ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ (2014-15)

Άσκηση 1

Ένας ακέραιος αριθμός μεγαλύτερος του 1 ονομάζεται *πρώτος* (prime) όταν έχει σαν μόνους διαιρέτες το 1 και τον εαυτό του. Για παράδειγμα, το 13 είναι πρώτος αριθμός, ενώ το 15 ($= 3 \times 5$) δεν είναι. Δεν είναι ιδιαίτερα δύσκολο να γράψουμε έναν αλγόριθμο που να ελέγχει αν ένας αριθμός είναι πρώτος, ο οποίος θα βασίζεται στον ορισμό για το πότε ένας αριθμός είναι πρώτος, μόνο που για μεγάλους αριθμούς ο αλγόριθμος αυτός θα είναι εξαιρετικά δαπανηρός σε χρόνο. Θα ονομάσουμε ένα τέτοιο αλγόριθμο *ντετερμινιστικό* (deterministic), υπονοώντας ότι μπορεί να αποφασίσει με απόλυτη βεβαιότητα αν ένας αριθμός είναι πρώτος ή όχι.

Από την άλλη πλευρά, υπάρχει και η μέθοδος του Fermat για τον έλεγχο αν ένας αριθμός είναι πρώτος. Σύμφωνα με μαθηματικό θεώρημα που διατύπωσε ο Fermat, αν ένας αριθμός p είναι πρώτος, τότε για κάθε ακέραιο a, τέτοιο ώστε $1 \leq a < p$, ισχύει ότι $a^{p-1} \bmod p = 1$ (με mod συμβολίζουμε το υπόλοιπο διαίρεσης). Οπότε, αν για δεδομένο p υπολογίσουμε το $a^{p-1} \bmod p$, για κάποιο θετικό ακέραιο a μικρότερο του p, και αυτό δεν ισούται με 1, τότε ο p δεν είναι πρώτος. Αν ισούται με 1, τότε **πιθανώς** είναι πρώτος. Αν κάνουμε αυτό τον έλεγχο για περισσότερα a, και για όλα αυτά το $a^{p-1} \bmod p$ ισούται με 1, τότε η πιθανότητα ο p να είναι πραγματικά πρώτος είναι εξαιρετικά μεγάλη, σχεδόν ίση με τη μονάδα. Αυτή η μέθοδος που μόλις περιγράψαμε δεν είναι τίποτε άλλο από έναν *πιθανοτικό* (probabilistic) αλγόριθμο για να ελεγχθεί αν ένας αριθμός είναι πρώτος. Υπάρχει ενδεχόμενο να χαρακτηρίσει σαν πρώτο έναν αριθμό p που δεν είναι πραγματικά (Fermat liar), αλλά αν τον εφαρμόσουμε αρκετές φορές, για διάφορα a, η πιθανότητα να συμβεί κάτι τέτοιο είναι εξαιρετικά μικρή.

Γράψτε ένα πρόγραμμα C (έστω ότι το πηγαίο αρχείο του ονομάζεται "`fermat.c`") το οποίο να υπολογίζει το πλήθος των πρώτων αριθμών που βρίσκονται μέσα σε δεδομένο διάστημα, τόσο ντετερμινιστικά, όσο και πιθανοτικά. Το διάστημα να καθορίζεται από δύο συμβολικές σταθερές `MINNUM` και `MAXNUM` που θα ορίσετε μέσα στο πρόγραμμά σας. Να ορίσετε επίσης και μία συμβολική σταθερά `MAXTRIES`, με τιμή ίση με το πλήθος των φορών που θα εκτελεί το πρόγραμμά σας τον πιθανοτικό αλγόριθμο για κάθε αριθμό του δεδομένου διαστήματος. Την πρώτη φορά, για κάθε αριθμό p που ελέγχεται πιθανοτικά αν είναι πρώτος, να επιλέγεται τυχαία μόνο ένα a μικρότερο του p, για την εφαρμογή του ελέγχου του Fermat. Τη δεύτερη φορά, να επιλέγονται δύο τυχαίοι a, για κάθε αριθμό p που ελέγχεται, κοκ. Συνολικά, θα πρέπει να γίνουν `MAXTRIES` έλεγχοι για κάθε αριθμό p, με τον τελευταίο να επιλέγει τυχαία `MAXTRIES` αριθμούς a.

Τόσο ο ντετερμινιστικός αλγόριθμος, όσο και οι `MAXTRIES` εφαρμογές του πιθανοτικού, θα πρέπει να χρονομετρηθούν, ο καθένας συνολικά για όλους τους αριθμούς του διαστήματος που ελέγχεται.

Προτάσεις/Υποδείξεις/Απαγορεύσεις:

1. Ως τιμες των συμβολικών σταθερών `MINNUM`, `MAXNUM` και `MAXTRIES` να θέσετε τις `1990000001`, `2000000000` και `10`, αντίστοιχα. Με αυτές τις τιμές θα ελεγχθεί το πρόγραμμα που θα παραδώσετε ως προς την αποδοτικότητά του.

