Αλγόριθμοι σε C

Μάθημα 3:

Ταξινόμηση Πίνακα

Δημήτρης Ψούνης

www.psounis.gr

Περιεχόμενα Μαθήματος

Α. Θεωρία

1. Αναζήτηση Στοιχείου σε Πίνακα

1. Ορισμός του Προβλήματος

Η **«Ταξινόμηση Πίνακα»** είναι το υπολογιστικό πρόβλημα στο οποίο:

- **Είσοδος:** Ένας Πίνακας Στοιχείων
- **Έξοδος:** Ο ίδιος Πίνακας με τα στοιχεία του Ταξινομημένα

- Τα στοιχεία ταξινομούνται:
  - Σε αύξουσα σειρά (από το μικρότερο στο μεγαλύτερο) [Σημείωση: Θα μελετήσουμε την ταξινόμηση σε αύξουσα σειρά στη θεωρία]
  - Σε φθίνουσα σειρά (από το μεγαλύτερο στο μικρότερο)
- Θα μελετήσουμε 5 αλγόριθμους:
  - Την Ταξινόμηση με Επιλογή (Selection Sort)
  - Την Ταξινόμηση με Εισαγωγή (Insertion Sort)
  - Την Ταξινόμηση Φυσαλίδας (Bubble Sort)
  - Την Ταξινόμηση με Συγχώνευση (Merge Sort)
  - Την Γρήγορη Ταξινόμηση (Quick Sort)

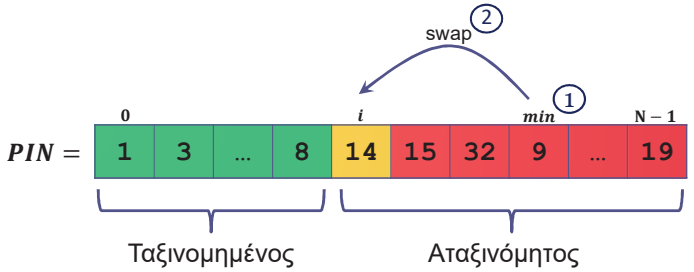
Α. Θεωρία

2. Ταξινόμηση με Επιλογή (Selection Sort)

1. Αλγοριθμική Ιδέα

ΙΔΕΑ της Selection Sort:

- Έχοντας ταξινομήσει τα στοιχεία 0..i-1
- επέλεξε το ελάχιστο ανάμεσα στα i έως N-1 και τοποθέτησε το στην θέση i



Υλοποίηση:

- Στο βήμα i (0…N-1)
  - Βρίσκω το ελάχιστο στις θέσεις i..N-1
  - Το κάνω swap με το στοιχείο στη θέση i

Α. Θεωρία

2. Ταξινόμηση με Επιλογή (Selection Sort)

2. Κώδικας σε C

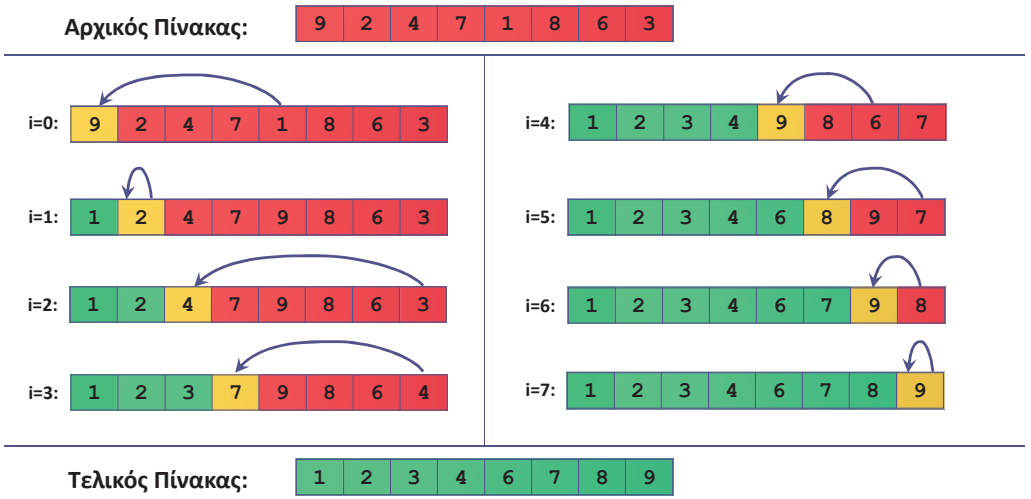
- Ακολουθεί ο κώδικας σε C: Ο pinakas είναι ένας πίνακας N ακεραίων:

```
/* Taksinomisi Selection Sort */

for (i=0; i<N; i++)
{
   pos=i;

   for (j=i+1; j<N; j++)
   {
      if (pinakas[j]<pinakas[pos])
         pos=j;
   }

   swap(&pinakas[i], &pinakas[pos]);
}
```


Α. Θεωρία

2. Ταξινόμηση με Εισαγωγή (Insertion Sort)

3. Ένα Παράδειγμα Εκτέλεσης



Α. Θεωρία

2. Ταξινόμηση με Επιλογή (Selection Sort)

4. Άσκηση

Αφού μελετήσετε το πρόγραμμα selection_sort.c κατασκευάσετε συνάρτηση που παίρνει ως όρισμα έναν πίνακα ακεραίων (και τη διάστασή του) και τον ταξινομεί με χρήση της selection sort.

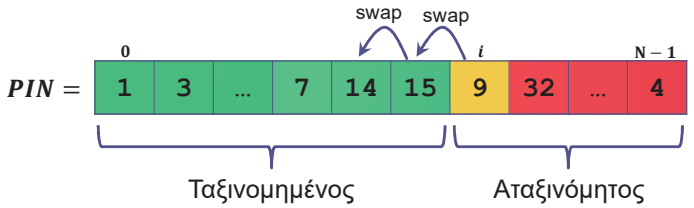
Α. Θεωρία

3. Ταξινόμηση με Εισαγωγή (Insertion Sort)

1. Αλγοριθμική Ιδέα

ΙΔΕΑ της Insertion Sort:

- Έχοντας ταξινομήσει τα στοιχεία 0..i-1
- Τοποθετεί το στοιχείο i ταξινομημένα στις θέσεις 0..i (με ανταλλαγές μέχρι να βρει μικρότερο)



Υλοποίηση:

