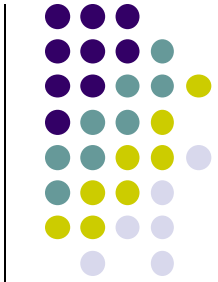
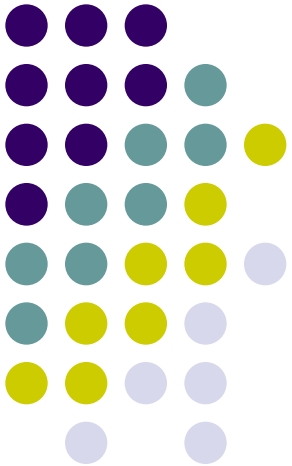


Κλασικοί Αλγόριθμοι ΙΙ

Γ’ Πληροφορική



Παραγωγή τυχαίων αριθμών

```
import random
num=random.randint(1,10)
```

Σημειώσεις

- Η **import random** εισάγει μια βιβλιοθήκη συναρτήσεων για την παραγωγή τυχαίων αριθμών.
- Η **συνάρτηση randint(1,10)** παίρνει ως παραμέτρους ένα κάτω (1) και ένα άνω όριο (10) και επιστρέφει έναν ακέραιο μεταξύ αυτών των ορίων, συμπεριλαμβανομένων και αυτών των δύο δηλαδή, έναν από τους 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Παραγωγή τυχαίων αριθμών

```
num=random.randrange(1,10)
```

- Η **συνάρτηση randrange(1,10)** επιστρέφει έναν τυχαίο ακέραιο αριθμό μεταξύ 1 και 9, δηλαδή, έναν από τους 1, 2, 3, 4, 5, 6, 7, 8, 9

```
num=random.randrange(10)
```

- Η **συνάρτηση randrange(10)** επιστρέφει έναν τυχαίο ακέραιο αριθμό μεταξύ 0 και 9, δηλαδή, έναν από τους 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Άσκηση

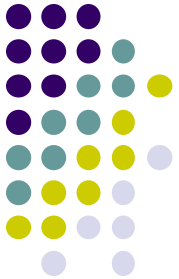
```
import random
num = random.randint(1,100)
print "Μπορείς να μαντέψεις ένα αριθμό απότο 1 έωστο 100"
guess = 0
while guess != num:
  guess = input("Δώσε αριθμό: ")
  if guess>num:
        print "Έδωσες μεγαλύτερο αριθμό"
  elif guess<num:
        print "Έδωσες μικρότερο αριθμό"
  else:
        print "Τον βρήκες!"
```

Ενα παιχνίδι

- Σκεφτείτε ένα αριθμό από το 1 έως το 100 και απαντήστε στην ερώτηση:
- Ο αριθμός που σκεφτήκατε είναι μεγαλύτερος ή μικρότερος του 50;
- Ας υποθέσουμε ότι ο αριθμός που σκεφτήκατε είναι μικρότερος του 50.
- Τότε θα βρίσκεται σίγουρα μεταξύ του 1 και του 50.
- Άρα με μια ερώτηση καταφέραμε να περιορίσουμε το χώρο αναζήτησης στο μισό.
- Με μια δεύτερη ερώτηση για τη σχέση του αριθμού μας με το 25 και ανεξάρτητα από την απάντηση, περιορίζουμε το πρόβλημα στο μισό του μισού, δηλ. Στο ¼ του αρχικού προβλήματος.

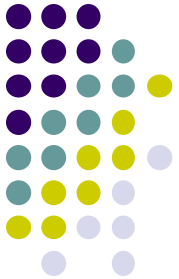
Ενα παιχνίδι (συνέχεια)

- Η προηγούμενη μέθοδος είναι γνωστή ως **Δυαδική αναζήτηση**, και
- εκμεταλεύεται τη διάταξη των στοιχείων ενός συνόλου,
- διαμεράζονταςκάθε φορά το σύνολο σε δύο ίσα μέρη, και
- εφαρμόζοντας πάλι την ίδια μέθοδο στο υποσύνολο στο οποίο ανήκει ο ζητούμενος αριθμός.



Δυαδική Αναζήτηση (binary search)

Δυαδική Αναζήτηση

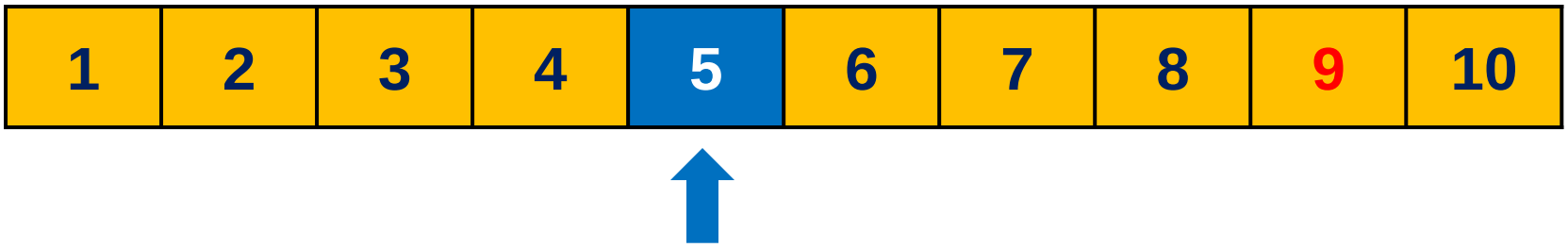


- **Η αναζήτηση εφαρμόζεται σε διατεταγμένα σύνολα.**
- Με τη δυαδική αναζήτηση ρωτάμε αν το στοιχείο που ψάχνουμε είναι στο μέσο. Η απάντηση που λαμβάνουμε είναι αν το στοιχείο που ψάχνουμε είναι ίσο (άρα το βρήκαμε!), μικρότερο ή μεγαλύτερο από αυτό που ρωτήσαμε.
- Όσο δεν έχουμε βρει το στοιχείο, συνεχίζουμε αποκλείοντας τα μισά στοιχεία και ψάχνοντας κάθε φορά στα υπόλοιπα μισά...

Παράδειγμα

- **Θα ακολουθήσει ένα παράδειγμα αναζήτησης του αριθμού 9 μέσα από το διάστημα από το 1 έως το 10.**

Δυαδική αναζήτηση

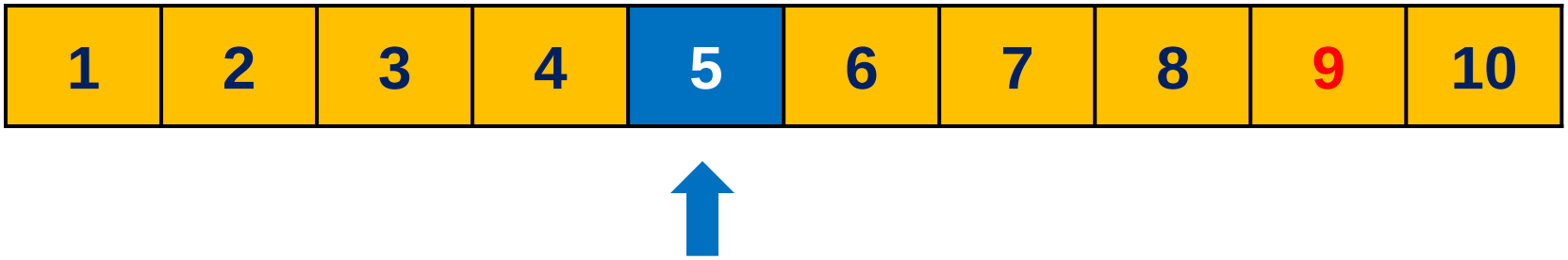


•Όταν αρχίζουμε το ***"μεσαίο"*** στοιχείο είναι το 5. Αυτό προκύπτει από την πράξη `(1+10)//2` όπου το 1 είναι το ***πρώτο*** στοιχείο και το 10 το ***τελευταίο***.

