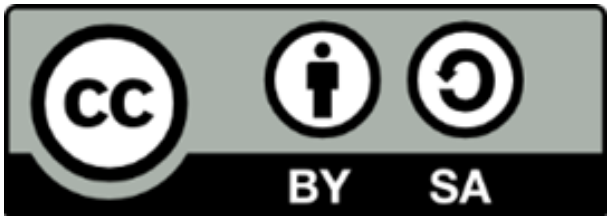
Υπολογιστικά & Διακριτά Μαθηματικά

Ενότητα 1: Εισαγωγή- Χαρακτηριστικά Παραδείγματα Αλγορίθμων

Στεφανίδης Γεώργιος
Τμήμα Εφαρμοσμένης Πληροφορικής

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.



Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Μακεδονίας» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Εισαγωγή

- Θα ξεκινήσουμε με έναν αλγόριθμο κι ένα πρόβλημα προκειμένου να μας δοθεί η ευκαιρία να αναφερθούμε στο αντικείμενο (αντικειμενικούς σκοπούς) του μαθήματος.
- Αρχικά παρουσιάζουμε έναν αλγόριθμο πολλαπλασιασμού δύο αριθμών, διαφορετικό απ' αυτόν που μάθαμε στο σχολείο.

➢ **(βλέπε: Παράδειγμα 1)**

Ο Πολλαπλασιασμός του Ρώσου Χωρικού (ΠΡΧ)[1]

- Για να βρεις το γινόμενο δύο ακεραίων M και N > 1:

1. Ξεκίνησε με δύο στήλες, τη μία σημειωμένη ως «A» και την άλλη ως «B»· γράψε την τιμή του M κάτω από το A και την τιμή του N κάτω από το B.
2. Επανέλαβε (Repeat)
   a) υπολόγισε μια νέα τιμή-A πολλαπλασιάζοντας την παλιά τιμή-A επί 2· και
   b) υπολόγισε μια νέα τιμή-B διαιρώντας την παλιά τιμή-B δια 2 και ελαττώνοντας το αποτέλεσμα κατά μισή μονάδα, αν απαιτείται, προκειμένου να προκύψει ακέραιος·

Μέχρι (Until) η τιμή-B να ισούται με ένα (1).

Ο Πολλαπλασιασμός του Ρώσου Χωρικού (ΠΡΧ)[2]

3. Διάβασε τις στήλες προς τα κάτω διαγράφοντας την τιμή-Α όποτε η τιμή-Β είναι άρτιος αριθμός.
4. Πρόσθεσε τις εναπομείνασες τιμές-Α και «επέστρεψε» (Return) το άθροισμα.

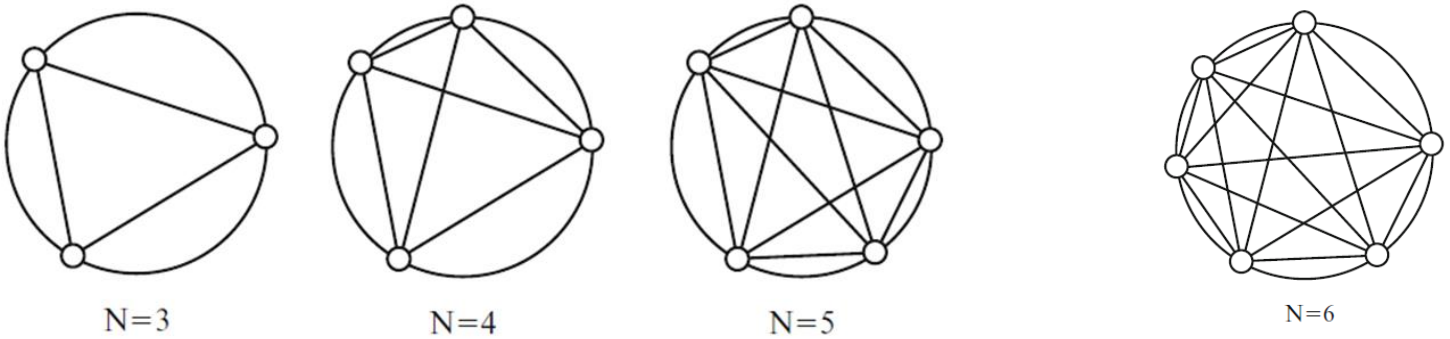
- **Παράδειγμα 2**
- **Άσκηση 1**

Ο Πολλαπλασιασμός του Ρώσου Χωρικού (ΠΡΧ)[3]