2. Για να δημιουργείτε τυχαίους αριθμούς a στο διάστημα $[1, p-1]$, να χρησιμοποιήσετε τη γεννήτρια τυχαίων αριθμών της C, μέσω της συνάρτησης `rand`. Η γεννήτρια να αρχικοποιηθεί μέσω της συνάρτησης `srand` με φύτρο τον τρέχοντα χρόνο, όπως επιδεικνύεται στο πρόγραμμα `http://www.di.uoa.gr/~ip/cprogs/gcdlcm.c`, στις σελίδες 44–45 των διαφανειών του μαθήματος, στο `http://www.di.uoa.gr/~ip/K04.pdf`. Λεπτομέρειες για τη χρήση των συναρτήσεων `srand` και `rand` μπορείτε να μάθετε δίνοντας στα μηχανήματα Linux του Τμήματος

"`man 3 srand`" ή "`man 3 rand`" (πιθανότατα, σε άλλα συστήματα Unix, να χρειάζεται "`man srand`" ή "`man rand`").

3. Η χρονομέτρηση των αλγορίθμων να γίνεται μέσω της συνάρτηση `clock`. Περισσότερες λεπτομέρειες για τη χρήση της συνάρτησης αυτής μπορείτε να πάρετε μέσω της εντολής "`man 3 clock`" στα μηχανήματα Linux του Τμήματος, ή να δείτε τη χρήση της στο πρόγραμμα `http://www.di.uoa.gr/~ip/cprogs/sorting.c`, στις σελίδες 167–169 των διαφανειών του μαθήματος.

4. Είναι προφανές ότι αν δοκιμάσετε να υπολογίσετε το $a^{p-1} \bmod p$ για μεγάλα p και a υπολογίζοντας πρώτα το a^{p-1}, το αποτέλεσμα δεν θα είναι σωστό λόγω υπερχείλισης. Για να λύσετε αυτό το πρόβλημα, σκεφτείτε ότι ισχύουν τα εξής:[1]

$$x^{2n+1} \bmod m = (x \cdot (x^{2n} \bmod m)) \bmod m$$

$$x^{2n} \bmod m = (x^2 \bmod m)^n \bmod m$$

5. Για να μπορέσετε να εφαρμόσετε τον πιθανοτικό αλγόριθμο για τους αριθμούς του διαστήματος που ορίζεται από τις τιμές των συμβολικών σταθερών που προτείνονται, θα χρειαστείτε, τουλάχιστον σε κάποιες ενδιάμεσες πράξεις, να χρησιμοποιήσετε ακεραίους των 8 bytes, κάτι που στη C σας εξασφαλίζει μόνο η χρήση του τύπου `long long`.

6. Στην άσκηση αυτή απαγορεύεται αυστηρά η χρήση πινάκων.

Μία ενδεικτική εκτέλεση[2] του προγράμματος φαίνεται στη συνέχεια:

```
linux29$ ./fermat
Checking range [1990000001,2000000000] for primes

Deterministic algorithm: Found 466646 primes in 90.07 secs

Trying Fermat test with seed 1414258936

Probabilistic algorithm: Found 466668 primes in 11.58 secs (tries = 1)
Probabilistic algorithm: Found 466648 primes in 12.13 secs (tries = 2)
Probabilistic algorithm: Found 466647 primes in 12.66 secs (tries = 3)
Probabilistic algorithm: Found 466647 primes in 13.19 secs (tries = 4)
Probabilistic algorithm: Found 466647 primes in 13.73 secs (tries = 5)
Probabilistic algorithm: Found 466646 primes in 14.27 secs (tries = 6)
Probabilistic algorithm: Found 466646 primes in 14.79 secs (tries = 7)
Probabilistic algorithm: Found 466647 primes in 15.34 secs (tries = 8)
Probabilistic algorithm: Found 466646 primes in 15.84 secs (tries = 9)
Probabilistic algorithm: Found 466646 primes in 16.38 secs (tries =10)
linux29$
```

Η παράδοση της άσκησης αυτής συνίσταται στην υποβολή του πηγαίου αρχείου `fermat.c` με διαδικασία που θα ανακοινωθεί σύντομα.

---

[1] Δείτε και το `http://en.wikipedia.org/wiki/Modular_exponentiation#Right-to-left_binary_method`

[2] Η συγκεκριμένη εκτέλεση έγινε σε μηχάνημα Linux του Τμήματος. Σε άλλες πλατφόρμες (π.χ. PC/Dev-C++) τα αποτελέσματα ενδέχεται να διαφέρουν, διότι η γεννήτρια τυχαίων αριθμών μπορεί να παράγει διαφορετική ακολουθία αριθμών, ακόμα και με το ίδιο "φύτρο".