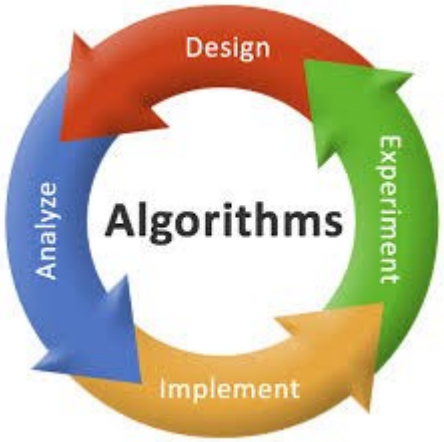
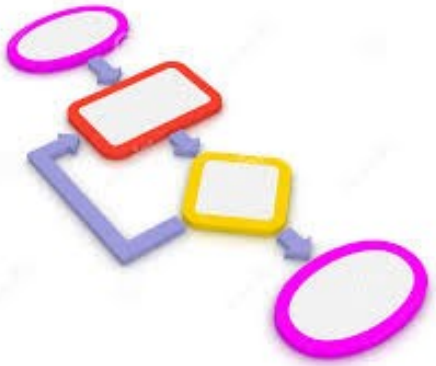




ΑΛΓΟΡΙΘΜΟΙ

Δρ. Χάρης Κουζινόπουλος

Εθνικό Κέντρο Έρευνας και Τεχνολογικής Ανάπτυξης

Πανεπιστήμιο Μακεδονίας

Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης

ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ

ΣΧΟΛΗ ΕΠΙΣΤΗΜΩΝ ΠΛΗΡΟΦΟΡΙΑΣ
Τμήμα Εφαρμοσμένης Πληροφορικής

CMOR
Computational Methodologies & Operations Research

Αλγόριθμοι Ταξινόμησης – [1]

Ταξινόμηση: Πιθανότατα το περισσότερο χρησιμοποιούμενο πρόβλημα της πληροφορικής.

Το πρόβλημα: Δίνεται διάνυσμα $T = [t_1\ t_2\ \ldots\ t_n]$ και ζητείται να ταξινομηθεί (κατά αύξουσα τάξη).

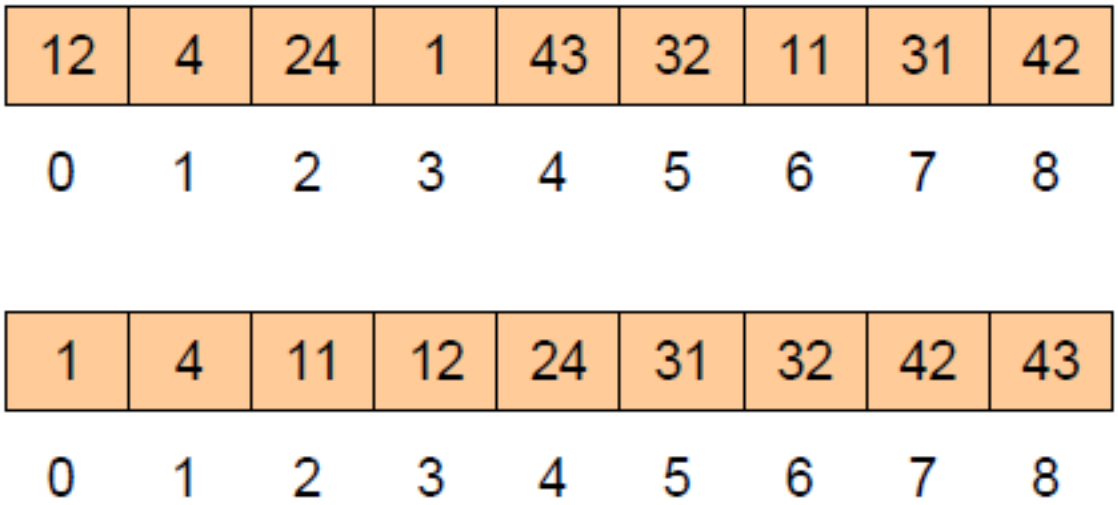
Τ ταξινομημένο κατά ***αύξουσα (φθίνουσα) τάξη***

$$t_1 \leq t_2 \leq t_3 \leq \ldots \leq t_{n-1} \leq t_n\ ,\ (t_1 \geq t_2 \geq t_3 \geq \ldots \geq t_{n-1} \geq t_n)$$

Κριτήρια αποδοτικότητας μιας μεθόδου ταξινόμησης αποτελούν **ο αριθμός των συγκρίσεων** που απαιτούνται και **ο αριθμός των μετακινήσεων** που γίνονται προκειμένου να επιτευχθεί η ταξινόμηση.