- Είναι ο αλγόριθμος ΠΡΧ ορθός (θα παρέχει τη σωστή απάντηση για κάθε ζεύγος τιμών στην είσοδο – δηλ. ∀ M, N > 1)?
- Είναι σίγουρο ότι ο «βρόχος» στο 2. (βήμα) θα τερματίζει, δηλ. ότι δεν θα είναι «ατέρμων βρόχος» (θα γίνει τελικά η τιμή-Β ίση με 1)?
- Ποια είναι η **πολυπλοκότητα** αυτού του αλγόριθμου?
  - Νομίζετε ότι είναι δυνατό να προβλέψουμε, πριν εφαρμόσουμε τον αλγόριθμο, πόσες φορές θα υποδιαιρεθεί η τιμή-Β?
  - Κάτι τέτοιο θα μας επέτρεπε να προσδιορίσουμε το πλήθος (αριθμό) των γραμμών που θα έχει ο πίνακας με τις στήλες και έτσι να έχουμε ένα άνω φράγμα για το πλήθος των προσθετέων στο 4. (βήμα)

Πρόβλημα[1]

- Έστω ότι υποδιαιρούμε έναν κυκλικό δίσκο σημειώνοντας N σημεία στην περιφέρειά του και συνδέοντάς τα ανά δύο με χορδές. Πόσα τμήματα P(N) θα δημιουργηθούν?



- Αν γράψουμε τον αριθμό των σημείων και τον αντίστοιχο αριθμό των τμημάτων, παίρνουμε:

N	*P(N)*
1	1
2	2
3	4
4	8
5	16
6	??

- $\Rightarrow P(N) = 2^{N-1}$ (?)
- $\rightarrow P(N) = ?$

Πρόβλημα[2]

- Θα δούμε ότι μπορούμε να αποδείξουμε ότι ο αλγόριθμος ΠΡΧ είναι όντως ορθός (δεν υπάρχουν περιπτώσεις στις οποίες να μην επιστρέφει το γινόμενο των N και M), ότι τερματίζει πάντα και μπορούμε να προσδιορίσουμε την πολυπλοκότητά του.
  - [Θα δούμε επίσης ότι ενώ το N πρέπει να είναι ακέραιος > 1, το M δεν χρειάζεται να είναι > 1, δεν χρειάζεται να είναι ακέραιος και μπορεί να είναι και αρνητικός αριθμός.]
- Θα επανέλθουμε στο πρόβλημα **καταμέτρησης** των τμημάτων ενός κυκλικού δίσκου, που προκύπτουν με τη σύνδεση, ανά δύο, N σημείων της περιφέρειάς του, και θα αποδείξουμε ότι υπάρχει ένας τύπος για το P(N) που ισούται με 2^{N-1} μόνο για N = 1, 2, 3, 4 και 5.

Πρόβλημα[3]

- Θα μας δοθεί η ευκαιρία να διαπιστώσουμε ότι η **επαγωγή** – η συλλογιστική (μέθοδος) που καταλήγει σε γενικό συμπέρασμα από μερικά παραδείγματα (περιπτώσεις) – δεν είναι επαρκής για την εδραίωση της (όλης) αλήθειας.

➢ Και στις δύο παραπάνω περιπτώσεις η μέθοδος της **μαθηματικής επαγωγής** θα παίξει τον καταλυτικό (απαραίτητο) ρόλο.

➢ Βασικός σκοπός του μαθήματος είναι η παροχή μεθόδων απόδειξης του τερματισμού και της ορθότητας ενός αλγόριθμου, όπως επίσης και απόδειξης της ορθότητας μιας (μαθηματικής) πρότασης ή τύπου. Και…

Πρόβλημα[4]

- να αποκτήσουμε το μαθηματικό υπόβαθρο που θα μας επιτρέπει να κατανοούμε ή να ανακαλύπτουμε τρόπους με τους οποίους βρίσκουμε **ταχύτερες** (?) μεθόδους (αλγόριθμους) υπολογισμού.
- **Παράδειγμα 3**

- Συνεχίζουμε την εισαγωγή αναφερόμενοι σε μερικούς αλγόριθμους ακεραίων και την πολυπλοκότητά τους.
- Θυμίζουμε αρχικά ορισμένες βασικές έννοιες από τη Θεωρία Αριθμών.

