Η Ομάδα μας 57
Κωνσταντίνος Χουλιαράς, 1505
Ιωάννης Αστλι, 1448

Εργασία στον μέγιστο κοινό διαιρέτη (gcd)

1 Η βασική ιδέα του Ευκλείδειου αλγόριθμου.

Για να υπολογίσουμε τον μέγιστο κοινό διαιρέτη δύο ακεραίων, $\gcd(a, b)$, a, $b \epsilon \mathbb{Z}_{>0}$, βλέπουμε πως από την Ευκλείδεια διαίρεση $a = b \cdot q + r$, $0 \leq r < b$, προκύπτει ότι *κάθε* διαιρέτης των a *και* b διαιρεί την διαφορά $a - b \cdot q$, και συνεπώς, διαιρεί και το υπόλοιπο r. Αυτό σημαίνει πως $\gcd(a, b) = \gcd(b, r)$, το οποίο μας δίνει έναν τρόπο για τον υπολογισμό του $\gcd(a, b)$. Πράγματι, θέτοντας $a_0 = a$, $a_1 = b$ και $a_2 = r$ έχουμε:

$$\gcd(a_0, a_1) = \gcd(a_1, a_2) = \cdots = \gcd(a_{k-1}, a_k) = \gcd(a_k, 0) = a_k.$$

Προσέξτε πως εδώ, αλλά και γενικά, υποθέτουμε ότι το υπόλοιπο r είναι *μη αρνητικό*!

Παράδειγμα 1.

διαιρετέος $=$	διαιρέτης $\times$	πηλίκο	$+$ υπόλοιπο
612 $=$	342 $\times$	1	$+$ 270
342 $=$	270 $\times$	1	$+$ 72
270 $=$	72 $\times$	3	$+$ 54
72 $=$	54 $\times$	1	$+$ 18
54 $=$	18 $\times$	3	$+$ 0

Ο μκδ των 612 και 342 είναι το 18, ο τελευταίος διαιρέτης που δίνει υπόλοιπο 0. Από τον παραπάνω πίνακα βλέπουμε πως χρειάστηκαν 5 διαιρέσεις για να υπολογίσουμε τον gcd $(612, 342)$.

1.1 Πρώτος Ευκλείδειος αλγόριθμος

Το παρακάτω πρόγραμμα είναι στην γλώσσα του Xcas, και περιγράφει τον Ευκλείδειο αλγόριθμο όπως τον χρησιμοποιούσαν γεωμετρικά (με ευθύγραμμα τμήματα) οι αρχαίοι Ελληνες.

```
> my_igcd1(a,b):={
    if(b==0) {return(a)};
    ifte(a>=b, my_igcd1(b,a-b), my_igcd1(a,b-a))
  }:;

> my_igcd1(612, 342)
```

18

Να προγραμματίσετε αυτόν τον αλγόριθμο στο Sympy και να τρέξετε το Παράδειγμα 1.

Απάντηση :

```
>>> def my_igcd1(a,b):
        if(b==0):
            return a
        if(a>=b):
            return my_igcd1(b,a-b)
        else :
            return my_igcd1(a,b-a)
>>> my_igcd1(612, 342)

   18

>>> my_igcd1(13,8)

   1
```

```
>>> my_igcd1(770,567)
    7
>>>
```

1.2 Δεύτερος Ευκλείδειος αλγόριθμος – μείωση του αριθμού των διαιρέσεων

Ο αριθμός των διαιρέσεων στο Παράδειγμα 1, μπορεί να ελαττωθεί αν επιτρέπονται *αρνητικά υπόλοιπα*. Για παράδειγμα, στο Sympy επιτρέπονται όταν ο διαιρέτης είναι αρνητικός

```
>>> from sympy import ZZ,
>>> [ZZ.quo(13, -7), ZZ.rem(13, -7)]
    [-2, -1]
>>> [ZZ.quo(13, 7), ZZ.rem(13, 7)]
    [1, 6]
```

ενώ στο Xcas δεν επιτρέπονται σε καμία περίπτωση.

```
> iquorem(13, -7)
```

$$[-1, 6]$$

```
> iquorem(13, 7)
```

$$[1, 6]$$

```
>
```

Δηλαδή, αν $q > 0$, τότε από την εξίσωση

$$a = b \cdot q + r_1 \tag{1}$$

μπορούμε να έχουμε και την εξίσωση

$$a = b \cdot (q+1) - r_2. \tag{2}$$

Στην γενική περίπτωση — όπου το πηλίκο q μπορεί να είναι και αρνητικό — ενεργούμε ως εξής: αν το υπόλοιπο είναι $\leq \left\lfloor \frac{|b|}{2} \right\rfloor$, χρησιμοποιούμε την εξίσωση (1); αν όμως το υπόλοιπο είναι $> \left\lfloor \frac{|b|}{2} \right\rfloor$ τότε το υπόλοιπο είναι $r = a - b \cdot q$, όπου το πηλίκο q αλλάζει ως εξής:

- εάν $q > 0$ θέτουμε $q = q + 1$;
- εάν $q < 0$ θέτουμε $q = q - 1$.