•Στο ερώτημα ***"είναι το 5;"*** η απάντηση που λαμβάνουμε μπορεί να είναι μια από τις:

- ***"είναι"*** οπότε το βρήκαμε,
- ***"το 5 είναι μικρότερο"*** ή
- ***"το 5 είναι μεγαλύτερο"***.

Δυαδική αναζήτηση

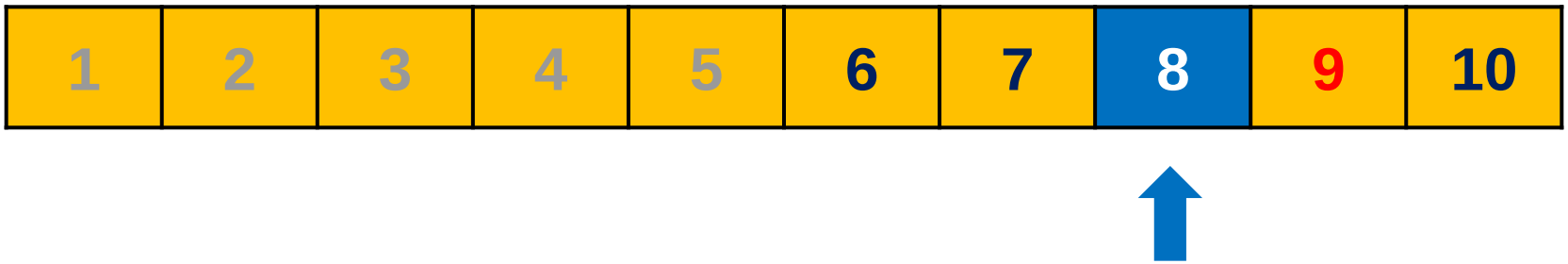


•Αφού η απάντηση που λαμβάνουμε είναι ***"το 5 είναι μικρότερο"*** γνωρίζουμε ότι το στοιχείο που ψάχνουμε είναι στο διάστημα **6 έως 10.**

•Άρα, αποκλείουμε τα στοιχεία από 1 έως και 5,

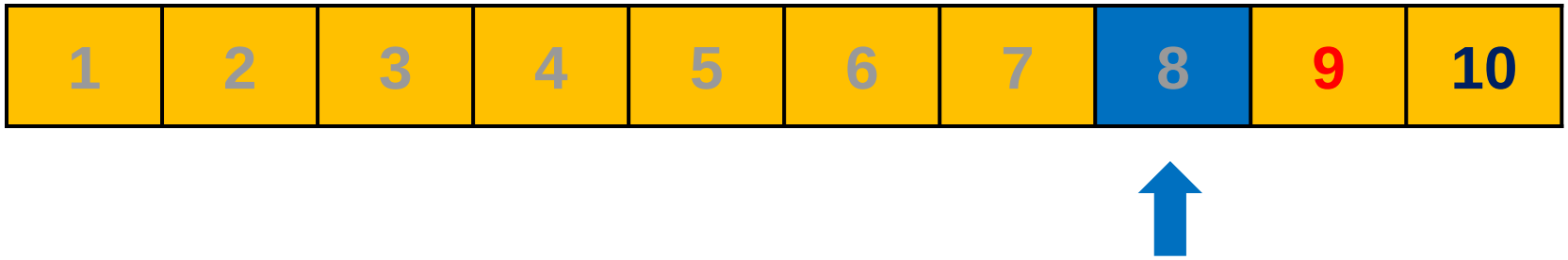
•και υπολογίζουμε τον νέο μέσο με τον τύπο **(6+10)//2**

Δυαδική αναζήτηση



- Τώρα θέτουμε ερώτημα ***"είναι το 8;"*** και
- η απάντηση που λαμβάνουμε είναι
- ***"το 8 είναι μικρότερο"***.
- Οπότε, αποκλείουμε και τα **6**, **7**, **8**

Δυαδική αναζήτηση

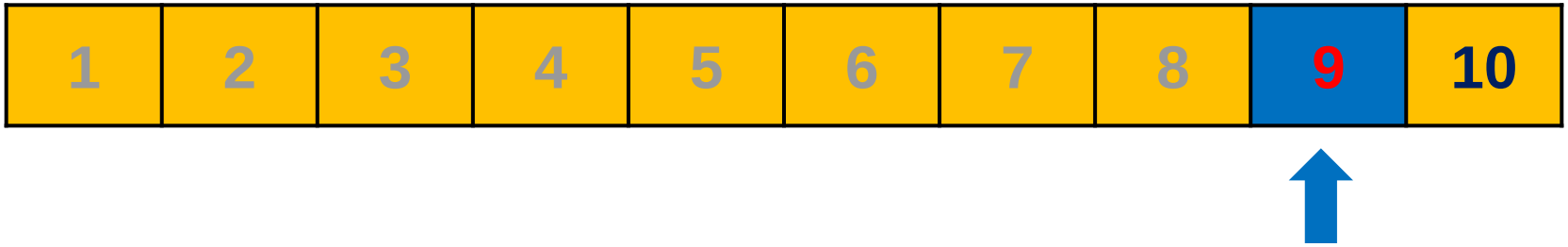


•Τώρα θέτουμε ερώτημα ***"είναι το 8;"*** και η απάντηση που λαμβάνουμε είναι ***"το 8 είναι μικρότερο"***.

•Οπότε, αποκλείουμε και τα **6**, **7**, **8** και

•συνεχίζουμε το ψάξιμο στα στοιχεία που περισσεύουν υπολογίζοντας τον νέο μέσο με τον τύπο **(9+10)//2**

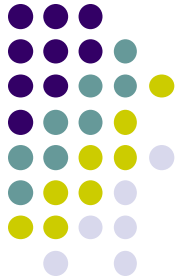
Δυαδική αναζήτηση



Στο ερώτημα ***"είναι το 9;"*** η απάντηση είναι θετική, οπότε τελειώσαμε ...