Διαιρετότητα[1]

- Έστω a, b, $d \in \mathbb{Z}$ (:= σύνολο ακεραίων). Τότε

1. Ο a **διαιρεί** τον b, συμβολικά $a \mid b$, αν υπάρχει $c \in \mathbb{Z}$ με $b = ac$. Αν $a \mid b$, τότε λέμε και ότι ο b είναι **πολλαπλάσιο** του a. Αν τώρα ο a δεν διαιρεί τον b, γράφουμε $a \nmid b$.

2. Αν $d \mid a$ και $d \geq 0$, ο d λέγεται **διαιρέτης** του a. Κάθε ακέραιος a έχει σαν *τετριμμένους διαιρέτες* τους 1 και a. Οι μη τετριμμένοι διαιρέτες του a λέγονται και *παράγοντες* του a.

   - Ένας ακέραιος $a > 1$ του οποίου οι μόνοι διαιρέτες είναι οι τετριμμένοι διαιρέτες 1 και a λέγεται ότι είναι *πρώτος αριθμός* ή απλά ***πρώτος***.
   - Ένας ακέραιος $a > 1$ ο οποίος δεν είναι πρώτος λέγεται ότι είναι *σύνθετος αριθμός* ή απλά ***σύνθετος***. Ο ακέραιος 1 δεν είναι πρώτος ούτε σύνθετος. Παρόμοια, ο ακέραιος 0 και όλοι οι αρνητικοί ακέραιοι δεν είναι ούτε πρώτοι ούτε σύνθετοι.

3. Ο $d \in \mathbb{N}$ λέγεται **μέγιστος κοινός διαιρέτης** των a και b, συμβολικά $\gcd(a, b)$, αν:

   - $d \mid a$ και $d \mid b$ · και
   - αν $t \in \mathbb{Z}$ διαιρεί τον a και b, τότε διαιρεί τον d.

Διαιρετότητα[2]

❑ **Θεώρημα 1** (Διαίρεση με υπόλοιπο)

Έστω $z, a \in \mathbb{Z}, a \neq 0$. Τότε υπάρχουν μοναδικοί ακέραιοι $q, r \in \mathbb{Z}$, τέτοιοι, ώστε

$$z = q \cdot a + r \quad \text{και} \quad 0 \leq r < |a|.$$

- Ο r λέγεται **υπόλοιπο** του z modulo a (δηλ. $r := z \bmod a$).
- Ο αριθμός q είναι το (ακέραιο) *πηλίκο* των z και a, συμβολικά z div a (δηλ. $q := z$ div a) και είναι $q = \lfloor z/a \rfloor$, όπου $\lfloor x \rfloor$ είναι ο μεγαλύτερος ακέραιος που δεν υπερβαίνει τον αριθμό x (:= κάτω ακέραιο μέρος του x – **floor**). Επίσης
  - $a \mid z \Leftrightarrow z \bmod a = 0$.

(floor :βλέπε floor.pdf)

Διαιρετότητα[3]

- Θεώρημα 2 (**Αναπαράσταση** ως προς βάση *b*)

Αν $x \in \mathbb{R}$, τότε

$$x = \sum_{k=-m}^{n-1} a_k b^k = \underbrace{a_{n-1}b^{n-1} + a_{n-2}b^{n-2} + \cdots + a_1 b^1 + a_0 b^0}_{\text{ακέραιο μέρος}} + \underbrace{a_{-1}b^{-1} + a_{-2}b^{-2} + \cdots + a_{-m+1}b^{-m+1} + a_{-m}b^{-m} + \cdots}_{\text{κλασματικό μέρος}}$$

- Θεώρημα 3 (Παραγοντοποίηση σε πρώτους)

Ένας σύνθετος $a \in \mathbb{Z}$ μπορεί να γραφεί κατά μοναδικό τρόπο ως γινόμενο της μορφής

$$a = p_1^{e_1} p_2^{e_2} \cdots p_s^{e_s} = \prod_{i=1}^{s} p_i^{e_i}$$

(Αναπαράσταση: βλέπε Αναπαράσταση.pdf)