Αλγόριθμοι Ταξινόμησης – [2]

Παράδειγμα: Ταξινόμηση ακεραίων



Ταξινόμηση με Επιλογή – [1]

Επέλεξε το μικρότερο από τα n στοιχεία του Τ και ενάλλαξέ το με το πρώτο, δηλ. το Τ(1).

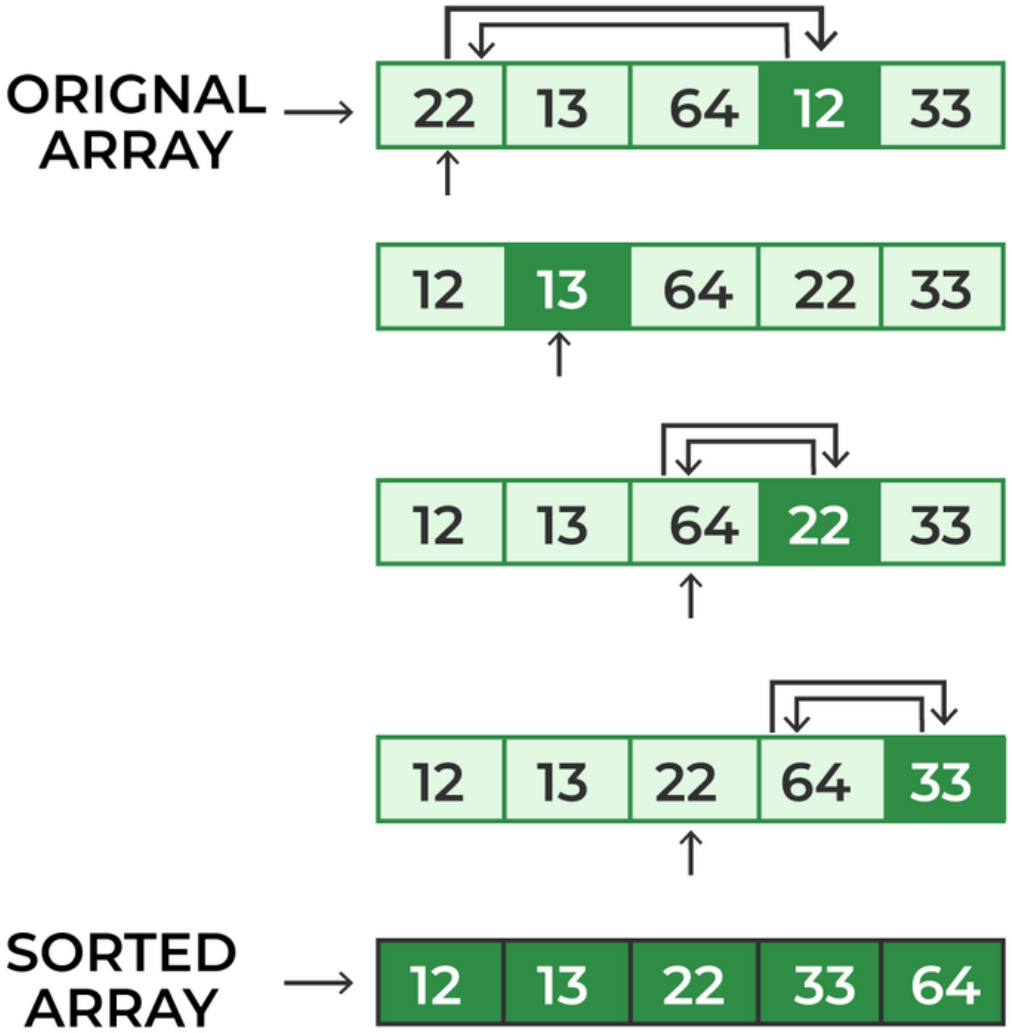
Από τα στοιχεία Τ(2), Τ(3), …, Τ(n) επέλεξε το μικρότερο και ενάλλαξέ το με το Τ(2), κ.ο.κ

Αλγόριθμος: min1
Είσοδος: Τ, n
Έξοδος: min, index

```
1   min ← T(1)
2   index ← 1
3   for i ← 2 to n
4       if T(i) < min
5           min ← T(i)
6           index ← i
```

Στον επόμενο ψευδοκώδικα χρησιμοποιείται ο min1 που υπολογίζει το ελάχιστο στοιχείο διανύσματος Χ και τη θέση του (index) στο Χ.