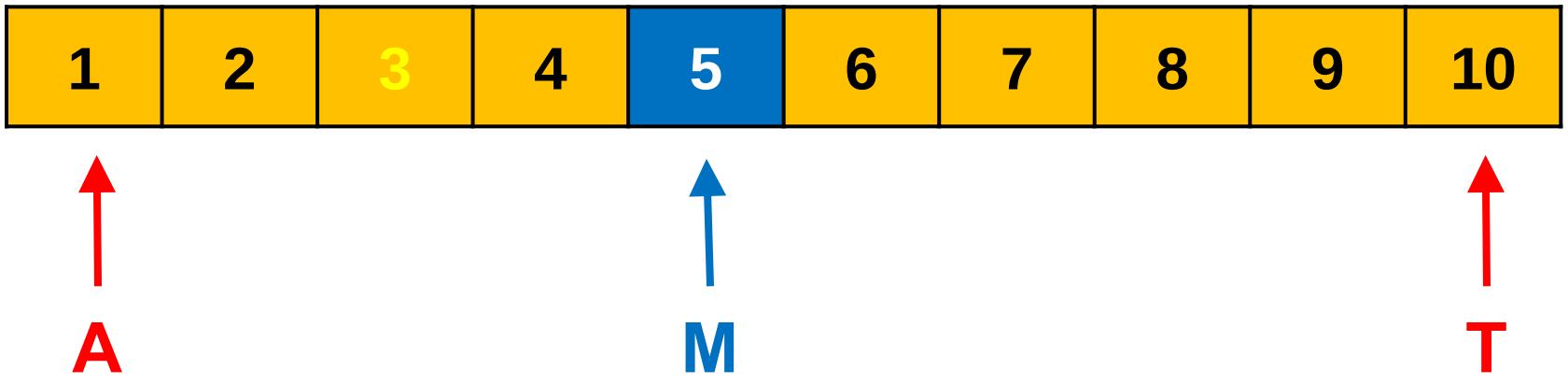
μετά από τρεις αναζητήσεις !!!

Δυαδική αναζήτηση



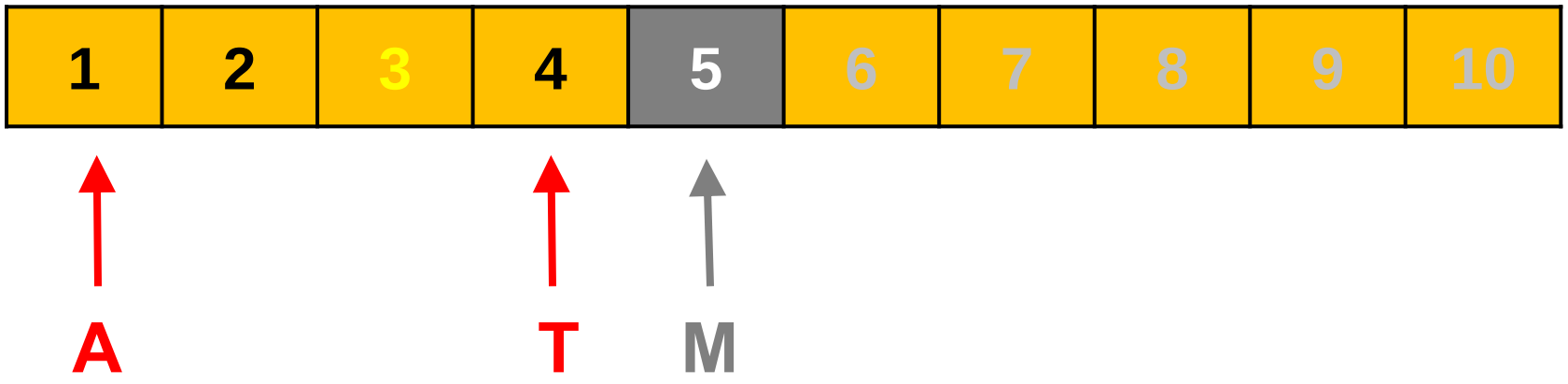
- Ο αλγόριθμος που μόλις περιγράψαμε είναι γνωστός ως ***αλγόριθμος δυαδικής αναζήτησης***, γιατί σε κάθε βήμα χωρίζει το χώρο αναζήτησης στο μισό.
- Για να υλοποιηθεί εκτός από τον εκάστοτε ***μέσο*** χρειάζεται να διαχειριζόμαστε την ***αρχή*** και το ***τέλος*** …
- Έστω ότι ο αριθμός που ψάχνουμε είναι ο **3**

Δυαδική αναζήτηση



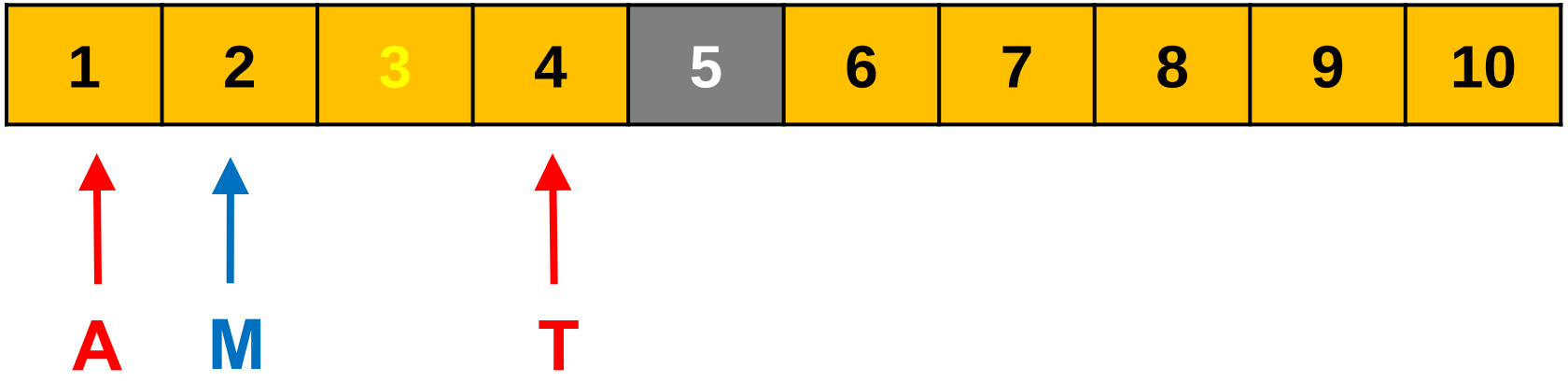
Την πρώτη φορά **A=1**, **T=10** και **M=(A+T)//2**
Επειδή **3<M**,