Πιστοποίηση Πρώτου

- Στηριζόμενοι στον ορισμό ενός πρώτου μπορούμε να γράψουμε έναν (πρωτόγονο) αλγόριθμο που να ελέγχει αν ένας ακέραιος είναι πρώτος ή σύνθετος.
- Αν στην είσοδο έχουμε έναν ακέραιο n>1, δεν έχουμε παρά να ελέγξουμε (δοκιμάσουμε) όλους τους ακεραίους μεταξύ του 2 και του n – 1.
- Αν κάποιος απ' αυτούς διαιρεί τον n τότε ο n δεν είναι πρώτος· αν κανένας απ' αυτούς δεν διαιρεί τον n τότε ο n πρέπει να είναι πρώτος (μέθοδος δοκιμαστικών διαιρέσεων).
- Θα υποθέσουμε ότι ο ακέραιος στην είσοδο είναι > 2 και θα χρησιμοποιήσουμε το t ως τη μεταβλητή για τους δοκιμαστικούς διαιρέτες.
- Ο Αλγόριθμος **Πιστοποίηση Πρώτου #1** είναι ο ακόλουθος:

Αλγόριθμος Πιστοποίηση Πρώτου[1]

➢ **Άσκηση 2**

- Ο αλγόριθμος αυτός είναι άμεση εφαρμογή του ορισμού ενός πρώτου και είναι σίγουρο ότι πιστοποιεί αν ο ακέραιος n είναι πρώτος ή όχι.
- Ο βρόχος επανάληψης είναι σίγουρο ότι θα τερματίσει μετά από το πολύ n – 2 επαναλήψεις.
- Η **χειρότερη περίπτωση** εμφανίζεται όταν ο n είναι πρώτος.
- Η **καλύτερη περίπτωση** εμφανίζεται όταν ο n είναι άρτιος (μία μόνο επανάληψη του βρόχου).

```
Begin
  t ← 1;

  Repeat
    t ← t + 1
  Until (t|n) or (t = n – 1);

  If (t|n) Then
    Output(t,"is a proper divisor of",n)
  Else
    Output(n,"is prime")
  End;
End.
```

Παράδειγμα

- Ας υποθέσουμε ότι θέλουμε να ελέγξουμε έναν ακέραιο n με 25 (δεκαδικά) ψηφία σε μια μηχανή που μπορεί να κάνει 10^9 επαναλήψεις του βρόχου ανά δευτερόλεπτο. Πάσο χρόνο νομίζετε ότι θα απαιτήσει ο αλγόριθμος?
- Αν ο n έχει 25 ψηφία, τότε $10^{24} = 1\underbrace{000...000}_{24\ \text{μηδενικά}} \le n \le \underbrace{9\,999...999}_{25\ \text{εννιάρια}} = 10^{25} - 1$
- Αν ο n ήταν πρώτος, το πλήθος των επαναλήψεων που είναι να γίνουν θα ήταν $n - 2 \ge 10^{24} - 2 > 10^{24} - 10^{23} = (10-1)10^{23} = 9 \times 10^{23}$
- Αυτό θα απαιτούσε περισσότερο από

$$9 \times 10^{23} / 10^9 \text{ sec} = 9 \times 10^{14} \text{ sec} = 900 \times 10^{12} / 60 \text{ min}$$
$$= 15 \times 10^{12} \text{ min} = 1500 \times 10^{10} / 60 \text{ ώρες}$$
$$= 25 \times 10^{10} \text{ ώρες} = 25 \times 10^{10} / 24 \text{ ημέρες}$$
$$> 10^{10} \text{ ημέρες} = 1000 \times 10^{7} / 365.2 \text{ έτη}$$
$$> 2 \times 10^{7} \text{ έτη} = 20\,000\,000 \text{ έτη}$$

Αλγόριθμος Πιστοποίηση Πρώτου[2]

- Πόσο μεγάλος μπορεί να είναι ένας (γνήσιος) διαιρέτης του n?
- Αν είναι $n = a \cdot b$ και $1 < a < n$, τότε $2 \leq a$, οπότε $2 \cdot b \leq a \cdot b = n$.
- Άρα, $b \leq n/2$.
- Επομένως μπορούμε να τερματίσουμε την αναζήτηση για έναν (γνήσιο) διαιρέτη του n μετά από $t = n/2$ διαιρέσεις, ή επειδή το t πρέπει να είναι ακέραιος, όταν το t πάρει την τιμή $\lfloor n/2 \rfloor$.