Ταξινόμηση με Επιλογή – [2]



Ταξινόμηση με Επιλογή – [3]

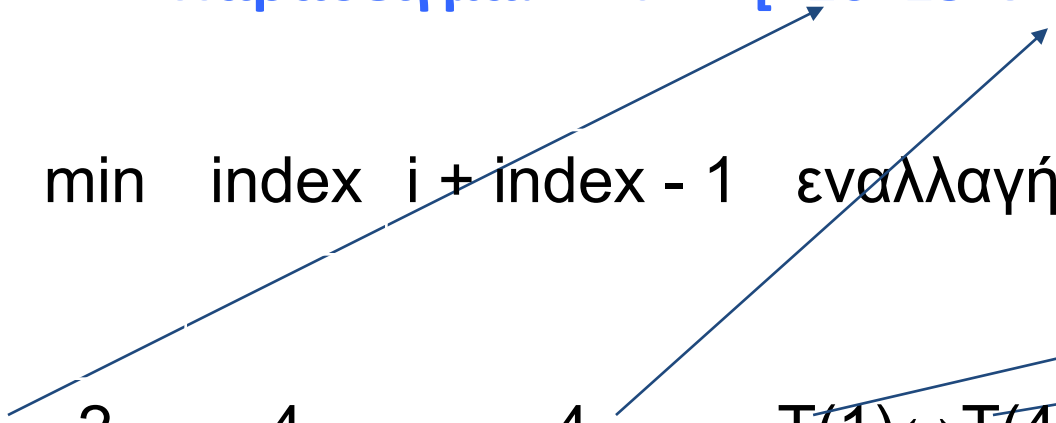
Αλγόριθμος: selectsort *Είσοδος*: T, n *Έξοδος*: T	
1	**for** i ← 1 **to** n-1
2	[min, index] ← min1(T(i:n))
3	T(i+index-1) ← T(i)
4	T(i) ← min

index = θέση ελάχιστου στοιχείου στο T(i:n) (=X).

Επειδή πλήθος (T(1:i–1))=i–1, το ελάχιστο στοιχείο X(index) του X=T(i:n) βρίσκεται στη θέση i+index–1 του διανύσματος T.

Ταξινόμηση με Επιλογή – [4]

Παράδειγμα. **T = [10 15 7 2 5 4]**



i	min	index	i + index - 1	εναλλαγή	Διάνυσμα T
					[10 15 7 2 5 4]
1	2	4	4	T(1)↔T(4)	[2 15 7 10 5 4]
2	4	5	6	T(2)↔T(6)	[2 4 7 10 5 15]
3	5	3	5	T(3)↔T(5)	[2 4 5 10 7 15]
4	7	2	5	T(4)↔T(5)	[2 4 5 7 10 15]
5	10	1	5	T(5)↔T(5)	[2 4 5 7 10 15]

Ταξινόμηση με Επιλογή – [5]

Άσκηση. Γράψτε τον κώδικα του αλγορίθμου selectsort ώστε να μη κάνει χρήση της συνάρτησης min1.

Αλγόριθμος: min1
Είσοδος: T, n
Έξοδος: min, index

```
1   min ← T(1)
2   index ← 1
3   for i ← 2 to n
4       if T(i) < min
5           min ← T(i)
6           index ← i
```

Αλγόριθμος: selectsort
Είσοδος: T, n
Έξοδος: T

```
1   for i ← 1 to  n-1
2       [min, index] ← min1(T(i:n))
3       T(i+index-1) ← T(i)
4       T(i) ← min
```

Ταξινόμηση με Επιλογή – [6]

Απάντηση

```
for i ← 1 to n-1
    min ← T(i)
    index ← i
    for j ← i+1 to n
        if T(j)<min
            min ← T(j)
            index ← j
    T(index) ← T(i)
    T(i) ← min
```

Ταξινόμηση με Επιλογή – [7]

Πρόβλημα. Τροποποιήστε τον αλγόριθμο της ταξινόμησης με επιλογή έτσι ώστε να καταγράφονται και οι θέσεις των ταξινομημένων στοιχείων στο αρχικό μη ταξινομημένο διάνυσμα.