Δυαδική αναζήτηση



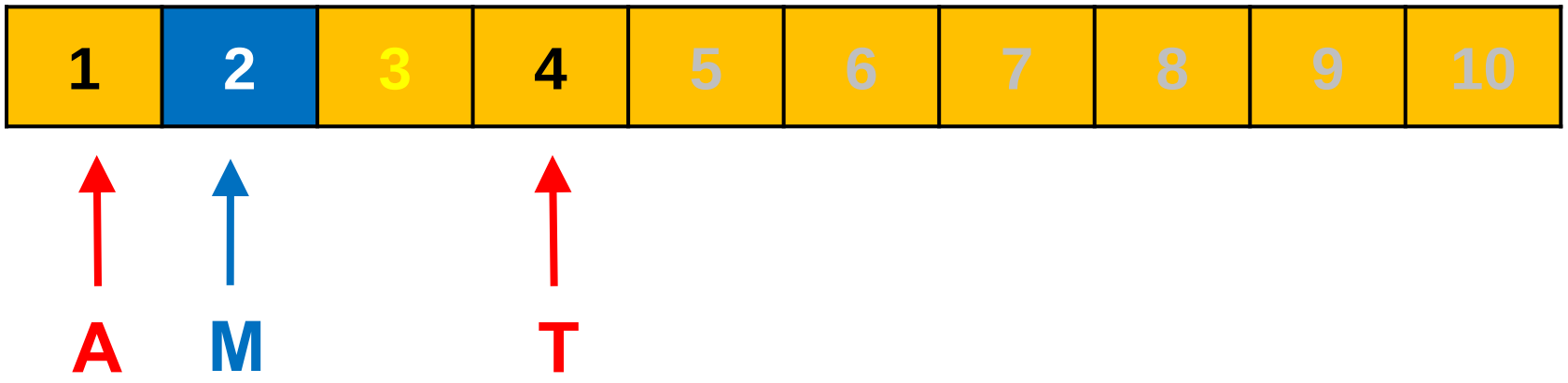
Την πρώτη φορά **A=1**, **T=10** και **M=(A+T)//2**
Επειδή **3<M**, το **T** γίνεται **T=M-1**

Δυαδική αναζήτηση



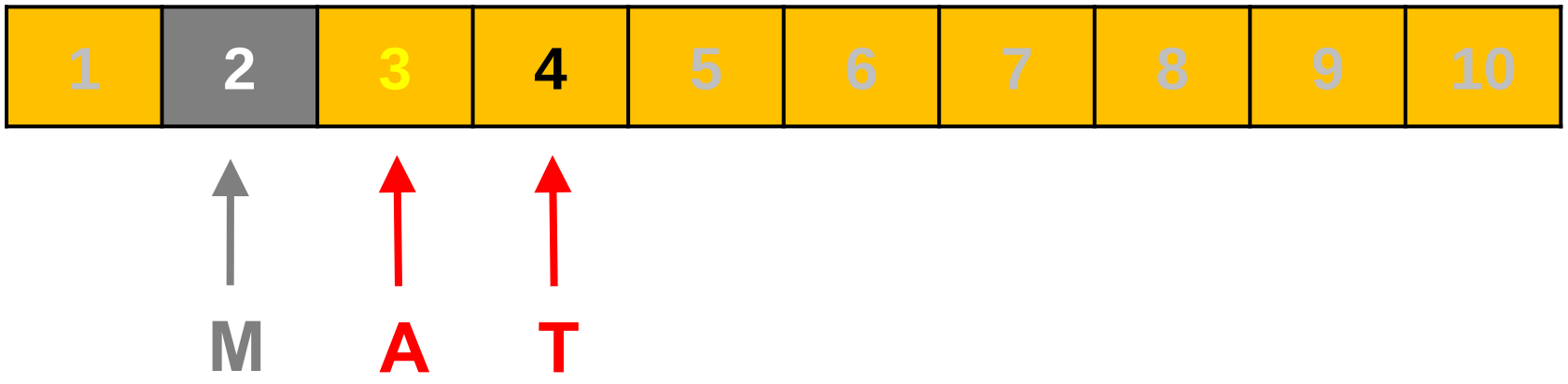
Την πρώτη φορά **A=1**, **T=10** και **M=(A+T)//2**
Επειδή **3<M**, το **T** γίνεται **T=M-1** και το
M=(A+T)//2

Δυαδική αναζήτηση



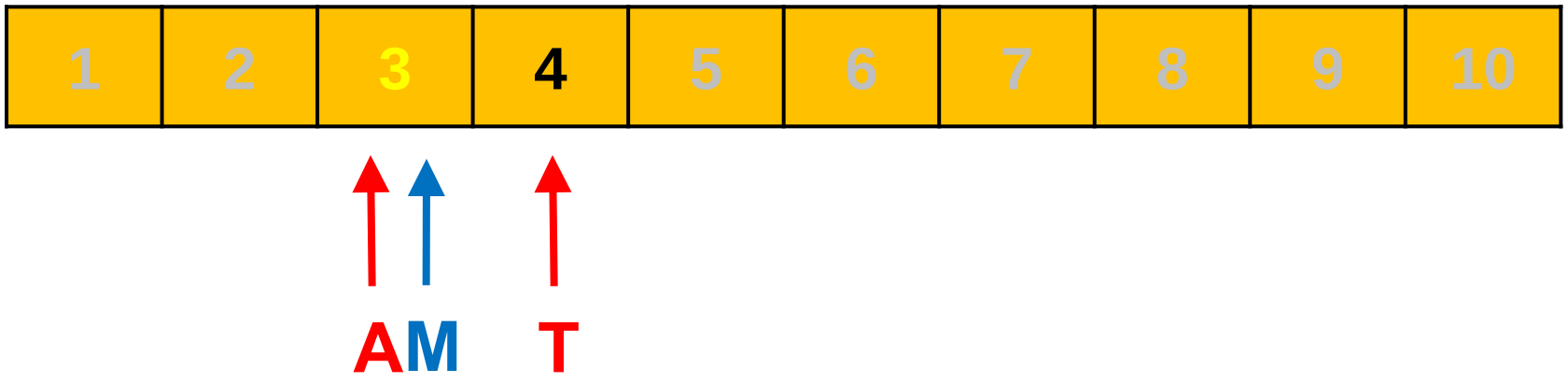
Την δεύτερη φορά **A=1**, **T=4** και **M=2**
Επειδή **3>M**,

Δυαδική αναζήτηση



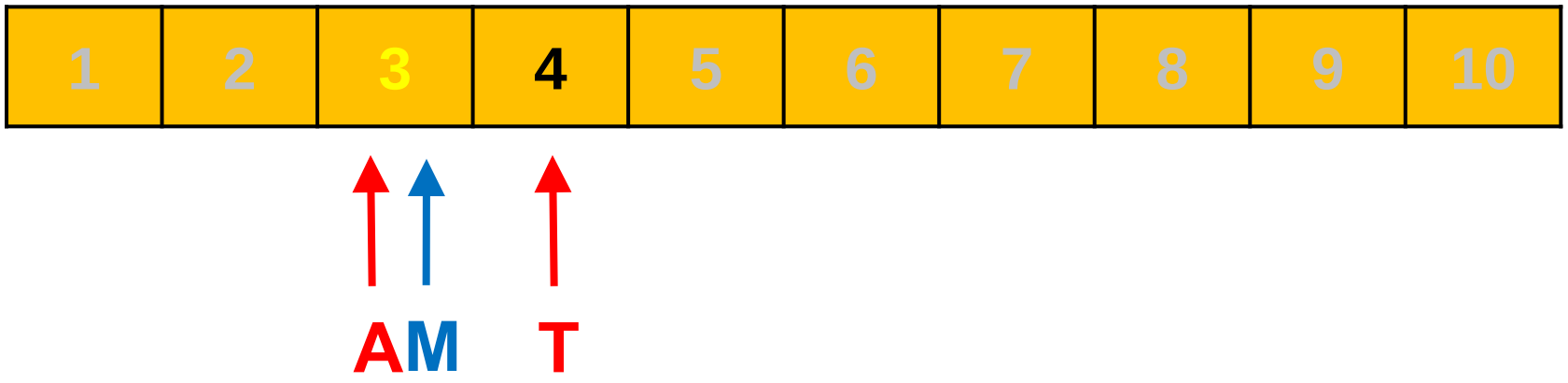
Την δεύτερη φορά **A=1**, **T=4** και **M=2**
Επειδή **3>M**, το **A** φίνεται **A=M+1**