Να προγραμματίσετε στο Sympy αυτόν τον αλγόριθμο και να μετρήσετε πόσες διαιρέσεις εκτελούνται στο Παράδειγμα 1.

```
>>> from sympy import ZZ, floor, Abs
    def gcdcounter(a,b,divisions):
        if(b==0):
            return a,divisions
        if (a>=b):
            if ZZ.rem(a,b)<=floor(Abs(b)/2):
                return
    gcdcounter(b,ZZ.rem(a,b),divisions+1)

            else:
                if (ZZ.quo(a,b)>0):
                    return
    gcdcounter(b,b*(ZZ.quo(a,b)+1)-a,divisions+1)
                else:
                    return
    gcdcounter(b,b*(ZZ.quo(a,b)-1)-a,divisions+1)
        else:
            if ZZ.rem(b,a)<=floor(Abs(a)/2):
                return
    gcdcounter(a,ZZ.rem(b,a),divisions+1)
            else:
                if (ZZ.quo(b,a)>0):
                    return
    gcdcounter(a,a*(ZZ.quo(b,a)+1)-b,divisions+1)
                else:
                    return
    gcdcounter(a,a*(ZZ.quo(b,a)-1)-b,divisions+1)
```

```
>>> gcd,divisions = gcdcounter(612,342,0)
    print(+str(gcd),+str(divisions))
```

```
>>> gcd,divisions = gcdcounter(770,567,0)
    print("Μέγιστος␣κοινός␣διαιρέτης␣των␣770␣και␣567␣=␣"+str(gcd),"\nΧ
```

```
>>> gcd,divisions = gcdcounter(13,8,0)
    print("Μέγιστος␣κοινός␣διαιρέτης␣των␣13␣και␣8␣=␣"+str(gcd),"\nΧρει
```

1.3 Σημαντική ιδιότητα των gcd's

Πρόταση 2. *Εάν $a, b \epsilon \mathbb{Z}$ και $b \neq 0$ τότε υπάρχουν $s, t \epsilon \mathbb{Z}$ έτσι ώστε*

$$\gcd(a, b) = a \cdot s + b \cdot t \tag{3}$$

Η εξίσωση (3) είναι γνωστή σαν ταυτότητα του Bezout.

Παράδειγμα 3. Για τους ακέραιους 215 και 5 έχουμε:

```
>>> from sympy import gcdex
>>> gcdex(612, 342)
```

```
(-5, 9, 18)
```

που σημαίνει, ότι χρησιμοποιώντας την εξίσωση (3) έχουμε:

```
>>> 18 == (-5) * 612 + 9 * 342
```

```
True
```

1.4 Επεκταμένος Ευκλείδειος Αλγόριθμος (Extended gcd)

Θα υπολογίσουμε τους συντελεστές s, t με δύο τρόπους τους οποίους θα προγραμματίσετε στο Sympy και θα τους δοκιμάσετε στο Παράδειγμα 1. Ιδιαίτερα στην *ανάδρομη αντικατάσταση να εκτυπώνετε όλα τα ενδιάμεσα αποτελέσματα.*

1.4.1 Ανάδρομη αντικατάσταση και συνδυασμός όρων

Παράλληλα με τον υπολογισμό του $\gcd(a,b)$ δημιουργούμε έναν ξεχωριστό πίνακα όπου κάθε υπόλοιπο εκφράζεται σαν διαφορά "διαιρετέος − διαιρέτης × πηλίκο". Αφού υπολογίσουμε τον $\gcd\ (a,\ b)$ (το τελευταίο μη μηδενικό υπόλοιπο) χρησιμοποιούμε τον ξεχωριστό πίνακα για την ανάδρομη αντικατάσταση και τον συνδυασμό των όρων για να υπολογίσουμε τους ακεραίους $s,\ t$ για να ισχύει η (3).

Παράδειγμα 4.

διαιρετέος =	διαιρέτης ×	πηλίκο	+ υπόλοιπο
612 =	342 ×	1	+ 270
342 =	270 ×	1	+ 72
270 =	72 ×	3	+ 54
72 =	54 ×	1	+ 18
54 =	18 ×	3	+ 0

Ο ξεχωριστός πίνακας.

