Δομές Δεδομένων και Αλγόριθμοι

Χρήστος Γκόγκος

ΤΕΙ Ηπείρου, Τμήμα Μηχανικών Πληροφορικής ΤΕ

Χειμερινό Εξάμηνο 2014-2015
(Παρουσίαση 5)

Απόδοση προγραμμάτων

- Συχνά χρειάζεται να εκτιμηθεί η απόδοση ενός προγράμματος δηλαδή πόσο γρήγορα εκτελείται και πόσους πόρους (μνήμη, αποθηκευτικό χώρο σε δευτερεύουσα μνήμη, εύρος ζώνης κ.α.) απαιτεί
- Μπορεί να χρειαστεί να συγκριθεί ένα πρόγραμμα με ένα άλλο πρόγραμμα που επιλύει το ίδιο πρόβλημα και να επιλεγεί το καλύτερο
- Η απόδοση ενός προγράμματος μπορεί να εξαρτάται από το είδος του υπολογιστή στο οποίο εκτελείται
- Η απόδοση ενός προγράμματος συνήθως εξαρτάται από το μέγεθος αλλά και την μορφή της εισόδου
- Η απόδοση ενός προγράμματος μπορεί να μετρηθεί πειραματικά ή θεωρητικά

Πειραματική μέτρηση απόδοσης προγράμματος

Πειραματική - εμπειρική μέτρηση

Καταστρώνεται ένα πείραμα κατά το οποίο το πρόγραμμα εκτελείται για διάφορα μεγέθη εισόδου και καταγράφεται ο χρόνος εκτέλεσης σε κάθε περίπτωση. Τα αποτελέσματα συνήθως απεικονίζονται διαγραμματικά και δείχνουν την συμπεριφορά του προγράμματος καθώς το μέγεθος του προβλήματος αυξάνεται.

Αν υπάρχουν περισσότερα του ενός υποψήφια εναλλακτικά προγράμματα μπορούν να προκύψουν συγκριτικά στοιχεία που να υποδεικνύουν το καταλληλότερο ανάμεσα τους

Αναλυτική μέτρηση απόδοσης προγράμματος

Θεωρητικό μοντέλο Η/Υ

Το πρόγραμμα εκτελείται σε ένα θεωρητικό μοντέλο υπολογιστή για το οποίο υποθέτουμε ότι οι βασικές λειτουργίες (αριθμητικές πράξεις, σύγκριση, μεταφορά) απαιτούν κάποιον χρόνο που μπορεί να θεωρηθεί σταθερός για κάθε μια στοιχειώδη ενέργεια.

Πολυπλοκότητα χώρου / χρόνου

Η πολυπλοκότητα χώρου αφορά τον απαιτούμενο χώρο που συνεπάγεται η εκτέλεση του προγράμματος
Ο απαιτούμενος χώρος ενός προγράμματος αποτελείται από:

- Χώρο εντολών
- Χώρο δεδομένων
- Χώρο στοίβας περιβάλλοντος εκτέλεσης

Η πολυπλοκότητα χρόνου έχει να κάνει με την καταμέτρηση των λειτουργιών που γίνονται κατά την εκτέλεση του προγράμματος

Αυτό που ενδιαφέρει είναι **ο ρυθμός αύξησης** του χώρου ή του χρόνου που απαιτεί η εκτέλεση ενός προγράμματος καθώς το μέγεθος του προβλήματος αυξάνεται.

