Αντικειμενοστρεφής Προγραμματισμός

Παναγιώτης Αδαμίδης
adamidis@ihu.gr

Ταξινόμηση (Sorting)

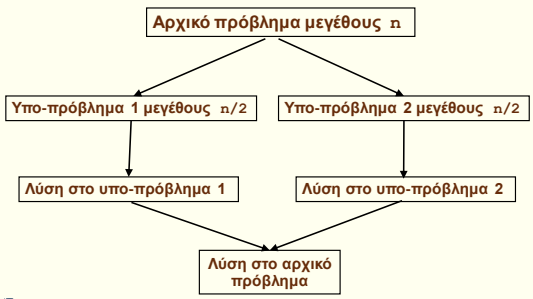
Αναδρομικές Μέθοδοι
(Merge Sort, Quick Sort)

Ταξινόμηση με συγχώνευση (Merge sort)

Γενικά

- Η ταξινόμηση με συγχώνευση ανήκει στην κατηγορία των αλγορίθμων «διαίρει και βασίλευε» (divide and conquer).
- Είναι από τους πιο αποτελεσματικούς αλγόριθμους ταξινόμησης, ιδιαίτερα σε δεδομένα που είναι αποθηκευμένα στη δευτερεύουσα μνήμη.
- Η χρονική πολυπλοκότητά της είναι O(nlogn), στη χειρότερη περίπτωση.
- Είναι ανεξάρτητη από τη διάταξη των δεδομένων.
- Επινοήθηκε από τον John von Neumann το 1945 για τον EDVAC.

Divide and Conquer



Divide and Conquer Merge Sort

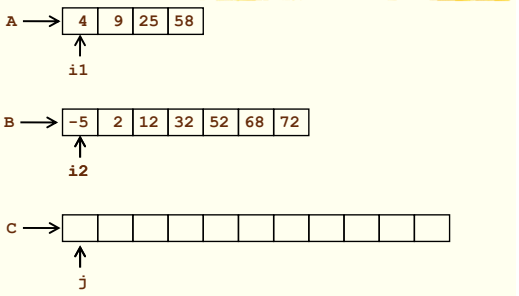
- Η ταξινόμηση με συγχώνευση χωρίζει τα δεδομένα στη μέση και μετά συγχωνεύει τα ταξινομημένα δεδομένα με τη σωστή σειρά.
- **Divide**: Διαιρεί τα μη ταξινομημένα δεδομένα στη μέση σε δύο υποσύνολα.
- **Conquer**: Ταξινόμηση των δύο υποσυνόλων αναδρομικά μέχρι το μέγεθος των υποσυνόλων να γίνει 1.
- **Συγχώνευση**: Συγχώνευση των δύο υποσυνόλων σε ένα ταξινομημένο.

Ταξινόμηση με Συγχώνευση Κώδικας

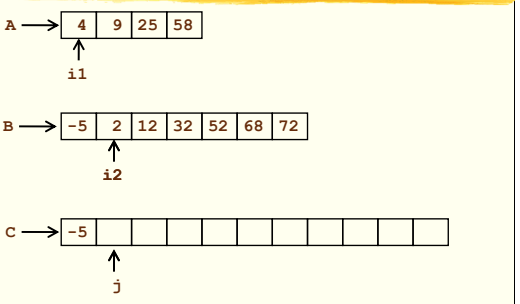
- Παράδειγμα μεθόδου:

```
public static void sort (int[] A) {
   mSort (A, 0, A.length-1);
}

public static void mSort (int[] A, int f, int l) {
   if (f==l) return;
   int mid=(f+l)/2;      // Μεσαία θέση
   mSort (A, f, mid);    // Αναδρομική κλήση, 1ο μισό
   mSort (A, mid+1, l); // Αναδρομική κλήση, 2ο μισό
   merge(A,f,l,mid);     // Συγχώνευση
}
```