- Στο βήμα i (1…N-1)
  - Κάνω συνεχή swap του στοιχείου i αριστερά, μέχρι να βρω κάποιο μικρότερο στοιχείο

Α. Θεωρία

3. Ταξινόμηση με Εισαγωγή (Insertion Sort)

2. Κώδικας σε C

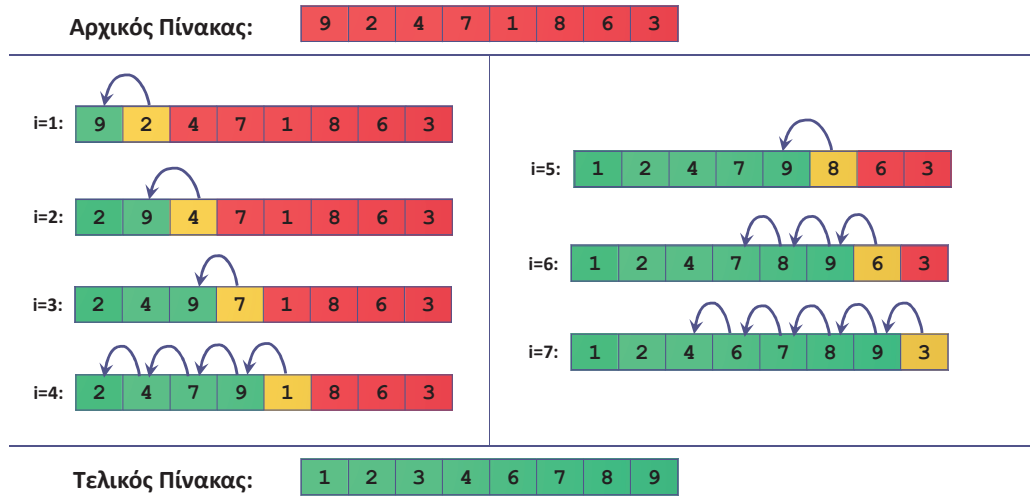
- Ακολουθεί ο κώδικας σε C: Ο pinakas είναι ένας πίνακας N ακεραίων:

```
/* Taksinomisi me Eisagogi */
for (i=1; i<N; i++)
{
   for (j=i; j>=1; j--)
   {
      if (pinakas[j]<pinakas[j-1])
         swap(&pinakas[j], &pinakas[j-1]);
      else
         break;
   }
}
```


Α. Θεωρία

3. Ταξινόμηση με Εισαγωγή (Insertion Sort)

3. Ένα Παράδειγμα Εκτέλεσης



Α. Θεωρία

3. Ταξινόμηση με Εισαγωγή (Insertion Sort)

4. Άσκηση

Αφού μελετήσετε το πρόγραμμα insertion_sort.c κατασκευάσετε συνάρτηση που παίρνει ως όρισμα έναν πίνακα ακεραίων (και τη διάστασή του) και τον ταξινομεί με χρήση της insertion sort.

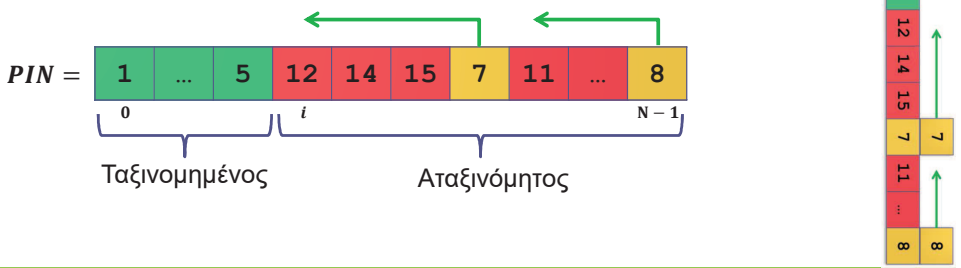
Α. Θεωρία

4. Ταξινόμηση Φυσαλίδας (Bubble Sort)

1. Αλγοριθμική Ιδέα

ΙΔΕΑ της Bubble Sort:

- Έχοντας ταξινομήσει τα στοιχεία 0..i-1
- Τα ελαφρύτερα (μικρότερα) στοιχεία αναδύονται με μία σάρωση στις θέσεις N-1…i



Υλοποίηση:

- Στο βήμα i (0…N-1)
  - Για j=N-1 έως i+1: Κάνω swap του PIN[j] με το PIN[j-1] εάν PIN[j]<PIN[j-1]

Α. Θεωρία

4. Ταξινόμηση Φυσαλίδας(Bubble Sort)

2. Κώδικας σε C

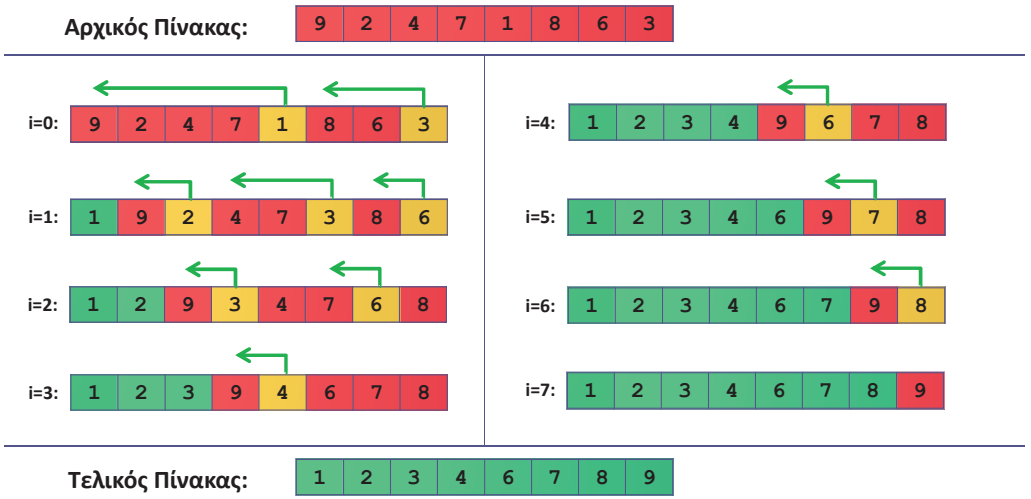
- Ακολουθεί ο κώδικας σε C: Ο pinakas είναι ένας πίνακας N ακεραίων:

```
/* Taksinomisi Fysalidas */

for (i=0; i<N; i++)
{
   for (j=N-1; j>=i+1; j--)
   {
      if (pinakas[j]<pinakas[j-1])
         swap(&pinakas[j], &pinakas[j-1]);
   }
}
```


Α. Θεωρία

4. Ταξινόμηση Φυσαλίδας(Bubble Sort)

3. Ένα Παράδειγμα Εκτέλεσης



Α. Θεωρία

4. Ταξινόμηση Φυσαλίδας(Bubble Sort)

4. Άσκηση

Αφού μελετήσετε το πρόγραμμα bubble_sort.c κατασκευάσετε συνάρτηση που παίρνει ως όρισμα έναν πίνακα ακεραίων (και τη διάστασή του) και τον ταξινομεί με χρήση της bubble sort.