Δυαδική αναζήτηση



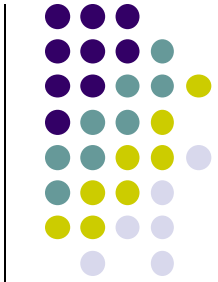
Την δεύτερη φορά **A=1**, **T=4** και **M=2**
Επειδή **3>M**, το **A** φίνεται **A=M+1, και το**
M=(A+T)//2

Δυαδική αναζήτηση



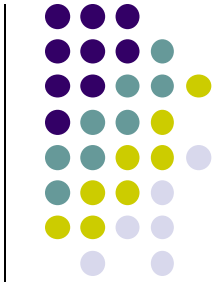
Επειδή **3=M**, η αναζήτηση ολοκληρώθηκε.

Δυαδική αναζήτηση



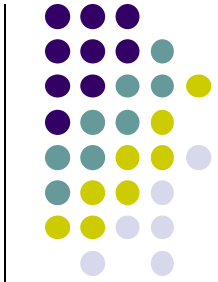
```
import random
number=random.randint(1, 10)
found=False
tr,a,t=0,1,10
while not found:
        tr=tr+1
        m=(a+t)/2
        if m==number:
          found=True
        elif m>number:
          t=m-1
        else: a=m+1
print "Ο αριθμός είναι ο:",m
print "Βρέθηκε μετά από",tr,"προσπάθειες"
```

Δυαδική αναζήτηση (με τη βοήθεια συνάρτησης)



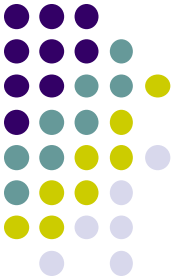
```
def binarySearch(array,key):
    first = 0
    last = len(arrray)-1
    found = False
    while first <= last and not found:
        mid = (first + last) / 2
        if array[mid] == key:
            found = True
        elif array[mid] < key:
            first = mid + 1
        else:
            last = mid-1
    return found
a = [20,24,34,45,50]
key=24
fou=binarySearch(a,key)
if fou==True:
    print "vrethike"
else:
    print "den vrethike"
```

Συνάρτηση Δυαδικής αναζήτησης (επιστρέφει και τη θέση του στοιχείου)



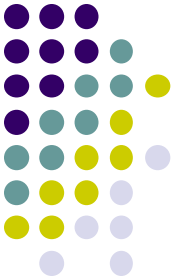
```
def binarySearch(array,key):
        first = 0
        last = len(arrray)-1
        pos = -1
        while first <= last and pos==-1:
                mid = (first + last) / 2
                if array[mid] == key:
                        pos = mid
                elif array[mid] < key:
                        first = mid + 1
                else:
                        last = mid-1
        return pos
```

Πολυμορφισμός

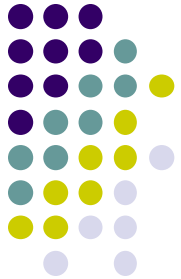


- Οι παραπάνω αλγόριθμοι ισχύουν για όλους τους τύπους δεδομένων για τους οποίους ορίζονται οι συγκριτικοί τελεστές ==, <.
- Δηλαδή μπορούν να χρησιμοποιηθούν για ακεραίους ή πραγματικούς αριθμούς, αλφαριθμητικά ή ακόμα και σύνθετους τύπους για τους οποίους έχουμς ορίσει τους συγκεκριμένους τελεστές.
- Αυτό το χαρακτηριστικό είναι γνωστό ως **πολυμορφισμός**.

Δυαδική αναζήτηση - Σύνοψη

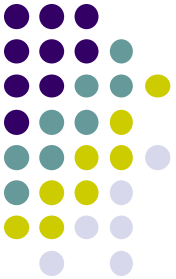


- Εκμεταλεύεται τη διάταξη των στοιχείων ενός συνόλου δεδομένων για τη γρήγορη εύρεση ενός στοιχείου
- Χρησιμοποιείται μόνο σε ταξινομημένες συλλογές δεδομένων
- Βρίσκει το ζητούμενο πολύ πιο γρήγορα από τη σειριακή αναζήτηση



Ταξινόμηση ευθείας ανταλλαγής ή ταξινόμηση φυσαλίδας (bubble short)

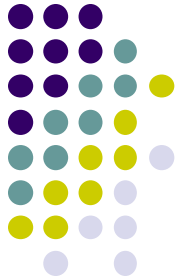
Ταξινόμηση “φυσαλίδας”



Έστω ότι θέλουμε να ταξινομήσουμε μία λίστα που έχει τα στοιχεία: **11**, **5**, **3**, **8**, **1**.

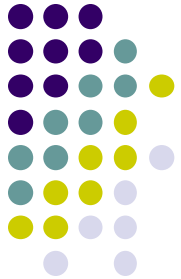
Ξεκινάμε από το τέλος και συγκρίνουμε, προχωρώντας προς την αρχή, δύο-δύο τα στοιχεία.
Αν δεν βρίσκονται στη σωστή σειρά, τα αντιμεταθέτουμε.
Επαναλαμβάνουμε τα παραπάνω μέχρι η λίστα να έχει πλήρως ταξινομηθεί!

Ταξινόμηση “φυσαλίδας” (1)



11
5
3
8
1

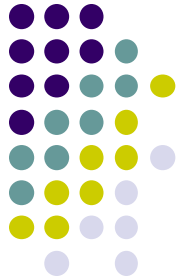
Ταξινόμηση “φυσαλίδας” (2)



11
5
3
8
1

πρώτα ελέγχουμε τα δύο τελευταία στοιχεία (4ο και 5ο)

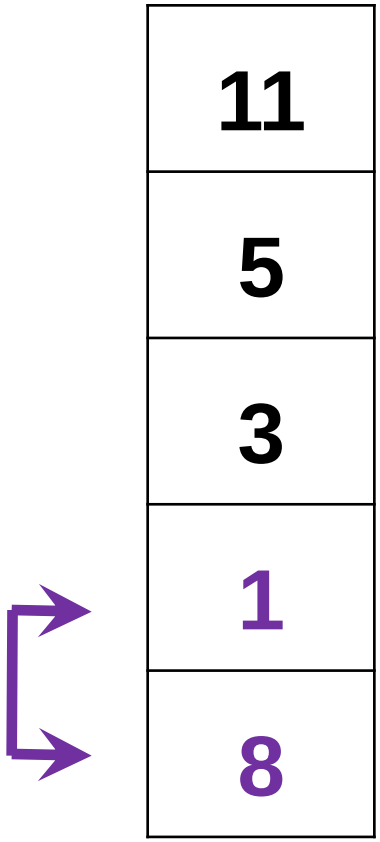
Ταξινόμηση “φυσαλίδας” (3)



