ΠΛΗ10

ΕΝΟΤΗΤΑ 2: Αλγόριθμοι και Ψευδογλώσσα

Μάθημα 2.7:
Αναδρομή

Δημήτρης Ψούνης

www.psounis.gr

Περιεχόμενα Μαθήματος

Α. Αναδρομή

1. Αναδρομικές Συναρτήσεις

- Ο όρος **<u>αναδρομή</u> αναφέρεται σε μια συνάρτηση που καλεί τον εαυτό της**!
- Άρα μια συνάρτηση (ή διαδικασία) που στο σώμα της καλεί τον εαυτό της, θα ονομάζεται **<u>αναδρομική συνάρτηση</u>** (αντίστοιχα **αναδρομική διαδικασία).**
- Η δημιουργία μιας αναδρομικής συνάρτησης είναι πολύ χρήσιμη, ιδίως όταν κατασκευάζουμε πράγματα που ορίζονται αναδρομικά!
- Ας δούμε ένα παράδειγμα:
  - Το παραγοντικό του φυσικού αριθμού n ορίζεται ως:
    - $n! = n \cdot (n-1) \cdot (n-2) \cdot \cdots 2 \cdot 1$
    - π.χ. έχουμε $1! = 1,\ \ 2! = 2 \cdot 1,\ \ 3! = 3 \cdot 2 \cdot 1 = 6,\ \ 4! = 4 \cdot 3 \cdot 2 \cdot 1 = 24$ κ.ο.κ.
  - Το παραγοντικό ορίζεται ωστόσο και αναδρομικά ως εξής:
    - $n! = n \cdot (n-1)!$ αν $n > 1$
    - $n! = 1,$ αν $n = 1$

Α. Αναδρομή

1. Αναδρομικές Συναρτήσεις

1. Υπολογισμός Παραγοντικού

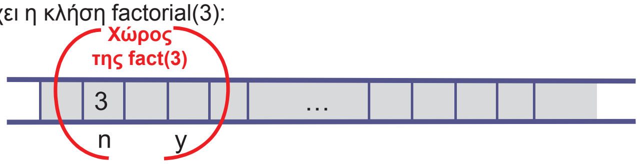
```
ΑΛΓΟΡΙΘΜΟΣ factorial
ΔΕΔΟΜΕΝΑ
   n,res: INTEGER;

ΣΥΝΑΡΤΗΣΗ factorial(n): INTEGER
ΔΙΕΠΑΦΗ
   ΕΙΣΟΔΟΣ
      n: INTEGER;
   ΕΞΟΔΟΣ
      factorial: INTEGER;
ΔΕΔΟΜΕΝΑ
         y:INTEGER;
ΑΡΧΗ
   ΕΑΝ (n=1) ΤΟΤΕ
      factorial:=1;
   ΑΛΛΙΩΣ
      y:=factorial(n-1);
      factorial:=n*y;
   ΕΑΝ-ΤΕΛΟΣ
ΤΕΛΟΣ-ΣΥΝΑΡΤΗΣΗΣ
```

```
ΑΡΧΗ

  ΤΥΠΩΣΕ(EOLN, "Δώσε το n: ");
  ΔΙΑΒΑΣΕ(n);

  res:=factorial(n);

  ΤΥΠΩΣΕ(n,"!=",res);

ΤΕΛΟΣ
```

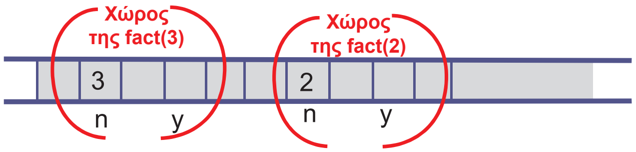
```
ΣΥΝΑΡΤΗΣΗ factorial(n): INTEGER
ΑΡΧΗ
   ΕΑΝ (n=1) ΤΟΤΕ
      factorial:=1;
   ΑΛΛΙΩΣ
      y:=factorial(n-1);
      factorial:=n*y;
   ΕΑΝ-ΤΕΛΟΣ
ΤΕΛΟΣ-ΣΥΝΑΡΤΗΣΗΣ
```

Α. Αναδρομή
1. Αναδρομικές Συναρτήσεις
1. Υπολογισμός Παραγοντικού

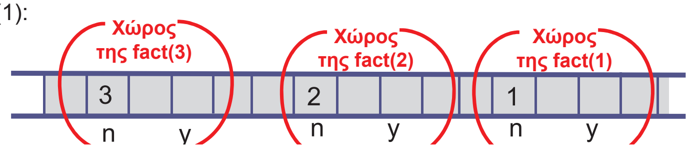
- Στον υπολογισμό μίας αναδρομικής συνάρτησης, κάθε αναδρομική κλήση έχει και το δικό της χώρο στη μνήμη.
- Ας δούμε πως τρέχει η κλήση factorial(3):



- Καλεί την factorial(2):



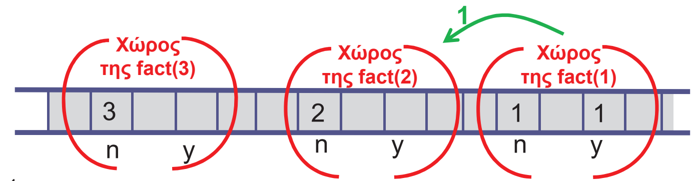
- Καλεί την factorial(1):



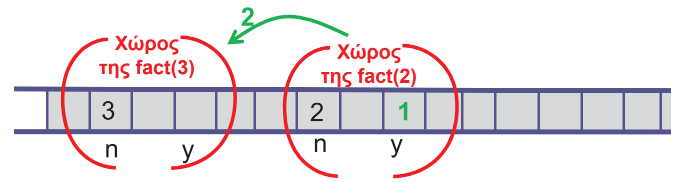

```
ΣΥΝΑΡΤΗΣΗ factorial(n): INTEGER
ΑΡΧΗ
   ΕΑΝ (n=1) ΤΟΤΕ
      factorial:=1;
   ΑΛΛΙΩΣ
      y:=factorial(n-1);
      factorial:=n*y;
   ΕΑΝ-ΤΕΛΟΣ
ΤΕΛΟΣ-ΣΥΝΑΡΤΗΣΗΣ
```