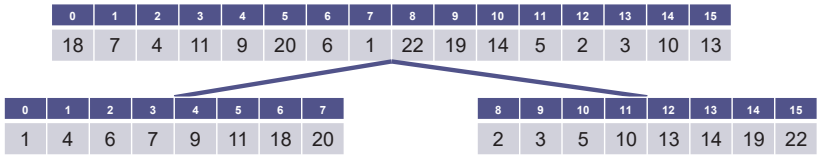
Α. Θεωρία

5. Ταξινόμηση με Συγχώνευση (Merge Sort)

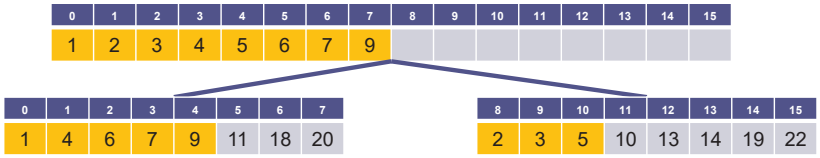
1. Αλγοριθμική Ιδέα

ΙΔΕΑ της Merge Sort:

1. (Αναδρομικά) **Ταξινόμησε** το αριστερό μισό του πίνακα
2. (Αναδρομικά) **Ταξινόμησε** το δεξί μισό του πίνακα



3. **Συγχώνευσε** τα δύο κομμάτια σε μία ταξινομημένη ακολουθία



Α. Θεωρία

5. Ταξινόμηση με Συγχώνευση (Merge Sort)

1. Αλγοριθμική Ιδέα

- Το σώμα του αλγορίθμου φαίνεται στον ακόλουθο κώδικα (ο πίνακας έχει δηλωθεί ως int pinakas[N] και οι μεταβλητές start,finish, middle είναι ακέραιες μεταβλητές):

```
void mergesort(int *pinakas, int start, int finish)
{
   int i,middle;

   if (start==finish) /* 1 stoixeio */
      return;
   else if (start==finish-1) /* 2 stoixeia */
   {
      if (pinakas[start]>pinakas[finish])
         swap(&pinakas[start], &pinakas[finish]);
   }
   else
   {
      middle=(start+finish)/2;
      mergesort(pinakas,start,middle);
      mergesort(pinakas,middle+1,finish);
      merge(pinakas,start,finish);
   }
}
```


Α. Θεωρία

5. Ταξινόμηση με Συγχώνευση (Merge Sort)

3. Ένα Παράδειγμα Εκτέλεσης

- Πίνακας προς ταξινόμηση:

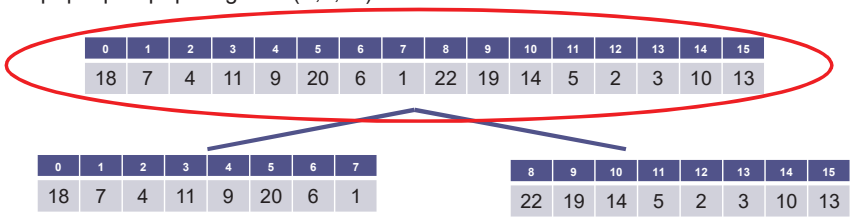
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
18	7	4	11	9	20	6	1	22	19	14	5	2	9	10	13

Α. Θεωρία

5. Ταξινόμηση με Συγχώνευση (Merge Sort)

3. Ένα Παράδειγμα Εκτέλεσης

- Αναδρομική Κλήση MergeSort(A,0,15)

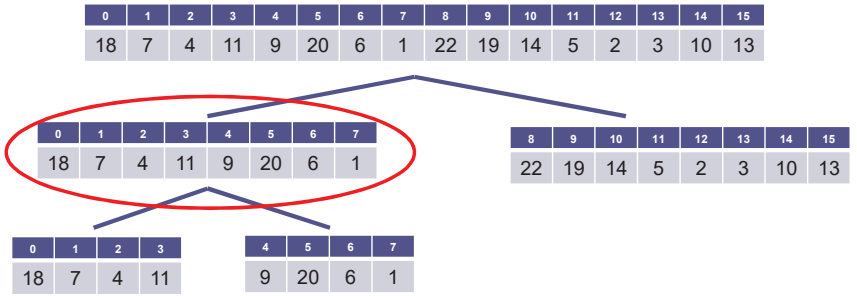


Α. Θεωρία

5. Ταξινόμηση με Συγχώνευση (Merge Sort)

3. Ένα Παράδειγμα Εκτέλεσης

- Αναδρομική Κλήση (A,0,7)

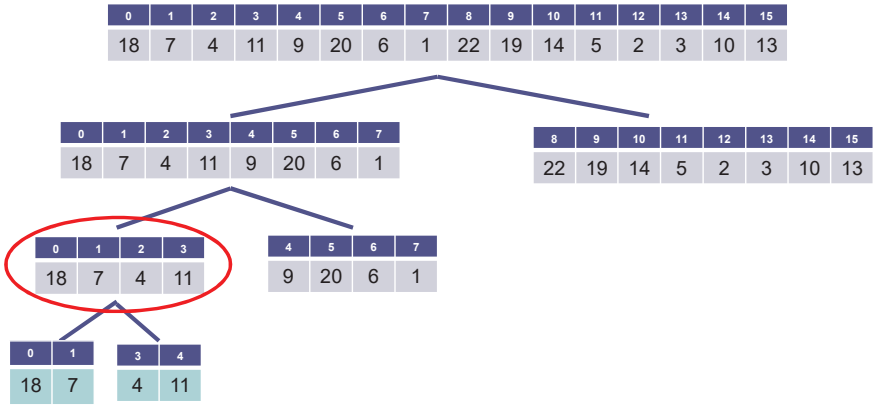


Α. Θεωρία

5. Ταξινόμηση με Συγχώνευση (Merge Sort)

