ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Εαρινό 2022

ΓΛΩΣΣΕΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ Ι

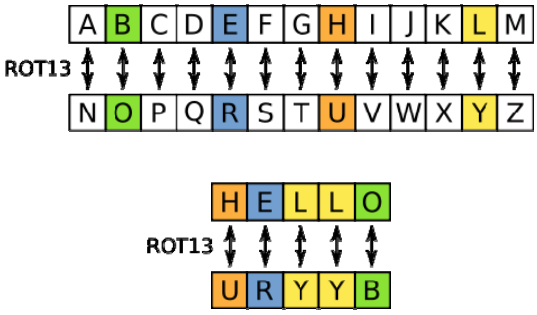
Άσκηση 1

Καταληκτική ημερομηνία και ώρα ηλεκτρονικής υποβολής: 9/5/2022, 23:59:59

Αποκρυπτογράφηση (0.25+0.25 = 0.5 βαθμοί)

Ένας παλιός και πολύ απλός τρόπος κρυπτογράφησης ενός μηνύματος είναι ο ROTN cipher (γνωστός και ως Ceasar cipher, από τον Ιούλιο Καίσαρα ο οποίος τον χρησιμοποιούσε για την αλληλογραφία του). Η «κρυπτογράφηση» απλά αντικαθιστά το κάθε γράμμα με το N-οστό επόμενό του στο αλφάβητο, αρχίζοντας κυκλικά από την αρχή για όσα γράμματα το N-οστό επόμενό του βρίσκεται μετά το τέλος του αλφαβήτου.

Η διπλανή εικόνα (από τη Wikipedia) δείχνει την αντιστοίχιση των κεφαλαίων γραμμάτων του Αγγλικού αλφάβητου με χρήση του ROT13, δηλαδή με περιστροφή των γραμμάτων προς τα δεξιά κατά 13 θέσεις (η οποία, στην περίπτωση αυτή, είναι ισοδύναμη με την περιστροφή προς τα αριστερά κατά 13 θέσεις), και ένα παράδειγμα κρυπτογράφησης και αποκρυπτογράφησης της λέξης HELLO.



Η κρυπτογράφηση με έναν ROTN cipher δεν είναι ασφαλής. Είναι πολύ εύκολο να την σπάσουμε, ειδικά αν είναι γνωστό το αλφάβητο. Μπορούμε να το κάνουμε ως εξής:

- Για κάθε N (π.χ. 0 ≤ N ≤ 25 για το Αγγλικό αλφάβητο), αποκρυπτογραφούμε το μήνυμα με τον αλγόριθμο ROTN. Προφανώς κάποιο από τα αποκρυπτογραφημένα μηνύματα που προκύπτουν είναι το αρχικό, απλά δεν ξέρουμε ποιο ακόμα.
- Στη συνέχεια, υπολογίζουμε το N που ελαχιστοποιεί την παρακάτω συνάρτηση εντροπίας:

$$H_N(f_N, f) = -\sum f_N(c) * \log f(c)$$

  όπου $f_N(c)$ είναι η συχνότητα εμφάνισης του χαρακτήρα *c* στο αποκρυπτογραφημένο κείμενο και $f(c)$ η συχνότητα εμφάνισης του χαρακτήρα *c* στο αλφάβητο του κειμένου (π.χ. στα Αγγλικά).
- Επιστρέφουμε ως έξοδο το κείμενο N.

Η άσκηση σας ζητάει να γράψετε δύο προγράμματα (ένα σε C/C++ και ένα σε ML) τα οποία να διαβάζουν ένα κρυπτογραφημένο κείμενο στα Αγγλικά από ένα αρχείο εισόδου όπως φαίνεται παρακάτω και να τυπώνουν το αποκρυπτογραφημένο κείμενο στην έξοδό τους (στο stdout).

