Κεφάλαιο 6ο
Κλασικοί Αλγόριθμοι Ι

6.1 Υπολογισμός Μέγιστου Κοινού Διαιρέτη

Δύο από τους πιο δημοφιλείς αλγόριθμους που έχουν επινοήσει αρχαίοι Έλληνες, είναι το κόσκινο του Ερατοσθένη για την εύρεση πρώτων αριθμών και ο αλγόριθμος υπολογισμού του Μέγιστου Κοινού Διαιρέτη του Ευκλείδη.
Ο αλγόριθμος του Ευκλείδη βασίζεται στην παρακάτω απλή, αλλά όχι προφανή ιδέα :
ΜΚΔ (α, β) = **ΜΚΔ** (β, α mod β)
Για παράδειγμα, ας προσπαθήσουμε, με επαναληπτική εφαρμογή της παραπάνω σχέσης, να βρούμε το μέγιστο κοινό διαιρέτη των αριθμών 100 και 36 :
ΜΚΔ (100, 36) = **ΜΚΔ** (36, 28) = **ΜΚΔ** (28, 8) = **ΜΚΔ** (8, 4) = **ΜΚΔ** (4, 0) = 4
Από τις παραπάνω σχέσεις φαίνεται ότι το κριτήριο διακοπής της επανάληψης θα μπορούσε να είναι το β = 0. Έτσι ο αλγόριθμος σε Python δίνεται παρακάτω :

```
def mkd (a, b) :
    while b > 0 :
        a , b = b , a % b
    return a
```

6.2 Σειριακή Αναζήτηση

Πολλές φορές χρειάζεται να αναζητήσουμε κάτι συγκεκριμένο μέσα σε μια συλλογή δεδομένων. Η αναζήτηση γίνεται συνήθως με βάση κάποιο χαρακτηριστικό των δεδομένων, όπως για παράδειγμα το επώνυμο, ο αριθμός ταυτότητας κ.λπ. Η πιο απλή μορφή αναζήτησης είναι η σειριακή ή γραμμική αναζήτηση σύμφωνα με την οποία εκτελούμε εξαντλητική αναζήτηση σε όλα τα στοιχεία της συλλογής, μέχρι να βρούμε αυτό που ψάχνουμε. Έτσι, στη χειρότερη περίπτωση, αν ξεκινήσουμε από το πρώτο στοιχείο και αυτό που ψάχνουμε είναι τελευταίο, θα κάνουμε τόσες συγκρίσεις, όσα είναι και τα στοιχεία που έχουμε. Μια πρώτη υλοποίηση του αλγόριθμου φαίνεται παρακάτω :

```
def linearSearch (array, key) :
    found = False
    for item in array :
        if item == key :
            found = True
    return found
```

Μια ακόμα έκδοση του αλγόριθμου, που αποδεικνύει ακόμα μια φορά την απλότητα αλλά και τη μινιμαλιστική έκφραση της Python, είναι η παρακάτω, η οποία εκμεταλλεύεται το γνωστό ιδίωμα της **for** και τη δυνατότητα βίαιης διακοπής του, χωρίς αυτό να θεωρείται πλέον κακή πρακτική

```
def linearSearch (array, key) :
    for item in array :
        if item == key :
            return True
    return False
```

Ωστόσο, πολλές φορές θέλουμε να βρούμε και τη θέση του στοιχείου στη συλλογή, ειδικά αν αυτή υλοποιείται με κάποια δομή δεδομένων που επιτρέπει την αποθήκευση των στοιχείων με κάποια διάταξη. Έτσι μπορούμε να προσθέσουμε και μία μεταβλητή θέση/position.
Τώρα όμως η λογική μεταβλητή είναι περιττή, αφού η μεταβλητή θέση μπορεί να παίξει και το ρόλο της λογικής μεταβλητής ως εξής : Αν η θέση παραμείνει –1, τότε σημαίνει ότι το στοιχείο δε βρέθηκε. Αν όμως βρεθεί, τότε θα πάρει την τιμή της θέσης του δείκτη

Αλγόριθμος Σειριακής Αναζήτησης (έκδοση 2.0)

```
def linearSearch (array, key) :
    pos = –1
    i = 1
    while pos < 0 and i <= N :
        if array[ i ] == key :
            pos = i
    i = i + 1
    return pos
```

6.3 Ταξινόμηση με Επιλογή

Το πρόβλημα της ταξινόμησης αναφέρεται στην αναδιάταξη των στοιχείων μιας λίστας, ώστε αυτά να βρεθούν σε αύξουσα ή σε φθίνουσα σειρά, σε σχέση με κάποια σχέση διάταξης. Πολλές φορές χρειάζεται να ταξινομήσουμε λίστες αριθμών, λέξεων ή εγγράφων.

Ένας από τους πιο απλούς αλγόριθμους ταξινόμησης είναι ο αλγόριθμος ταξινόμησης με επιλογή (selection sort). Η λογική του αλγόριθμου είναι η παρακάτω :
Βήμα 1. Βρίσκουμε το μικρότερο στοιχείο του πίνακα και το τοποθετούμε στην πρώτη θέση.
Βήμα 2. Βρίσκουμε το αμέσως μικρότερο στοιχείο του πίνακα και το τοποθετούμε στη δεύτερη θέση.
Βήμα 3. Συνεχίζουμε βρίσκοντας κάθε φορά το μικρότερο στοιχείο από τα στοιχεία του πίνακα που απομένουν και το τοποθετούμε στο τέλος των ήδη ταξινομημένων στοιχείων.

Μπορεί να υλοποιηθεί σε Python με την παρακάτω συνάρτηση :

```
def selectionSort (array) :
    N = len (array)
    for i in range (0, N) :
      pos = posMin (i, N, array)
      array[i], array[pos] = array[pos], array[i]
```

Ο αλγόριθμος χρησιμοποιεί μία συνάρτηση για να υπολογίζει τη θέση του ελάχιστου στοιχείου του πίνακα.

```
def posMin (start, size, array) :
    pos = start
    for i in range (start, size) :
        if array[ i ] < array[ pos ] :
            pos = i
    return pos
```

Η συνάρτηση posMin δέχεται το μέγεθος της λίστας size. Το τελευταίο στοιχείο της λίστας σε αυτή την περίπτωση είναι το size-1.

Παρακάτω φαίνεται το περιεχόμενο ενός πίνακα 7 αριθμών, για κάθε βήμα της εκτέλεσης του αλγόριθμου ταξινόμησης της επιλογής :

	i=0	i=1	i=2	i=3	i=4	i=5	i=6
0	60	20	20	20	20	20	20
1	38	38	32	32	32	32	32
2	98	98	98	38	38	38	38
3	54	54	54	54	54	54	54
4	32	32	38	98	98	60	60
5	90	90	90	90	90	90	90
6	20	60	60	60	60	98	98