3. Ένα Παράδειγμα Εκτέλεσης

- Αναδρομική Κλήση (A,0,3)

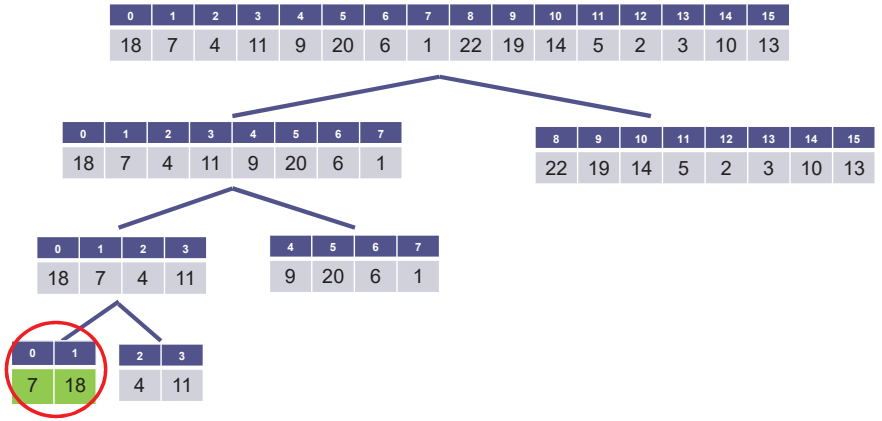


Α. Θεωρία

5. Ταξινόμηση με Συγχώνευση (Merge Sort)

3. Ένα Παράδειγμα Εκτέλεσης

- Αναδρομική Κλήση (A,0,1): Ταξινομηση του υποπίνακα

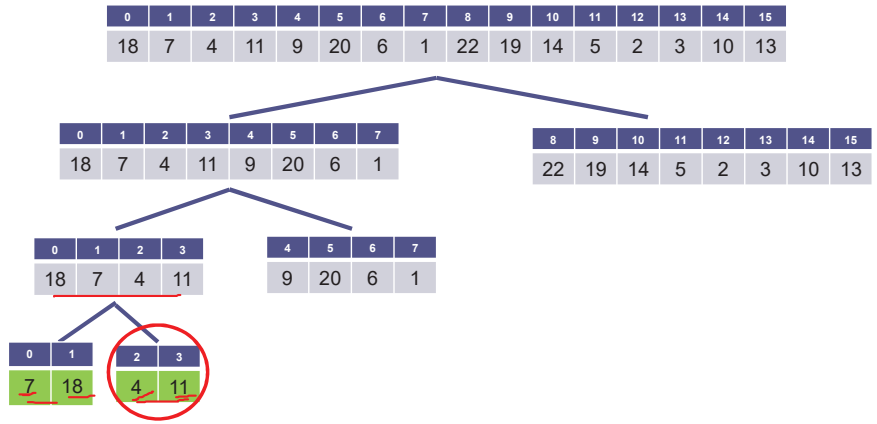


Α. Θεωρία

5. Ταξινόμηση με Συγχώνευση (Merge Sort)

3. Ένα Παράδειγμα Εκτέλεσης

- Αναδρομική Κλήση (A,2,3): Ταξινομηση του υποπίνακα

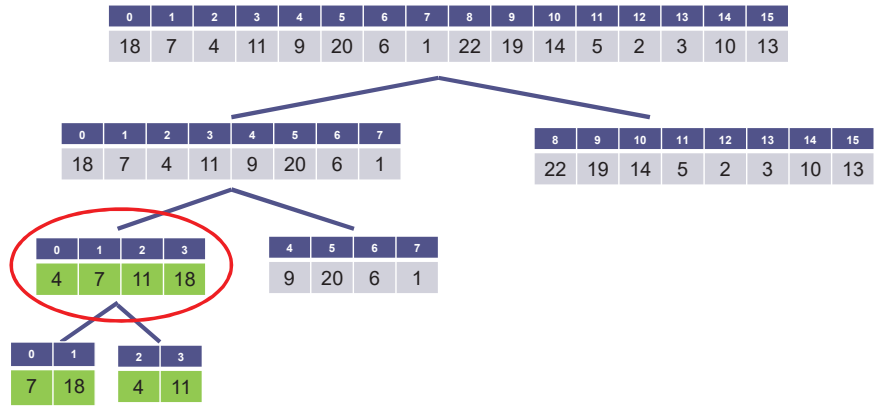


Α. Θεωρία

5. Ταξινόμηση με Συγχώνευση (Merge Sort)

3. Ένα Παράδειγμα Εκτέλεσης

- Αναδρομική Κλήση (A,0,3): Συγχώνευση των δύο υποπινάκων

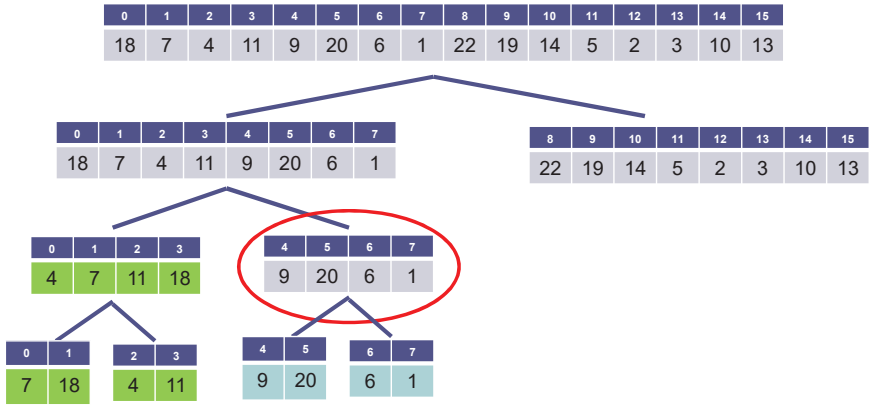


Α. Θεωρία

5. Ταξινόμηση με Συγχώνευση (Merge Sort)

3. Ένα Παράδειγμα Εκτέλεσης

- Αναδρομική Κλήση (A,4,7)