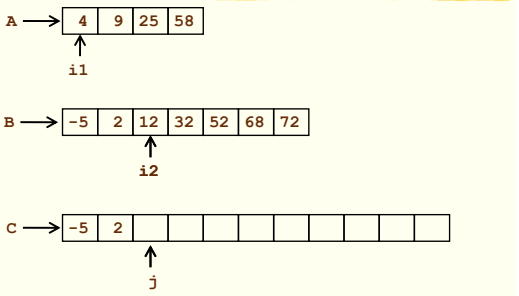
Συγχώνευση
Παράδειγμα λειτουργίας



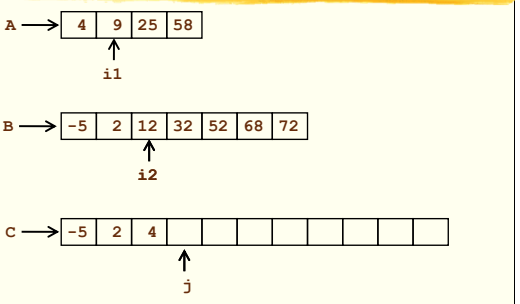
Συγχώνευση
Παράδειγμα λειτουργίας



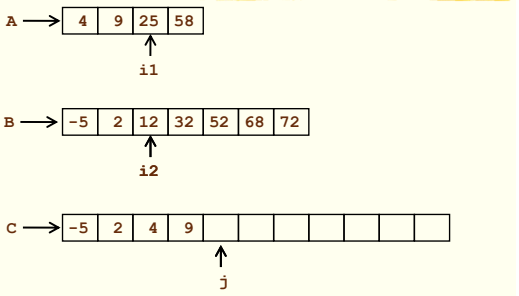
Συγχώνευση
Παράδειγμα λειτουργίας



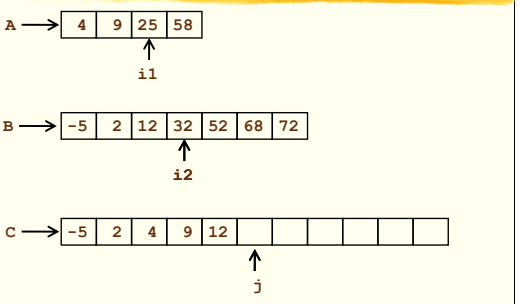
Συγχώνευση
Παράδειγμα λειτουργίας



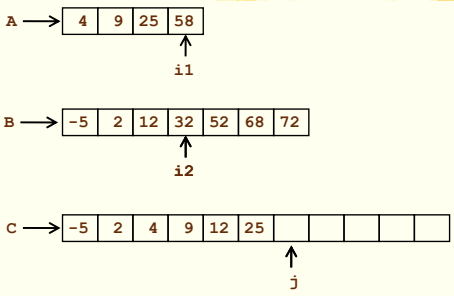
Συγχώνευση
Παράδειγμα λειτουργίας



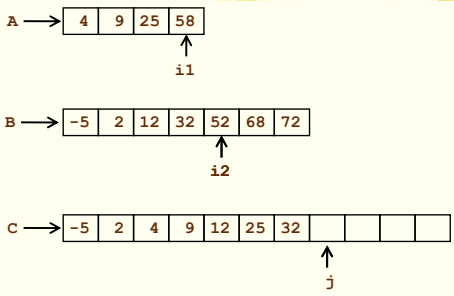
Συγχώνευση
Παράδειγμα λειτουργίας



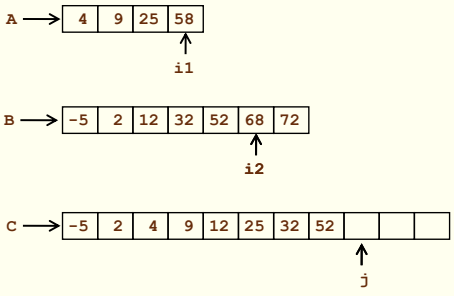
Συγχώνευση
Παράδειγμα λειτουργίας



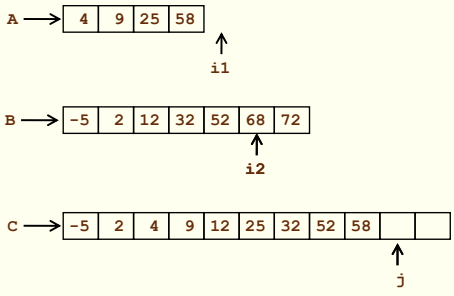
Συγχώνευση
Παράδειγμα λειτουργίας



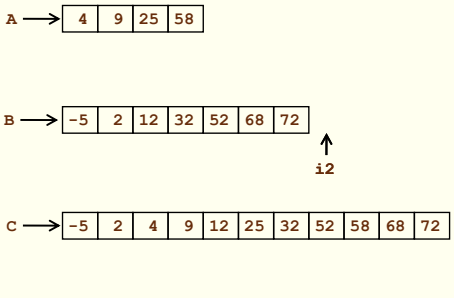
Συγχώνευση
Παράδειγμα λειτουργίας



Συγχώνευση
Παράδειγμα λειτουργίας



Συγχώνευση
Παράδειγμα λειτουργίας



Συγχώνευση Πινάκων: Κώδικας 1

```
public static void merge (int A[], int[] B, int[] C) {
     int i1=0, i2=0, j=0;

     while (i1<=A.length && i2<=B.length) {
            if (A[i1] < B[i2]) {
                C[j]=A[i1];
                i1++;
            }
            else {
                C[j]=B[i2];
                i2++;