11
5
3
8
1

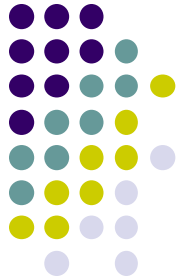
εάν το 5ο είναι μικρότερο του 4ου, τα αντιμεταθέτουμε

Ταξινόμηση “φυσαλίδας” (4)



εάν το 5ο είναι μικρότερο του 4ου, τα αντιμεταθέτουμε

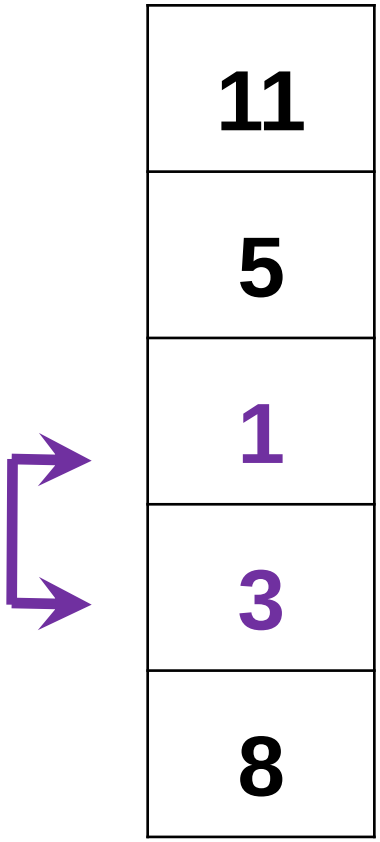
Ταξινόμηση “φυσαλίδας” (5)



11
5
3
1
8

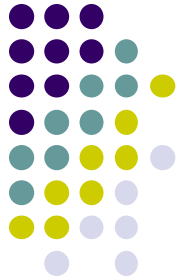
συνεχίζουμε με τη σύγκριση του 3ου με το 4ο

Ταξινόμηση “φυσαλίδας” (6)



συνεχίζουμε με τη σύγκριση του 3ου με το 4ο και αντιμεταθέτουμε και πάλι

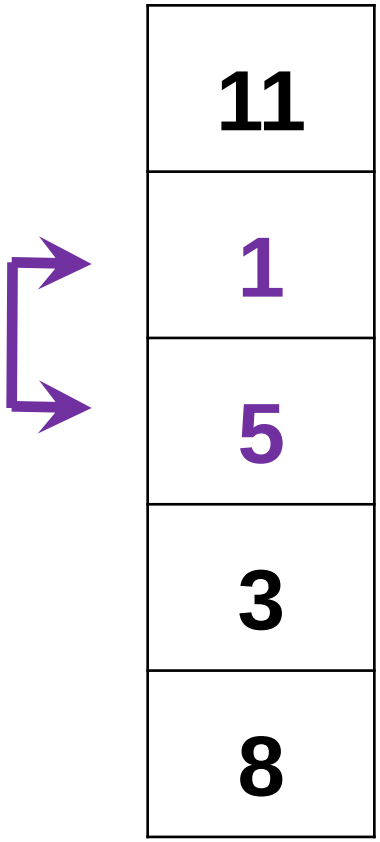
Ταξινόμηση “φυσαλίδας” (7)



11
5
1
3
8

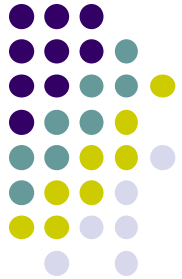
συνεχίζουμε με τη σύγκριση του 2ου με το 3ο

Ταξινόμηση “φυσαλίδας” (8)



συνεχίζουμε με τη σύγκριση του 2ου με το 3ο και αντιμεταθέτουμε και πάλι

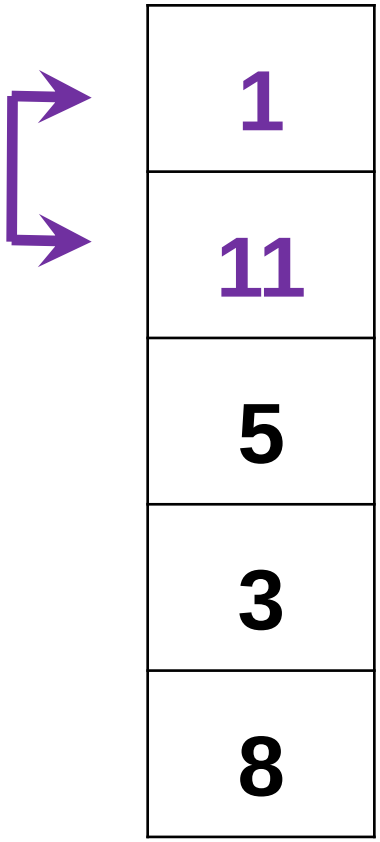
Ταξινόμηση “φυσαλίδας” (9)



11
1
5
3
8

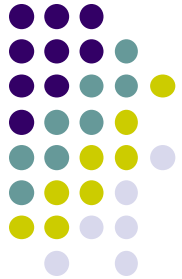
συνεχίζουμε με τη σύγκριση του 1ου με το 2ο

Ταξινόμηση “φυσαλίδας” (10)



συνεχίζουμε με τη σύγκριση του 1ου με το 2ο και αντιμεταθέτουμε και πάλι

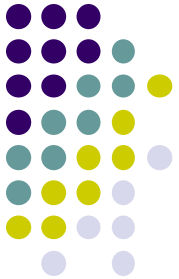
Ταξινόμηση “φυσαλίδας” (11)



1
11
5
3
8

έχουμε κάνει 4 συγκρίσεις

Ταξινόμηση “φυσαλίδας” (12)

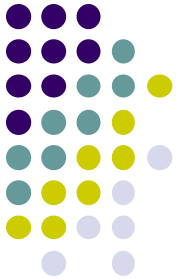


1
11
5
3
8

μετά από ένα πέρασμα
το μικρότερο στοιχείο
έχει ανέβει πάνω-πάνω
σαν φυσαλίδα στο νερό

έχουμε κάνει 4 συγκρίσεις
και στην 1η θέση έχει τοποθετηθεί το μικρότερο στοιχείο

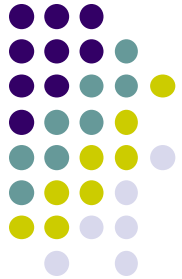
Ταξινόμηση “φυσαλίδας” (13)



1
11
5
3
8

συνεχίζουμε τις συγκρίσεις πάλι από τα κάτω

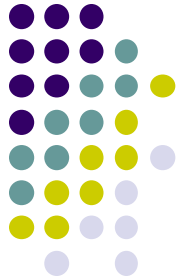
Ταξινόμηση “φυσαλίδας” (14)