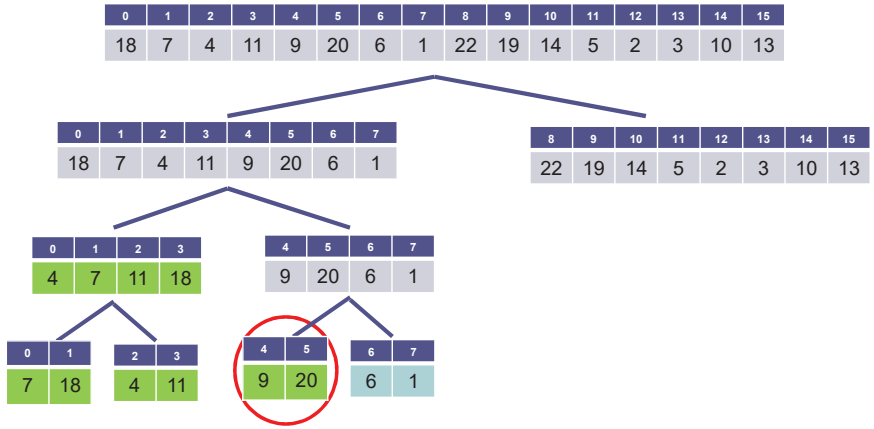


Α. Θεωρία

5. Ταξινόμηση με Συγχώνευση (Merge Sort)

3. Ένα Παράδειγμα Εκτέλεσης

- Αναδρομική Κλήση (A,4,5): Ταξινομηση του υποπίνακα

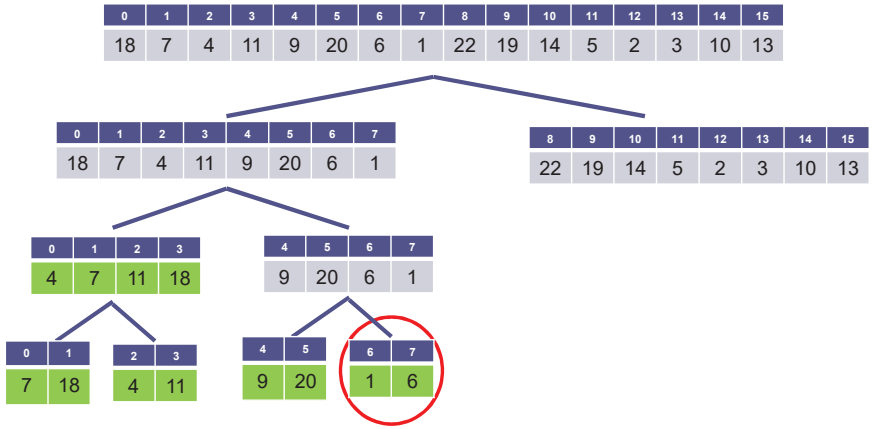


Α. Θεωρία

5. Ταξινόμηση με Συγχώνευση (Merge Sort)

3. Ένα Παράδειγμα Εκτέλεσης

- Αναδρομική Κλήση (A,6,7): Ταξινομηση του υποπίνακα

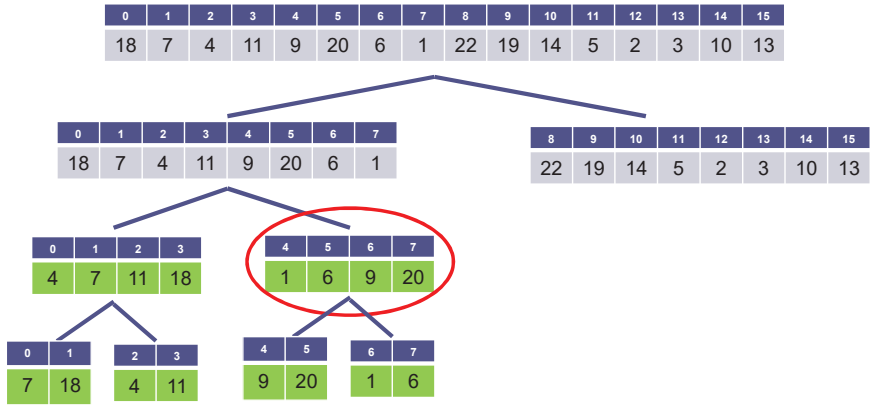


Α. Θεωρία

5. Ταξινόμηση με Συγχώνευση (Merge Sort)

3. Ένα Παράδειγμα Εκτέλεσης

- Αναδρομική Κλήση (A,4,7): Συγχώνευση των δύο υποπινάκων

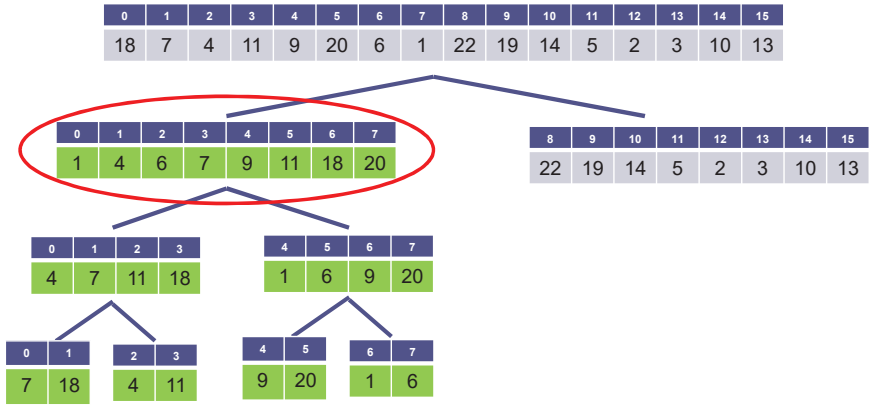


Α. Θεωρία

5. Ταξινόμηση με Συγχώνευση (Merge Sort)