            }
            j++;
     }
```

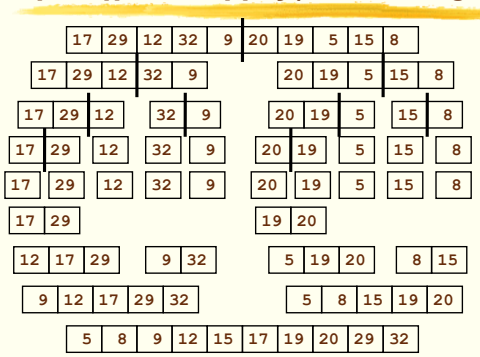
Συγχώνευση Πινάκων: Κώδικας 2

```
    // Copy the rest elements of array A
    while (i1<A.length) {
           C[j]=A[i1];
           i1++;
           j++;
    }

     // Copy the rest elements of array B
    while (i2<B.length) {
           C[j]=B[i2];
           i2++;
           j++;
    }
}
```

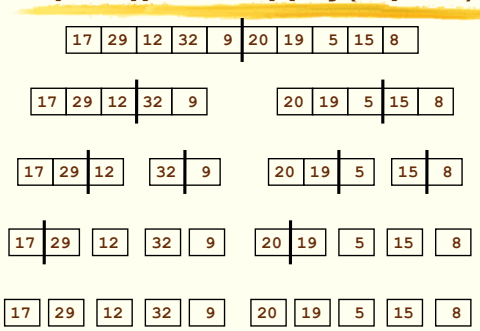
Συγχώνευση Πινάκων: Κώδικας (Συμπύκνωση)

```
public static void merge (int A[], int B, int C) {
     int i1=0, i2=0, j=0;

     while (i1<=A.length && i2<=B.length) {
            if (A[i1] < B[i2]) C[j++]=A[i1++];
            else C[j++]=B[i2++];
     }

     // Copy the rest elements of array A
     while (i1<A.length)
            C[j++]=A[i1++];

      // Copy the rest elements of array B
     while (i2<B.length)
            C[j++]=B[i2++];
}
```

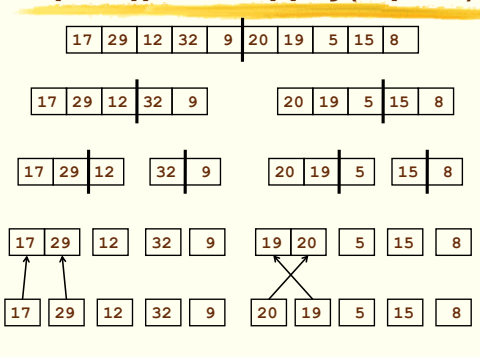
Ταξινόμηση με Συγχώνευση
Παράδειγμα λειτουργίας (double storage)



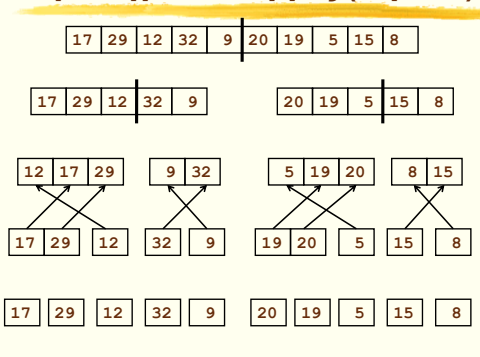
Ταξινόμηση με Συγχώνευση
Παράδειγμα λειτουργίας (in place)