1
11
5
3
8

συνεχίζουμε τις συγκρίσεις πάλι από τα κάτω
τα στοιχεία στην 4η και 5η θέση είναι διατεταγμένα

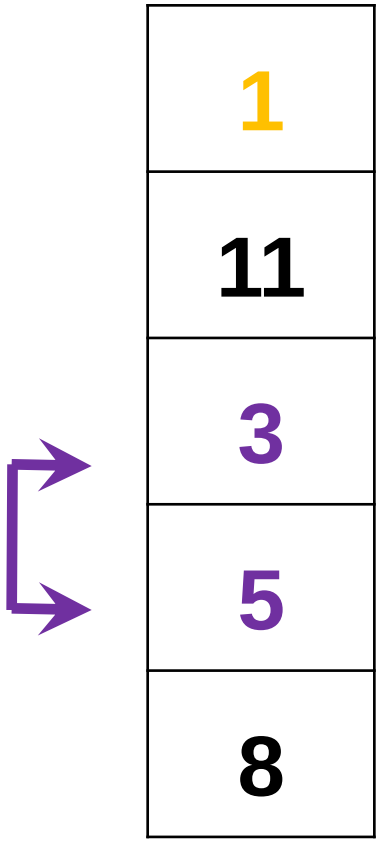
Ταξινόμηση “φυσαλίδας” (15)



1
11
5
3
8

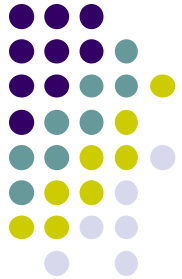
συνεχίζουμε με τη σύγκριση του 3ου με το 4ο

Ταξινόμηση “φυσαλίδας” (16)



συνεχίζουμε με τη σύγκριση του 3ου με το 4ο και αντιμεταθέτουμε και πάλι

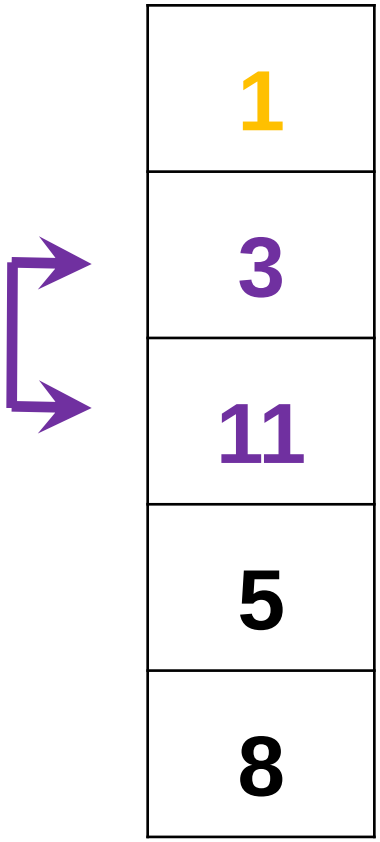
Ταξινόμηση “φυσαλίδας” (17)



1
11
3
5
8

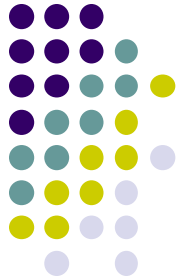
συνεχίζουμε με τη σύγκριση του 2ου με το 3ο

Ταξινόμηση “φυσαλίδας” (18)



συνεχίζουμε με τη σύγκριση του 2ου με το 3ο και αντιμεταθέτουμε και πάλι

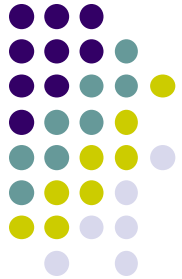
Ταξινόμηση “φυσαλίδας” (19)



1
3
11
5
8

αν συνεχίζαμε με τη σύγκριση του 1ου με το 2ο, δεν θα κάναμε αντιμετάθεση

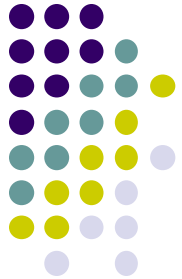
Ταξινόμηση “φυσαλίδας” (20)



1
3
11
5
8

στην ουσία, μετά από δύο “περάσματα”,
στις δύο πρώτες θέσεις έχουν τοποθετηθεί τα δύο μικρότερα στοιχεία

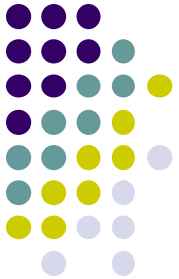
Ταξινόμηση “φυσαλίδας” (21)



1
3
11
5
8

θα συνεχίσουμε με τα υπόλοιπα τρία…

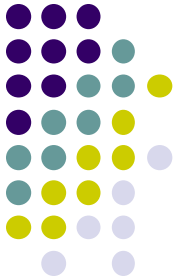
Ταξινόμηση “φυσαλίδας” (22)



1
3
11
5
8

εξετάζουμε τα στοιχεία στην 4η και 5η θέση και είναι διατεγμένα

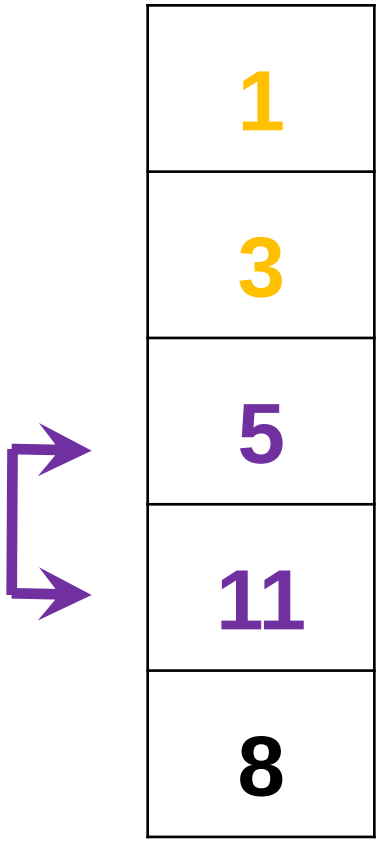
Ταξινόμηση “φυσαλίδας” (23)



1
3
11
5
8

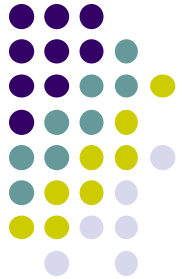
εξετάζουμε τα στοιχεία στην 3η και 4η θέση

Ταξινόμηση “φυσαλίδας” (24)



συνεχίζουμε με τη σύγκριση του 3ου με το 4ο και αντιμεταθέτουμε και πάλι

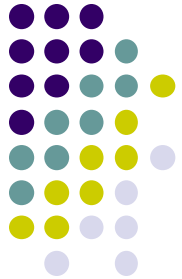
Ταξινόμηση “φυσαλίδας” (25)



1
3
5
11
8

τα τρία πρώτα στοιχεία έχουν μπει στη θέση του μετά από τρία “περάσματα και

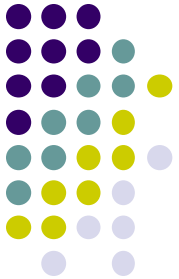
Ταξινόμηση “φυσαλίδας” (26)