Για παράδειγμα, αν Τ είναι το αρχικό διάνυσμα και Χ το τελικό διάνυσμα, που προκύπτει από την ταξινόμηση του Τ και είναι Χ(k) = Τ(m), τότε στο διάνυσμα των αρχικών θέσεων, theseis, θα είναι theseis(k) = m.

Ταξινόμηση με Επιλογή – [8]

Απάντηση

```
theseis ← 1:n
for i ← 1 to n-1
    min ← T(i)
    index ← i
    for j ← i+1 to n
        if T(j)<min
            min ← T(j)
            index ← j
    T(index) ← T(i)
    T(i) ← min
     temp ← theseis(i)
     theseis(i) ← theseis(index)
     theseis(index) ← temp
```

Ταξινόμηση φυσαλίδας

Αποτελείται από στάδια: το πολύ n – 1

Σε κάθε στάδιο ελέγχεται, αν το τρέχον διάνυσμα είναι ταξινομημένο, δηλαδή αν είναι:

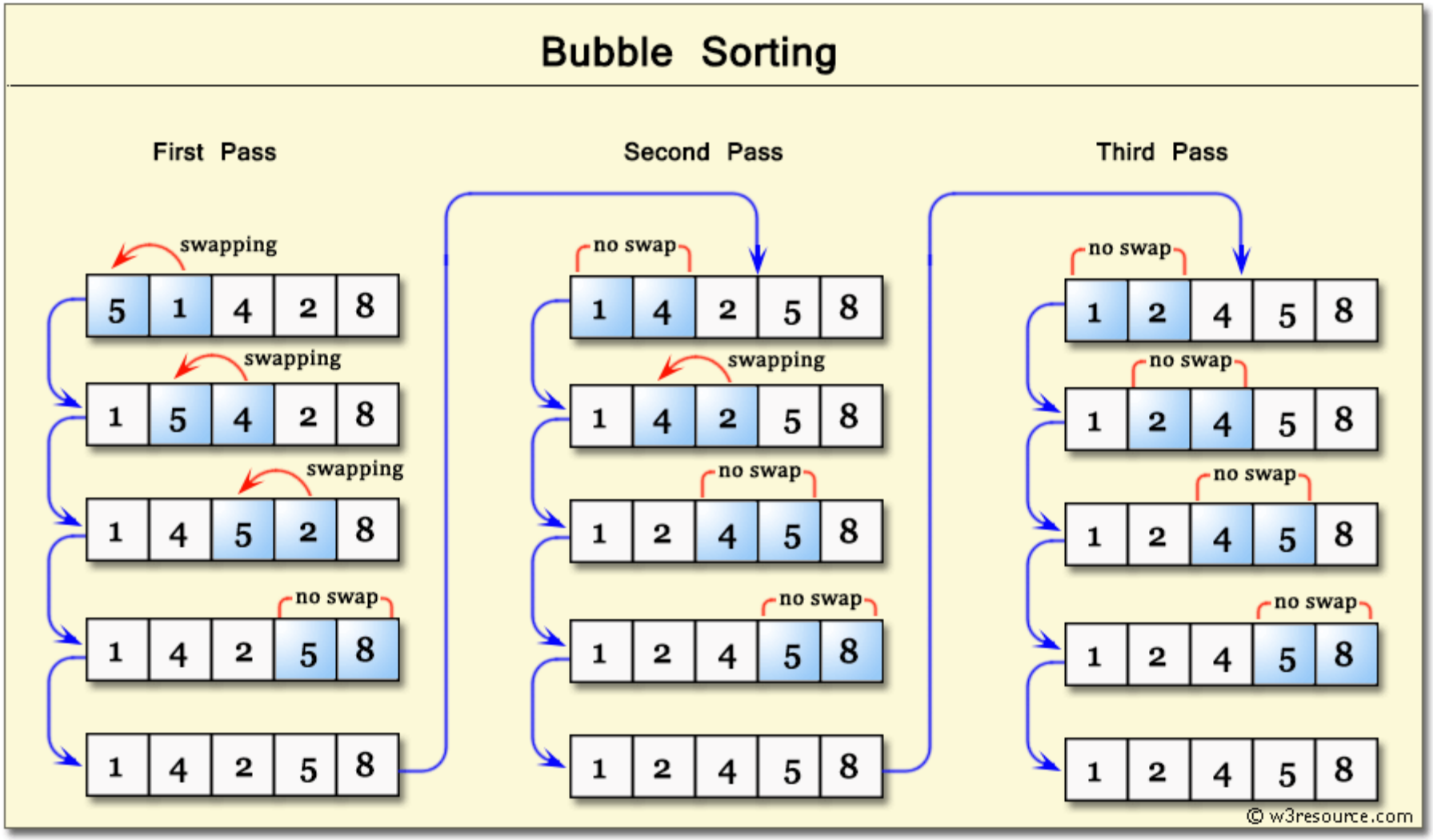
$$T(j) \leq T(j+1), \quad j = 1, 2, \ldots n-1$$

Αν πράγματι είναι, οι υπολογισμοί σταματούν.