Αριθμός εκτελέσεων εντολών σε μια απλή for

```
for (int i = 0; i < 10; i++) {
    command1
}

...

for (int i = a; i < b; i++) {
    command2
}
```

Στο πρώτο τμήμα κώδικα η συνθήκη i<10 θα εξεταστεί 11 φορές και η εντολή command1 θα εκτελεστεί 10 φορές
Στο δεύτερο τμήμα κώδικα η συνθήκη i<b θα εξεταστεί $b - a + 1$ φορές και η εντολή command2 θα εκτελεστεί $b - a$ φορές

Ανάλυση χρόνου εκτέλεσης find_max (1/2)

Το πρόβλημα που επιλύει η find_max

Βρίσκει τη μεγαλύτερη τιμή ενός πίνακα n θέσεων

```
template<class T> T find_max(T a[], int  n) {
    T max = a[0];                    //  c1
    for  (int  i  = 1;  i  < n;  i++) {  //  c2
        if  (a[i] > max)                 //  c3
            max = a[i];              //  c4
    }
    return  max;                     //  c5
}
```

c1 έως c5 είναι οι χρόνοι εκτέλεσης των εντολών κάθε φορά που εκτελούνται

Ανάλυση χρόνου εκτέλεσης find_max (2/2)

```
template<class T> T find_max(T a[], int
    n) {
  T max = a[0];                    // c1
  for (int i = 1; i < n; i++) {    // c2
    if (a[i] > max)                // c3
      max = a[i];                  // c4
  }
  return max;                      // c5
}
```

- Η εντολή με χρόνο εκτέλεσης $c1$ εκτελείται 1 φορά
- Η εντολή με χρόνο εκτέλεσης $c2$ εκτελείται n φορές
- Η εντολή με χρόνο εκτέλεσης $c3$ εκτελείται n-1 φορές
- Η εντολή με χρόνο εκτέλεσης $c4$ εκτελείται n-1 φορές
- Η εντολή με χρόνο εκτέλεσης $c5$ εκτελείται 1 φορά

Εξετάζουμε το χρόνο της χειρότερης περίπτωσης

$$T(n) = c1 + c2n + c3(n-1) + c4(n-1) + c5$$
$$= (c2 + c3 + c4)n + (c1 - c3 - c4 + c5) \Rightarrow an + b$$
$$a = (c2 + c3 + c4), b = (c1 - c3 - c4 + c5)$$

Άρα ο χρόνος εκτέλεσης $T(n)$ είναι $an + b$ δηλαδή περιγράφεται από μια γραμμική συνάρτηση του μεγέθους n της ακολουθίας εισόδου

Ο αλγόριθμος ταξινόμησης με επιλογή (selection_sort)

Περιγραφή λειτουργίας

Ο αλγόριθμος εντοπίζει το μικρότερο στοιχείο του πίνακα και το αντιμεταθέτει με το πρώτο στοιχείο του. Στη συνέχεια εντοπίζει το μικρότερο στοιχείο από το δεύτερο μέχρι το τελευταίο στοιχείο του πίνακα και το αντιμεταθέτει με το στοιχείο που βρίσκεται στην δεύτερη θέση. Η διαδικασία συνεχίζεται εντοπίζοντας στην εναπομείνασα ακολουθία κάθε φορά το μικρότερο στοιχείο και κάνοντας αντιμετάθεση με το αριστερότερο στοιχείο της ακολουθίας που εξετάστηκε για εύρεση του μικρότερου.

```
template<class T> void selection_sort(T a [],  int  n)  {
    for  ( int  i  = 0;  i  < n  –  1;  i++) {     // c1
        int  pmin = i;                    // c2
        for  ( int  j  = i  + 1;  j  < n;  j++)   // c3
            if  (a[j ]  < a[pmin])            // c4
                pmin = j;                 // c5
            swap(a[i],  a[pmin]);         // c6
    }
}
```

c1 έως c6 είναι οι χρόνοι εκτέλεσης των εντολών κάθε φορά που εκτελούνται

Παράδειγμα εκτέλεσης του αλγορίθμου selection_sort

Έστω η ακολουθία τιμών: 16, 22, 11, 9, 7, 18

Η ταξινόμηση προχωρά ως εξής:

$\rightarrow$ |16, 22, 11, 9, 7, 18

Παράδειγμα εκτέλεσης του αλγορίθμου selection_sort

Έστω η ακολουθία τιμών: 16, 22, 11, 9, 7, 18

Η ταξινόμηση προχωρά ως εξής:

$\rightarrow$ |16, 22, 11, 9, 7, 18

$\rightarrow$ 7,| 22, 11, 9, 16, 18

Παράδειγμα εκτέλεσης του αλγορίθμου selection_sort

Έστω η ακολουθία τιμών: 16, 22, 11, 9, 7, 18

Η ταξινόμηση προχωρά ως εξής:

$\rightarrow$ |16, 22, 11, 9, 7, 18

$\rightarrow$ 7,| 22, 11, 9, 16, 18

$\rightarrow$ 7, 9,| 11, 22, 16, 18

Παράδειγμα εκτέλεσης του αλγορίθμου selection_sort

Έστω η ακολουθία τιμών: 16, 22, 11, 9, 7, 18

Η ταξινόμηση προχωρά ως εξής:

$\rightarrow$ |16, 22, 11, 9, 7, 18

$\rightarrow$ 7,| 22, 11, 9, 16, 18

$\rightarrow$ 7, 9,| 11, 22, 16, 18

$\rightarrow$ 7, 9, 11,| 22, 16, 18

Παράδειγμα εκτέλεσης του αλγορίθμου selection_sort

Έστω η ακολουθία τιμών: 16, 22, 11, 9, 7, 18

Η ταξινόμηση προχωρά ως εξής:

$\rightarrow$ |16, 22, 11, 9, 7, 18

$\rightarrow$ 7,| 22, 11, 9, 16, 18

$\rightarrow$ 7, 9,| 11, 22, 16, 18

$\rightarrow$ 7, 9, 11,| 22, 16, 18

$\rightarrow$ 7, 9, 11, 16,| 22, 18

Παράδειγμα εκτέλεσης του αλγορίθμου selection_sort

Έστω η ακολουθία τιμών: 16, 22, 11, 9, 7, 18

Η ταξινόμηση προχωρά ως εξής:

$\rightarrow$ |16, 22, 11, 9, 7, 18

$\rightarrow$ 7,| 22, 11, 9, 16, 18

$\rightarrow$ 7, 9,| 11, 22, 16, 18

$\rightarrow$ 7, 9, 11,| 22, 16, 18

$\rightarrow$ 7, 9, 11, 16,| 22, 18

$\rightarrow$ 7, 9, 11, 16, 18,| 22

Ανάλυση χρόνου εκτέλεσης της ταξινόμησης με επιλογή selection_sort (1/2)

```
template<class T> void
    selection_sort(T a[], int n) {
  for (int i = 0; i < n - 1;
       i++) {     // c1
    int pmin = i;
         // c2
    for (int j = i + 1; j < n;
         j++)   // c3
      if (a[j] < a[pmin])
            // c4
        pmin = j;
            // c5
      swap(a[i], a[pmin]);
          // c6
  }
}
```

- Η εντολή με χρόνο εκτέλεσης $c1$ εκτελείται n φορές
- Η εντολή με χρόνο εκτέλεσης $c2$ εκτελείται $n-1$ φορές
- Η εντολή με χρόνο εκτέλεσης $c3$ εκτελείται $\sum_{i=0}^{n-2} n-(i+1)+1 = \sum_{i=0}^{n-2} n-i$ φορές
- Η εντολή με χρόνο εκτέλεσης $c4$ εκτελείται $\sum_{i=0}^{n-2} n-(i+1) = \sum_{i=0}^{n-2} n-i-1$ φορές
- Η εντολή με χρόνο εκτέλεσης $c5$ εκτελείται $\sum_{i=0}^{n-2} n-(i+1) = \sum_{i=0}^{n-2} n-i-1$ φορές
- Η εντολή με χρόνο εκτέλεσης $c6$ εκτελείται $n-1$ φορές

$$\sum_{i=0}^{n-2} n-i = n+(n-1)+(n-2)+\ldots+2 = \frac{n(n+1)}{2} - 1 = \frac{n^2}{2} + \frac{n}{2} - 1$$