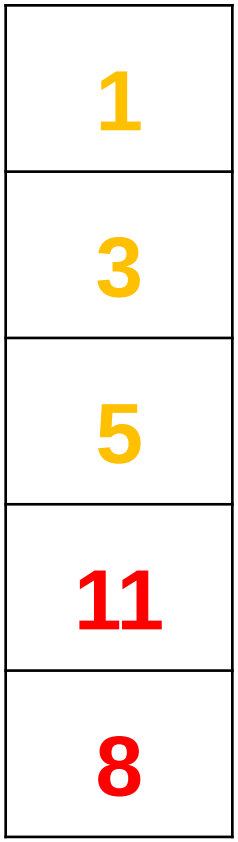


1
3
5
11
8

Έχουν απομείνει μόνο δύο

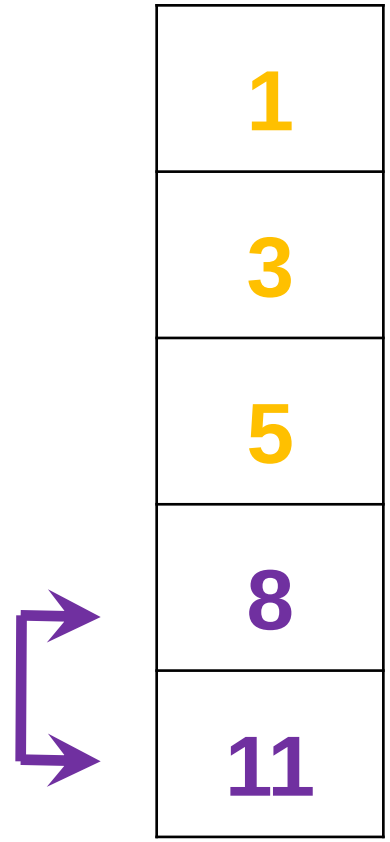
Ταξινόμηση “φυσαλίδας” (27)





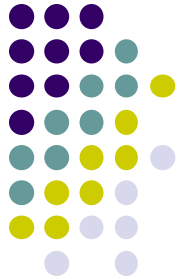
Εξετάζουμε τα στοιχεία στη $4^{η}$ και $5^{η}$ θέση

Ταξινόμηση “φυσαλίδας” (28)



Εξετάζουμε το 4ο στοιχείο με το 5ο και τα αντιμεταθέτουμε

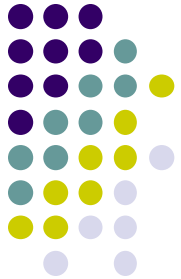
Ταξινόμηση “φυσαλίδας” (29)



1
3
5
8
11

Τέλος, όλα τα στοιχεία είναι διατεταγμένα!

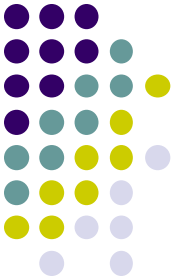
Ταξινόμηση “φυσαλίδας” (30)



Αρχική λίστα	1ο πέρασμα	2ο πέρασμα	3ο πέρασμ	4ο πέρασμ
11	1	1	1	1
5	11	3	3	3
3	5	11	5	5
8	3	5	11	8
1	8	8	8	11

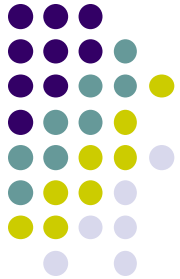
Περάσματα: 4 (N-1)

1ο πέρασμα: 4 συγκρίσεις
2ο πέρασμα: 3 συγκρίσεις
3ο πέρασμα: 2 συγκρίσεις
4ο πέρασμα: 1 σύγκριση



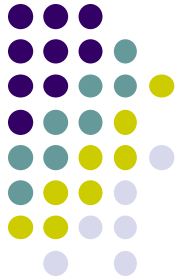
Αρχική λίστα	1ο πέρασμα	2ο πέρασμα	3ο πέρασμ	4ο πέρασμ
11	1	1	1	1
5	11	3	3	3
3	5	11	5	5
8	3	5	11	8
1	8	8	8	11

Ταξινόμηση “φυσαλίδας”



```
array = [11, 5, 3, 8, 1]
N = len(array)
for i in range(N-1):
    for j in range(N-1,i,-1):
        if array[j] < array[j-1]:
            array[j],array[j-1] = array[j-1],array[j]
print array
```

Άσκηση 2 - Ταξινόμηση “φυσαλίδας” (με συνάρτηση)

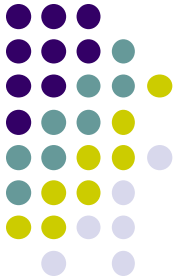


```
def BubbleSort(array):
    N = len(array)
    for i in range(N-1):
        for j in range(N-1,i,-1):
            if array[j] < array[j-1]:
                array[j], array[j-1] = array[j-1], array[j]
    return array
a = [10, 5, 2, 9, 1]
print BubbleSort(a)
```

Άσκηση 3

- **Να μετατρέψετε τον αλγόριθμο φυσαλίδας (στη μορφή με το for) ώστε να κάνει φθίνουσα ταξινόμηση.**

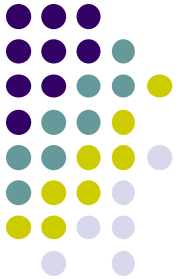
Άσκηση 3 - “φυσαλίδας”- φθίνουσα



Η γενικευμένη του μορφή είναι:

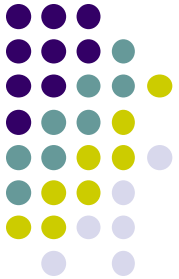
```
array = [10, 5, 2, 9, 1]
N = len(array)
for i in range(N-1):
    for j in range(N-1,i,-1):
        if array[j] > array[j-1]:
            array[j],array[j-1] = array[j-1],array[j]
print array
```

Δραστηριότητα 4 -Ταξινόμηση “βελτιωμένη φυσαλίδας”



- Να δώσετε τη βελτιωμένη έκδοση του αλγορίθμου ταξινόμησης ευθείας ανταλλαγής η οποία τερματίζει, όταν διαπιστώσει ότι η λίστα είναι ταξινομημένη, ώστε να αποφεύγονται περιττές συγκρίσεις.
- Υπόδειξη: Χρησιμοποιήστε μια λογική μεταβλητή η οποία θα αλλάζει τιμή, αν υπάρχουν τουλάχιστον δύο στοιχεία τα οποία δεν βρίσκονται στην επιθυμητή σειρά, καθώς η “φυσαλίδα ανεβαίνει στην επιφάνεια”.

Ασκηση 4 - Ταξινόμηση “βελτιωμένη φυσαλίδας”



```
array=[12, 6, 24, 5, 13, 22]
N=len(array)
sorted=False
i=1
while i<N and not sorted:
    sorted=True
    for j in range(N-1,i-1,-1):
        if array[j] < array[j-1]:
            array[j],array[j-1] = array[j-1],array[j]
            sorted=False
    i=i+1
print array
```