ΠΑΡΑΡΤΗΜΑ

Α) ΜΕΤΑΤΡΟΠΕΣ ΜΕΤΑΞΥ ΔΟΜΩΝ ΕΠΑΝΑΛΗΨΗΣ

*Μετατροπή **ΑΡΧΗ ΕΠΑΝΑΛΗΨΗΣ** … **ΜΕΧΡΙΣ_ΟΤΟΥ** σε **ΟΣΟ... ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ** και αντιστρόφως*

1.

<εντολές>

Αρχή_επανάληψης

<εντολές>

Μέχρις_ότου **<συνθήκη>**

ΟΣΟ **Όχι <συνθήκη>** ΕΠΑΝΑΛΑΒΕ

<εντολές>

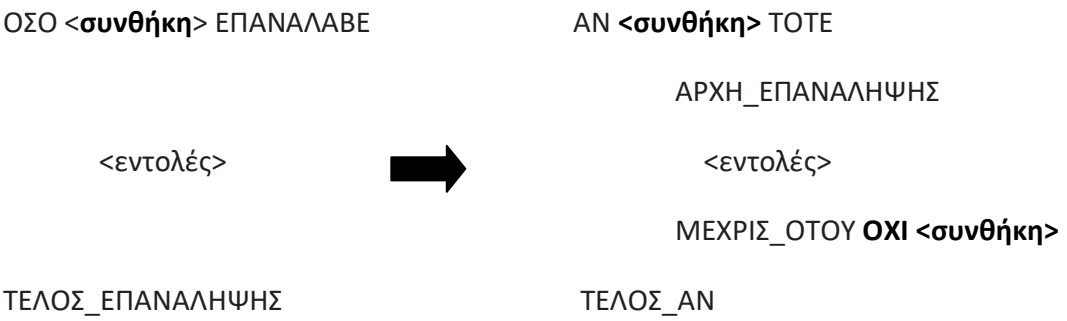
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

• Η εντολές στη δομή επανάληψης ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ…ΜΕΧΡΙΣ_ΟΤΟΥ, εκτελούνται όσο η **<συνθήκη>** μετά το Μέχρις_ότου είναι Ψευδής, ενώ στην δομή επανάληψης ΟΣΟ…ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ, εκτελούνται όσο η <συνθήκη> ανάμεσα στο ΟΣΟ και το ΕΠΑΝΑΛΑΒΕ είναι Αληθής. Γι' αυτό, κατά την μετατροπή από την μια δομή επανάληψης στην άλλη, αρκεί να γράφουμε την άρνηση της <συνθήκη> της πρώτης στη δεύτερη ή να προτάξουμε τον τελεστή ΟΧΙ στην συνθήκη.

• Επίσης, η ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ…ΜΕΧΡΙΣ_ΟΤΟΥ εκτελεί τουλάχιστον μια φορά όλες τις εντολές της, γι' αυτό όταν την μετατρέπουμε στην

ΟΣΟ…ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ πρέπει, όλες τις εντολές που περιέχει, να τις γράψουμε μια φορά πριν την ΟΣΟ…ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ και άλλη μια φορά μέσα σ' αυτήν.

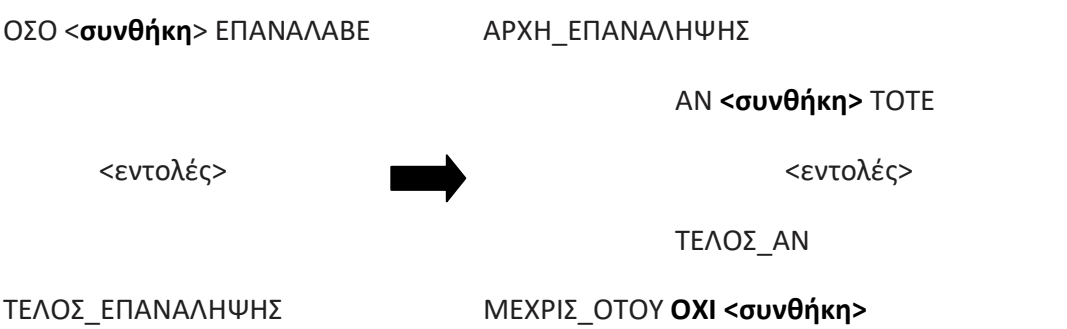
2.



