Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών
Τμήμα Πληροφορικής και Τηλεπικοινωνιών

Κατατακτήριες Εξετάσεις 2024

Εισαγωγή στον Προγραμματισμό (09/12/2024)

Απαντήστε και στα 4 θέματα, τα οποία είναι βαθμολογικά ισοδύναμα. Τα προγράμματα που θα γράψετε πρέπει να είναι δομημένα, διατυπωμένα ευκρινώς και με επαρκή τεκμηρίωση ώστε να είναι κατανοητά.

Θέμα 1 - Προσεγγίζοντας το π (25 Μονάδες)

Ο Άγγλος μαθηματικός John Wallis θεωρείται πως είναι ένας από τους πρώτους που εισήγαγε την έννοια του απείρου (∞) στα μαθηματικά. Το 1656 δημοσίευσε ένα αποτέλεσμα σύμφωνα με το οποίο μπορούμε να προσεγγίσουμε το π χρησιμοποιώντας το ακόλουθο απειρογινόμενο όρων t_i, γνωστό και ως γινόμενο Wallis προς τιμήν του:

$$\frac{\pi}{2} = \underbrace{\frac{2}{1}\cdot\frac{2}{3}}_{t_1}\cdot\underbrace{\frac{4}{3}\cdot\frac{4}{5}}_{t_2}\cdot\underbrace{\frac{6}{5}\cdot\frac{6}{7}}_{t_3}\cdot\underbrace{\frac{8}{7}\cdot\frac{8}{9}}_{t_4}\cdots$$

Χρησιμοποιώντας την παραπάνω σχέση μπορούμε να προσεγγίσουμε και εμείς το π όπως ο Wallis. Μάλιστα, όσους περισσότερους όρους t_i χρησιμοποιούμε, τόσο καλύτερη η προσέγγισή μας. Γράψτε ένα πρόγραμμα C το οποίο παίρνει έναν θετικό ακέραιο ως όρισμα από την γραμμή εντολών που αντιπροσωπεύει πόσους πρώτους όρους t_i να χρησιμοποιήσει στον υπολογισμό του π και στην συνέχεια εκτυπώνει την προσέγγιση του π με 8 δεκαδικά ψηφία. Παραδείγματα εκτέλεσης ακολουθούν:

```
$ ./wallis
Usage: ./wallis <number of terms to use>
$ echo $?
1
$ ./wallis 1
Using the first 1 terms, pi = 2.66666667
$ ./wallis 2
Using the first 2 terms, pi = 2.84444444
$ ./wallis 4
Using the first 4 terms, pi = 2.97215420
$ ./wallis 100
Using the first 100 terms, pi = 3.13378749
$ ./wallis 100000000
Using the first 100000000 terms, pi = 3.14159264
```

Θέμα 2 - Οι Καλύτεροι Αριθμοί, Παμψηφεί (25 Μονάδες)

Στα μαθηματικά, ένας φυσικός αριθμός n λέγεται *πανψηφιακός* (pandigital) ως προς μια βάση b όταν περιέχει *τουλάχιστον* μία φορά όλα τα ψηφία από το 0 μέχρι το $b-1$. Για παράδειγμα, ο αριθμός 1234567890 είναι πανψηφιακός με βάση το 10 όπως και ο αριθμός 90123456789 (τα ψηφία μπορούν να επαναλαμβάνονται και η σειρά τους δεν έχει σημασία).

Γράψτε ένα πρόγραμμα C το οποίο βρίσκει και τυπώνει όλους τους πανψηφιακούς αριθμούς με βάση το 10 σε ένα εύρος φυσικών αριθμών $[low, high]$ (το εύρος είναι κλειστό, δηλαδή τα άκρα συμπεριλαμβάνονται), όπου οι αριθμοί δίνονται από την γραμμή εντολών. Οι αριθμοί του εύρους δεν θα υπερβούν το $2^{64}-1$. Ακολουθεί παράδειγμα εκτέλεσης για να βρούμε όλους τους πανψηφιακούς αριθμούς στο διάστημα $[1000000000, 1300000000]$:

```
$ ./pandigital 1000000000 1300000000
All pandigital numbers in the range 1000000000 to 1300000000 follow:
1023456789
1023456798
1023456879
1023456897
...
1298765304
1298765340
1298765403
1298765430
```

Θέμα 3 - Η Τριπλέτα Στόχος (25 Μονάδες)

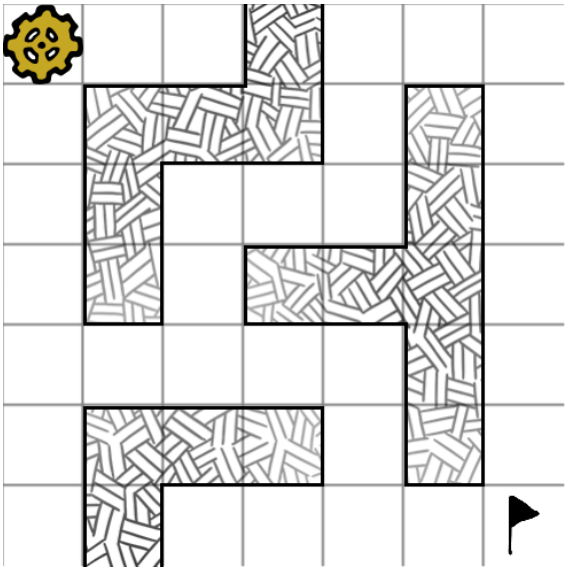
Γράψτε ένα πρόγραμμα το οποίο παίρνει ως όρισμα από την γραμμή εντολών έναν ακέραιο-στόχο (goal) και από την πρότυπη είσοδο (stdin) ένα σύνολο διαφορετικών υποψηφίων ακεραίων (candidates) και τυπώνει όλες τις πιθανές τριπλέτες υποψηφίων ακεραίων των οποίων το άθροισμα ισούται με τον στόχο. Ο κάθε υποψήφιος ακέραιος μπορεί να χρησιμοποιηθεί μέχρι μία φορά σε κάθε τριπλέτα. Για παράδειγμα αν δοθεί ο στόχος 10 και οι ακέραιοι 6, 1, 7, 2, 3 ως υποψήφιοι για τις τριπλέτες υπάρχουν ακριβώς δύο συνδυασμοί που οδηγούν στον στόχο: $1+3+6=10$ και $1+2+7=10$ (ο συνδυασμός $6+2+2=10$ *δεν* είναι δεκτός καθώς ο ακέραιος 2 επαναλαμβάνεται).