3. Ένα Παράδειγμα Εκτέλεσης

- Αναδρομική Κλήση (A,0,7): Συγχώνευση των δύο υποπινάκων

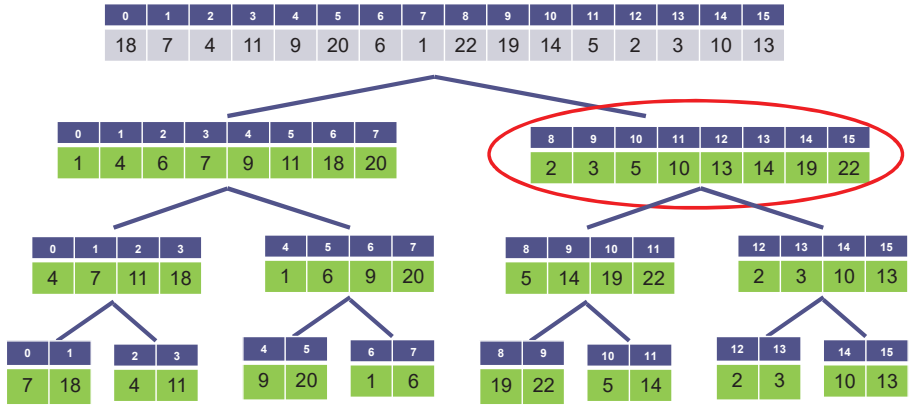


Α. Θεωρία

5. Ταξινόμηση με Συγχώνευση (Merge Sort)

3. Ένα Παράδειγμα Εκτέλεσης

- Αντίστοιχα θα γίνουν όλες οι αναδρομικές κλήσεις στο (8,15)

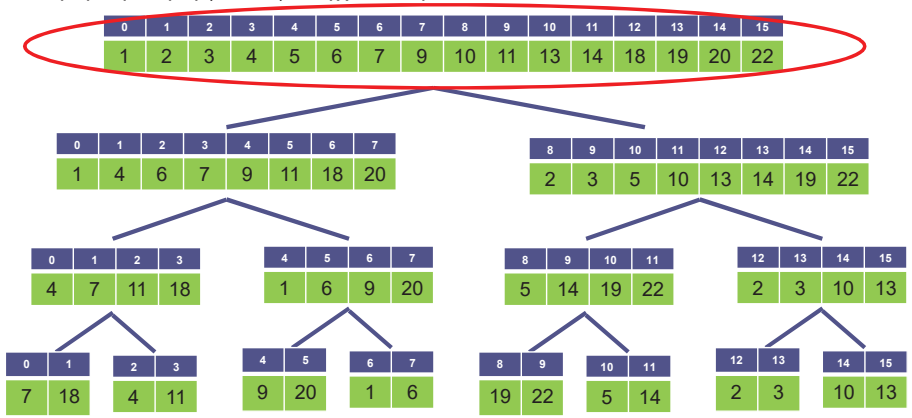


Α. Θεωρία

5. Ταξινόμηση με Συγχώνευση (Merge Sort)

3. Ένα Παράδειγμα Εκτέλεσης

- Αναδρομική Κλήση (A,0,15): Συγχώνευση των δύο υποπινάκων



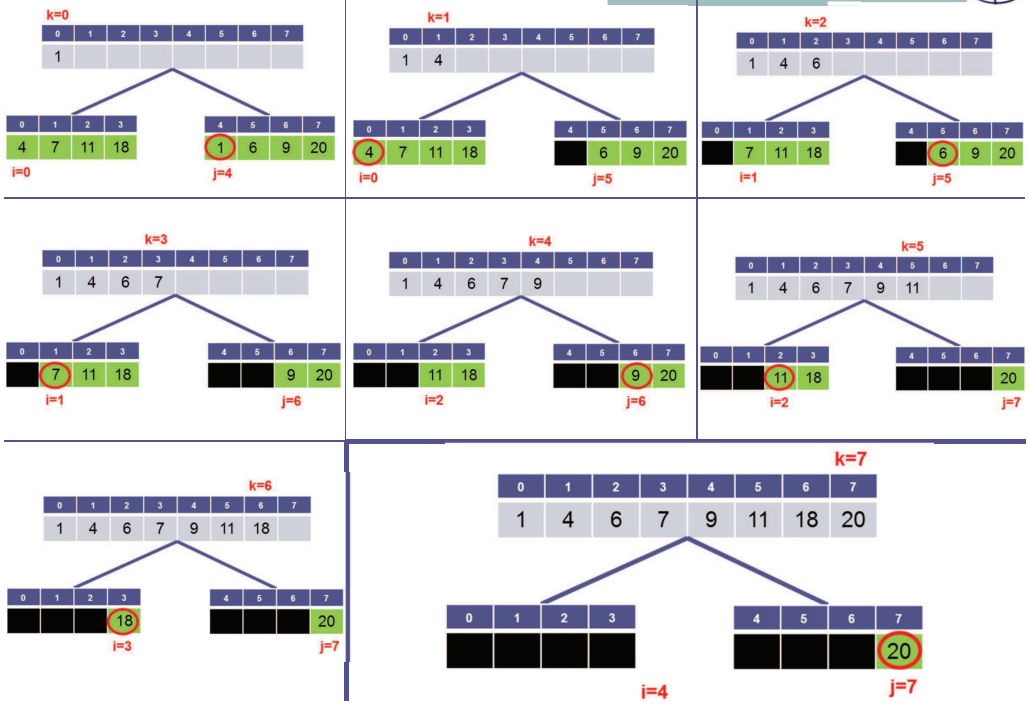
Α. Θεωρία

5. Ταξινόμηση με Συγχώνευση (Merge Sort)

4. Η διαδικασία merge

- Η διαδικασία merge δουλεύει ως εξής:
  - Σαρώνει τους δύο πίνακες ταυτόχρονα από αριστερά προς τα δεξιά.
  - Συγκρίνει τα δύο τρέχοντα στοιχεία των πινάκων
  - Επιλέγει το μικρότερο και το βάζει στην επόμενη θέση της ταξινομημένης ακολουθίας
- Όταν εξαντληθούν τα στοιχεία του ένος από τους δύο πίνακες, βάζουμε όσα στοιχεία απέμειναν από τον άλλο πίνακα στο τέλος της ταξινομημένης ακολουθίας.

- Στο παράδειγμα βλέπουμε μερικά βήματα και τους μετρητές που χρησιμοποιούνται.