• Η δομή επανάληψης ΟΣΟ…ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ εκτελείται όσο η <συνθήκη> ανάμεσα στο ΟΣΟ και το ΕΠΑΝΑΛΑΒΕ είναι Αληθής, ενώ η δομή επανάληψης ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ…ΜΕΧΡΙΣ_ΟΤΟΥ εκτελείται όσο η συνθήκη είναι μετά το Μέχρις_ότου Ψευδής. Γι' αυτό κατά την μετατροπή από την μια δομή στην άλλη αρκεί να γράψουμε την άρνηση της <συνθήκης> της πρώτης στη δεύτερη ή να προτάξουμε τον τελεστή ΟΧΙ στην συνθήκη.

• Επίσης, η ΟΣΟ…ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ μπορεί να μην εκτελεστεί καμία φορά σε αντίθεση με την ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ…ΜΕΧΡΙΣ_ΟΤΟΥ που θα εκτελεστεί τουλάχιστον μια φορά, γι' αυτό πριν την δομή ΑΡΧΗ_ ΕΠΑΝΑΛΗΨΗΣ…ΜΕΧΡΙΣ_ΟΤΟΥ θα χρησιμοποιηθεί μια εντολή ελέγχου, που αν ισχύει η <συνθήκη> θα εκτελεστεί η ΑΡΧΗ_ ΕΠΑΝΑΛΗΨΗΣ…ΜΕΧΡΙΣ_ΟΤΟΥ.

*Ως μη βέλτιστη λύση (η συνθήκη ελέγχεται δύο φορές), για την ίδια μετατροπή μπορεί να δοθεί και η παρακάτω:



*Μετατροπή από **ΓΙΑ…** σε **ΟΣΟ... ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ***

Περίπτωση τιμή1<= τιμή2 και β>0	*Περίπτωση τιμή1>= τιμή2 και β<0*
ΓΙΑ **<μεταβλητή>** ΑΠΟ **τιμή1** ΜΕΧΡΙ **τιμή2** ΜΕ_ΒΗΜΑ **β** <εντολές> ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ ↓ **<μεταβλητή> ← τιμή1** ΟΣΟ **<μεταβλητή> <=τιμή2** ΕΠΑΝΑΛΑΒΕ <εντολές> **<μεταβλητή> ← <μεταβλητή> + β** ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ	ΓΙΑ **<μεταβλητή>** ΑΠΟ **τιμή1** ΜΕΧΡΙ **τιμή2** ΜΕ_ΒΗΜΑ **β** <εντολές> ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ ↓ **<μεταβλητή> ← τιμή1** ΟΣΟ **<μεταβλητή> >= τιμή2** ΕΠΑΝΑΛΑΒΕ <εντολές> **<μεταβλητή> ← <μεταβλητή> + β** ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

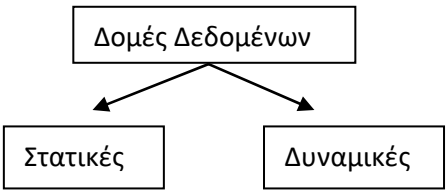
Πριν την εντολή ΟΣΟ…ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ εκχωρούμε στην <μεταβλητή> της ΓΙΑ…την αρχική τιμή δηλ. την τιμή1.

• Στη συνθήκη της ΟΣΟ…ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ συγκρίνουμε την <μεταβλητή> με την τιμή2. Η σύγκριση εξαρτάται από το βήμα αν είναι θετικό ή αρνητικό όπως βλέπουμε παραπάνω.

• Πριν το ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ της ΟΣΟ…αυξάνουμε την τιμή της μεταβλητής όσο είναι η τιμή του βήματος. Στη περίπτωση που το βήμα δεν υπάρχει, τότε η <μεταβλητή> αυξάνεται κατά 1.