$$\sum_{i=0}^{n-2} n-i-1 = (n-1)+(n-2)+\ldots+1 = \frac{(n-1)n}{2} = \frac{n^2}{2} - \frac{n}{2}$$

$$T(n) = c1n + c2(n-1) + c3(\tfrac{n^2}{2} + \tfrac{n}{2} - 1) + c4(\tfrac{n^2}{2} - \tfrac{n}{2}) + c5(\tfrac{n^2}{2} - \tfrac{n}{2}) + c6(n-1) = an^2 + bn + c$$

$$a = \tfrac{c3+c4+c5}{2}, b = (c1 + c2 + \tfrac{c3}{2} - \tfrac{c4}{2} - \tfrac{c5}{2} + c6), c = (-c2 - c3 - c6)$$

Ανάλυση χρόνου εκτέλεσης της ταξινόμησης με επιλογή selection_sort (2/2)

Συνάρτηση χρόνου εκτέλεσης

Άρα ο χρόνος εκτέλεσης $T(n)$ είναι $an^2 + bn + c$ δηλαδή περιγράφεται από μια τετραγωνική συνάρτηση του μεγέθους n της ακολουθίας εισόδου

Ο αλγόριθμος ταξινόμησης με εισαγωγή (insertion_sort)

Περιγραφή λειτουργίας

Ξεκινώντας από το αριστερό άκρο κάθε τιμή τοποθετείται στην σωστή θέση σε σχέση με αυτές που βρίσκονται αριστερά της

```
template<class T>
void insertion_sort(T a[], int n) {
    for (int i = 1; i < n; i++) {    // c1
        T key = a[i];                // c2
        int j = i - 1;               // c3
        while ((j >= 0) && (key < a[j])) {  // c4
            a[j + 1] = a[j];         // c5
            j--;                     // c6
        }
        a[j + 1] = key;              // c7
    }
}
```

c1 έως c7 είναι οι χρόνοι εκτέλεσης των εντολών κάθε φορά που εκτελούνται

Παράδειγμα εκτέλεσης του αλγορίθμου insertion_sort

Έστω η ακολουθία τιμών: 16, 22, 11, 9, 7, 18

Η ταξινόμηση με εισαγωγή προχωρά ως εξής:

$\rightarrow$ |16, 22, 11, 9, 7, 18

Παράδειγμα εκτέλεσης του αλγορίθμου insertion_sort

Έστω η ακολουθία τιμών: 16, 22, 11, 9, 7, 18

Η ταξινόμηση με εισαγωγή προχωρά ως εξής:

$\rightarrow$ |16, 22, 11, 9, 7, 18

$\rightarrow$ 16,| 22, 11, 9, 7, 18

Παράδειγμα εκτέλεσης του αλγορίθμου insertion_sort

Έστω η ακολουθία τιμών: 16, 22, 11, 9, 7, 18

Η ταξινόμηση με εισαγωγή προχωρά ως εξής:

$\rightarrow$ |16, 22, 11, 9, 7, 18

$\rightarrow$ 16,| 22, 11, 9, 7, 18

$\rightarrow$ 16, 22,| 11, 9, 7, 18

Παράδειγμα εκτέλεσης του αλγορίθμου insertion_sort

Έστω η ακολουθία τιμών: 16, 22, 11, 9, 7, 18

Η ταξινόμηση με εισαγωγή προχωρά ως εξής:

$\rightarrow$ |16, 22, 11, 9, 7, 18

$\rightarrow$ 16,| 22, 11, 9, 7, 18

$\rightarrow$ 16, 22,| 11, 9, 7, 18

$\rightarrow$ 11, 16, 22,| 9, 7, 18

Παράδειγμα εκτέλεσης του αλγορίθμου insertion_sort

Έστω η ακολουθία τιμών: 16, 22, 11, 9, 7, 18

Η ταξινόμηση με εισαγωγή προχωρά ως εξής:

$\rightarrow$ |16, 22, 11, 9, 7, 18

$\rightarrow$ 16,| 22, 11, 9, 7, 18

$\rightarrow$ 16, 22,| 11, 9, 7, 18

$\rightarrow$ 11, 16, 22,| 9, 7, 18

$\rightarrow$ 9, 11, 16, 22,| 7, 18

Παράδειγμα εκτέλεσης του αλγορίθμου insertion_sort

Έστω η ακολουθία τιμών: 16, 22, 11, 9, 7, 18

Η ταξινόμηση με εισαγωγή προχωρά ως εξής:

$\rightarrow$ |16, 22, 11, 9, 7, 18

$\rightarrow$ 16,| 22, 11, 9, 7, 18

$\rightarrow$ 16, 22,| 11, 9, 7, 18

$\rightarrow$ 11, 16, 22,| 9, 7, 18

$\rightarrow$ 9, 11, 16, 22,| 7, 18

$\rightarrow$ 7, 9, 11, 16, 22,| 18

Παράδειγμα εκτέλεσης του αλγορίθμου insertion_sort

Έστω η ακολουθία τιμών: 16, 22, 11, 9, 7, 18

Η ταξινόμηση με εισαγωγή προχωρά ως εξής:

$\rightarrow$ |16, 22, 11, 9, 7, 18

$\rightarrow$ 16,| 22, 11, 9, 7, 18

$\rightarrow$ 16, 22,| 11, 9, 7, 18

$\rightarrow$ 11, 16, 22,| 9, 7, 18

$\rightarrow$ 9, 11, 16, 22,| 7, 18

$\rightarrow$ 7, 9, 11, 16, 22,| 18

$\rightarrow$ 7, 9, 11, 16, 18, 22

Ανάλυση χρόνου εκτέλεσης της ταξινόμησης με εισαγωγή

```
template<class T>
void insertion_sort (T a [], int n)
    {
  for (int i = 1; i < n; i++) {
        // c1
      T key = a[i ];
            // c2
      int j = i - 1;
            // c3
      while ((j >= 0) && (key <
           a[j ])) { // c4
         a[j + 1] = a[j ];   // c5
         j--;                // c6
      }
      a[j + 1] = key;
            // c7
  }
}
```

- Η εντολή με χρόνο εκτέλεσης $c1$ εκτελείται n φορές
- Η εντολή με χρόνο εκτέλεσης $c2$ εκτελείται $n-1$ φορές
- Η εντολή με χρόνο εκτέλεσης $c3$ εκτελείται $n-1$ φορές
- Η εντολή με χρόνο εκτέλεσης $c4$ εκτελείται $\sum_{i=1}^{n-1} i+1$ φορές
- Η εντολή με χρόνο εκτέλεσης $c5$ εκτελείται $\sum_{i=1}^{n-1} i$ φορές
- Η εντολή με χρόνο εκτέλεσης $c6$ εκτελείται $\sum_{i=1}^{n-1} i$ φορές
- Η εντολή με χρόνο εκτέλεσης $c7$ εκτελείται $n-1$ φορές

$$\sum_{i=1}^{n-1} i+1 = \sum_{i=2}^{n} i = \frac{n(n+1)}{2} - 1 = \frac{n^2}{2} + \frac{n}{2} - 1 \quad \sum_{i=1}^{n-1} i = \frac{(n-1)n}{2} = \frac{n^2}{2} - \frac{n}{2}$$

$$T(n) = c1n + c2(n-1) + c3(n-1) + c4 * (\frac{n^2}{2} + \frac{n}{2} - 1) + c5(\frac{n^2}{2} - \frac{n}{2}) + c6(\frac{n^2}{2} - \frac{n}{2}) + c7(n-1)$$

$$= an^2 + bn + c, a = \frac{c4+c5+c6}{2}, b = (c1 + c2 + c3 + \frac{c4}{2} - \frac{c5}{2} - \frac{c6}{2} + c7), c = (-c2 - c3 - c4 - c7)$$