Α. Αναδρομή
1. Αναδρομικές Συναρτήσεις
1. Υπολογισμός Παραγοντικού

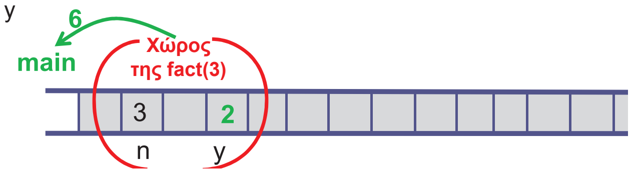
- Η factorial(1) επιστρέφει 1:



- Η factorial(2) αποθηκεύει το 1 στο y
  - έπειτα επιστρέφει 2*1.



- Η factorial(2) αποθηκεύει το 1 στο y
  - έπειτα επιστρέφει 3*2




```
ΣΥΝΑΡΤΗΣΗ factorial(n): INTEGER
ΑΡΧΗ
   ΕΑΝ (n=1) ΤΟΤΕ
      factorial:=1;
   ΑΛΛΙΩΣ
      y:=factorial(n-1);
      factorial:=n*y;
   ΕΑΝ-ΤΕΛΟΣ
ΤΕΛΟΣ-ΣΥΝΑΡΤΗΣΗΣ
```


Α. Αναδρομή
1. Αναδρομικές Συναρτήσεις
1. Υπολογισμός Παραγοντικού

- Για να απεικονίσουμε τις αναδρομικές κλήσεις που γίνονται προτιμάται μία παράσταση όπου κάθε αναδρομική κλήση στοιχίζεται λίγο δεξιότερα.
- Με τον τρόπο αυτό μπορούμε να παρακολουθήσουμε αρκετά ικανοποιητικά την εκτέλεση ενός αναδρομικού κώδικα:

```
ΚΛΗΣΗ factorial(3)
(3=1) ΟΧΙ
y:=factorial(2)
    ΚΛΗΣΗ factorial(2)
    (2=1) ΟΧΙ
    y:=factorial(1)
        ΚΛΗΣΗ factorial(1)
        (1=1) ΝΑΙ
        Επιστρέφει 1
    y=1
    Επιστρέφει 2*1=2
y=2
Επιστρέφει 3*2=6
```


Α. Αναδρομή
1. Αναδρομικές Συναρτήσεις
2. Δυαδική Αναζήτηση (Binary Search)

- Κλασικό παράδειγμα αλγορίθμου που μπορεί να υλοποιηθεί αναδρομικά είναι η **δυαδική αναζήτηση** (Binary Search).

- Ο αλγόριθμος που είδαμε να υλοποιείται με επανάληψη στους πίνακες, τώρα μπορεί να αναδιατυπωθεί με αναδρομή.

- Σκεπτικό δυαδικής αναζήτησης με αναδρομή του στοιχείου x σε έναν ταξινομημένο σε αύξουσα σειρά πίνακα:

  - Αν το μεσαίο στοιχείο είναι το x, το στοιχείο βρέθηκε!
  - Αν το x είναι μικρότερο από το μεσαίο στοιχείο τότε αναδρομικά ψάχνει στο κομμάτι του πίνακα από την αρχή μέχρι το μεσαίο στοιχείο
  - Αν το x είναι μεγαλύτερο από το μεσαίο στοιχείο τότε αναδρομικά ψάχνει στο κομμάτι του πίνακα από το μεσαίο στοιχείο μέχρι το τέλος

Α. Αναδρομή

1. Αναδρομικές Συναρτήσεις

2. Δυαδική Αναζήτηση (Binary Search)

- Μία υλοποίηση σε ψευδογλώσσα του αλγορίθμου (με αναδρομή) είναι η ακόλουθη:

```
ΣΥΝΑΡΤΗΣΗ binary_search(PIN,x,start,finish): BOOLEAN
ΔΙΕΠΑΦΗ
 -
ΔΕΔΟΜΕΝΑ
  check:BOOLEAN;
  middle:INTEGER;
ΑΡΧΗ
  ΕΑΝ (start>finish) ΤΟΤΕ
    check:=FALSE;
  ΑΛΛΙΩΣ
    middle:=(start+finish) DIV 2;
    ΕΑΝ (x=PIN[middle]) ΤΟΤΕ
      check:=TRUE;
    ΑΛΛΙΩΣ
      ΕΑΝ (x>PIN[middle]) ΤΟΤΕ
        check:=binary_search(PIN,x,middle+1,finish);
      ΑΛΛΙΩΣ
        check:=binary_search(PIN,x,start,middle-1);
      ΕΑΝ-ΤΕΛΟΣ
    ΕΑΝ-ΤΕΛΟΣ
  ΕΑΝ-ΤΕΛΟΣ
  binary_search:=check;
ΤΕΛΟΣ-ΣΥΝΑΡΤΗΣΗΣ
```


Α. Αναδρομή

1. Αναδρομικές Συναρτήσεις

2. Δυαδική Αναζήτηση (Binary Search)

- Εκτελούμε τον αλγόριθμο ψάχνοντας το στοιχείο 11 στον πίνακα:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2	3	5	7	11	13	17	21	23	27	31	33	37	41	43

- Κλήση: BinarySearch(A,11,1,15): middle=(1+15) div 2=8. x<A[middle]

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2	3	5	7	11	13	17	21	23	27	31	33	37	41	43
start														finish

  - Αναδρομική Κλήση: BinarySearch(A,11,1,7) : middle=(1+7) div 2=4 x>A[middle]

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2	3	5	7	11	13	17	21	23	27	31	33	37	41	43
start						finish								

  - Αναδρομική Κλήση: BinarySearch(A,11,5,7) : middle=(5+7) div 2=6 x<A[middle]