• Η μετατροπή της ΟΣΟ…σε ΓΙΑ..., γίνεται μόνο στην περίπτωση που στην ΟΣΟ… είναι γνωστός ο αριθμός των επαναλήψεων, σε οποιαδήποτε άλλη περίπτωση δεν μετατρέπεται η ΟΣΟ…σε ΓΙΑ....

Β) ΣΤΑΤΙΚΕΣ ΚΑΙ ΔΥΝΑΜΙΚΕΣ ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ



Χαρακτηριστικά των Στατικών και Δυναμικών δομών δεδομένων

Στατικές δομές: Αποθηκεύονται σε συνεχόμενες θέσεις μνήμης και έχουν σταθερό μέγεθος, το οποίο καθορίζεται στην αρχή του προγράμματος. Οι στατικές δομές υλοποιούνται με πίνακες.

Δυναμικές δομές: Δεν αποθηκεύονται σε συνεχόμενες θέσεις μνήμης, δεν έχουν σταθερό μέγεθος, αλλά ο αριθμός των κόμβων τους αυξάνεται και μειώνεται, όταν στη δομή αντίστοιχα εισάγονται ή διαγράφονται δεδομένα. Το μέγεθος της μνήμης καθορίζεται κατά την στιγμή της εκτέλεσης του προγράμματος. Με δυναμικές δομές υλοποιούνται οι λίστες, τα δένδρα και οι γράφοι.

Πέρα από την διαφορά τους στην αποθήκευση στην κύρια μνήμη, οι μαθητές θα πρέπει να κατανοήσουν ότι για τις στατικές δομές (όπως αντιμετωπίζονται και στο βιβλίο) πρέπει να ορίζουν το μέγεθός τους, πριν από την έναρξη του προγράμματος, στο τμήμα δηλώσεων. Αντίθετα, για τις δυναμικές δομές μπορούμε να ορίζουμε και να τροποποιούμε το μέγεθος τους μέσα από το πρόγραμμα.

Πρέπει να τονιστεί, ότι μια δομή δεδομένων δεν είναι εγγενώς στατική ή δυναμική, αλλά εξαρτάται από τις δυνατότητες της γλώσσας προγραμματισμού που χρησιμοποιούμε και από τον τρόπο υλοποίησης της δομής στη γλώσσα αυτή. Οποιαδήποτε γλώσσα προγραμματισμού δεν υποστηρίζει όλες τις δομές δεδομένων, με τις σύγχρονες γλώσσες προγραμματισμού να υποστηρίζουν δυναμικές δομές δεδομένων.

Η γλώσσα προγραμματισμού ΓΛΩΣΣΑ, που χρησιμοποιείται στο βιβλίο, υποστηρίζει μόνο στατικές δομές.

Να σημειωθεί ότι οι πράξεις, των δομών της παραγράφου 3.2, αναφέρονται γενικά και αποκτούν πιο συγκεκριμένη σημασία, ανάλογα με τη δομή στην οποία αναφερόμαστε. Για παράδειγμα σε μια δομή πίνακα, κατά την πράξη της ταξινόμησης, δεν αναδιατάσσονται οι κόμβοι του αλλά το περιεχόμενο των κόμβων.

Επισημαίνεται ότι οι πίνακες στο βιβλίο της Β΄ τάξης αντιμετωπίζονται ως δυναμικές δομές, ενώ στο βιβλίο της Γ' τάξης ορίζονται ως στατικές δομές. Συνεπώς για τη Γ' τάξη και τη ΓΛΩΣΣΑ, η δομή του πίνακα είναι στατική και για να χρησιμοποιηθεί ένας πίνακας θα πρέπει να έχει πρώτα δηλωθεί, τόσο ο πίνακας, όσο και το μέγεθός του. Επίσης και οι δομές ουρά και στοίβα θεωρούνται στατικές δομές για τη ΓΛΩΣΣΑ, επειδή υλοποιούνται με πίνακες.