Οι συχνότητες εμφάνισης των 26 γραμμάτων του Αγγλικού αλφάβητου είναι οι εξής:

```
0.08167, 0.01492, 0.02782, 0.04253, 0.12702, 0.02228, 0.02015,
0.06094, 0.06966, 0.00153, 0.00772, 0.04025, 0.02406, 0.06749,
0.07507, 0.01929, 0.00095, 0.05987, 0.06327, 0.09056, 0.02758,
0.00978, 0.02360, 0.00150, 0.01974, 0.00074
```

δηλαδή το A (ανεξάρτητα του αν είναι μικρό ή κεφαλαίο) εμφανίζεται με συχνότητα 8.167% σε ένα τυπικό Αγγλικό κείμενο, ενώ το Z με συχνότητα 0.074%.

Μπορείτε να υποθέσετε ότι τα κείμενα δε θα έχουν πάνω από 10.000 χαρακτήρες. Προσέξτε, ότι

στο κείμενο μπορεί να υπάρχουν και άλλοι χαρακτήρες εκτός από γράμματα του αλφαβήτου (spaces, new lines, ερωτηματικά, θαυμαστικά, κοκ) και οι χαρακτήρες αυτοί δεν αλλάζουν κατά την κρυπτογράφησή τους. Επίσης, προσέξτε ότι η αποκρυπτογράφηση διατηρεί το κατά πόσο οι χαρακτήρες του κειμένου εμφανίζονται με πεζά ή με κεφαλαία.

Παρακάτω δίνονται κάποια παραδείγματα σε C/C++ και σε ML.

Σε C/C++, MLton, ή σε OCaml

```
$ ./decrypt msg0.txt
See
$ ./decrypt msg1.txt
Why did the chicken cross the road?
To get to the other side!
$
```

Σε SML/NJ

```
- decrypt "msg0.txt";
See
val it = () : unit
- decrypt "msg1.txt";
Why did the chicken cross the road?
To get to the other side!
val it = () : unit
```

όπου τα αρχεία με τα δεδομένα εισόδου είναι τα εξής (η εντολή **cat** είναι εντολή του Unix):

```
$ cat msg0.txt
Lxx
```

```
$ cat msg1.txt
Jul qvq gur puvpxra pebff gur ebnq?
Gb trg gb gur bgure fvqr!
```

Ακριβή μου βενζίνη (0.25+0.25 = 0.5 βαθμοί)

Ως γνωστόν, λόγω πολέμου (αλλά όχι μόνο!), η τιμή της βενζίνης έχει ανέβει στα ύψη τελευταία. Κάποιοι συνάνθρωποί μας έχουν αναγκαστεί να περιορίσουν την επίσκεψή τους στα πρατήρια βενζίνης στα απολύτως απαραίτητα για τη μετακίνησή τους αντί για τα συνηθισμένα πριν από λίγα χρόνια «βάλε 20», «βάλε 30», ... (ή το προ κρίσης «γέμισέ το» μου). Άλλοι μελετούν το χάρτη και προσπαθούν να βρουν τις πιο οικονομικές διαδρομές για τη μετακίνησή τους, αλλά ταυτόχρονα φροντίζουν να έχουν το ντεπόζιτο του αυτοκινήτου τους έτοιμο για κάθε ενδεχόμενη μετακίνηση. Στην άσκηση αυτή θα ασχοληθούμε με αυτήν την κατηγορία οδηγών.

Συγκεκριμένα, μας δίνεται ένας χάρτης με το οδικό δίκτυο της χώρας. Υπάρχουν N πόλεις (αριθμημένες από 1 μέχρι N) και M δρόμοι διπλής κατεύθυνσης που συνδέουν ζεύγη διαφορετικών πόλεων. Για κάθε δρόμο γνωρίζουμε την ποσότητα βενζίνης (σε λίτρα) που χρειάζεται ένα αυτοκίνητο για να ταξιδέψει από το ένα άκρο του στο άλλο. Θεωρήστε δεδομένο ότι υπάρχει τουλάχιστον μία διαδρομή, αποτελούμενη πιθανώς από περισσότερους του ενός δρόμους, που συνδέει οποιαδήποτε πόλη με οποιαδήποτε άλλη.