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2	3	5	7	11	13	17	21	23	27	31	33	37	41	43
				start		finish								

  - Αναδρομική Κλήση: BinarySearch(A,11,5,5) : middle=(5+5) div 2=5 x=A[middle]

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2	3	5	7	11	13	17	21	23	27	31	33	37	41	43
				start=finish										

Α. Αναδρομή

1. Αναδρομικές Συναρτήσεις

2. Δυαδική Αναζήτηση (Binary Search)

- **Άσκηση:** Κατασκευάστε ένα πρόγραμμα που να αναδεικνύει την χρήση της δυαδικής αναζήτησης και παρουσιάστε ένα παράδειγμα εκτέλεσης για τον πίνακα A=[1 3 5 7 9] αναζητώντας το στοιχείο 7

- Σημείωση: Στο σημείο αυτό συνίσταται να μελετήσετε πρώτα τις δύο εφαρμογές (αριθμοί Fibonacci και αλγόριθμος Ευκλείδη) και έπειτα να προχωρήσετε με την θεωρία του μαθήματος.

Α. Αναδρομή

2. Αναδρομικές Διαδικασίες

1. Ταξινόμηση με Συγχώνευση (Merge Sort)

- Ένας ακόμη κλασικός αλγόριθμος ταξινόμησης που μάλιστα λειτουργεί με αναδρομή, είναι η **Ταξινόμηση με Συγχώνευση (Merge Sort)**.

- Ο αλγόριθμος λειτουργεί ως εξής:
  - Ταξινομεί το αριστερό κομμάτι του πίνακα
  - Ταξινομεί το δεξί κομμάτι του πίνακα
  - Συγχωνεύει τα δύο ταξινομημένα πλέον κομμάτια σε μία ταξινομημένη ακολουθία
- Η ταξινόμηση κάθε κομματιού γίνεται με αναδρομική κλήση της ίδιας διαδικασίας.

Α. Αναδρομή

2. Αναδρομικές Διαδικασίες

1. Ταξινόμηση με Συγχώνευση (Merge Sort)

- Η υλοποίηση σε ψευδογλώσσα είναι η ακόλουθη:

```
ΔΙΑΔΙΚΑΣΙΑ merge_sort(%PIN,start,finish)
ΔΙΕΠΑΦΗ
  ΕΙΣΟΔΟΣ
    PIN: ARRAY[1..MAX_N] OF INTEGER;
    start,finish: INTEGER;
  ΕΞΟΔΟΣ
    PIN: ARRAY[1..MAX_N] OF INTEGER;
ΔΕΔΟΜΕΝΑ
  middle: INTEGER;
ΑΡΧΗ
  ΕΑΝ (start<finish) ΤΟΤΕ
    ΕΑΝ (start+1=finish) ΤΟΤΕ /* 2 στοιχεία */
      ΕΑΝ (PIN[start]>PIN[finish]) ΤΟΤΕ
        ΥΠΟΛΟΓΙΣΕ swap(%PIN[start],%PIN[finish]);
      ΕΑΝ-ΤΕΛΟΣ
    ΑΛΛΙΩΣ  /* περισσότερα από 2 στοιχεία */
      middle:=(start+finish) DIV 2;
      ΥΠΟΛΟΓΙΣΕ merge_sort(%PIN,start,middle);
      ΥΠΟΛΟΓΙΣΕ merge_sort(%PIN,middle+1,finish);
      ΥΠΟΛΟΓΙΣΕ merge(%PIN,start,finish);
    ΕΑΝ-ΤΕΛΟΣ
  ΕΑΝ-ΤΕΛΟΣ
ΤΕΛΟΣ-ΔΙΑΔΙΚΑΣΙΑΣ
```


Α. Αναδρομή

2. Αναδρομικές Διαδικασίες

1. Ταξινόμηση με Συγχώνευση (Merge Sort)

- Αναδρομική Κλήση MergeSort(A,1,16)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
18	7	4	11	9	20	6	1	22	19	14	5	2	9	10	13

Α. Αναδρομή

2. Αναδρομικές Διαδικασίες

1. Ταξινόμηση με Συγχώνευση (Merge Sort)

- Αναδρομική Κλήση MergeSort(A,1,16)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
18	7	4	11	9	20	6	1	22	19	14	5	2	3	10	13

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

9	10	11	12	13	14	15	16
22	19	14	5	2	3	10	13

Α. Αναδρομή

2. Αναδρομικές Διαδικασίες

1. Ταξινόμηση με Συγχώνευση (Merge Sort)

- Αναδρομική Κλήση (A,1,8)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
18	7	4	11	9	20	6	1	22	19	14	5	2	3	10	13

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

9	10	11	12	13	14	15	16
22	19	14	5	2	3	10	13

1	2	3	4
18	7	4	11

5	6	7	8
9	20	6	1

Α. Αναδρομή
2. Αναδρομικές Διαδικασίες
1. Ταξινόμηση με Συγχώνευση (Merge Sort)

- Αναδρομική Κλήση (Α,1,4)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
18	7	4	11	9	20	6	1	22	19	14	5	2	3	10	13

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

9	10	11	12	13	14	15	16
22	19	14	5	2	3	10	13

1	2	3	4
18	7	4	11

5	6	7	8
9	20	6	1

1	2
18	7

3	4
4	11

Α. Αναδρομή
2. Αναδρομικές Διαδικασίες
1. Ταξινόμηση με Συγχώνευση (Merge Sort)

- Αναδρομική Κλήση (Α,1,2): Ταξινομηση του υποπίνακα

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
18	7	4	11	9	20	6	1	22	19	14	5	2	3	10	13

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

9	10	11	12	13	14	15	16
22	19	14	5	2	3	10	13

1	2	3	4
18	7	4	11

5	6	7	8
9	20	6	1

1	2
7	18

3	4
4	11

Α. Αναδρομή
2. Αναδρομικές Διαδικασίες
1. Ταξινόμηση με Συγχώνευση (Merge Sort)