Αν δεν είναι, τότε κατά τη διάρκεια του σταδίου και για όλα τα j για τα οποία ισχύει:

$$T(j) > T(j+1), \text{ εναλλαγή}(T(j), T(j+1))$$

Ταξινόμηση φυσαλίδας



Ταξινόμηση φυσαλίδας

Σημαντική παρατήρηση. Στο τέλος του σταδίου k τα τελευταία k στοιχεία είναι ταξινομημένα.

Προέλευση ονοματολογίας: «ταξινόμηση με τη μέθοδο της φυσαλίδας - bubblesort»

Στον ψευδοκώδικα, όπου

t = πλήθος εναλλαγών σε ένα στάδιο, **(Αν t = 0 ⇒ STOP)**

i = δείκτης: τα στοιχεία i+1, i+2, ..., n είναι ταξινομημένα

Ταξινόμηση φυσαλίδας

Αλγόριθμος: bubblesort
Είσοδος: T, n
Έξοδος: T

```
1  i ← n
2  t ← -1
3  while i ≥ 2 and t ≠ 0
4     t ← 0
5          for  j ← 1 to i - 1
6               if T(j) > T(j+1)
7                    [T(j) , T(j+1)]  ← swap(T(j), T(j+1))
8                    t ← t+1
9          i ← i – 1
```

Μεταβλητές Εισόδου/Εξόδου

```
ΟΝΟΜΑ ΑΡΧΕΙΟΥ: 03-bubblesort.c
```

```
ΜΕΤΑΒΛΗΤΕΣ ΕΙΣΟΔΟΥ:
Τ – Πίνακας (τύπου double) με τους αριθμούς προς ταξινόμηση.
Ν – Η διάσταση του πίνακα Τ (τύπου int).
```

```
ΜΕΤΑΒΛΗΤΕΣ ΕΞΟΔΟΥ:
Τ – Πίνακας (τύπου double) με τους αριθμούς ταξινομημένους.
```

Filename: 03-bubblesort.c

Ταξινόμηση με Εισαγωγή – [1]

Περιγραφή της μεθόδου: Αν τα πρώτα i στοιχεία του T είναι ταξινομημένα, κάνε εισαγωγή του T(i + 1) στο T(1:i) ώστε το διάνυσμα T(1:i + 1) να ταξινομηθεί.

Αιτιολογία ονοματολογίας: **Μέθοδος του χαρτοπαίκτη.**

Αλγόριθμος: insertsort *Δεδομένα*: T, n *Αποτελέσματα*: T	
1 2	**για** i από 2 **μέχρι** n T(1:i) ← insert(T(1:i-1), T(i))

Ταξινόμηση με Εισαγωγή – [2]

Εφαρμογή. T = [15 10 7 2 5 4]

i = 2	T = [15 <u>10</u> 7 2 5 4]
i = 3	T = [10 15 <u>7</u> 2 5 4]
i = 4	T = [7 10 15 <u>2</u> 5 4]
i = 5	T = [2 7 10 15 <u>5</u> 4]
i = 6	T = [2 5 7 10 15 <u>4</u>]
	T = [2 4 5 7 10 15]

Μεταβλητές Εισόδου/Εξόδου

ΟΝΟΜΑ ΑΡΧΕΙΟΥ: 04-insert.c

ΜΕΤΑΒΛΗΤΕΣ ΕΙΣΟΔΟΥ:
Τ – Πίνακας (τύπου double) με τους αριθμούς ταξινομημένους.
Ν – Η διάσταση του πίνακα Τ (τύπου int).
a – Ο αριθμός προς εισαγωγή στο Τ (τύπου double).

ΜΕΤΑΒΛΗΤΕΣ ΕΞΟΔΟΥ:
Τ – Πίνακας (τύπου double) με τους αριθμούς ταξινομημένους.