υπόλοιπο=διαιρετέος−διαιρέτης×υπόλοιπο
270=612−342×1
72=342−270×1
54=270−72×3
18=72−54×1

Με ανάδρομη αντικατάσταση και συνδυασμό όρων έχουμε:

$$18 = 72 - 54 = (\text{αντικαθιστού}\mu\varepsilon\,72\text{και}54)$$

$$= (342 - 270 \times 1) - (270 - 72 \times 3) = (\text{αντικαθιστού}\mu\varepsilon\,270\text{και}72)$$

$$= (342 - (612 - 342 \times 1) \times 1) - ((612 - 342 \times 1) - (342 - 270 \times 1) \times 3)$$

$$=(\text{συνδυάζου}\mu\varepsilon\,\text{όρους}) = 612 \times (-2) + 342 \times 6 - 270 \times 3 = (\text{αντικαθιστού}\mu\varepsilon\,270)$$

$$=612 \times (-2) + 342 \times 6 - (621 - 342 \times 1) \times 3 = (\text{συνδυάζου}\mu\varepsilon\,\text{όρους})$$

$$=612 \times (-5) + 342 \times 9$$

Δηλαδή έχουμε $s = -5$ και $t = 9$ όπως είδαμε και προηγουμένως.

Απάντηση. 1.4.1

```
>>> from sympy import ZZ
    def my_gcd2(a,b,array):
        if a == 0 :
            print("\n\tΕπομένως␣ο␣μκδ("+str(array[0][1])+","+str(array
            array.reverse()
            temp ="\t={0}-␣{1}*{2}".format(array[0][1],array[0][2],arr
            print(temp)
            for i in range(1,len(array)):
                temp
    =temp.replace(str(array[i][0]),"({0}␣-␣{1}*{2})").format(array[i][
                print(temp)
            return (b,0,1)
        else:
            if ((ZZ.quo(b,a) > 0) and (ZZ.rem(b,a) > 0)) :
                temp = [ZZ.rem(b,a),b,a,ZZ.quo(b,a)]
                array.append(temp)
            if ZZ.quo(b,a):
                print("\t{0}␣={1}*{2}␣+␣{3}".format(b,ZZ.quo(b,a),a,ZZ

            g,x,y = my_gcd2(ZZ.rem(b,a),a,array)
            return (g,y-ZZ.quo(b,a)*x,x)
    print("\nA.\n")
    a = 13
    b = 8
    print("\t>μκδ("+str(a)+","+str(b)+");")
    g,x,y = my_gcd2(a,b,[])
    print("\n\tΔηλαδή␣s␣=␣{2}␣και␣t␣=␣{4}.\n\n\t{0}␣=␣{1}*({2})␣+␣{3}*
    a = 612
    b = 342
    print("\t>μκδ("+str(a)+","+str(b)+");")
    g,x,y = my_gcd2(a,b,[])
    print("\n\tΔηλαδή␣s␣=␣{2}␣και␣t␣=␣{4}.\n\n\t{0}␣=␣{1}*({2})␣+␣{3}*
    a = 770
    b = 567
    print("\t>μκδ("+str(a)+","+str(b)+");")
    g,x,y = my_gcd2(a,b,[])
    print("\n\tΔηλαδή␣s␣=␣{2}␣και␣t␣=␣{4}.\n\n\t{0}␣=␣{1}*({2})␣+␣{3}*
```

1.4.2 Ταυτόχρονος υπολογισμός των s και t

Καθώς υπολογίζουμε τον $\gcd(a,b)$ εκφράζουμε το κάθε υπόλοιπο r_i σαν:

$$r_i = a \cdot s_i + b \cdot t_i \tag{4}$$

για κάποια s_i και t_i που πρέπει να υπολογισθούν.

Με αυτόν τον τρόπο το τελευταίο μη μηδενικό υπόλοιπο — που είαι ο $\gcd(a,b)$ — θα εκφρασθεί σαν γραμμικός συνδυασμός των a και b, και το πρόβλημα λύθηκε.

Για να υπολογίσουμε τα s_i και t_i θέτουμε $r_0 = a$, $r_1 = b$ και εκφράζουμε τα r_0 και r_1 σαν (4). Προφανώς,

$$r_0 = a = a \cdot 1 + b \cdot 0$$

και

$$r_1 = b = a \cdot 0 + b \cdot 1$$

Δηλαδή για το πρώτο υπόλοιπο $r_0 = a$ έχουμε $s_0 = 1, t_0 = 0$ ενώ για το δεύτερο υπόλοιπο $r_1 = b$ έχουμε $s_1 = 0$, $t_1 = 1$. Για τα άλλα υπόλοιπα ξέρουμε πως ισχύει:

$$r_i = r_{i-2} - r_{i-1} \cdot q_i$$

Στην εξίσωση αυτή αντικαθιστούμε τα r_{i-2} και r_{i-1} με τις αντίστοιχες εκφράσεις από την (4) και έχουμε

$$r_i = (a \cdot s_{i-2} + b \cdot t_{i-2}) - (a \cdot s_{i-1} + b \cdot t_{i-1}) \cdot q_i$$