- Αναδρομική Κλήση (Α,3,4): Ταξινομηση του υποπίνακα

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
18	7	4	11	9	20	6	1	22	19	14	5	2	3	10	13

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

9	10	11	12	13	14	15	16
22	19	14	5	2	3	10	13

1	2	3	4
18	7	4	11

5	6	7	8
9	20	6	1

1	2
7	18

3	4
4	11

Α. Αναδρομή
2. Αναδρομικές Διαδικασίες
1. Ταξινόμηση με Συγχώνευση (Merge Sort)

- Αναδρομική Κλήση (Α,1,4): Συγχώνευση των δύο υποπινάκων

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
18	7	4	11	9	20	6	1	22	19	14	5	2	3	10	13

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

9	10	11	12	13	14	15	16
22	19	14	5	2	3	10	13

1	2	3	4
4	7	11	18

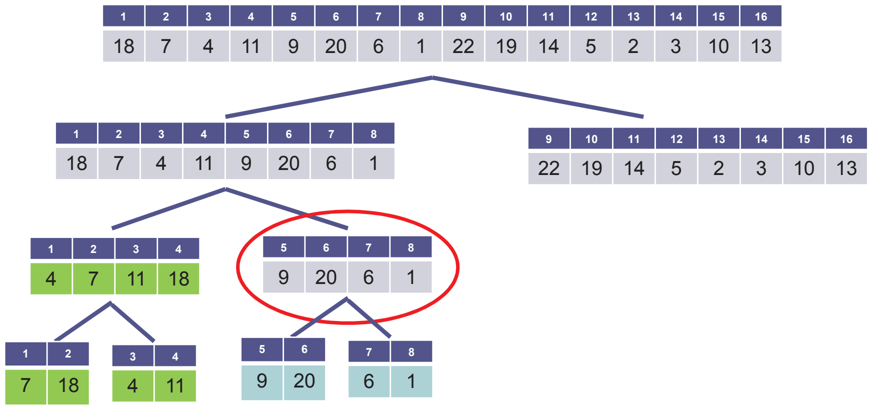
5	6	7	8
9	20	6	1

1	2
7	18

3	4
4	11

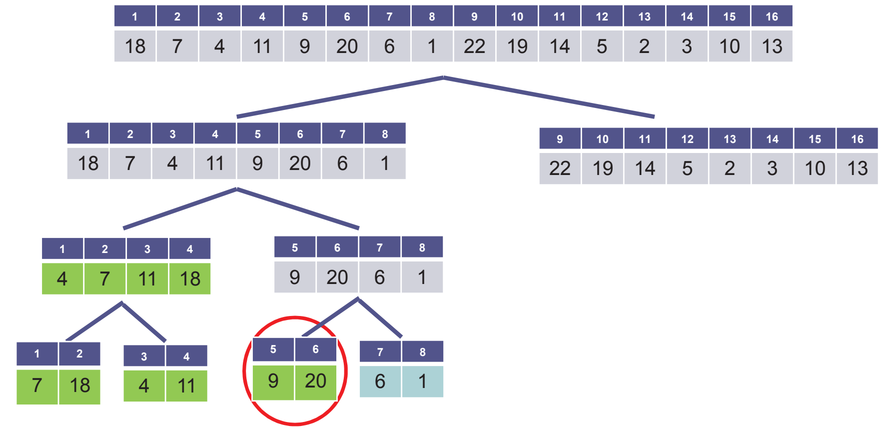
Α. Αναδρομή
2. Αναδρομικές Διαδικασίες
1. Ταξινόμηση με Συγχώνευση (Merge Sort)

- Αναδρομική Κλήση (Α,5,8)



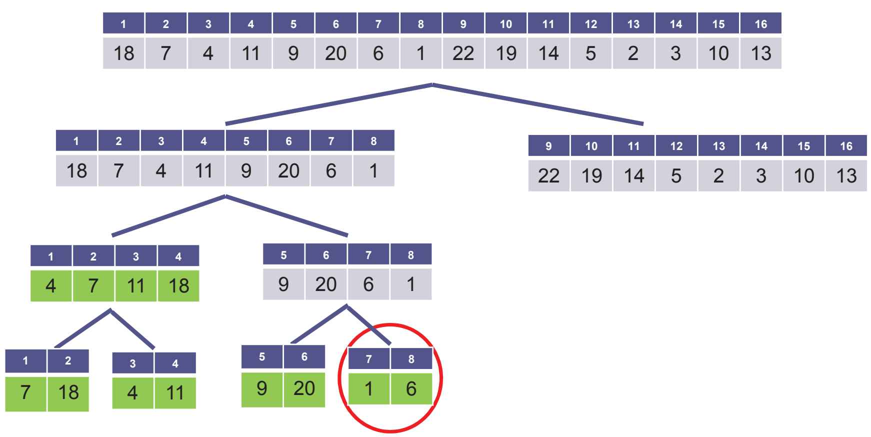
Α. Αναδρομή
2. Αναδρομικές Διαδικασίες
1. Ταξινόμηση με Συγχώνευση (Merge Sort)

- Αναδρομική Κλήση (Α,5,6): Ταξινομηση του υποπίνακα



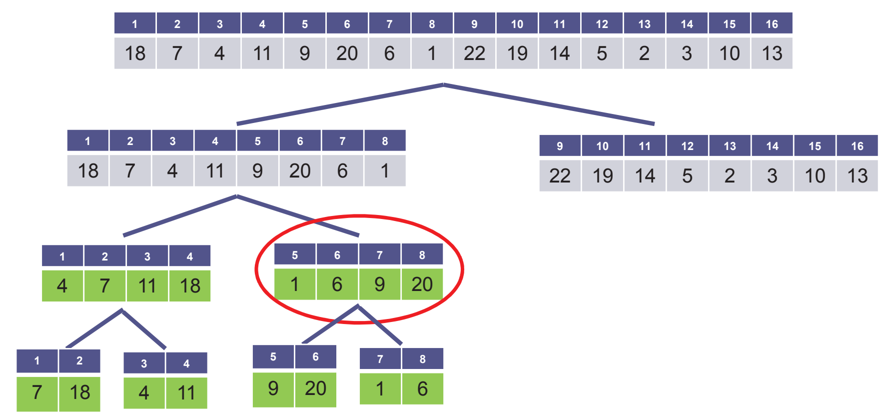
Α. Αναδρομή
2. Αναδρομικές Διαδικασίες
1. Ταξινόμηση με Συγχώνευση (Merge Sort)

