1}\end{matrix} \rightarrow \begin{matrix}\mathbf{0}0\\\mathbf{0}1\\\mathbf{1}0\\\mathbf{1}1\end{matrix}$$

ενώ με μη-ευσταθή ταξινόμηση

$$\begin{matrix}01\\11\\10\\00\end{matrix} \rightarrow \begin{matrix}1\mathbf{0}\\0\mathbf{0}\\0\mathbf{1}\\1\mathbf{1}\end{matrix} \rightarrow \begin{matrix}\mathbf{0}1\\\mathbf{0}0\\\mathbf{1}0\\\mathbf{1}1\end{matrix}$$

Ταξινόμηση Βάσης (lsd)

Χρόνος Εκτέλεσης

Ο χρόνος εκτέλεσης εάν υποθέσουμε βάση R και χρήση του γραμμικού αλγόριθμου ταξινόμησης με δοχεία είναι $O(w \cdot n)$ όπου

- n είναι το μέγεθος της εισόδου που θέλουμε να ταξινομήσουμε και
- w είναι ο αριθμός των ψηφίων που χρειάζονται τα στοιχεία της εισόδου για να αναπαρασταθούν σε βάση R.

Έστω για παράδειγμα πως θέλουμε να ταξινομήσουμε n συμβολοσειρές (strings) όπου κάθε συμβολοσειρά έχει μήκος το πολύ $\mathcal{L}$, και υποθέσουμε πως οι χαρακτήρες χρησιμοποιούν την κωδικοποίηση Extended-ASCII (1 byte για κάθε χαρακτήρα).

Ταξινόμηση Βάσης (lsd)

Παράδειγμα Συμβολοσειρών

Μπορούμε να χρησιμοποιήσουμε για βάση ένα byte, δηλαδή $R = 256$. Τα ψηφία μας σε αυτήν την περίπτωση είναι όλοι οι χαρακτήρες του πίνακα ASCII.

Για να υλοποιήσουμε την ταξινόμηση με δοχεία, χρειαζόμαστε 256 δοχεία, από 0 έως και 255.

Ο αλγόριθμος λοιπόν θα ταξινομήσει τις συμβολοσειρές ως προς κάθε γράμμα (ξεκινώντας από το δεξιότερο) σε χρόνο $O(\mathcal{L} \cdot n)$.

Ταξινόμηση Βάσης (lsd)

Παράδειγμα Συμβολοσειρών

now		ac**e**		w**as**		**ace**
for		we**e**		r**aw**		**ago**
tip		ow**l**		a**ce**		**bet**
ace		ag**o**		we**e**		**for**
hut		ti**p**		b**et**		**hut**
bet	$\rightarrow$	fo**r**	$\rightarrow$	a**go**	$\rightarrow$	**joy**
owl		wa**s**		ti**p**		**now**
joy		hu**t**		fo**r**		**owl**
wee		be**t**		n**ow**		**raw**
was		no**w**		j**oy**		**tip**
raw		ra**w**		h**ut**		**was**
ago		jo**y**		o**wl**		**wee**

Ταξινόμηση Βάσης (msd)

Υπάρχει και η έκδοση ταξινόμησης βάσης **most significant digit** που ξεκινάμε από το πιο σημαντικό ψηφίο της κωδικοποίησης.

Ο αλγόριθμος ταξινομεί αρχικά όλα τα κλειδιά σε R δοχεία με βάση το πιο σημαντικό ψηφίο.

Στην συνέχεια ταξινομεί αναδρομικά τα περιεχόμενα των δοχείων με βάση τα κλειδιά με ένα λιγότερο ψηφίο.

Ταξινόμηση Βάσης (msd)

Παράδειγμα Συμβολοσειρών