Γ) ΔΥΑΔΙΚΗ ΑΝΑΖΗΤΗΣΗ

Ο αλγόριθμος της δυαδικής αναζήτησης (binary search) εφαρμόζεται μόνο σε πίνακες που έχουν ταξινομημένα στοιχεία. Αν τα στοιχεία δεν είναι ταξινομημένα τότε δεν μπορεί να εφαρμοστεί.

Ο αλγόριθμος λειτουργεί ως εξής:

Βρίσκουμε το μεσαίο στοιχείο του ταξινομημένου πίνακα.

Εάν το προς αναζήτηση στοιχείο είναι ίσο με το μεσαίο στοιχείο τότε σταματάμε την αναζήτηση αφού το στοιχείο βρέθηκε

Εάν δεν βρέθηκε, τότε ελέγχουμε αν το στοιχείο που αναζητούμε είναι μικρότερο ή μεγαλύτερο από το μεσαίο στοιχείο του πίνακα. Αν είναι μικρότερο, περιορίζουμε την αναζήτηση στο πρώτο μισό του πίνακα (με την προϋπόθεση ότι τα στοιχεία είναι διατεταγμένα κατά αύξουσα σειρά), ενώ αν είναι μεγαλύτερο περιορίζουμε την αναζήτηση στο δεύτερο μισό του πίνακα.

Η διαδικασία αυτή λοιπόν επαναλαμβάνεται για το κατάλληλο πρώτο ή δεύτερο μισό πίνακα, μετά για το 1/4 του πίνακα κ.ο.κ. μέχρι, είτε να βρεθεί το στοιχείο, είτε να μην είναι δυνατό να χωρισθεί ο πίνακας περαιτέρω σε δύο νέα μέρη.

Αλγόριθμος δυαδικής αναζήτησης

αλγόριθμος Δυαδική_αναζήτηση !Α μονοδιάστατος πίνακας Ν θέσεων, S το στοιχείο που αναζητούμε

δεδομένα // N, A, S //

Left ← 1 ! αριστερό όριο

Right ← N ! δεξιό όριο

K ← 0 ! θέση του στοιχείου

F ← FALSE

όσο (Left<=Right) και (f=FALSE) **επανάλαβε**

M ← (Left+Right) div 2

αν A[M]=S **τότε**

K← M;

F ← TRUE;

αλλιώς

αν A[M]<S **τότε**

Left ← M+1;

αλλιώς

Right← M-1;

Τέλος_αν

Τέλος_αν

Τέλος_επανάληψης

Αν F = TRUE **τότε**

Εμφάνισε "Το στοιχείο,", S , "υπάρχει στη θέση:", M

Αλλιώς

Εμφάνισε "Το στοιχείο,", S , " δεν υπάρχει στον πίνακα"