- Αναδρομική Κλήση (Α,7,8): Ταξινομηση του υποπίνακα



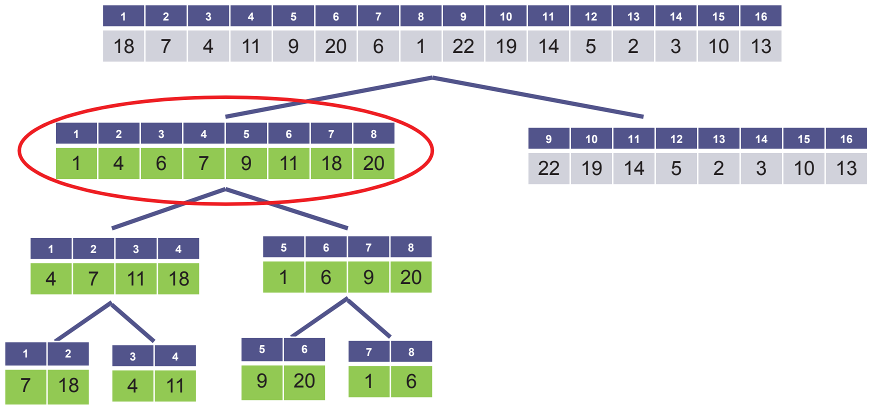
Α. Αναδρομή
2. Αναδρομικές Διαδικασίες
1. Ταξινόμηση με Συγχώνευση (Merge Sort)

- Αναδρομική Κλήση (Α,5,8): Συγχώνευση των δύο υποπινάκων



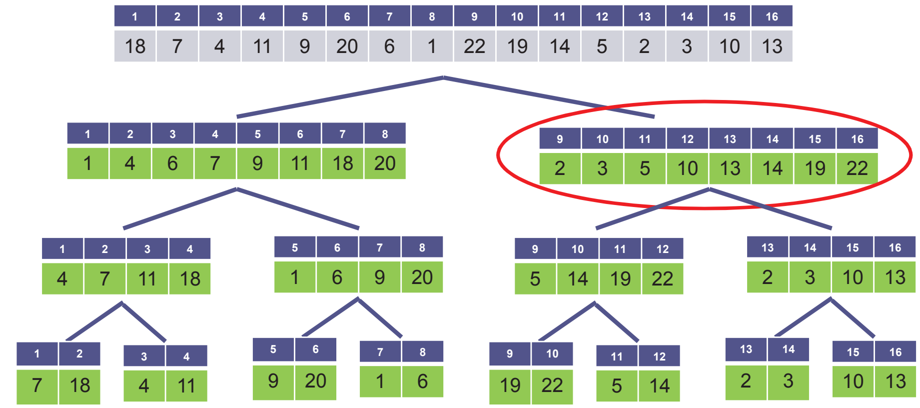
Α. Αναδρομή
2. Αναδρομικές Διαδικασίες
1. Ταξινόμηση με Συγχώνευση (Merge Sort)

- Αναδρομική Κλήση (A,1,8): Συγχώνευση των δύο υποπινάκων



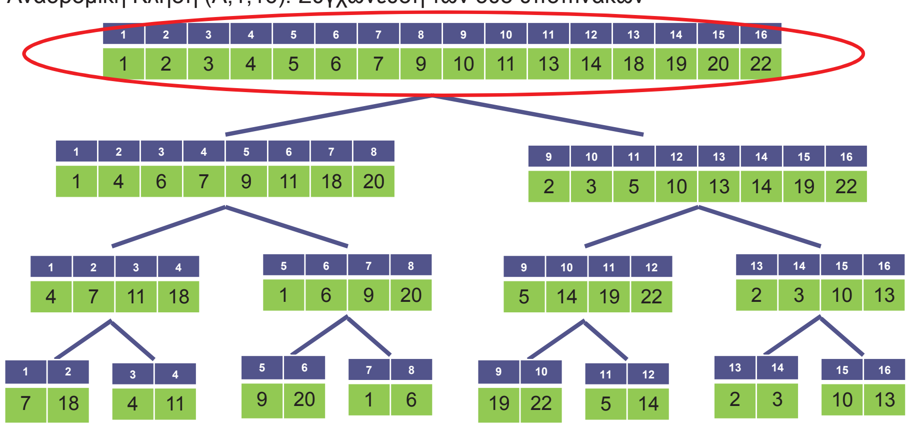
Α. Αναδρομή
2. Αναδρομικές Διαδικασίες
1. Ταξινόμηση με Συγχώνευση (Merge Sort)

- Αντίστοιχα θα γίνουν όλες οι αναδρομικές κλήσεις στο (9,16)



Α. Αναδρομή
2. Αναδρομικές Διαδικασίες
1. Ταξινόμηση με Συγχώνευση (Merge Sort)