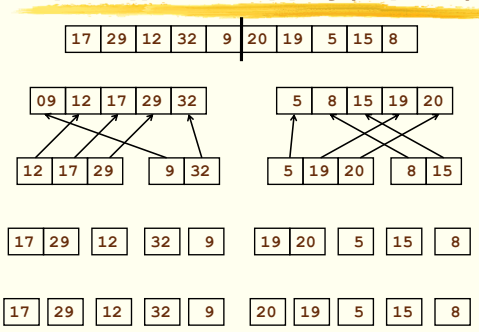
Ταξινόμηση με Συγχώνευση
Παράδειγμα λειτουργίας (in place)



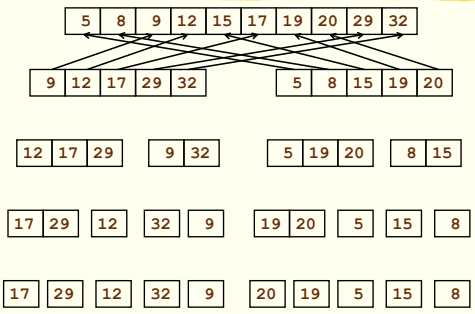
Ταξινόμηση με Συγχώνευση
Παράδειγμα λειτουργίας (in place)



Ταξινόμηση με Συγχώνευση Παράδειγμα λειτουργίας (in place)



Ταξινόμηση με Συγχώνευση Παράδειγμα λειτουργίας (in place)



Ερωτήσεις;

Γρήγορη Ταξινόμηση (Quick Sort)

Γενικά

- Η γρήγορη ταξινόμηση ανήκει στην κατηγορία των αλγορίθμων «διαίρει και βασίλευε» (divide and conquer).
- Είναι από τους πιο αποτελεσματικούς αλγόριθμους ταξινόμησης.
- Η χρονική πολυπλοκότητά της είναι O(nlogn).
- Επινοήθηκε από τον Tony Hoare το 1960.
- Κάποιες φορές αναφέρεται και ως partition exchange sort.

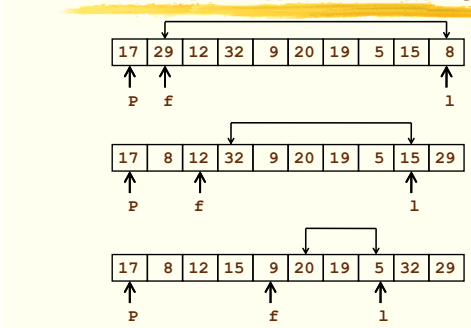
Divide and Conquer Quick Sort

- Η γρήγορη ταξινόμηση χωρίζει τον πίνακα σε δύο υπο-πίνακες έτσι ό αριστερός να έχει τα μικρότερα και ο δεξιός τα μεγαλύτερα στοιχεία.
- **Divide**: Διαιρεί τα μη ταξινομημένα δεδομένα στη μέση γύρω από μια κεντρική τιμή (pivot) έτσι ώστε τα μικρότερα να είναι αριστερά και τα μεγαλύτερα δεξιά της κεντρικής τιμής.
- **Conquer**: Αναδρομική επανάληψη της διαδικασίας με τον ίδιο τρόπο μέχρι το μέγεθος των υπο-πινάκων να γίνει 1.

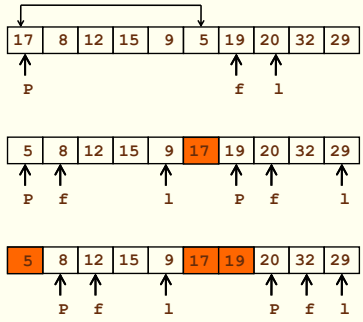
Quick Sort: Λειτουργία

- Αρχή με την επιλογή του Κεντρικού Στοιχείου (Pivot), το οποίο πάει στην 1η θέση.
- Ορίζουμε δύο δείκτες. Ένας στο πρώτο (1ο) στοιχείο (first) μετά το Pivot και ένας στο τελευταίο (last).
- Σύγκριση της τιμής που δείχνει ο δείκτης first με το Pivot. Εάν το Pivot είναι μεγαλύτερο συνεχίζουμε με το επόμενο στοιχείο μέχρι να βρεθεί ένα στοιχείο που θα είναι μεγαλύτερο του Pivot.
- Σύγκριση της τιμής που δείχνει ο δείκτης last με το Pivot. Εάν το Pivot είναι μικρότερο συνεχίζουμε με το προηγούμενο στοιχείο μέχρι να βρεθεί ένα στοιχείο που θα είναι μικρότερο του Pivot.
- Ανταλλαγή των στοιχείων που δείχνουν τα first και last.
- Η διαδικασία συνεχίζεται για όσο ισχύει: first<last.
- Αλλαγή του Pivot με το στοιχείο τη θέση last ή first.
- Αναδρομική επανάληψη της διαδικασίας για τα στοιχεία που βρίσκονται αριστερά του Pivot και για τα στοιχεία που δεξιά του Pivot.