Α. Θεωρία

5. Ταξινόμηση με Συγχώνευση (Merge Sort)

4. Η διαδικασία merge

- Μία υλοποίηση της merge είναι η ακόλουθη:

```
void merge(int *pinakas, int start, int finish)
{
   int C[SIZE];
   int i,j,k;
   int middle, n, m;

   middle=(start+finish)/2;

   /* 1os pinakas PIN[start..middle]=PIN[i..n] */
   i=start;
   n=middle;

   /* 2os pinakas PIN[middle+1..finish]=PIN[j...m] */
   j=middle+1;
   m=finish;

   /* C: sigxwneumenos pinakas */
   k=0;
```

(συνεχίζεται…)

Α. Θεωρία

5. Ταξινόμηση με Συγχώνευση (Merge Sort)

4. Η διαδικασία merge

- Μία υλοποίηση της merge είναι η ακόλουθη:

```
(συνέχεια…)

   /* 1. Sigxwneusi twn dio pinakwn */
   while (i<=n && j<=m)
   {
      if (pinakas[i]<pinakas[j])
      {
         C[k]=pinakas[i];
         k++;
         i++;
      }
      else
      {
         C[k]=pinakas[j];
         k++;
         j++;
      }
   }
```

(συνεχίζεται…)

Α. Θεωρία

5. Ταξινόμηση με Συγχώνευση (Merge Sort)

4. Η διαδικασία merge

- Μία υλοποίηση της merge είναι η ακόλουθη:

```
(συνέχεια…)

   /* 2. Antigrafi tou pinaka pou perissevei sto telos tou sigxwneumenou pinaka */

   if (i==n+1) /* Eksantlithike o 1os pinakas */
   {
      while (j<=m)
      {
         C[k]=pinakas[j];
         k++;
         j++;
      }
   }
   else /* Eksantlithike o 2os pinakas */
   {
      while (i<=n)
      {
         C[k]=pinakas[i];
         k++;
         i++;
      }
   }
```

(συνεχίζεται…)

A. Θεωρία

5. Ταξινόμηση με Συγχώνευση (Merge Sort)

4. Η διαδικασία merge

- Μία υλοποίηση της merge είναι η ακόλουθη:

```
(συνέχεια…)

   /* 3. Antigrafi tou C ston pinakas */

   k=0;

   i=start;
   while (i<=finish)
   {
      pinakas[i]=C[k];
      i++;
      k++;
   }
}
```

(συνεχίζεται…)

A. Θεωρία

5. Ταξινόμηση με Συγχώνευση (Merge Sort)

5. Άσκηση

- **Άσκηση:** Μελετήστε το πρόγραμμα mergesort.c και “καθαρίστε” το από τις εκτυπώσεις ώστε η mergesort να είναι μια αυτόνομη συνάρτηση.

A. Θεωρία

6. Γρήγορη Ταξινόμηση (Quick Sort)

1. Ιδέα του Αλγορίθμου

ΙΔΕΑ της QuickSort:

1. Επιλέγει ένα Στοιχείο του Πίνακα (Οδηγό Στοιχείο – Εδώ το στοιχείο που είναι στην πρώτη θέση)
2. Χωρίζει τον πίνακα σε τρία μέρη:
   - Τα στοιχεία που είναι μικρότερα του οδηγού στοιχείου
   - Το οδηγό στοιχείο
   - Τα στοιχεία που είναι μεγαλύτερα ή ίσα του οδηγού στοιχείου

0	1	2	3	4	5	6	7	8	9
18	7	4	11	9	20	6	1	22	19
6	7	4	11	9	1	18	20	22	19

3. Επαναλαμβάνει αναδρομικά στους δύο υποπίνακες που προέκυψαν.

0	1	2	3	4	5
6	7	4	11	9	1

6
18

7	8	9
20	22	19

A. Θεωρία

6. Γρήγορη Ταξινόμηση (Quick Sort)

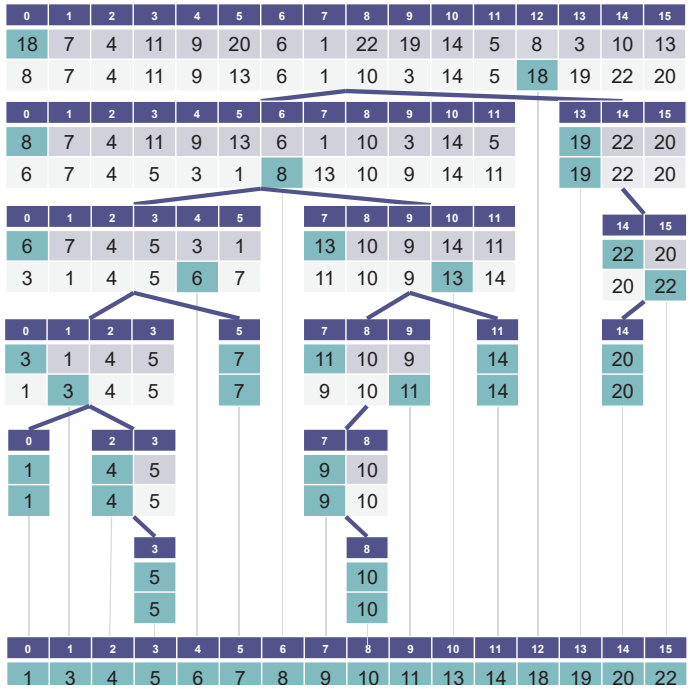
2. Κώδικας σε C

- Η υλοποίηση σε γλώσσα C είναι η ακόλουθη:

```
void quicksort(int *pinakas, int start, int finish)
{
   int pos;

   if (start<finish)
   {
      pos=partition(pinakas,start,finish);
      quicksort(pinakas,start,pos-1);
      quicksort(pinakas,pos+1,finish);
   }
}
```

- Όπου partition είναι μια συνάρτηση που πραγματοποιεί τον χωρισμό του πίνακα σε μικρότερα και μεγαλύτερα (ή ίσα) στοιχεία σε σχέση με το οδηγό στοιχείο.
  - Επιστρέφει τη θέση του τελευταίου μικρότερου στοιχείου από το οδηγό στοιχείο.
  - Συνεπώς οι δύο πίνακες στους οποίους γίνεται η αναδρομή είναι:
    - start..pos
    - pos+1..finish
- Στην επόμενη διαφάνεια βλέπουμε ένα παράδειγμα εκτέλεσης.