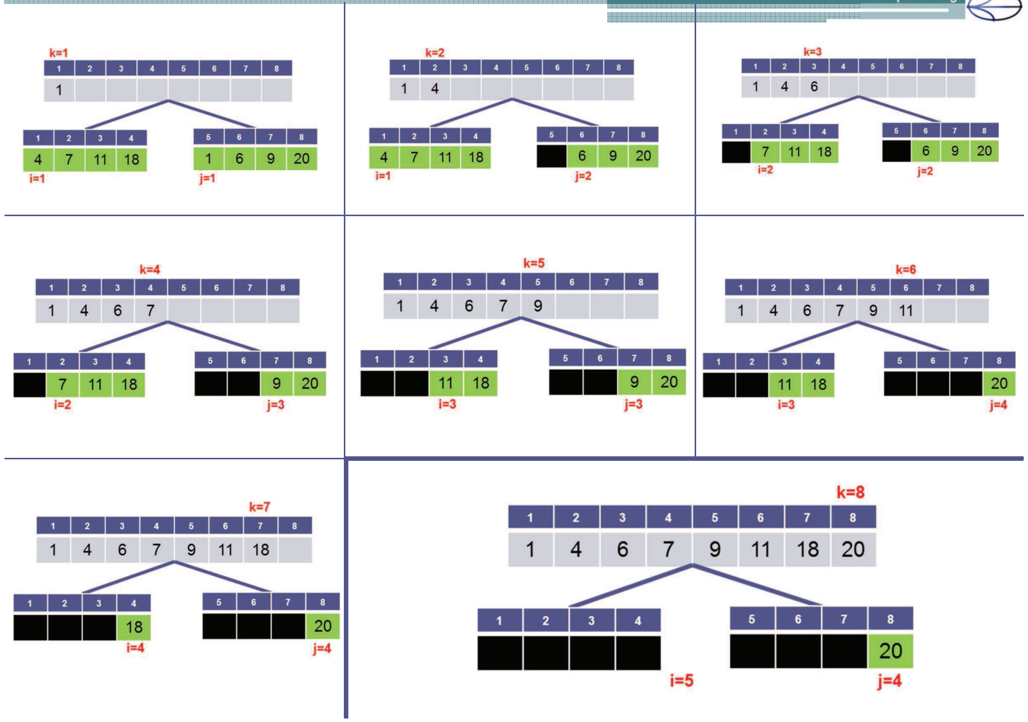
- Αναδρομική Κλήση (A,1,16): Συγχώνευση των δύο υποπινάκων



Α. Αναδρομή
2. Αναδρομικές Διαδικασίες
1. Ταξινόμηση με Συγχώνευση (Merge Sort)

- 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. Αναδρομή

2. Αναδρομικές Διαδικασίες

1. Ταξινόμηση με Συγχώνευση (Merge Sort)

- Μία υλοποίηση της merge είναι η ακόλουθη:

```
ΔΙΑΔΙΚΑΣΙΑ merge(%PIN,start,finish)
ΔΙΕΠΑΦΗ
  ΕΙΣΟΔΟΣ
    PIN: ARRAY[1..MAX_N] OF INTEGER;
    start,finish: INTEGER;
  ΕΞΟΔΟΣ
    PIN: ARRAY[1..MAX_N] OF INTEGER;
ΔΕΔΟΜΕΝΑ
  C: ARRAY[1..MAX_N] OF INTEGER;
  middle: INTEGER;
  i,j,k:INTEGER;
  n: INTEGER;
  m: INTEGER;
ΑΡΧΗ
  middle:=(start+finish) DIV 2;
  /* 1ος πίνακας PIN[start...middle]*/
  i:=start;
  n:=middle;
  /* 2ος πίνακας PIN[middle+1...finish]*/
  j:=middle+1;
  m:=finish;
  /* C: συγχωνευμένος πίνακας */
  k:=1;                                                   (συνεχίζεται…)
```


Α. Αναδρομή

2. Αναδρομικές Διαδικασίες

1. Ταξινόμηση με Συγχώνευση (Merge Sort)

- Μία υλοποίηση της merge είναι η ακόλουθη:

```
(συνέχεια…)

  /* 1. Συγχώνευση των δύο πινάκων */

  ΕΝΟΣΩ (i<=n AND j<=m) ΕΠΑΝΑΛΑΒΕ
    ΕΑΝ (PIN[i]<PIN[j]) ΤΟΤΕ
      C[k]:=PIN[i];
      k:=k+1;
      i:=i+1;
    ΑΛΛΙΩΣ
      C[k]:=PIN[j];
      k:=k+1;
      j:=j+1;
    ΕΑΝ-ΤΕΛΟΣ
  ΕΝΟΣΩ-ΤΕΛΟΣ
                                                  (συνεχίζεται…)
```


Α. Αναδρομή

2. Αναδρομικές Διαδικασίες

1. Ταξινόμηση με Συγχώνευση (Merge Sort)

```
  /* 2. Αντιγραφή του πίνακα που «περισσεύει στο τέλος του συγχωνευμένου πίνακα */
  ΕΑΝ (i=n+1) ΤΟΤΕ /* Εξαντλήθηκε ο 1ος πίνακας */
    ΕΝΟΣΩ (j<=m) ΕΠΑΝΑΛΑΒΕ
      C[k]:=PIN[j];
      k:=k+1;
      j:=j+1;
    ΕΝΟΣΩ-ΤΕΛΟΣ
  ΑΛΛΙΩΣ /* Εξαντλήθηκε ο 2ος πίνακας */
    ΕΝΟΣΩ (i<=n) ΕΠΑΝΑΛΑΒΕ
      C[k]:=PIN[i];
      k:=k+1;
      i:=i+1;
    ΕΝΟΣΩ-ΤΕΛΟΣ
  ΕΑΝ-ΤΕΛΟΣ

  /* 3. Αντιγραφή του C στον PIN */
  k:=1;
  i:=start;
  ΕΝΟΣΩ (i<=finish) ΕΠΑΝΑΛΑΒΕ
    PIN[i]:=C[k];
    i:=i+1;
    k:=k+1;
  ΕΝΟΣΩ-ΤΕΛΟΣ
ΤΕΛΟΣ-ΔΙΑΔΙΚΑΣΙΑΣ
```

Α. Αναδρομή
2. Αναδρομικές Διαδικασίες
1. Ταξινόμηση με Συγχώνευση (Merge Sort)

- **Άσκηση:** Κατασκευάστε ένα πρόγραμμα που να αναδεικνύει την χρήση της merge sort και παρουσιάστε ένα παράδειγμα εκτέλεσης για τον πίνακα A=[6 2 4 1 8 10 11 12]