Προκύπτει λοιπόν η εξής βελτιωμένη μορφή του αλγορίθμου μας

```
Begin
  t ← 1;

  Repeat
    t ← t + 1
  Until (t|n) or (t = ⌊n/2⌋);

  If (t|n) Then
    Output(t, "is a proper divisor of", n)
  Else
    Output( n, "is prime" )
  End;
End.
```

Αλγόριθμος Πιστοποίηση Πρώτου[3]

- Η 2η εκδοχή του αλγορίθμου μας θα εκτελεστεί περίπου δύο φορές ταχύτερα από την 1η εκδοχή στη χειρότερη περίπτωση, αλλά ο έλεγχος ενός 25-ψήφιου ακεραίου θα απαιτούσε περισσότερο από 10.000.000 έτη. Επανερχόμαστε λοιπόν!
- Ο αλγόριθμος #2 στην πραγματικότητα αναζητεί τον **μικρότερο** (γνήσιο) διαιρέτη του n. Πόσο μεγάλος μπορεί να είναι ο μικρότερος (γνήσιος) διαιρέτης του n?
- Αν $n = a \cdot b$ και $1 < a \leq b < n$, τότε $a \leq b$, οπότε $a \cdot a \leq a \cdot b = n$. Άρα,

$$a \leq \sqrt{n} \Leftrightarrow a \leq \lfloor \sqrt{n} \rfloor$$

Παίρνουμε λοιπόν τον αλγόριθμο #3

Άσκηση 3

```
Begin
  t ← 1;

  Repeat
    t ← t + 1
  Until (t|n) or (t = ⌊√n⌋);

  If (t|n) Then
    Output(t,"is a proper divisor of",n)
  Else
    Output(n,"is prime")
  End;
End.
```

Αλγόριθμος Πιστοποίηση Πρώτου[4]

- Αυτή η εκδοχή του αλγόριθμου είναι ταχύτερη όταν ο *n* είναι πρώτος, αλλά πόση βελτίωση έχει γίνει?
- Αν ο *n* έχει 25 ψηφία, το πλήθος των επαναλήψεων του βρόχου στον αλγόριθμο #3 θα είναι μικρότερο από

$$\sqrt{n} \text{ που είναι μικρότερο από } \sqrt{10^{25}} = 10^{12} \times \sqrt{10}$$

- Αυτό θα απαιτήσει λιγότερο από $\left(10^{12} \times \sqrt{10}\right) / 10^{9} \text{ sec} = 10^{3} \times \sqrt{10} \text{ sec}$

$$= 1000 \times (3.162277\ldots) \text{sec}$$
$$< 3163 \text{sec}$$
$$= 52.7166\ldots \text{min}$$
$$< 53 \text{min}$$

- Είναι εύλογο να σκεφτούμε ότι μπορούμε να σχεδιάσουμε ταχύτερο έλεγχο?
- Δεν έχουμε παρά να αναλογιστούμε ότι μέχρι τώρα στηριχθήκαμε μόνο στον ορισμό ενός πρώτου αριθμού, ενώ η Θεωρία Αριθμών μας παρέχει μια πληθώρα ιδιοτήτων των πρώτων αριθμών…

Ανάλυση σε Γινόμενο Πρώτων Παραγόντων

- Ο αλγόριθμος πιστοποίησης πρώτου θα βρει τον μικρότερο (γνήσιο) διαιρέτη του ακεραίου *n* που ως γνωστόν πρέπει να είναι πρώτος – άρα πρέπει να είναι ο p_1 (βλ. Θ.2) – ή θα μας πει ότι ο *n* είναι πρώτος.
- Μπορούμε λοιπόν να υιοθετήσουμε την εξής στρατηγική:
  - Βρίσκουμε τον μικρότερο πρώτο παράγοντα p του n //$p = n$ ή $p = \lfloor \sqrt{n} \rfloor$
  - Αν $p=n$ τότε STOP //έχουμε βρει όλους τους πρώτους παράγοντες του n
  - Διαφορετικά, έστω $Q \leftarrow n$ DIV p

    τώρα η παραγοντοποίηση του *n* είναι

    $p \times$(πρώτοι παράγοντες του Q)

  //Γνωρίζουμε επίσης ότι ο μικρότερος πρώτος παράγοντας του *Q* είναι το πολύ *p*,
  //και επομένως, αν $p > \lfloor \sqrt{Q} \rfloor$, τότε ο Q είναι επίσης πρώτος.
- Παραθέτουμε σε μορφή ψευδοκώδικα τον αλγόριθμο. Εμπεριέχει τον έλεγχο για πρώτο (Αλγ.#3) και χρησιμοποιούμε μια μεταβλητή Q για τον ακέραιο που μένει να παραγοντοποιηθεί.