Filename: 04-insert.c

Μεταβλητές Εισόδου/Εξόδου

```
ΟΝΟΜΑ ΑΡΧΕΙΟΥ: 05-insertsort.c
```

```
ΜΕΤΑΒΛΗΤΕΣ ΕΙΣΟΔΟΥ:
Τ – Πίνακας (τύπου double) με τους αριθμούς προς ταξινόμηση.
Ν – Η διάσταση του πίνακα Τ (τύπου int).
```

```
ΜΕΤΑΒΛΗΤΕΣ ΕΞΟΔΟΥ:
Τ – Πίνακας (τύπου double) με τους αριθμούς ταξινομημένους.
```

Filename: 05-insertsort.c

Ταξινόμηση με Εισαγωγή – [3]

Αλγόριθμος: insertsort
Είσοδος: T, n
Έξοδος: T

```
1   for i ← 2 to n
2       T(1:i) ← insert(T(1:i-1), T(i))
```

Αλγόριθμος: insert
Είσοδος: T, m, a
Έξοδος: T

```
1           T(m+1) ← a
2           i ← m+1
3           while i > 1 and T(i-1) > T(i)
4                   temp ← T(i-1)
5                   T(i-1) ← T(i)
6                   T(i) ← temp
7                   i ← i-1
```

Πρόβλημα για το σπίτι – [10]

Γράψτε τον κώδικα του αλγόριθμου insertsort έτσι ώστε να μη κάνει χρήση της συνάρτησης insert. Στον νέο κώδικα δώστε το όνομα **insertsort2.c**

Αναζήτηση – [1]

Σειριακή αναζήτηση (Linear search)

Σειριακή αναζήτηση με τυχαία επιλογή (Linear search with random choice)

Αναζήτηση με βήμα Κ (Search with step K)

Δυαδική αναζήτηση (Binary search)

Αναζήτηση – [2]

Δίνεται μια λίστα (γραμμικός πίνακας) Τ από n στοιχεία και ένα επιπλέον στοιχείο, a, και το πρόβλημα είναι να αποφασίσουμε αν το a είναι στη λίστα Τ. Αν το a είναι στην λίστα Τ, θα πρέπει να προσδιορίσουμε τουλάχιστον ένα στοιχείο της λίστας Τ το οποίο να είναι ίσο με αυτό.

Εφαρμογές Αναζήτησης

Εφαρμογή	Στόχος	Κλειδί	Τιμή
Λεξικό	Εύρεση ορισμού	Λέξη	Ορισμός
Ευρετήριο Βιβλίου	Εύρεση σχετικών σελίδων	Όρος	Λίστα με αριθμούς σελίδων
Κοινή Χρήση Αρχείου	Εύρεση mp3 για κατέβασμα	Όνομα τραγουδιού	Computer ID
Αναζήτηση στο Διαδίκτυο	Εύρεση σχετικών ιστοσελίδων στο web	Λέξη κλειδί	Λίστα ονόματα ιστοσελίδων
Διαχείριση Τραπεζικού λογαριασμού	Επεξεργασία συναλλαγών	Αριθμός λογαριασμού	Λεπτομέρειες συναλλαγής

Σειριακή Αναζήτηση – [1]

Το πρόβλημα. Δίνεται διάνυσμα Τ με n στοιχεία και αριθμός a. Ζητείται να βρεθεί, αν ένα από τα στοιχεία του Τ είναι ο αριθμός a.

a = -7

Τ =

10	4	-8	5	3	-19	-7	1	12

Σειριακή Αναζήτηση – [2]

Διατρέχουμε όλα τα στοιχεία του Τ και όταν ο αριθμός a βρεθεί, οι υπολογισμοί σταματούν.

Αλγόριθμος: linsearch
Είσοδος: T, n, a
Έξοδος: found, index

```
1  found ← 0
2  index ← 0
3  k ← 1
4  while k ≤ n and  found = 0
5        if T(k) = a
6              found ← 1
7              index ← k
8        k ← k+1
```

Σειριακή Αναζήτηση – [3]

Έστω Τ=[12 2 4 6 5 5] και a=5. Το n=6.

Αρχικά τίθεται found=0, index=0, k=1

Επανάληψη	k	Συνθήκη όσο (k ≤ 6 και found=0)	Συνθήκη αν (T(k)=5)	found	index
1η	1	αληθής	T(1)=5 (ψευδής)	0	0
2η	2	αληθής	T(2)=5 (ψευδής)	0	0
3η	3	αληθής	T(3)=5 (ψευδής)	0	0
4η	4	αληθής	T(4)=5 (ψευδής)	0	0
5η	5	αληθής	T(5)=5 (αληθής)	1	5
6η	6	ψευδής			

Σειριακή Αναζήτηση – [4]

Ο αλγόριθμος επιστρέφει index=5, found=1.