Α. Αναδρομή
2. Αναδρομικές Διαδικασίες
2. Γρήγορη Ταξινόμηση (Quick Sort)

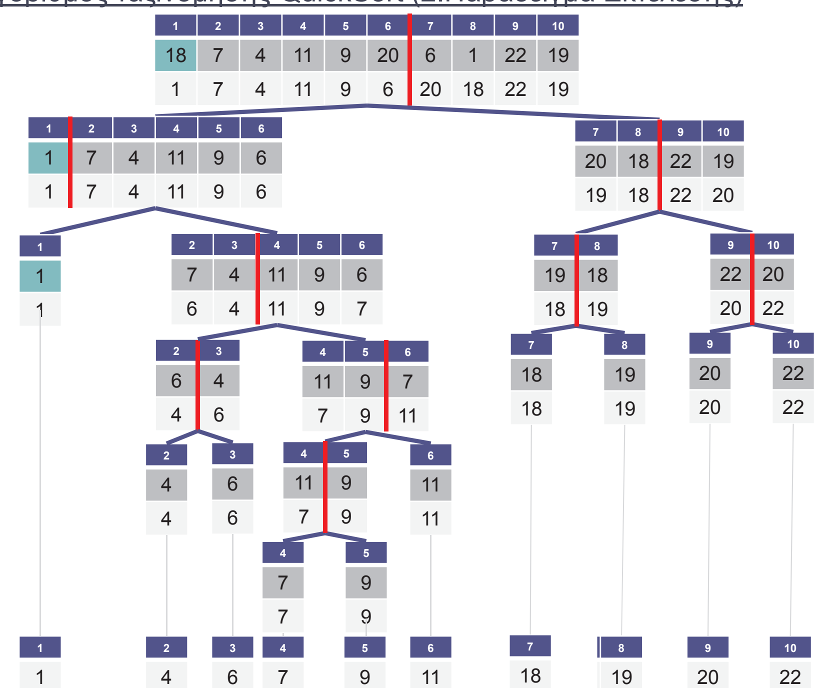
- Τελευταίος αλγόριθμος ταξινόμησης που θα μελετήσουμε (και λειτουργεί με αναδρομή), είναι η **Γρήγορη Ταξινόμηση (Quick Sort)**.
- Ο αλγόριθμος λειτουργεί ως εξής:
  - Επιλέγει ένα Στοιχείο του Πίνακα (Οδηγό Στοιχείο – Εδώ το στοιχείο που είναι στην πρώτη θέση)
  - Χωρίζει τον πίνακα σε δύο μέρη:
    - Τα στοιχεία που είναι μικρότερα του οδηγού στοιχείου
    - Τα στοιχεία που είναι μεγαλύτερα ή ίσα του οδηγού στοιχείου
  - Επαναλαμβάνει αναδρομικά στους δύο υποπίνακες που προέκυψαν.

Α. Αναδρομή
2. Αναδρομικές Διαδικασίες
2. Γρήγορη Ταξινόμηση (Quick Sort)

- Η υλοποίηση σε ψευδογλώσσα είναι η ακόλουθη:

```
ΔΙΑΔΙΚΑΣΙΑ quick_sort(%PIN,start,finish)
ΔΙΕΠΑΦΗ
  ΕΙΣΟΔΟΣ
    PIN: ARRAY[1..MAX_N] OF INTEGER;
    start,finish: INTEGER;
  ΕΞΟΔΟΣ
    PIN: ARRAY[1..MAX_N] OF INTEGER;
ΑΡΧΗ
  ΕΑΝ (start<finish) ΤΟΤΕ
      pos=partition(%PIN,start,finish); /* Διαμέριση του πίνακα */
      ΥΠΟΛΟΓΙΣΕ quick_sort(%PIN,start,pos); /* Αναδρομή στον αριστερό υποπίνακα */
      ΥΠΟΛΟΓΙΣΕ quick_sort(%PIN,pos+1,finish);/* Αναδρομή στον δεξιό υποπίνακα */
  ΕΑΝ-ΤΕΛΟΣ
ΤΕΛΟΣ-ΔΙΑΔΙΚΑΣΙΑΣ
```


Ο αλγόριθμος ταξινόμησης QuickSort (2.Παράδειγμα Εκτέλεσης)

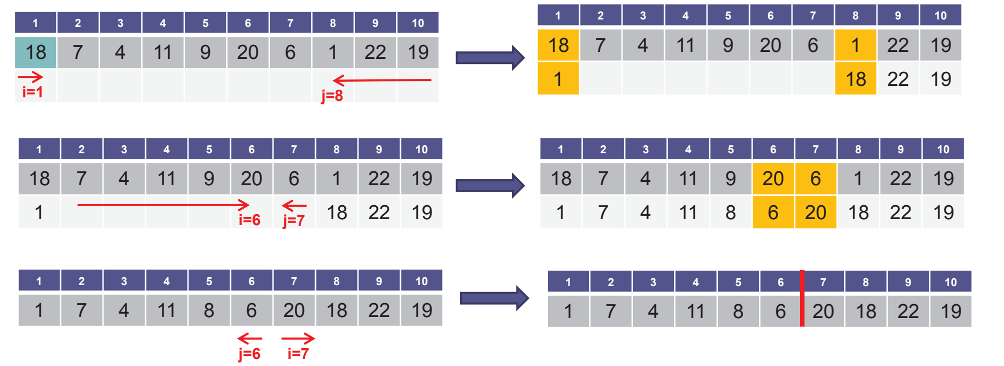


Α. Αναδρομή
2. Αναδρομικές Διαδικασίες
2. Γρήγορη Ταξινόμηση (Quick Sort)

