factor

Παραγοντοποίηση

Σε αυτήν την άσκηση, θα ασχοληθούμε (και πάλι!) με ένα
πρόβλημα που είχαμε δει στο Δημοτικό, το πρόβλημα της παραγοντοποίησης!
Μάθαμε στα διακριτά μαθηματικά πρόσφατα, ότι κάθε μη πρώτος (σύνθετο-
ς/composite) αριθμός, μπορεί να γραφτεί σαν γινόμενο δύο ή περισσότερων πρώτων
αριθμών. Επομένως η παραγοντοποίηση (factorization) ενός φυσικού αριθμού
n είναι το πρόβλημα της εύρεσης των πρώτων παραγόντων που το γινόμενό τους
μας δίνει το n. Για παράδειγμα, η παραγοντοποίηση του αριθμού 42 είναι 2 ·3 ·7.
Συγκεκριμένα, σε αυτήν την άσκηση θα ασχοληθούμε με την παραγοντοποίηση μιας
συγκεκριμένης κατηγορίας σύνθετων αριθμών, τους ημιπρώτους (semiprimes).

Ορισμός 6. Ένας φυσικός αριθμός λέγεται ημιπρώτος (semiprime), όταν είναι το
γινόμενο ακριβώς δύο πρώτων αριθμών. Για παράδειγμα, ο αριθμός 46 είναι semiprime
(2 ·23) όπως και ο αριθμός 9 (3 ·3). Αντίθετα, ο αριθμός 42 δεν είναι semiprime.
Το πρόβλημα της παραγοντοποίησης ημιπρώτων φαίνεται απλό: το μόνο που
πρέπει να κάνουμε είναι να βρούμε δύο αριθμούς που το γινόμενό τους να μας δίνει
τον αριθμό με τον οποίο ξεκινήσαμε. Παρόλα αυτά, κανένας/καμία δεν έχει βρει —
μέχρι στιγμής! — μια αποδοτική λύση σε αυτό το πρόβλημα παρόλα τα χρηματικά
έπαθλα που έχουν ανακυρηχθεί. Από την μία, αυτό είναι καλό, η ασφάλεια
της κρυπτογράφησης που είδαμε στον αλγόριθμο RSA παραπάνω στηρίζεται στο ότι
η παραγοντοποίηση μεγάλων αριθμών είναι ένα δύσκολο πρόβλημα και επομένως
μπορούμε να συνεχίσουμε να χρησιμοποιούμε αυτόν τον αλγόριθμο για να μπαίνουμε
στα αγαπημένα μας site. Από την άλλη, συνεχίζουμε και ψάχνουμε για πιο γρήγορες
λύσεις, μιας και υποψιαζόμαστε ότι είναι πολύ πιθανό ένας γρήγορος αλγόριθμος να
υπάρχει και απλά να μην έχουμε βρει ακόμα!
Σε αυτήν την άσκηση λοιπόν, καλείστε να γράψετε ένα αποδοτικό πρόγραμμα το
οποίο να παραγοντοποιεί ημιπρώτους (semiprimes). Ακολουθούν οι τεχνικές προδια-
γραφές.
Τεχνικές Προδιαγραφές
• C Filepath: factor.c
• Το πρόγραμμά θα πρέπει να παίρνει ένα όρισμα από την γραμμή εντολών στην
μορφή ./factor semiprime, με το πρώτο (semiprime) να είναι ο ημιπρώτος που
θέλουμε να παραγοντοποιήσουμε. Αν το πρόγραμμα εκτελεστεί με ορίσματα που
δεν ακολουθούν τις παραπάνω προδιαγραφές, πρέπει να εκτυπώσει αντίστοιχο
μήνυμα όπως στα παρακάτω παραδείγματα και να επιστρέφει με κωδικό εξόδου
(exit code) 1.