Παρατηρούμε ότι ο αλγόριθμος επιστρέφει την πρώτη τιμή T(k), την οποία βρίσκει ίση με το a. Στο παράδειγμά μας το T(5).

Μεταβλητές Εισόδου/Εξόδου

ΟΝΟΜΑ ΑΡΧΕΙΟΥ: 06-linsearch.c

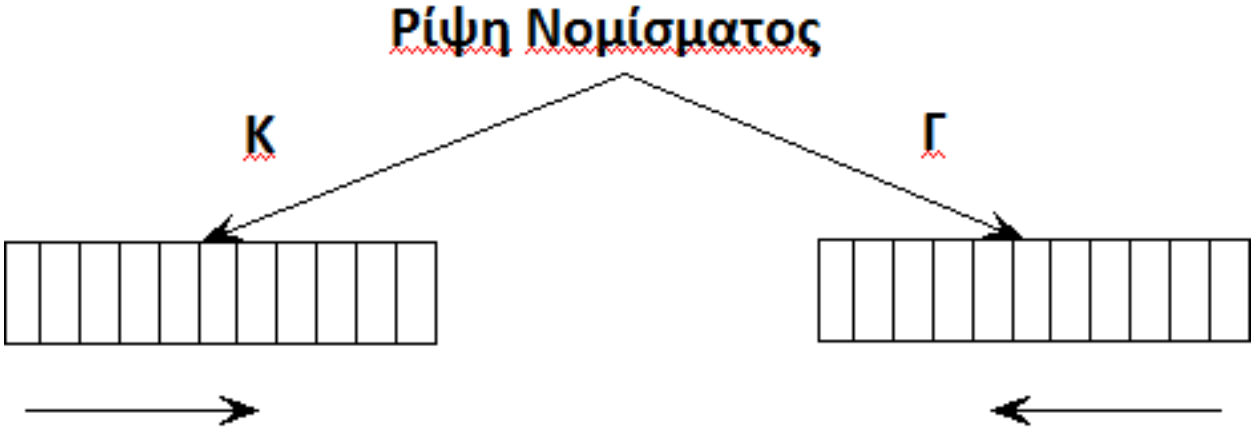
ΜΕΤΑΒΛΗΤΕΣ ΕΙΣΟΔΟΥ:
Τ – Πίνακας (τύπου double) στον οποίο θα πραγματοποιηθεί η αναζήτηση.
Ν – Το πλήθος των στοιχείων του Τ (τύπου int)
a – Ο αριθμός προς αναζήτηση (τύπου double).

ΜΕΤΑΒΛΗΤΕΣ ΕΞΟΔΟΥ:
Found – (τύπου int) αν found=0 ο αριθμός δεν βρέθηκε. Αν found=1 ο αριθμός βρέθηκε.
Index – Επιστρέφει τη θέση στην οποία βρέθηκε ο αριθμός a (τύπου int).

Filename: 06-linsearch.c

Σειριακή Αναζήτηση με τυχαία επιλογή – [1]

Έστω ότι ο πίνακας Τ **δεν είναι ταξινομημένος**.



Ανάλογα με το αποτέλεσμα της ρίψης θα ξεκινάει η αναζήτηση από τα αριστερά ή από τα δεξιά του πίνακα Τ.

Σειριακή Αναζήτηση με τυχαία επιλογή – [2]

Αλγόριθμος: linsearch_RandomChoice
Είσοδος: T, n, a
Έξοδος: found, index

```
1   found ← 0
2   index ← 0
3   Coin ← {0 ∨ 1}
4   if Coin==0 /* Κορώνα */
5           index ← 1
6           while n≥index and T(index)≠a
7                   index ← index+1
8           if index>n
9                   STOP
10          else found ← 1
11  else /* Γράμματα */
12          index ← n
13          while index≥1 and T(index)≠a
14                  index ← index-1
15          if 1>index
16                  STOP
17          else found ← 1
```

Σειριακή Αναζήτηση με τυχαία επιλογή – [3]

Έστω T=[12 2 4 6 5 5] και a=5. Το n=6.