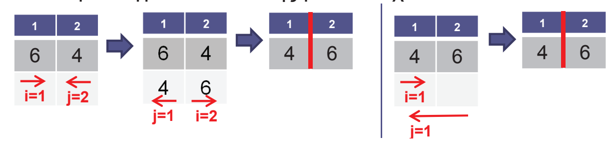
- Η διαμέριση των στοιχείων στα μικρότερα και μεγαλύτερα ή ίσα μπορεί να γίνει με πολλούς τρόπους:
  - Π.χ. ένας απλός τρόπος είναι να χρησιμοποιήσουμε έναν βοηθητικό πίνακα που να βάζουμε στα αριστερά τα μικρότερα στοιχεία και στα δεξιά τα μεγαλύτερα στοιχεία του οδηγού.
- Ωστόσο εμείς θα μελετήσουμε έναν τρόπο διαμέρισης, που αναφέρεται ως σχήμα του Hoare:
  - Σαρώνουμε τον πίνακα από αριστερά ψάχνοντας για ένα στοιχείο που είναι μεγαλύτερο (ή ίσο) 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.
- Θα μελετήσουμε το σχήμα του Hoare με παραδείγματα (επόμενη διαφάνεια)

Α. Αναδρομή
2. Αναδρομικές Διαδικασίες
2. Γρήγορη Ταξινόμηση (Quick Sort)

- Παράδειγμα εκτέλεσης της partition:



- Παραδείγματα εκτέλεσης με 2 στοιχεία:



και με 1 στοιχειο:

Α. Αναδρομή
2. Αναδρομικές Διαδικασίες
2. Γρήγορη Ταξινόμηση (Quick Sort)

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

```
ΣΥΝΑΡΤΗΣΗ
partition(%PIN,start,finish):INTEGER
ΔΙΕΠΑΦΗ
  ΕΙΣΟΔΟΣ
    PIN: ARRAY[1..MAX_N] OF INTEGER;

    start,finish: INTEGER;
  ΕΞΟΔΟΣ
    PIN: ARRAY[1..MAX_N] OF INTEGER;

    partition: INTEGER;
ΔΕΔΟΜΕΝΑ
   pivot,i,j: INTEGER;
   stop: BOOLEAN;
ΑΡΧΗ
  pivot:=PIN[start];
  i:=start-1;
  j:=finish+1;
  stop:=FALSE;
  ΕΝΟΣΩ (stop=FALSE) ΕΠΑΝΑΛΑΒΕ
    ΕΠΑΝΑΛΑΒΕ
      j:=j-1;
    ΜΕΧΡΙ (PIN[j]<=pivot)

    ΕΠΑΝΑΛΑΒΕ
      i:=i+1;
    ΜΕΧΡΙ (PIN[i]>=pivot)

    ΕΑΝ (i<j) ΤΟΤΕ
      ΥΠΟΛΟΓΙΣΕ swap(%PIN[i],%PIN[j]);
    ΑΛΛΙΩΣ
      stop:=TRUE;
      partition:=j;
    ΕΑΝ-ΤΕΛΟΣ
  ΕΝΟΣΩ-ΤΕΛΟΣ
ΤΕΛΟΣ-ΣΥΝΑΡΤΗΣΗΣ
```


Α. Αναδρομή
2. Αναδρομικές Διαδικασίες
2. Γρήγορη Ταξινόμηση (Quick Sort)

- **Άσκηση:** Κατασκευάστε ένα πρόγραμμα που να αναδεικνύει την χρήση της merge sort και παρουσιάστε ένα παράδειγμα εκτέλεσης για τον πίνακα Α=[6 2 4 1 8 10 11 12]

Β. Ασκήσεις
Εφαρμογή 1 (Αναδρομή: Η ακολουθία FIbonacci)

- Η ακολουθία fibonacci ορίζεται ως:
  - $F_n=F_{n-1}+F_{n-2}$, για n>2
  - $F_2=1$
  - $F_1=1$
- Για παράδειγμα έχουμε $F_1=1,F_2=1,F_3=2,F_4=3,F_5=5,F_6=8$ κ.ο.κ.
- Ορίστε την συνάρτηση fibonacci(n) που δέχεται ως όρισμα έναν φυσικό και επιστρέφει το n-οστό fibonacci.
- Έπειτα κατασκευάστε έναν αλγόριθμο που διαβάζει από τον χρήστη έναν ακέραιο και υπολογίζει και επιστρέφει τον αριθμό fibonacci του αριθμού που εισήγαγε ο χρήστης.

Β. Ασκήσεις
Εφαρμογή 2 (Αναδρομή: ΜΚΔ με τον αλγόριθμο του Ευκλείδη)

- Ο αλγόριθμος του Ευκλείδη για την εύρεση του Μέγιστου Κοινού Διαιρέτη δύο (φυσικών) αριθμών:
  - Ξεκινά με ένα ζεύγος φυσικών και σχηματίζει ένα νέο ζευγάρι με τον μικρότερο αριθμό και την διαφορά του μικρότερου από τον μεγαλύτερο αριθμό.
  - Η διαδικασία επαναλαμβάνεται εωσότου οι αριθμοί γίνουν ίσοι. Ο αριθμός αυτός είναι ο ΜΚΔ των αρχικών αριθμών.
- Μαθηματικά ο ΜΚΔ(a,b) όπου a,b είναι φυσικοί:
  - Είναι ίσο με a, αν a=b
  - Είναι ίσο με ΜΚΔ(a,b-a), αν a<b
  - Είναι ίσο με ΜΚΔ(a-b,b), αλλιώς
- Κατασκευάστε έναν αλγόριθμο που θα υλοποιεί με μία αναδρομική συνάρτηση τον υπολογισμό του ΜΚΔ και θα ζητάει από το χρήστη να εισάγει τους δύο φυσικούς, θα κάνει κατάλληλη κλήση της συνάρτησης και θα τυπώνει τον ΜΚΔ των αριθμών.

Β. Ασκήσεις
Εφαρμογή 3 (Πρόγραμμα: Ταξινόμηση Πίνακα)

- Επεκτείνετε το πρόγραμμα της ταξινόμησης πινάκων ώστε να ενσωματώνει και τους αλγορίθμους της ταξινόμησης με συγχώνευση (Merge Sort) και της γρήγορης ταξινόμησης (Quick Sort).