απ' όπου προκύπτει

$$r_i = a \cdot (s_{i-2} - s_{i-1} \cdot q_i) + b \cdot (t_{i-2} - t_{i-1} \cdot q_i)$$

ή

$$r_i = a \cdot s_i + b \cdot t_i$$

Αυτό σημαίνει πως οι τιμές των s_i και t_i υπολογίζονται ακριβώς όπως τα υπόλοιπα r_i. Δηλαδή έχουμε:

$$r_i = r_{i-2} - r_{i-1} \cdot q_i \tag{5}$$

$$s_i = s_{i-2} - s_{i-1} \cdot q_i \tag{6}$$

$$t_i = t_{i-2} - t_{i-1} \cdot q_i \tag{7}$$

Παράδειγμα 5.

q_i	r_0		r_1	s_0		s_1	t_0		t_1
1	612	$\swarrow$	342	1	$\swarrow$	0	0	$\swarrow$	1
1	342	$\swarrow$	270 $\natural$	0	$\swarrow$	1 $\natural$	1	$\swarrow$	-1 $\natural$
3	270	$\swarrow$	72 $\natural$	1	$\swarrow$	-1 $\natural$	-1	$\swarrow$	2 $\natural$
1	72	$\swarrow$	54 $\natural$	-1	$\swarrow$	4 $\natural$	2	$\swarrow$	-7 $\natural$
3	54	$\swarrow$	18 $\natural$	4	$\swarrow$	-5 $\natural$	-7	$\swarrow$	9 $\natural$
-	18		0	-5		-	9		-

Συνεπώς $18 = 612 \cdot (-5) + 342 \cdot 9$

Απάντηση. 1.4.2

```
>>> from sympy import ZZ,floor

    def extendedgcd(a, b):
        if a == 0:
            return (b, 0, 1)
        else:
            g, s, t = extendedgcd(ZZ.rem(b,a), a)
            return (g,(s-(floor(ZZ.quo(b,a)*t))), t)

    a = 13
    b = 8
    g,s,t = extendedgcd(a,b)
    print("ΜΚΔ␣των␣"+str(a)+"␣και␣"+str(b)+"␣=␣"+str(g)+"\ns="+str(s)+

    a = 612
    b = 342
    g,s,t = extendedgcd(a,b)
    print("ΜΚΔ␣των␣"+str(a)+"␣και␣"+str(b)+"␣=␣"+str(g)+"\ns="+str(s)+

    a = 770
    b = 567
    g,s,t = extendedgcd(a,b)
    print("ΜΚΔ␣των␣"+str(a)+"␣και␣"+str(b)+"␣=␣"+str(g)+"\ns="+str(s)+
```

.

1.5 Υπολογισμός του $m^{-1} \bmod n$

Με τον επεκταμένο Ευκλείδειο αλγόριθμο μπορούμε να υπολογίσουμε τον πολλαπλασιαστικό αντίστροφο ενός ακεραίου m modulo n. Υπενθυμίζουμε πως το αντίστροφο m^{-1} mod n υπάρχει μόνο στην περίπτωση που $\gcd(m,n)=1$.

Για τον υπολογισμό του αντιστρόφου εφαρμόζουμε τον επεκταμένο Ευκλείδειο αλγόριθμο στους m,n και έχουμε $\gcd(m,n)=1=m\cdot s+n\cdot t$, απ' όπου βλέπουμε πως $m^{-1} \bmod n = s$.

Να προγραμματίσετε αυτό το πρόγραμμα στο Sympy και με την βοήθειάτου να υπολογίσετε όλα τα πολλαπλασιαστικά αντίστροφα στο σώμα $\mathbb{Z}_{29}$.

Απάντηση. 1.5

```
>>> from sympy import ZZ,floor

    def extendedgcd(a, b):
        if a == 0:
            return (b, 0, 1)
        else:
            g, y, x = extendedgcd(ZZ.rem(b,a), a)
            return (g,(x-(floor(ZZ.quo(b,a)*y))), y)

    def inversion(a, m):
        g, x,_ = extendedgcd(a, m)
        if g != 1:
            raise Exception('Δεν␣υπάρχει␣αντίστροφος')
        else:
            return ZZ.rem(x,m)


    for i in range(1,29):
        print ("Με␣m:␣"+str(i)+"␣και␣n:29\nΤο␣πολλαπλασιαστικό␣αντίστρο
    +"\n")
```

[1]

1. Λόγο αδυναμίας σύνδεσης του sympy με το Texmacs μέχρι και την τελευταία μέρα της προθεσμίας δεν είμαστε σε θέση να ξέρουμε αν παράγονται τα σωστά αποτελέσματα στο περιβάλλον του Texmacs . Ολα τα προγράμματα δοκιμάστηκαν στο Thonny .