Αρχικά τίθεται found←0, index←0, και Coin←0 (index←1)

Επανάληψη	**Συνθήκη όσο (n≥index και T(index)≠a)**	**Συνθήκη αν (index>6)**	**found**	**index**
1η	αληθής	ψευδής	0	1
2η	αληθής	ψευδής	0	2
3η	αληθής	ψευδής	0	3
4η	αληθής	ψευδής	0	4
5η	αληθής	ψευδής	1	5
6η	ψευδής			

Αναζήτηση με βήμα Κ – [1]

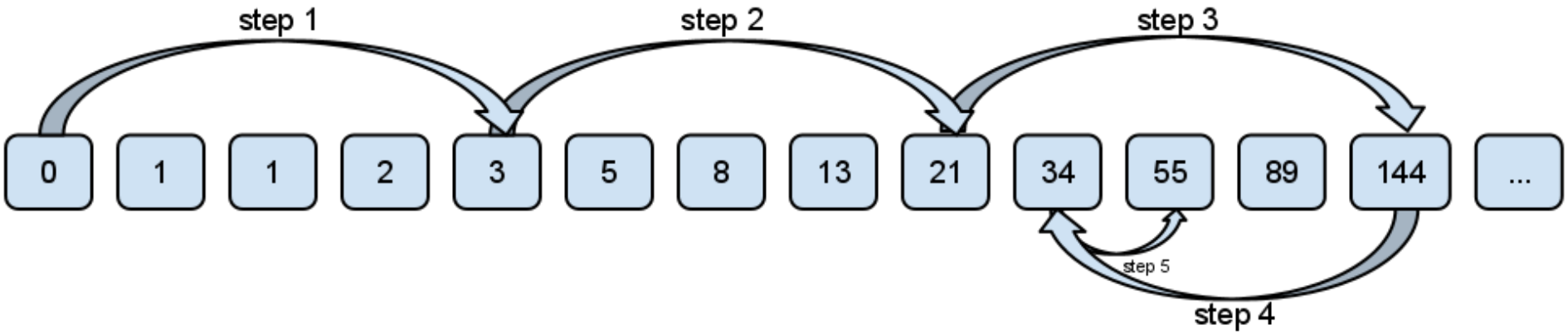
Έστω ότι ο πίνακας Τ **είναι ταξινομημένος σε αύξουσα διάταξη**.

Επιλογή ενός αριθμού Κ<n και εξέταση σε κάθε επανάληψη κατά πόσο το στοιχείο a είναι μεγαλύτερο από το επόμενο Κ-οστό στοιχείο του πίνακα Τ. Δηλαδή, θα εξεταστούν τα στοιχεία Τ(Κ), Τ(2Κ), Τ(3Κ), …

Αν βρεθεί ένα στοιχείο το οποίο είναι μεγαλύτερο ή ίσο του a, τότε χρησιμοποιείται η σειριακή αναζήτηση για τα Κ στοιχεία μεταξύ αυτού και του τελευταίου στοιχείου που εξετάστηκε.

Αναζήτηση με βήμα Κ – [2]

Αν ξεπεραστεί το τέλος του πίνακα Τ, εφαρμόζεται η σειριακή αναζήτηση για τα στοιχεία από το τελευταίο στοιχείο που εξετάστηκε μέχρι το τέλος του πίνακα Τ.



Αναζήτηση με βήμα K – [3]

Αλγόριθμος: Search_StepK *Δεδομένα*: T, i, j, K, a *Αποτελέσματα*: found, index	
1	index ← i+K-1
2	**όσο** j>index **και** a>T(index)
3	index ← index+K
4	**αν** j>index
5	j ← index
6	Σειριακή_Αναζήτηση(T, index-K+1, j, a)

Ποιες σχέσεις συνδέουν τα i, j, και K έτσι ώστε να δουλεύει ο παραπάνω αλγόριθμος?

$$j \geq i \geq 0, \quad j-i+1 \geq K \geq 1$$

Αναζήτηση με βήμα K (4)

Έστω T=[3 5 6 8 10 11 13 14 17 18 20 23 26] και

a=20, K=4, i=1, j=13, index=1+4-1=4

Επανάληψη	Συνθήκη όσο (j≥index και a>T(index))	Συνθήκη αν (j>index)	j	index
1η	αληθής	--	13	8
2η	αληθής	--	13	12
3η	ψευδής	αληθής	12	12

Κλήση της Σειριακή_αναζήτηση(T, index-K+1, j, a)

Σειριακή_αναζήτηση(T, 9, 12, 20)