• Όλοι οι δεκαδικοί ακέραιοι που θα δοθούν στο πρόγραμμά σας θα είναι ημιπρώτοι και στο εύρος: [0,2127]. Για οποιαδήποτε άλλη είσοδο, το πρόγραμμά σας πρέπει να τερματίζει με κωδικό εξόδου 1.
• Το αρχείο C που θα υποβληθεί πρέπει να μεταγλωττίζεται χωρίς ειδοποιήσεις για λάθη και με κωδικό επιστροφής (exit code) που να είναι 0.
`gcc -O3 -Wall -Wextra -Werror -o factor factor.c -lm`
• Μορφοποίηση: το αρχείο που θα υποβληθεί πρέπει να έχει μορφοποίηση σύμφωνη με το C/C++ στυλ της Google. Για να μορφοποιήσετε το αρχείο σας, μπορείτε να τρέξετε την ακόλουθη εντολή: `clang-format -i -style=Google factor.c`
• README Filepath: factor/README.md
• Ένα αρχείο που να περιέχει ένα στοιχείο εισόδου και ένα εξόδου *διαφορετικά* από αυτά της εκφώνησης. Συγκεκριμένα προτείνουμε να βάλετε έναν συνδυασμό που θεωρείται ότι είναι δύσκολος να γίνει σωστός κατά την υλοποίηση.
– input Filepath: factor/test/input.txt
– output Filepath: factor/test/output.txt
Για παράδειγμα, το περιεχόμενο του input.txt αρχείου μπορεί να είναι: "93" και του αντίστοιχου output.txt: "3 31". Προσοχή: αυτό το παράδειγμα δεν θα γίνει δεκτό από την άσκηση επειδή αυτό το input-output ζευγάρι υπάρχει ήδη παρακάτω, επομένως πρέπει να διαλέξετε κάποιο άλλο.
• Πρέπει να ολοκληρώνει την εκτέλεση μέσα σε: 10 δευτερόλεπτα.
• Δεν επιτρέπεται χρήση προϋπολογισμένων αποτελεσμάτων. Το πρόγραμμά σας πρέπει να υπολογίζει το αποτέλεσμα χωρίς "πρότερη γνώση", δηλαδή χωρίς να έχετε ήδη κωδικοποιήσει παραγοντοποιήσεις ημιπρώτων που έχετε βρει από προηγούμενους υπολογισμούς σας μέσα στον κώδικα.
• Καθώς αυτή η εργασία είναι Bonus, θα ελέγξουμε και ποιοτικά χαρακτηριστικά της υποβολής. Μια ιδιαίτερα δυσνόητη ή μη διαχειρίσιμη λύση (π.χ., 6 nested if-else, 35 μεταβλητές στην ίδια συνάρτηση κτλ) θα οδηγήσει σε αφαίρεση μονάδων κατά την κρίση του εξεταστή.

Στο αρχείο README.md πρέπει να προσθέσετε οποιεσδήποτε παρατηρήσεις κάνατε κατά την διεκπεραίωση της άσκησης. Ο κώδικας απαιτείται να είναι καλά τεκμηριωμένος με σχόλια καθώς αυτό θα είναι μέρος της βαθμολόγησης. Οι 10 γρηγορότερες λύσεις θα μοιραστούν ένα bonus (extra) 500 μονάδων κατανεμημένων αναλογικά με τον παράγοντα $\frac{1}{T}$ (μέχρι το max: 200 μονάδες ανά υποβολή), όπου T είναι ο συνολικός χρόνος που απαιτεί ο αλγόριθμος της υλοποίησης για να παραγοντοποιήσει όλους τους ακεραίους που του δόθηκαν. Αν δούμε μια ιδιαίτερα ενδιαφέρουσα/αποδοτική λύση, θα ζητήσουμε μια παρουσίαση από το παιδί που την υλοποίησε.

Παρακάτω παραθέτουμε την αλληλεπίδραση με μια ενδεικτική λύση:

```
$ ./factor 93
Factors: 3 31
$ echo $?
0
$ ./factor 9827348119
Factors: 613 16031563
$ # level: very hard
$ ./factor 2524891914334062643
Factors: 1175747593 2147477851
$ # level: extremely hard
$ ./factor 809724910412139638697047
Factors: 783108713587 1033987869581
$ # level: this is impossible
$ ./factor 661621452399004520128701898758039 61
Factors: 206547667773749927 320323855277677343
12
```