Έστω ότι το ντεπόζιτο βενζίνης του αυτοκινήτου που μας ενδιαφέρει χωράει L λίτρα. Μπορούμε να το γεμίσουμε σε οποιοδήποτε βενζινάδικο, δυστυχώς όμως όλα τα βενζινάδικα βρίσκονται στις πόλεις — δεν υπάρχουν πλέον βενζινάδικα κατά μήκος των δρόμων! Επομένως, για να κινηθούμε κατά μήκος ενός δρόμου, πρέπει η ποσότητα βενζίνης που απαιτείται για το ταξίδι να είναι μικρότερη ή ίση του L.

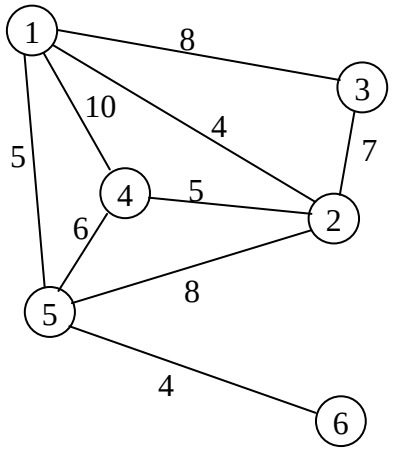
Η άσκηση σας ζητάει να γράψετε δύο προγράμματα (ένα σε C/C++ και ένα σε ML) που να βρίσκουν την ελάχιστη δυνατή τιμή του L που επιτρέπει στον οδηγό του αυτοκινήτου να κινηθεί από οποιαδήποτε πόλη της χώρας σε οποιαδήποτε άλλη.

Τα στοιχεία εισόδου διαβάζονται από ένα αρχείο όπως φαίνεται στο παράδειγμα που ακολουθεί. Η πρώτη γραμμή του αρχείου έχει δύο ακέραιους N ($2 \le N \le 10.000$) και M ($1 \le N \le 100.000$), χωρισμένους με ένα κενό διάστημα: το πλήθος των πόλεων και το πλήθος των δρόμων, αντίστοιχα. Κάθε μία από τις επόμενες M γραμμές περιγράφει έναν δρόμο διπλής κατεύθυνσης. Περιέχει τρεις ακέραιους αριθμούς U_i, V_i ($1 \le U_i \le N$, $1 \le V_i \le N$) και W_i ($1 \le W_i \le 100.000$), χωρισμένους ανά δύο με ένα κενό διάστημα. Μία τέτοια γραμμή σημαίνει ότι μπορούμε να πάμε από την πόλη U_i στην πόλη V_i (ή και αντίστροφα) και για το ταξίδι αυτό χρειαζόμαστε W_i λίτρα βενζίνης. Μπορείτε να υποθέσετε ότι υπάρχει το πολύ ένας δρόμος διπλής κατεύθυνσης που συνδέει δύο οποιεσδήποτε πόλεις.

Δείτε το παρακάτω παράδειγμα εισόδου και χάρτη.

map.txt

```
6 9
2 1 4
3 2 7
4 5 6
1 3 8
1 4 10
5 2 8
5 6 4
1 5 5
4 2 5
```



Στο παραπάνω παράδειγμα, δεν υπάρχει τρόπος να πάει κανείς από την πόλη 3 στην πόλη 4 χωρίς να περάσει από δρόμο που να χρειάζεται τουλάχιστον 7 λίτρα βενζίνη, επομένως το ελάχιστο L δεν μπορεί να είναι μικρότερο από 7. Εξετάζοντας όλα τα ζεύγη πόλεων αφετηρίας και προορισμού, διαπιστώνουμε ότι το 7 μας αρκεί για να μετακινηθούμε οπουδήποτε.