Τέλος_αν

Η δυαδική αναζήτηση να διδαχθεί ως άσκηση και να υλοποιηθεί με πρόγραμμα, όπως παρακάτω σε ταξινομημένο πίνακα 20 θέσεων. Πέρα από το τμήμα δηλώσεων, το πρόγραμμα έχει ένα επιπλέον τμήμα για το "γέμισμα" του πίνακα με στοιχεία (υποθέτουμε ότι ο πίνακας γεμίζει με σωστά ταξινομημένα στοιχεία σε αύξουσα σειρά).

```
ΠΡΟΓΡΑΜΜΑ δυαδική_αναζήτηση
ΜΕΤΑΒΛΗΤΕΣ
 ΑΚΕΡΑΙΕΣ: A[20], Left, Right, M, k, S, i

 ΛΟΓΙΚΕΣ: f
 ΑΡΧΗ

        ΓΡΑΨΕ 'Οι αριθμοί που θα δοθούν πρέπει να είναι ταξινομημένοι κατά αύξουσα
τάξη'
        ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 20
              ΓΡΑΨΕ 'Δώσε το', i, ' στοιχείο του πίνακα'
              ΔΙΑΒΑΣΕ A[i]
        ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

        ΓΡΑΨΕ 'Δωσε τιμή για αναζήτηση: '
        ΔΙΑΒΑΣΕ S
         Left <- 1
        Right <- 20
         k <- 0
        f <- ΨΕΥΔΗΣ
        ΟΣΟ (Left <= Right) ΚΑΙ (f = ΨΕΥΔΗΣ) ΕΠΑΝΑΛΑΒΕ
              M <- (Left + Right) DIV 2
              ΑΝ A[M] = S ΤΟΤΕ
                    k <- M
                    f <- ΑΛΗΘΗΣ
              ΑΛΛΙΩΣ
                    ΑΝ A[M] < S ΤΟΤΕ
                          Left <- M + 1
                     ΑΛΛΙΩΣ
                          Right <- M - 1
                    ΤΕΛΟΣ_ΑΝ
              ΤΕΛΟΣ_ΑΝ
        ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
        ΑΝ f = ΑΛΗΘΗΣ ΤΟΤΕ
              ΓΡΑΨΕ "Το στοιχείο,", S, "υπάρχει στη θέση:", M
        ΑΛΛΙΩΣ
              ΓΡΑΨΕ "Το στοιχείο,", S, " δεν υπάρχει στον πίνακα"
        ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ δυαδική_αναζήτηση
```

Παράδειγμα

Δίνεται ο πίνακας

1	2	5	8	9	15	22	27	35	37	38	40	43	45	47

Αναζήτηση του στοιχείου 38 (υπάρχει στον πίνακα)

Βήμα 1	1	2	5	8	9	15	22	27	35	37	38	40	43	45	47
Βήμα 2	1	2	5	8	9	15	22	27	35	37	38	40	43	45	47
Βήμα 3	1	2	5	8	9	15	22	27	35	37	38	40	43	45	47
Βήμα 4	1	2	5	8	9	15	22	27	35	37	38	40	43	45	47

Με κίτρινο σημειώνεται το στοιχείο του πίνακα που εξετάζεται (στο μέσον)

Με πράσινο σημειώνεται το τμήμα του πίνακα που απομένει για αναζήτηση

Με κόκκινο σημειώνεται το τμήμα του πίνακα που έχει αποκλειστεί

(Τα ίδια χρώματα χρησιμοποιούνται και στο επόμενο παράδειγμα)

Αναζήτηση του στοιχείου 39 (δεν υπάρχει στον πίνακα)

Βήμα 1	1	2	5	8	9	15	22	27	35	37	38	40	43	45	47
Βήμα 2	1	2	5	8	9	15	22	27	35	37	38	40	43	45	47
Βήμα 3	1	2	5	8	9	15	22	27	35	37	38	40	43	45	47
Βήμα 4	1	2	5	8	9	15	22	27	35	37	38	40	43	45	47
Βήμα 5	1	2	5	8	9	15	22	27	35	37	38	40	43	45	47

Αριθμός συγκρίσεων στη δυαδική αναζήτηση

Στοιχεία Ν	**Συγκρίσεις**
10	4
100	7
1.000	10
10.000	14
100.000	17
1.000.000	20
10.000.000	24
100.000.000	27
1.000.000.000	30

Δ) ΤΑΞΙΝΟΜΗΣΗ ΜΕ ΕΠΙΛΟΓΗ (SELECTION SORT)

Η ταξινόμηση με επιλογή (selection sort), αποτελεί βασικό τρόπο ταξινόμησης, που υλοποιείται σε ένα μονοδιάστατο πίνακα σε τρία βήματα:

1. Επιλογή του ελάχιστου στοιχείου
2. Ανταλλαγή του ελάχιστου με το πρώτο στοιχείο
3. Επανάληψη των βημάτων 1 και 2 για τα υπόλοιπα στοιχεία του πίνακα

Ο Αλγόριθμος ταξινόμησης με επιλογή είναι ο παρακάτω.