Αλγόριθμος Παραγοντοποίησης σε Πρώτους[1]

```
Begin
  Q ← n;
  t ← 2;

  While (t <= ⌊√Q⌋) Do
    If (t|Q) Then
      Output(t, "×");
      Q ← Q DIV t
    Else
      t ← t + 1
    End;
  End;

  If (Q = n) Then
    Output(n, "is prime")
  Else
    Output(Q, " = ", n)
  End;
End.
```

// Σε κάθε επανάληψη του βρόχου-While είτε
// μειώνεται το Q είτε αυξάνει το t (αλλά όχι και τα
// δύο), οπότε τελικά $t>\lfloor\sqrt{Q}\rfloor$ και η συνθήκη ελέγχου
// του βρόχου καθίσταται ψευδής.

// Για θετικούς ακεραίους t και Q, η ($t \le \lfloor\sqrt{Q}\rfloor$)
// ισοδυναμεί με την ($t \times t \le \lfloor Q \rfloor$), που είναι πιο
// κατάλληλη για ένα πρόγραμμα.

- Διαφορά μεταξύ ενός βρόχου-While και ενός βρόχου-Repeat ?
  - το σώμα του βρόχου-Repeat εκτελείται πάντα (τουλάχιστο) μία φορά, αλλά αν η συνθήκη ενός βρόχου-While δεν ικανοποιείται την πρώτη φορά που ελέγχεται, το σώμα του βρόχου δεν εκτελείται ποτέ. Στην περίπτωσή μας αυτό θα συμβεί αν στην είσοδο ο ακέραιος n είναι 2 ή 3.
  - **Άσκηση 4**

Αλγόριθμος Παραγοντοποίησης σε Πρώτους[2]

- Ο Αλγόριθμος Παραγοντοποίησης σε Πρώτους τερματίζει και είναι ορθός. Το πλήθος των δοκιμαστικών διαιρέσεων είναι το πολύ $\lfloor \sqrt{n} \rfloor - 1$· η χειρότερη περίπτωση συμβαίνει όταν δεν βρίσκονται πρώτοι παράγοντες επειδή ο n είναι πρώτος.
- Τελικά πόσοι πρώτοι παράγοντες του n υπάρχουν?
- Αν θεωρήσουμε ότι $n = p_1 \times p_2 \times \ldots \times p_k$ πόσο μεγάλο μπορεί να είναι το k?

➢ Οι λογάριθμοι κάνουν την εμφάνισή τους!

- Έχουμε, $n = p_1 \times p_2 \times \ldots \times p_k$ // $\forall i,\ p_i \geq 2$

$$\geq 2 \times 2 \times \ldots \times 2 = 2^k \Rightarrow$$

$$\lg n \geq \lg(2^k) \Rightarrow k \lg 2 \leq \lg n \Rightarrow k \leq \lg n \Leftrightarrow \mathbf{k \leq \lfloor \lg n \rfloor}.$$

Αλγόριθμος Παραγοντοποίησης σε Πρώτους[3]

- Η λογαριθμική συνάρτηση (με βάση 2) είναι (↑) αλλά με πολύ αργό ρυθμό αύξησης.
- Αν η συνάρτηση πολυπλοκότητας ενός αλγόριθμου (το πλήθος των βημάτων που απαιτεί για να τερματίσει όταν η είσοδος είναι μεγέθους n) είναι λογαριθμική, ο αλγόριθμος θα είναι **πολύ αποδοτικός**.
- Τέτοιοι αλγόριθμοι είναι ο ΠΡΧ και ο αλγόριθμος διχοτόμησης (εύρεση ριζών συνάρτησης).

n	$\sqrt{n}$	$\lg(n)$
4	2	2
16	4	4
64	8	6
256	16	8
1024	32	10
4096	64	12
16384	128	14
65536	256	16
262 144	512	18
1 048 576	1024	20

Αλγόριθμος του Ευκλείδη[1]

- Θέλουμε έναν αποδοτικό αλγόριθμο για την εύρεση του Μέγιστου Κοινού Διαιρέτη δύο θετικών ακεραίων x και y, συμβολικά gcd(x,y).