Τα προγράμματά σας σε C/C++ και σε ML θα πρέπει να δουλεύουν ως εξής.

Σε C/C++, MLton, ή σε OCaml

```
$ ./min_fill map.txt
7
```

Σε SML/NJ

```
- min_fill "map.txt";
7
val it = () : unit
```

Περαιτέρω οδηγίες για τις ασκήσεις

- Μπορείτε να δουλέψετε σε ομάδες το πολύ δύο ατόμων, τόσο σε αυτή όσο και στις επόμενες σειρές ασκήσεων. Όμως, έχετε υπ' όψη σας ότι, αν δεν περάσετε το μάθημα φέτος, οι βαθμοί των προγραμματιστικών ασκήσεων κρατούνται μόνο για όσους δεν τις έκαναν σε ομάδα αλλά τις έκαναν μόνοι τους. (Για όλους τους υπόλοιπους, οι βαθμοί των ασκήσεων κρατούνται.)

- Δεν επιτρέπεται να μοιράζεστε τα προγράμματά σας με συμφοιτητές εκτός της ομάδας σας ή να τα βάλετε σε μέρος που άλλοι μπορούν να τα βρουν (π.χ. σε κάποια σελίδα στο διαδίκτυο, σε ιστοσελίδες συζητήσεων, …). Σε περίπτωση που παρατηρηθούν «περίεργες» ομοιότητες σε προγράμματα, ο βαθμός των εμπλεκόμενων φοιτητών σε *όλες τις σειρές ασκήσεων* γίνεται αυτόματα μηδέν ανεξάρτητα από το ποια ομάδα... «εμπνεύστηκε» από την άλλη.

- Μπορείτε να χρησιμοποιήσετε «βοηθητικό» κώδικα (π.χ. κώδικα ταξινόμησης, κάποιο κώδικα που διαχειρίζεται κάποια δομή δεδομένων) που βρήκατε στο διαδίκτυο στα προγράμματά σας, με την προϋπόθεση ότι το πρόγραμμά σας περιέχει σε σχόλια την παραδοχή για την προέλευση αυτού του κώδικα και ένα σύνδεσμο (link) σε αυτόν.

- Τα προγράμματα σε C/C++ πρέπει να είναι σε ένα αρχείο και να μπορούν να μεταγλωττιστούν χωρίς warnings με gcc/g++ (version ≥ 10.2.1) με εντολές της μορφής, π.χ.

```
gcc –std=c99 -Wall –Werror -O3 –o decrypt decrypt.c
g++ –std=c++11 -Wall –Werror -O3 –o decrypt decrypt.cpp
```

- Τα προγράμματα σε ML πρέπει επίσης να είναι σε ένα αρχείο και να δουλεύουν σε SML/NJ ≥ v110.79 ή σε MLton ≥ 20210117 ή σε Objective Caml version ≥ 4.11.1. Το σύστημα ηλεκτρονικής υποβολής σας επιτρέπει να επιλέξετε μεταξύ αυτών των διαλέκτων της ML.

- Η υποβολή των προγραμμάτων θα γίνει ηλεκτρονικά μέσω του helios και για να μπορέσετε να τα υποβάλλετε και να βαθμολογηθείτε για αυτά, τα μέλη της ομάδας σας (και οι δύο) θα πρέπει να έχετε εγγραφεί στο μάθημα στο helios. Θα υπάρξει σχετική ανακοίνωση για την ακριβή διαδικασία υποβολής όταν ανοίξει το σύστημα. Τα προγράμματά σας πρέπει να διαβάζουν την είσοδο όπως αναφέρεται και δεν πρέπει να έχουν κάποιου άλλους είδους έξοδο διότι δεν θα γίνουν δεκτά από το σύστημα στο οποίο θα υποβληθούν.