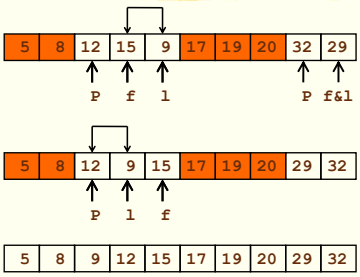
Γρήγορη Ταξινόμηση 1ο Παράδειγμα λειτουργίας 1



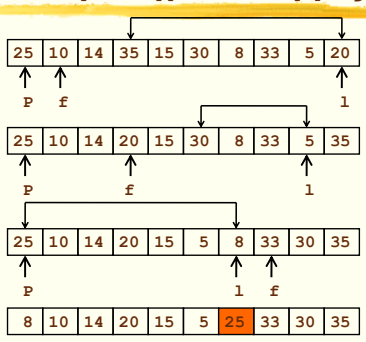
Γρήγορη Ταξινόμηση 1ο Παράδειγμα λειτουργίας 2



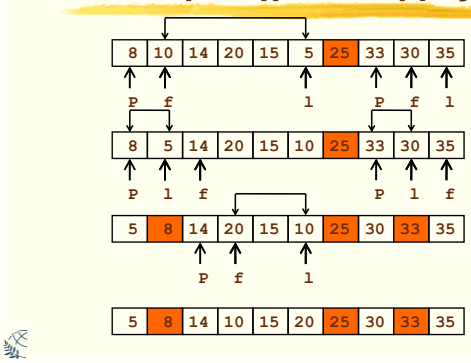
Γρήγορη Ταξινόμηση 1ο Παράδειγμα λειτουργίας 3



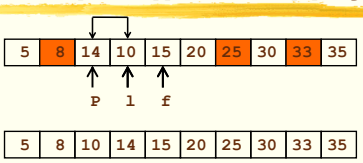
Γρήγορη Ταξινόμηση 2ο Παράδειγμα λειτουργίας 1



Γρήγορη Ταξινόμηση 2ο Παράδειγμα λειτουργίας 2



Γρήγορη Ταξινόμηση 2ο Παράδειγμα λειτουργίας 3



Κώδικας: Quick Sort

```
public static void sort (int[] A)  {
        qSort (A, 0, A.length-1);
}

public static void qSort(int[] A, int f,int l)
{
   if (l-f>0) {
      int pivot_index = partition (A, f, l);
      qSort (A, f, pivot_index-1);
      qSort (A, pivot_index+1, l);
   }
}
```

Κώδικας: Διαίρεση πίνακα – Τοποθέτηση Pivot 1

```
public static int partition (int A[],
                             int f, int l) {
    int retValue=0;
    int lowerLimit = f;
    int mid=(f+l)/2;
    swap(A,f,mid);
    int pivot =A[f];
    f++;

    while (f<l) {
        while (A[f] <= pivot && f< l) f++;
        while (A[l] >= pivot && f<= l) l--;
        if (f<l) swap (A,f,l);
    }
```

Κώδικας: Διαίρεση πίνακα & Τοποθέτηση Pivot 2

```
        if (pivot > A[f]) {
            swap (A,f,lowerLimit);
            retValue=f;
        }
        else {
            if (pivot >= A[l]) {
            swap (A,l,lowerLimit);
            retValue=l;
            }
        }
        return retValue;
    } //  method partition
```

Αλγόριθμοι Ταξινόμησης: Σύγκριση Επίδοσης

- Οι χρόνοι είναι σε ms

Πλήθος Στοιχείων	BUBBLE	SELECTION	INSERTION	MERGE	QUICK
1000	1.187	0.469	0.188	0.125	0.078
2000	4.687	1.687	0.719	0.203	0.172
5000	32.17	9.79	4.05	0.65	0.46
10000	151.65	38.18	16.38	1.37	1.03
20000	620.32	98.08	64.17	2.98	2.05
100000	3003.56	661.22	284.25	5.63	4.20

Ερωτήσεις;