Αλγόριθμος Selection_Sort

Δεδομένα // table, n //

Για i **από** 1 **μέχρι** n-1

k ← i

x ← table[i]

Για j από i+1 **μέχρι** n

Αν x > table[j] **Τότε**

k ← j

x ← table[j]

Τέλος_Επανάληψης

table[k] ← table[i]

table[i] ← x

Τέλος_ επανάληψης

Η υλοποίηση του αλγορίθμου ταξινόμησης με επιλογή, να διδαχθεί ως άσκηση και να υλοποιηθεί με πρόγραμμα όπως παρακάτω. Πέρα από το τμήμα δηλώσεων, το πρόγραμμα έχει δύο επιπλέον τμήματα, ένα τμήμα για το "γέμισμα" του πίνακα με στοιχεία και ένα τμήμα για την εκτύπωση του ταξινομημένου πίνακα.

```
ΠΡΟΓΡΑΜΜΑ Selection_Sort
ΜΕΤΑΒΛΗΤΕΣ
        ΑΚΕΡΑΙΕΣ: A[20], K1, x, i, j
ΑΡΧΗ
      ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 20
            ΓΡΑΨΕ 'Δώσε το', i, ' στοιχείο του πίνακα'
```

```
            ΔΙΑΒΑΣΕ A[i]
        ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

        ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 19
            K1 <- i
            x <- A[i]
            ΓΙΑ j ΑΠΟ i + 1 ΜΕΧΡΙ 20
                ΑΝ x > A[j] ΤΟΤΕ
                    K1 <- j
                    x <- A[j]
                ΤΕΛΟΣ_ΑΝ
            ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
            A[K1] <- A[i]
            A[i] <- x
        ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

        ΓΡΑΨΕ 'Εκτύπωση με ταξινομημένα τα στοιχεία'
        ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 20
            ΓΡΑΨΕ A[i]
        ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ Selection_Sort
```

Παράδειγμα

Αν υποθέσουμε ότι έχουμε το πίνακα Α[8] με στοιχεία τους αριθμούς 46, 55, 12, 42, 94, 18, 06, 67. Δηλαδή σε μορφή μονοδιάστατου πίνακα:

46	55	12	42	94	18	06	67

τότε παρακάτω φαίνεται πως μετακινούνται τα στοιχεία με τον αλγόριθμο SelectionSort

Βήμα 1 (εύρεση του ελάχιστου των στοιχείων και ανταλλαγή με το πρώτο)

46	55	12	42	94	18	06	67

Βήμα 2 (επανάληψη της ανωτέρω διαδικασίας αλλά στο τμήμα του πίνακα από το δεύτερο στοιχείο και κάτω)

06	55	12	42	94	18	46	67

Βήμα 3 (επανάληψη της ανωτέρω διαδικασίας αλλά στο τμήμα του πίνακα από το τρίτο στοιχείο και κάτω)

06	12	55	42	94	18	46	67

Βήμα 4 (επανάληψη της ανωτέρω διαδικασίας αλλά στο τμήμα του πίνακα από το τέταρτο στοιχείο και κάτω)

06	12	18	42	94	55	46	67

Βήμα 5 (επανάληψη της ανωτέρω διαδικασίας αλλά στο τμήμα του πίνακα από το πέμπτο στοιχείο και κάτω)

06	12	18	42	94	55	46	67

Βήμα 6 (επανάληψη της ανωτέρω διαδικασίας αλλά στο τμήμα του πίνακα από το έκτο στοιχείο και κάτω)

06	12	18	42	46	55	94	67

Βήμα 7 (επανάληψη της ανωτέρω διαδικασίας αλλά στο τμήμα του πίνακα από το έβδομο στοιχείο και κάτω)

06	12	18	42	46	55	94	67

Τελική μορφή ταξινομημένου πίνακα (δεν χρειάζεται $8^{η}$ επανάληψη σύγκρισης, αφού όταν απομένουν δύο μόνο κελιά και στο πρώτο θέσεις τον μικρότερο αριθμό, τότε στο δεύτερο αναγκαστικά τίθεται ο μεγαλύτερος)

06	12	18	42	46	55	67	94