Ο αλγόριθμός σας πρέπει να έχει χρονική πολυπλοκότητα γρηγορότερη από $O(n^3)$, όπου n είναι ο αριθμός των ακεραίων που σας δίνονται. Παραδείγματα εκτέλεσης ακολουθούν:

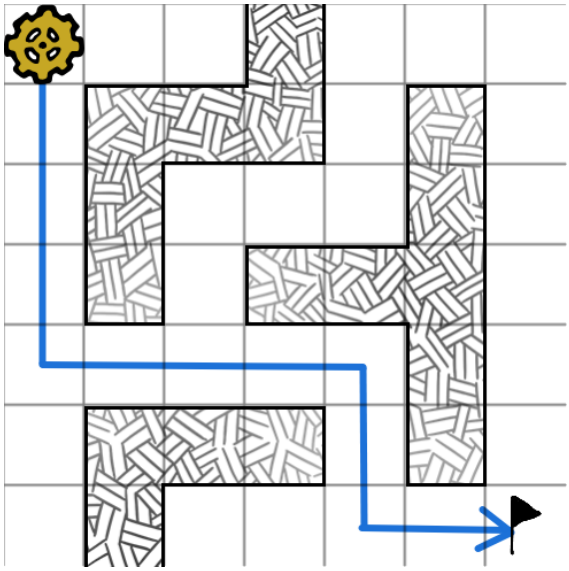
```
$ ./triplet 10
Provide the numbers: 6 1 7 2 3
Triplet found: 1 + 2 + 7 = 10
Triplet found: 1 + 3 + 6 = 10
$ ./triplet 42
Provide the numbers: 1 2 3
No triplet leads to 42
$ ./triplet 42
Provide the numbers: 19 21 3 5 12 11
Triplet found: 11 + 12 + 19 = 42
```

```
$ ./triplet 42
Provide the numbers: 42 27 25 12 31 5 26 40 34 3 18
Triplet found: 3 + 5 + 34 = 42
Triplet found: 3 + 12 + 27 = 42
Triplet found: 5 + 12 + 25 = 42
```

Θέμα 4 - Λύσε τον Λαβύρινθο (25 Μονάδες)



(i) Αρχικός λαβύρινθος 7x7, ξεκινάμε στο (0, 0) και θέλουμε να φτάσουμε στο (6, 6).



(ii) Ενδεικτική λύση. Επιτρέπονται κινήσεις μόνο προς τα κάτω (D) και δεξιά (R).

Σχήμα 1: Οπτικοποίηση ενός λαβυρίνθου και της λύσης του. Παρατηρήστε ότι το ρομπότ μας ξεκινάει πάντα την διαδρομή του στο (0,0) και ολοκληρώνει την διαδρομή του στο (N-1, N-1) όπου N η διάσταση του λαβυρίνθου.

Σε αυτό το πρόβλημα θα προγραμματίσουμε ένα ρομπότ το οποίο μπορεί να βρίσκει την έξοδο σε μια κατηγορία τετραγωνικών λαβυρίνθων, όπου η είσοδος είναι πάνω αριστερά και η έξοδος κάτω δεξιά. Για λόγους οικονομίας, το ρομπότ μας μπορεί να κινηθεί μόνο προς τα κάτω (D - Down) ή προς τα δεξιά (R - Right) σε κάθε βήμα. Το Σχήμα 1 δείχνει ένα παράδειγμα με το ρομπότ μας να βρίσκει το μονοπάτι DDDDRRRRDDRR και να φτάνει στην έξοδο.

Γράψτε ένα πρόγραμμα C που διαβάζει τον πίνακα με τα περιεχόμενα του λαβυρίνθου και τυπώνει (εφόσον υπάρχει) ένα βέλτιστο μονοπάτι για να φτάσει το ρομπότ μας στην έξοδο. Το πρόγραμμά σας πρέπει να διαβάζει τον λαβύρινθο από την πρότυπη είσοδο όπου θα δίνονται: (1) στην πρώτη γραμμή η διάσταση του τετραγωνικού πλέγματος, και (2) στην συνέχεια τα περιεχόμενα του λαβυρίνθου όπου "1" θα συμβολίζει ένα κελί με τοίχο και "0" θα συμβολίζει το κενό. Το πρόγραμμά σας πρέπει να είναι αποδοτικό σε χρονική και χωρική πολυπλοκότητα. Αφού ολοκληρώσετε το πρόγραμμά σας, καταγράψτε και εξηγήστε την πολυπλοκότητά του ως προς τον χρόνο και τον χώρο μνήμης που απαιτεί. Ακολουθούν ενδεικτικές εκτελέσεις:

```
$ cat maze.txt
7
0001000
0111010
0100010
0101110
0000010
0111010
0100000
$ ./robot < maze.txt
Path: DDDDRRRRDDRR
$ cat nopath.txt
7
0001000
0111010
0100010
0101110
0000010
0111110
0100000
$ ./robot < nopath.txt
No path found
```