Γρήγορη Ταξινόμηση QuickSort (3.Παράδειγμα Εκτέλεσης)



Α. Θεωρία

6. Γρήγορη Ταξινόμηση (Quick Sort)

4. Η διαδικασία partition

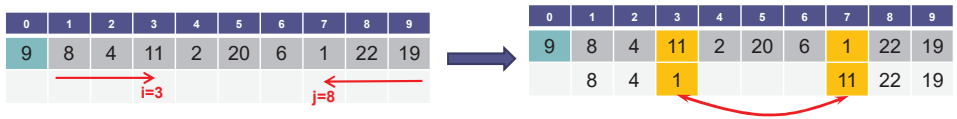
- Η διαμέριση των στοιχείων στα μικρότερα και μεγαλύτερα ή ίσα μπορεί να γίνει με πολλούς τρόπους:
  - Π.χ. ένας απλός τρόπος είναι να χρησιμοποιήσουμε έναν βοηθητικό πίνακα που να βάζουμε στα αριστερά τα μικρότερα στοιχεία και στα δεξιά τα μεγαλύτερα στοιχεία του οδηγού.
- Ωστόσο εμείς θα μελετήσουμε έναν τρόπο διαμέρισης, που είναι μια μικρή παραλλαγή του σχήματος Hoare:
  - Σαρώνουμε τον πίνακα από αριστερά ψάχνοντας για ένα στοιχείο που είναι μεγαλύτερο του οδηγού
  - Σαρώνουμε τον πίνακα από δεξιά ψάχνοντας για ένα στοιχείο που είναι μικρότερο (ή ίσο) του οδηγού
  - Ανταλλάσσουμε τα δύο στοιχεία και επαναλαμβάνουμε μέχρι να γίνει ο χωρισμός των στοιχείων.
- Μελετούμε το παράδειγμα της επόμενης διαφάνειας

Α. Θεωρία

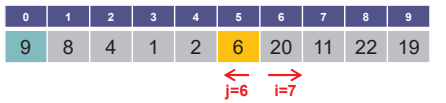
6. Γρήγορη Ταξινόμηση (Quick Sort)

4. Η διαδικασία partition

- Παράδειγμα εκτέλεσης της partition:



- Τερματισμός:



- Ανταλλαγή του οδηγού στοιχείου με το «δεξιό» δείκτη

Α. Θεωρία

6. Γρήγορη Ταξινόμηση (Quick Sort)

4. Η διαδικασία partition

- Η υλοποίηση της partition με το σχήμα του Hoare σε ψευδογλώσσα είναι η ακόλουθη:

```
int partition(int *pinakas, int start, int finish)
{
   int pivot, i, j;

   pivot=pinakas[start];

   i=start+1;
   j=finish;
   while(1)
   {
      while(pinakas[i]<=pivot && i<=finish)
         i++;

      while(pinakas[j]>pivot && j>=start)
         j--;

      if (i<j)
         swap(&pinakas[i],&pinakas[j]);
      else
      {
         swap(&pinakas[start],&pinakas[j]);
         return j;
      }
   }
}
```

Α. Θεωρία

6. Γρήγορη Ταξινόμηση (Quick Sort)

5. Άσκηση

- **Άσκηση:** Μελετήστε το πρόγραμμα quicksort.c και "καθαρίστε" το από τις εκτυπώσεις ώστε η quicksort να είναι μια αυτόνομη συνάρτηση.

Β. Ασκήσεις

Εφαρμογή 1: Ταξινόμηση Πραγματικών Αριθμών

- Μετατρέψτε την bubble sort και την quick sort έτσι ώστε να πραγματοποιούν ταξινόμηση σε τύπο δεδομένων double.

Β. Ασκήσεις

Εφαρμογή 2: Ταξινόμηση σε Φθίνουσα Σειρά

- Μετατρέψτε την insertion sort και την merge sort έτσι ώστε να πραγματοποιούν ταξινόμηση σε φθίνουσα σειρά.

Β. Ασκήσεις

Εφαρμογή 3: Quick Sort που χρησιμοποιεί Insertion Sort

- Προτείνεται η εξής υβριδική μέθοδος ταξινόμησης:
  - Εκτελούμε τον αλγόριθμο Quick Sort
  - Αν ωστόσο ο πίνακας έχει το πολύ 5 στοιχεία, τότε αυτός ταξινομείται με χρήση της insertion sort
- Κατασκευάστε την νέα μέθοδο ταξινόμησης με όνομα hybrid sort και αναδείξτε τη χρήση της με μία κατάλληλη συνάρτηση main.

Β. Ασκήσεις
Εφαρμογή 4.1: Εκτέλεση όλων των αλγορίθμων

- Δημιουργήστε ένα πρόγραμμα το οποίο:
  - Δημιουργείται ένας πίνακας 100 ακεραίων αριθμών. Οι τιμές του πίνακα να είναι ακέραιοι στο εύρος [0…1000]
  - Καλούνται διαδοχικά οι 5 αλγόριθμοι ταξινόμησης που μελετήσαμε και προβάλλεται το αποτέλεσμα της ταξινόμησης του πίνακα.

Β. Ασκήσεις
Εφαρμογή 4.2: Πλήθος Συγκρίσεων των Αλγορίθμων Αναζήτησης

- Τροποποιήστε τους 3 μη αναδρομικούς αλγορίθμους, ώστε να μετράνε το πλήθος των συγκρίσεων που πραγματοποιούνται.
- Μετά την εκτέλεση του αντίστοιχου αλγορίθμου να τυπώνεται το πλήθος των συγκρίσεων που πραγματοποιήθηκαν.
  - Σημείωση: Όταν λέμε συγκρίσεις εννοούμε τις συγκρίσεις που γίνονται μεταξύ στοιχείων του πίνακα.

Β. Ασκήσεις
Εφαρμογή 4.3: Πλήθος Συγκρίσεων των Αλγορίθμων Αναζήτησης

- Τροποποιήστε τους 2 αναδρομικούς αλγορίθμους, ώστε να μετράνε το πλήθος των συγκρίσεων που πραγματοποιούνται.
- Μετά την εκτέλεση του αντίστοιχου αλγορίθμου να τυπώνεται το πλήθος των συγκρίσεων που πραγματοποιήθηκαν.
  - Σημείωση: Όταν λέμε συγκρίσεις εννοούμε τις συγκρίσεις που γίνονται μεταξύ στοιχείων του πίνακα.

Β. Ασκήσεις
Εφαρμογή 4.4: Πειραματική Μελέτη των Αλγορίθμων Αναζήτησης

- Πειραματιστείτε δοκιμάζοντας διαφορετικές διαστάσεις στον πίνακα.
- Ποιος αλγόριθμος δείχνει να επικρατεί των άλλων;