Ο αλγόριθμος ταξινόμησης φυσαλίδας (bubble_sort)

Περιγραφή λειτουργίας

Γειτονικά στοιχεία συγκρίνονται από το τέλος προς την αρχή και αν δεν είναι στην ορθή σειρά αντιμετατίθενται. Η διαδικασία επαναλαμβάνεται μέχρι να ταξινομηθούν όλα τα στοιχεία του πίνακα.

```
template<class T>
void bubble_sort(T a[], int n) {
    for (int i = 1; i < n; i++)
        for (int j = n - 1; j >= i; j--)
            if (a[j - 1] > a[j])
                swap(a[j], a[j - 1]);
}
```

Παράδειγμα εκτέλεσης του αλγορίθμου bubble_sort

Έστω η ακολουθία τιμών: 16, 22, 11, 9, 7, 18
Η ταξινόμηση φυσαλίδας προχωρά ως εξής:
$\rightarrow$ 16, 22, 11, 9, 7, 18

Παράδειγμα εκτέλεσης του αλγορίθμου bubble_sort

Έστω η ακολουθία τιμών: 16, 22, 11, 9, 7, 18
Η ταξινόμηση φυσαλίδας προχωρά ως εξής:
$\rightarrow$ 16, 22, 11, 9, 7, 18
$\rightarrow$ 16, 22, 11, 7, 9, 18

Παράδειγμα εκτέλεσης του αλγορίθμου bubble_sort

Έστω η ακολουθία τιμών: 16, 22, 11, 9, 7, 18
Η ταξινόμηση φυσαλίδας προχωρά ως εξής:
$\rightarrow$ 16, 22, 11, 9, 7, 18
$\rightarrow$ 16, 22, 11, 7, 9, 18
$\rightarrow$ 16, 22, 7, 11, 9, 18

Παράδειγμα εκτέλεσης του αλγορίθμου bubble_sort

Έστω η ακολουθία τιμών: 16, 22, 11, 9, 7, 18
Η ταξινόμηση φυσαλίδας προχωρά ως εξής:

$\rightarrow$ 16, 22, 11, 9, 7, 18
$\rightarrow$ 16, 22, 11, 7, 9, 18
$\rightarrow$ 16, 22, 7, 11, 9, 18
$\rightarrow$ 16, 7, 22, 11, 9, 18

Παράδειγμα εκτέλεσης του αλγορίθμου bubble_sort

Έστω η ακολουθία τιμών: 16, 22, 11, 9, 7, 18
Η ταξινόμηση φυσαλίδας προχωρά ως εξής:

$\rightarrow$ 16, 22, 11, 9, 7, 18
$\rightarrow$ 16, 22, 11, 7, 9, 18
$\rightarrow$ 16, 22, 7, 11, 9, 18
$\rightarrow$ 16, 7, 22, 11, 9, 18
$\rightarrow$ 7, 16, 22, 11, 9, 18

Παράδειγμα εκτέλεσης του αλγορίθμου bubble_sort

Έστω η ακολουθία τιμών: 16, 22, 11, 9, 7, 18
Η ταξινόμηση φυσαλίδας προχωρά ως εξής:

$\rightarrow$ 16, 22, 11, 9, 7, 18
$\rightarrow$ 16, 22, 11, 7, 9, 18
$\rightarrow$ 16, 22, 7, 11, 9, 18
$\rightarrow$ 16, 7, 22, 11, 9, 18
$\rightarrow$ 7, 16, 22, 11, 9, 18
$\rightarrow$ 7, 16, 22, 9, 11, 18

Παράδειγμα εκτέλεσης του αλγορίθμου bubble_sort

Έστω η ακολουθία τιμών: 16, 22, 11, 9, 7, 18
Η ταξινόμηση φυσαλίδας προχωρά ως εξής:

$\rightarrow$ 16, 22, 11, 9, 7, 18
$\rightarrow$ 16, 22, 11, 7, 9, 18
$\rightarrow$ 16, 22, 7, 11, 9, 18
$\rightarrow$ 16, 7, 22, 11, 9, 18
$\rightarrow$ 7, 16, 22, 11, 9, 18
$\rightarrow$ 7, 16, 22, 9, 11, 18
$\rightarrow$ 7, 16, 9, 22, 11, 18

Παράδειγμα εκτέλεσης του αλγορίθμου bubble_sort

Έστω η ακολουθία τιμών: 16, 22, 11, 9, 7, 18
Η ταξινόμηση φυσαλίδας προχωρά ως εξής:

$\rightarrow$ 16, 22, 11, 9, 7, 18
$\rightarrow$ 16, 22, 11, 7, 9, 18
$\rightarrow$ 16, 22, 7, 11, 9, 18
$\rightarrow$ 16, 7, 22, 11, 9, 18
$\rightarrow$ 7, 16, 22, 11, 9, 18
$\rightarrow$ 7, 16, 22, 9, 11, 18
$\rightarrow$ 7, 16, 9, 22, 11, 18
$\rightarrow$ 7, 9, 16, 22, 11, 18

Παράδειγμα εκτέλεσης του αλγορίθμου bubble_sort

Έστω η ακολουθία τιμών: 16, 22, 11, 9, 7, 18
Η ταξινόμηση φυσαλίδας προχωρά ως εξής:

$\rightarrow$ 16, 22, 11, 9, 7, 18
$\rightarrow$ 16, 22, 11, 7, 9, 18
$\rightarrow$ 16, 22, 7, 11, 9, 18
$\rightarrow$ 16, 7, 22, 11, 9, 18
$\rightarrow$ 7, 16, 22, 11, 9, 18
$\rightarrow$ 7, 16, 22, 9, 11, 18
$\rightarrow$ 7, 16, 9, 22, 11, 18
$\rightarrow$ 7, 9, 16, 22, 11, 18
$\rightarrow$ 7, 9, 16, 11, 22, 18

Παράδειγμα εκτέλεσης του αλγορίθμου bubble_sort

Έστω η ακολουθία τιμών: 16, 22, 11, 9, 7, 18
Η ταξινόμηση φυσαλίδας προχωρά ως εξής:

$\rightarrow$ 16, 22, 11, 9, 7, 18
$\rightarrow$ 16, 22, 11, 7, 9, 18
$\rightarrow$ 16, 22, 7, 11, 9, 18
$\rightarrow$ 16, 7, 22, 11, 9, 18
$\rightarrow$ 7, 16, 22, 11, 9, 18
$\rightarrow$ 7, 16, 22, 9, 11, 18
$\rightarrow$ 7, 16, 9, 22, 11, 18
$\rightarrow$ 7, 9, 16, 22, 11, 18
$\rightarrow$ 7, 9, 16, 11, 22, 18
$\rightarrow$ 7, 9, 11,16, 22, 18

Παράδειγμα εκτέλεσης του αλγορίθμου bubble_sort

Έστω η ακολουθία τιμών: 16, 22, 11, 9, 7, 18
Η ταξινόμηση φυσαλίδας προχωρά ως εξής:

$\rightarrow$ 16, 22, 11, 9, 7, 18
$\rightarrow$ 16, 22, 11, 7, 9, 18
$\rightarrow$ 16, 22, 7, 11, 9, 18
$\rightarrow$ 16, 7, 22, 11, 9, 18
$\rightarrow$ 7, 16, 22, 11, 9, 18
$\rightarrow$ 7, 16, 22, 9, 11, 18
$\rightarrow$ 7, 16, 9, 22, 11, 18
$\rightarrow$ 7, 9, 16, 22, 11, 18
$\rightarrow$ 7, 9, 16, 11, 22, 18
$\rightarrow$ 7, 9, 11,16, 22, 18
$\rightarrow$ 7, 9, 11,16, 18, 22