  // θα είναι gcd(x,y) = gcd(y,x)?
- Έστω $x \geq y \geq 1$. Επειδή $1|x$ και $1|y$, και αφού ο y είναι ο μεγαλύτερος ακέραιος που διαιρεί τον y, θα είναι $1 \leq \gcd(x,y) \leq y$.
- Αν $y|x$, τότε (επειδή $y|y$) gcd(x,y) = y.

// αυτή η περίπτωση είναι εύκολη και δεν περιλαμβάνει αναζήτηση
// διαιρετών. Τι γίνεται με την άλλη περίπτωση?

- Διαφορετικά ($y \nmid x$) $x > y$ και

  $x = yq + r$, $0 < r < y$ //$q \geq 1$ και $r = x \bmod y$
- Ο Ευκλείδης απέδειξε ότι σ' αυτή την περίπτωση,

  gcd(x,y) = gcd(y,r) //θα το αποδείξουμε στη συνέχεια του μαθήματος

// πρόκειται για ένα παράδειγμα *αναδρομής*· ο ΜΚΔ των x και y θα βρεθεί (εν καιρώ)
// βρίσκοντας τον ΜΚΔ δύο άλλων (αλλά μικρότερων) ακεραίων, y και r.

Αλγόριθμος του Ευκλείδη[2]

- Μπορούμε να κατασκευάσουμε γι' αυτό το πρόβλημα έναν επαναληπτικό αλγόριθμο παρατηρώντας ότι ο ΜΚΔ είναι μια συνάρτηση με δύο «παραμέτρους», έστω *A* και *B*. Τις *αρχικοποιούμε με* (αναθέτουμε αρχικές τιμές) τους ακεραίους x και y, αντίστοιχα· ανανεώνονται οποτεδήποτε υπολογίζουμε ένα θετικό υπόλοιπο.

❑ Αλγόριθμος Ευκλείδη

```
Begin
  A ← x;
  B ← y;
  R ← A MOD B;
  While ( R > 0 ) Do
    A ← B;
    B ← R;
    R ← A MOD B;
  End;
  Output("GCD(", x, ",", y, ")=", B);          // ή Return(B)
End.
```

- ***Άσκηση 5***

Αλγόριθμος του Ευκλείδη[3]

- Είναι εγγυημένο ότι ο αλγόριθμος αυτός θα τερματίζει?
- Αυτός ο αλγόριθμος παράγει μια ακολουθία ακέραιων τιμών για τα *A*, *B* και *R*, όπου

$A_1 = x,$ $B_1 = y$ και $0 \le R_1 < y = B_1$.

αν $0 < R_1$, $A_2 = B_1,$ $B_2 = R_1$ και $0 \le R_2 < R_1 = B_2$. και

αν $0 < R_2$, $A_3 = B_2 = R_1,$ $B_3 = R_2$ και $0 \le R_3 < R_2 = B_3$.

- Οι τιμές-*R* ελαττώνονται, αλλά ποτέ δεν γίνονται αρνητικές, οπότε τελικά κάποιο $R_k = 0$.

// πόσο μεγάλο μπορεί να είναι αυτό το k?

Αλγόριθμος του Ευκλείδη[4]

- Θα δούμε αργότερα ότι αν απαιτούνται k επαναλήψεις, τότε

$$y \geq \left(\frac{1+\sqrt{5}}{2}\right)^k \quad \text{και απ' αυτό,} \quad k \leq \left\lfloor \frac{3}{2} \lg y \right\rfloor$$

// Παράξενο? Είναι δυνατό η συνάρτηση πολυπλοκότητας του Αλγόριθμου του
//Ευκλείδη να σχετίζεται με έναν τέτοιο παράξενο αριθμό?

- Ο Αλγόριθμος του Ευκλείδη είναι πολύ αποδοτικός. Δουλεύει κι όταν $x < y$, αλλά τότε απαιτεί μία επιπλέον επανάληψη. Πώς? Γιατί?

Τέλος Ενότητας

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης