ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
Εθνικόν και Καποδιστριακόν Πανεπιστήμιον Αθηνών
ΙΔΡΥΘΕΝ ΤΟ 1837

Εισαγωγή στον Προγραμματισμό

Εργασία #2

Δεκέμβριος 2024

Με τις πρώτες μας εργασίες γράψαμε προγράμματα που διαχειρίζονται ακεραίους και εξοικειωθήκαμε με τις βασικές προγραμματιστικές δομές. Σε αυτήν την εργασία θα αρχίσουμε να λύνουμε πιο ρεαλιστικά προβλήματα και ταυτόχρονα θα συνεχίσουμε να χτίζουμε την εμπειρία μας με άλλους τύπους, προγράμματα και μεθόδους εισόδου και εξόδου. Συγκεκριμένα, στοχεύουμε σε εμπειρία με τα ακόλουθα:

1. Χρήση πινάκων, χαρακτήρων και συμβολοσειρών
2. Χρήση δεικτών και δυναμικής μνήμης
3. Διάβασμα/γράψιμο δεδομένων από/σε αρχεία
4. Μεταγλώττιση προγραμμάτων με πάνω από ένα αρχεία
5. Έκθεση σε προγραμματιστικές τεχνικές

Υποβολή Εργασίας. Όπως πάντα, όλες οι υποβολές των εργασιών θα γίνουν μέσω GitHub. Προκειμένου να ξεκινήσεις, μπορείς να δεχτείς αυτήν την πρόσκληση [1].

1. Προβλέποντας το Μέλλον (future - 50 Μονάδες)

Είναι δύσκολο—ή ίσως αδύνατο—να προβλέψουμε το μέλλον. Παρ'όλες τις δυσκολίες όμως, ένα σημαντικό μέρος του πληθυσμού ασχολείται καθημερινά με τις προβλέψεις [6]. Οι προβλέψεις αυτές ποικίλουν ανάλογα με το πεδίο εφαρμογής, από τον καιρό μέχρι ... καφεμαντεία και από το ποια θα είναι η επόμενη λέξη σε αυτήν την πρόταση [8] μέχρι την ύπαρξη σωματιδίων στον χώρο της θεωρητικής φυσικής. Ανεξαρτήτως του πεδίου εφαρμογής, όλες οι προβλέψεις αξιολογούνται για την ποιότητά τους με ένα κοινό κριτήριο: το πόσο ακριβείς είναι. Όσο πιο ακριβείς οι προβλέψεις, τόσο πιο αξιόπιστη θεωρείται η πηγή τους. Αν αυτή η ακρίβεια διατηρηθεί σε βάθος χρόνου, τότε η διαδικασία πρόβλεψης μπορεί να προαχθεί σε αποδεκτό θεώρημα ή νόμο της φυσικής [12].

Σε αυτήν την άσκηση, θα υλοποιήσουμε έναν αλγόριθμο που χρησιμοποιείται κατεξοχήν για να κάνουμε προβλέψεις [3], τον *κινούμενο μέσο όρο (Simple Moving Average - SMA)* [9]. Ο κινούμενος μέσος όρος είναι παρεμφερής με τον κανονικό μέσο όρο, με μια διαφορά: ο κινούμενος μέσος όρος απαιτεί και ένα "παράθυρο" (window), δηλαδή τον αριθμό των τιμών (μετρώντας από το τέλος) που θέλουμε να λάβουμε υπόψη μας στον υπολογισμό του μέσου όρου. Για παράδειγμα, ας υποθέσουμε ότι έχουμε αυτές τις 10 τιμές:

$$9 \quad 7 \quad 7 \quad 1 \quad 4 \quad 4 \quad 4 \quad 38 \quad 8 \quad 4$$

Ο μέσος όρος αυτών των τιμών είναι $\frac{9+7+7+1+4+4+4+38+8+4}{10} = 8.60$. Για να υπολογίσουμε τον κινούμενο μέσο, πρέπει να επιλέξουμε ένα παράθυρο, έστω 3. Τότε ο κινούμενος μέσος με παράθυρο 3 για αυτά τα στοιχεία λαμβάνει υπόψη του μόνο τα τελευταία 3:

$$9 \quad 7 \quad 7 \quad 1 \quad 4 \quad 4 \quad 4 \quad \underbrace{38 \quad 8 \quad 4}_{SMA_3}$$

και επομένως $SMA_3 = \frac{38+8+4}{3} = 16.67$. Αντίστοιχα, μπορούν να οριστούν κινούμενοι μέσοι με μικρότερα παράθυρα (π.χ., παράθυρο 1 σημαίνει μόνο το τελευταίο στοιχείο) ή μεταλύτερα παράθυρα (π.χ., παράθυρο 10 θα λάμβανε υπόψη του όλες τις τιμές παραπάνω).

Σε αυτήν την άσκηση λοιπόν, καλείστε να υλοποιήσετε ένα πρόγραμμα το οποίο θα υπολογίζει τον κινούμενο μέσο όρο για μια σειρά από δεδομένα και να τον εκτυπώνει ως πρόβλεψη. Οι τεχνικές προδιαγραφές ακολουθούν.

Τεχνικές Προδιαγραφές

- Repository Name: progintro/hw2-<YourUsername>
- C Filepath: future/src/future.c
- Το πρόγραμμά θα πρέπει να παίρνει ως κυρίως όρισμα το όνομα του αρχείου που περιέχει τα δεδομένα μας. Προαιρετικά, μπορεί να παίρνει και άλλα δύο ορίσματα προκειμένου να ορίσουμε το μέγεθος του "παραθύρου" που θέλουμε για τον κινούμενο μέσο όρο. Για παράδειγμα αν το τρέξουμε ως `./future data.txt --window 42`, σημαίνει υπολόγισε τον κινούμενο μέσο όρο με παράθυρο 42 για τα δεδομένα του αρχείου data.txt. Αν δεν δοθεί παράμετρος για το μέγεθος του παραθύρου, τότε το (default) παράθυρο πρέπει να είναι μεγέθους 50. 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.
- Το αρχείο με τα δεδομένα θα περιέχει αριθμούς κινητής υποδιαστολής με μέχρι δύο ψηφία ακριβείας. Γενικά, δεν θα σας ζητηθεί να χρησιμοποιήσετε μεγαλύτερη ακρίβεια από double.

- Αν το παράθυρο είναι μικρότερο από 1 ή μεγαλύτερο από τον συνολικό αριθμό δεδομένων, το πρόγραμμά σας πρέπει να τερματίζει με μήνυμα σφάλματος όπως στα παραδείγματα. Το μήνυμά σας πρέπει να τυπώνεται στο stderr του προγράμματος. Το μέγεθος του παραθύρου δεν θα υπερβεί το 10^{18}.
- Το αρχείο C που θα υποβληθεί πρέπει να μεταγλωττίζεται χωρίς ειδοποιήσεις για λάθη και με κωδικό επιστροφής (exit code) που να είναι 0. Συγκεκριμένα, το αρχείο σας **πρέπει** να μπορεί να μεταγλωττιστεί επιτυχώς με την ακόλουθη εντολή σε ένα από τα μηχανήματα του εργαστηρίου (linuxXY.di.uoa.gr):

  ```
  gcc -Os -Wall -Wextra -Werror -pedantic -o future future.c
  ```

- README Filepath: future/README.md
- Πρέπει να ολοκληρώνει την εκτέλεση μέσα σε: 1 δευτερόλεπτο.

Παραδείγματα εκτέλεσης ακολουθούν:

```
$ ./future
Usage: ./future <filename> [--window N (default: 50)]
$ cat values.txt
9 7 7 1 4 4 4 38 8 4
$ ./future values.txt --window 10
8.60
$ echo $?
0
$ ./future values.txt --window 3
16.67
$ ./future values.txt --window 1
4.00
$ ./future values.txt --window 0
Window too small!
$ echo $?
1
$ ./future values.txt --window 0 2> out
$ cat out
Window too small!
$ ./future values.txt --window 12
Window too large!
$ echo $?
1
$ ./future values.txt --window 1000000000000
Failed to allocate window memory
```

Φυσικά οι ίδιοι έλεγχοι μπορούν να γίνουν με πραγματικά δεδομένα και να δείτε αν οι προβλέψεις του προγράμματός μας είναι καλές (Το αρχείο dow_jones.txt βρίσκεται στο https://github.com/progintro/data):

```
$ ./future dow_jones.txt
43471.71
$ ./future dow_jones.txt --window 200
40677.19
$ ./future dow_jones.txt --window 1000
35273.61
$ wc -l dow_jones.txt
8302 dow_jones.txt
$ ./future dow_jones.txt --window 8000
15447.33
```

Στο αρχείο README.md πρέπει να προσθέσετε οποιεσδήποτε παρατηρήσεις σας κατά την διεκπεραίωση της άσκησης. Ο κώδικας απαιτείται να είναι καλά τεκμηριωμένος με σχόλια καθώς αυτό θα είναι μέρος της βαθμολόγησης.

Bonus (Προαιρετικό)

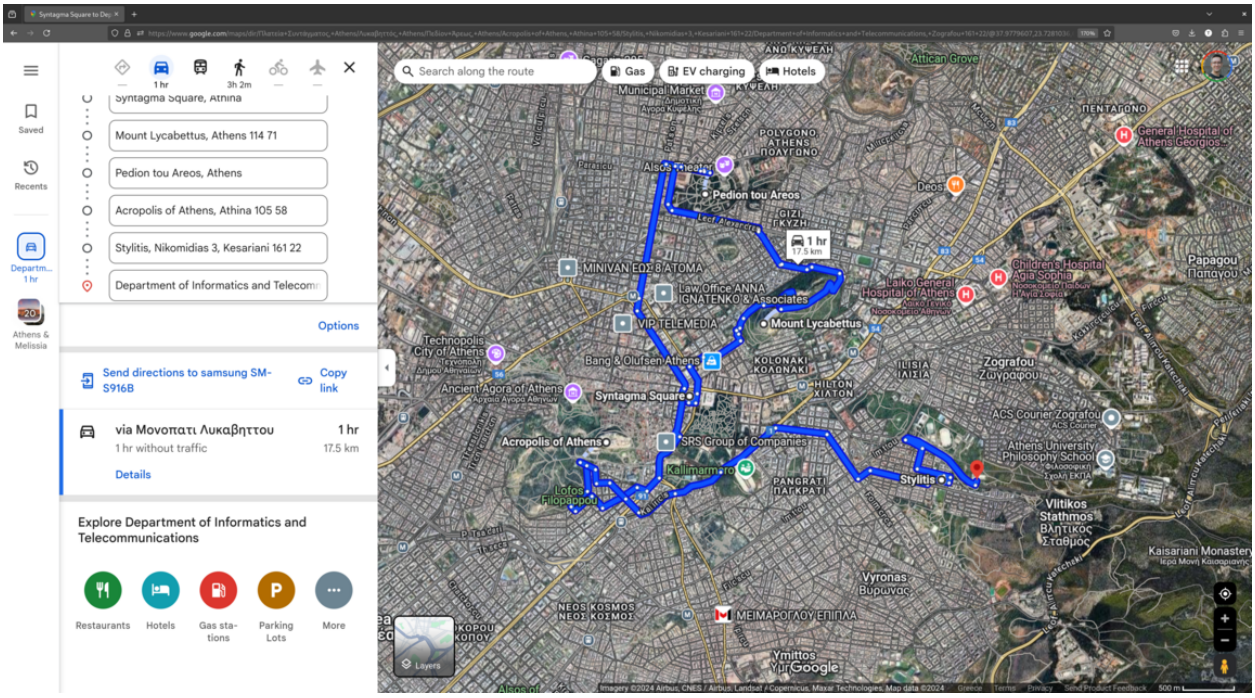
Πιστεύετε πως μπορείτε να φτιάξετε αλγορίθμους που προβλέπουν πως θα κινηθεί ένα οικονομικό asset καλύτερα από τον κινούμενο μέσο; Αν ναι, προσθέστε ένα easter egg option στο πρόγραμμά σας ονόματι `--compete` και θα μπείτε αυτόματα στον διαγωνισμό μας. Η αξιολόγηση θα γίνει με πραγματικά δεδομένα από χρηματιστήρια του κόσμου. Η διεπαφή θα είναι παρόμοια με παραπάνω, απλά υπολογίζετε την καλύτερή σας πρόβλεψη για την επόμενη τιμή σε μια σειρά δεδομένων:

```
$ ./future dow_jones.txt --compete
43455.32
```

Οι τρεις καλύτερες υποβολές θα λάβουν 100%, 70% και 40% έξτρα βαθμολογία σε αυτήν την άσκηση. Αν σας φάνηκε ενδιαφέρον αυτό το πρόβλημα, υπάρχουν πολλά σχετικά προβλήματα με πραγματικές εφαρμογές στον χώρο των οικονομικών [5, 2].

2. Το Καλύτερο GPS (jabbamaps - 50 Μονάδες)

Οι διαδρομές είναι κουραστικές και ιδιαίτερα μάλιστα όταν χρειάζεται να επισκεφτείς πάνω από ένα μέρη. Ευτυχώς, οι ομάδες λογισμικού το έχουν λάβει υπόψη τους και πολλά προϊόντα σήμερα σου επιτρέπουν να προσθέσεις όλες τις στάσεις που θες να καλύψεις και σου επιστρέφουν αμέσως την βέλτιστη (κατά την άποψή τους) διαδρομή—ένα παράδειγμα φαίνεται στην Εικόνα 1. Όμως, έχει σημασία με ποια σειρά θα καλύψουμε αυτές τις στάσεις; Υπάρχει περίπτωση με μια απλή αναδιάταξη να

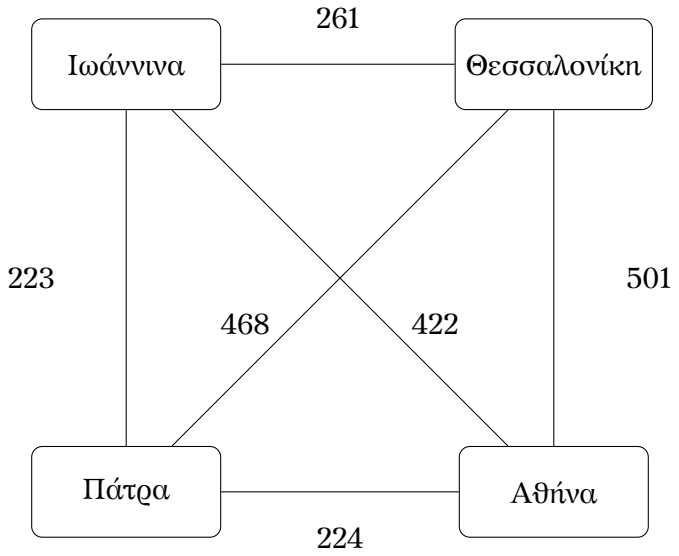


Εικόνα 1: Σύγχρονες εφαρμογές όπως το Google Maps επιτρέπουν να έχεις πολλές στάσεις μέχρι να φτάσεις στον προορισμό σου. Δυστυχώς όμως δεν σου προτείνουν να αναδιατάξεις τις στάσεις σου ακόμη και αν αυτό σου γλύτωνε χρόνο!

γλυτώσουμε κόπο και χρόνο; Αυτό θα είναι το πρόβλημα με το οποίο θα ασχοληθούμε σε αυτήν την άσκηση.

Η Εικόνα 2 δείχνει μια απλοποιημένη εκδοχή του προβλήματός μας για 4 στάσεις. Έστω ότι θέλουμε να κάνουμε τον γύρο τις Ελλάδας και συγκεκριμένα να καλύψουμε τέσσερις πόλεις: Αθήνα, Θεσσαλονίκη, Γιάννενα και Πάτρα. Από κάθε πόλη μπορούμε να κινηθούμε προς οποιαδήποτε άλλη, αλλά κάθε κίνησή μας έχει ένα κόστος (έστω χιλιομετρικό). Για παράδειγμα, για να πάμε από την Αθήνα στην Θεσσαλονίκη έχουμε κόστος 501, ενώ για να πάμε στην Πάτρα έχουμε κόστος 224. Θεωρούμε ότι η κατεύθυνση δεν έχει σημασία και το χιλιομετρικό κόστος είναι το ίδιο ανεξαρτήτως κατεύθυνσης.

Έστω ότι ξεκινάμε από την Αθήνα (η πόλη από την οποία ξεκινάμε δεν έχει σημασία, καθώς πρέπει να επισκεφτούμε όλες τις πόλεις), έχουμε 3 διαφορετικές επιλογές για την πόλη που θα επισκεφτούμε στην συνέχεια. Προσέξτε ότι οποιαδήποτε πόλη επιλέξουμε έχει συνέπειες και για τις επόμενες επιλογές μας. Ποιο είναι το μονοπάτι με το ελάχιστο κόστος; Αν επιλέξω να κάνω το Αθήνα $\to$ Θεσσαλονίκη $\to$ Ιωάννινα $\to$ Πάτρα θα χρειαστώ $501 + 261 + 223 = 985$ χιλιόμετρα. Αντίθετα, αν πάω Αθήνα $\to$ Πάτρα $\to$ Ιωάννινα $\to$ Θεσσαλονίκη χρειάζομαι $224 + 223 + 261 = 708$ χιλιόμετρα. Με μια αλλαγή στην σειρά επίσκεψης γλύτωσα 277 χιλιόμετρα! Είναι όμως αυτή η βέλτιστη επιλογή; Πόσες δυνατές διαφορετικές διατάξεις υπάρχουν στην σειρά με



Εικόνα 2: Από κάθε πόλη μπορείς να αποφασίσεις να κινηθείς σε οποιαδήποτε άλλη πόλη με κάποιο χιλιομετρικό κόστος. Ποια είναι η πιο αποδοτική (από απόψεως κόστους) σειρά με την οποία μπορείς να τις επισκεφτείς;

την οποία μπορώ να επισκεφτώ τις πόλεις;

Το πρόβλημα στο οποίο πρέπει να αποφασίσουμε με ποια σειρά θα επισκεφτούμε μια σειρά πόλεων λέγεται *Travelling Salesman Problem* [11], ελληνιστί το πρόβλημα του πλανόδιου πωλητή—επειδή βρίσκει την βέλτιστη διαδρομή που θα έπρεπε να κάνει ένας πωλητής με την πραμάτεια του για να επισκεφτεί όλες τις πόλεις. Ο ίδιος αλγόριθμος μπορεί να χρησιμοποιηθεί για βελτιστοποίηση σχεδίασης κυκλωμάτων, παράδοσης παραγγελιών, ακόμα και προβλημάτων ανεφοδιασμού—με νεοφυείς εταιρείες ακόμα και στην χώρα μας.

Φτάσαμε επομένως στο ζητούμενο αυτής της άσκησης: να γράψετε ένα πρόγραμμα το οποίο θα διαβάζει έναν χάρτη με τις αποστάσεις όλων των ζευγών πόλεων που θέλουμε να επισκεφτούμε και θα μας υπολογίζει το μονοπάτι ελαχίστου κόστους, εύκολα, γρήγορα και πάνω απ'όλα τζάμπα! Οι τεχνικές προδιαγραφές ακολουθούν.

Τεχνικές Προδιαγραφές

- Repository Name: progintro/hw2-<YourUsername>
- C Filepath: jabbamaps/src/jabbamaps.c
- Το πρόγραμμά θα πρέπει να παίρνει ακριβώς ένα όρισμα, το όνομα του αρχείου που περιέχει τα δεδομένα του χάρτη. Αν το πρόγραμμα εκτελεστεί με ορίσματα που δεν ακολουθούν τις παραπάνω προδιαγραφές, πρέπει να εκτυπώσει αντί-

στοιχο μήνυμα όπως στα παρακάτω παραδείγματα και να επιστρέφει με κωδικό εξόδου (exit code) 1.

- Το αρχείο με τα δεδομένα του χάρτη θα έχει σε κάθε γραμμή του ένα ζεύγος πόλεων και στην συνέχεια την απόστασή τους. Συγκεκριμένα, κάθε γραμμή θα είναι της μορφής `city1-city2: distance`, όπου city1 και city2 είναι το ζεύγος των πόλεων και distance είναι η απόστασή τους. Τα ονόματα των πόλεων δεν θα έχουν τους χαρακτήρες '-' ή ':' και η απόστασή τους θα είναι πάντα ένας ακέραιος αριθμός.

- Η πρώτη πόλη που θα διαβάσουμε στον χάρτη θέλουμε να είναι η πόλη από την οποία θα ξεκινήσουμε την διαδρομή μας. Το συνολικό κόστος δεν θα πρέπει να περιλαμβάνει το κόστος επιστροφής στην αρχική μας πόλη.

- Οι αποστάσεις των πόλεων δεν θα υπερβαίνουν το 2^{31}.

- Οι πόλεις που θα πρέπει να επισκεφτούμε δεν θα είναι πάνω από 64.

- Το μόνοπάτι σας πρέπει να επισκεφτεί *κάθε* πόλη *ακριβώς* μία φορά.

- Το αρχείο C που θα υποβληθεί πρέπει να μεταγλωττίζεται χωρίς ειδοποιήσεις για λάθη και με κωδικό επιστροφής (exit code) που να είναι 0. Συγκεκριμένα, το αρχείο σας **πρέπει** να μπορεί να μεταγλωττιστεί επιτυχώς με την ακόλουθη εντολή σε ένα από τα μηχανήματα του εργαστηρίου (linuxXY.di.uoa.gr):

  ```
  gcc -m32 -Ofast -Wall -Wextra -Werror -pedantic -o jabbamaps jabbamaps.c
  ```

- README Filepath: jabbamaps/README.md

- Πρέπει να ολοκληρώνει την εκτέλεση μέσα σε: 30 δευτερόλεπτα.

Ενδεικτικές εκτελέσεις ακολουθούν σε κάποιους από τους χάρτες που υπάρχουν στο https://github.com/progintro/data ακολουθούν:

```
$ cat map4.txt
Athens-Thessaloniki: 501
Athens-Ioannina: 422
Athens-Patras: 224
Patras-Thessaloniki: 468
Patras-Ioannina: 223
Thessaloniki-Ioannina: 261
 ./jabbamaps map4.txt
We will visit the cities in the following order:
Athens -(224)-> Patras -(223)-> Ioannina -(261)-> Thessaloniki
Total cost: 708
```

```
$ ./jabbamaps map7.txt
We will visit the cities in the following order:
Athens -(211)-> Amfissa -(122)-> Patras -(223)-> Ioannina -(128)->
  Trikala -(123)-> Volos -(211)-> Thessaloniki
Total cost: 1018
```

Ιδανικά, επιθυμούμε το πρόγραμμά μας να δουλεύει και για μεγαλύτερους χάρτες, όσο ασυνήθιστα και να είναι τα ονόματα των πόλεων ή οι αποστάσεις μεταξύ τους. Για παράδειγμα:

```
$ ./jabbamaps tatooine.txt
We will visit the cities in the following order:
Republic City -(65)-> Aldera -(124)-> Anchorhead -(44)-> Lessu -(45)->
  Mos Pelgo -(32)-> Canto Bight -(65)-> Mos Espa -(80)-> Coronet City -(53)->
  Hanna City -(50)-> Sern Prime -(62)-> NiJedha -(20)-> Kachirho -(66)->
  Tipoca City -(141)-> Sundari -(51)-> Galactic City -(29)->
  Capital City (Lothal City) -(157)-> Mos Eisley -(317)-> Stalgasin Hive -(26)->
  Coral City -(25)-> Otoh Gunga -(13)-> Theed -(18)-> Cloud City -(136)-> Eriadu City
Total cost: 1619
```

Όσο περισσότερες οι πόλεις, τόσοι περισσότεροι οι συνδυασμοί που πρέπει να εξετάσουμε και το πρόβλημα πλέον δεν είναι εύκολο να επιβεβαιωθεί "με το μάτι". Είναι η λύση που βρήκαμε η βέλτιστη ή μήπως υπάρχει καλύτερη;

Ως συνήθως, υπενθυμίζουμε πως στο αρχείο README.md πρέπει να προσθέσετε οποιεσδήποτε παρατηρήσεις σας κατά την διεκπεραίωση της άσκησης. Ο κώδικας απαιτείται να είναι καλά τεκμηριωμένος με σχόλια καθώς αυτό θα είναι μέρος της βαθμολόγησης.

3. Το Δικό σου Chatbot - (jason - 50 Μονάδες)

Η τεχνητή νοημοσύνη έχει μπει για τα καλά στην ζωή μας. Τα κινητά μας, οι virtual assistants, οι διαφημίσεις που μας σερβίρονται και ένα σωρό άλλα προϊόντα εξαρτώνται όλο και περισσότερο από αυτήν την τεχνολογία. Η έκρηξη φαίνεται να ξεκίνησε το 2022 [4], με τις πρώτες ανοιχτές υπηρεσίες να κάνουν την τεχνολογία ευρέως διαθέσιμη [8]. Από τότε, χρήστες και προγραμματιστές/τριες ψάχνουν και βρίσκουν συνεχώς διαφορετικές εφαρμογές για αυτήν την τεχνολογία: chatbots που προσφέρουν support, εργαλεία που πραγματοποιούν ανάλυση δεδομένων, μέχρι και λύτες για προβλήματα μαθηματικών ολυμπιάδων (ή εργασιών).

Πόσο δύσκολο είναι όμως να φτιάξουμε ένα εργαλείο που να μπορεί να αξιοποιήσει τις υπηρεσίες τεχνητής νοημοσύνης και να προσφέρει λύσεις σε χρήστες; Αυτή είναι η ερώτηση που θα μας απασχολήσει σε αυτήν την άσκηση.

Προκειμένου να μιλήσουμε με απομακρυσμένες υπηρεσίες, πρέπει να μπορούμε να μιλήσουμε την γλώσσα τους. Τυχαίνει σήμερα πάμπολλες εφαρμογές στο internet να

μεταφέρουν δεδομένα χρησιμοποιώντας μια μορφοποίηση (data format) που λέγεται JSON (JavaScript Object Notation) [7]. Για να δούμε ένα παράδειγμα. Έστω ότι έχουμε ένα αρχείο `person.json` με τα ακόλουθα περιεχόμενα:

```
{
  "first_name": "John",
  "last_name": "Smith",
  "age": 27,
  "address": {
    "street_address": "21 2nd Street",
    "city": "New York",
    "state": "NY",
  },
  "phone_numbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    }
  ],
  "children": [
    "Catherine",
    "Thomas",
    "Trevor"
  ],
  "spouse": null
}
```

Το παραπάνω παράδειγμα JSON παρουσιάζει ιεραρχικά τα δεδομένα για ένα άτομο. Παρατηρούμε ότι όλα τα δεδομένα είναι μέσα σε ένα object (περικλείονται από αγκύλες { και }) και μέσα σε αυτό έχει διάφορα πεδία που κλείνονται μέσα σε διπλά quotes ": first_name, last_name, ..., children, spouse. Πέρα από αυτό παρατηρούμε ότι κάθε πεδίο μπορεί να έχει ως τιμή (ότι ακολουθεί την άνω και κάτω τελεία ":") ένα από τα ακόλουθα:

1. έναν αριθμό (π.χ., "age": 27)
2. ένα string ("first_name": "John")
3. μια λίστα από δεδομένα ("children": ["Catherine", "Thomas", "Trevor"])
4. ένα object με δικά του πεδία ("address": { ... })

Παρατηρήστε ότι ο παραπάνω ορισμός είναι αυτο-αναφορικός (ένα JSON object μπορεί να περιέχει ένα πεδίο τύπου JSON object, που μπορεί να περιέχει ένα JSON object κοκ).

Τι σχέση έχουν όλα αυτά με την τεχνητή νοημοσύνη; Προκειμένου να χρησιμοποιήσουμε τις υπηρεσίες τεχνητής νοημοσύνης πρέπει να μπορούμε να εξάγουμε κάποια συγκεκριμένα δεδομένα από ένα JSON. Για παράδειγμα, δείτε το παρακάτω παράδειγμα-απάντηση του gpt-4o-mini μοντέλου τεχνητής νοημοσύνης στην ερώτηση "Tell me a joke":


```
{
  "id": "chatcmpl-Ag5hb4g6PYiVpvBp3tuiJjqX9KjqN",
  "object": "chat.completion",
  "created": 1734595059,
  "model": "gpt-4o-mini-2024-07-18",
  "choices": [
    {
      "index": 0,
      "message": {
        "role": "assistant",
        "content": "Why do programmers always mix up Halloween
                    and Christmas?\nBecause Oct 31 == Dec 25!\n",
        "refusal": null
      },
      "logprobs": null,
      "finish_reason": "stop"
    }
  ],
  "usage": {
    "prompt_tokens": 10,
    "completion_tokens": 23,
    "total_tokens": 33,
    "prompt_tokens_details": {
      "cached_tokens": 0,
      "audio_tokens": 0
    },
    "completion_tokens_details": {
      "reasoning_tokens": 0,
      "audio_tokens": 0,
      "accepted_prediction_tokens": 0,
      "rejected_prediction_tokens": 0
    }
  },
  "system_fingerprint": "fp_6fc10e10eb"
}
```

Το παραπάνω JSON φαίνεται χαοτικό στην αρχή, αλλά αν το κοιτάξετε καλύτερα θα βρείτε το περιεχόμενο της απάντησης. Για να την ανιχνεύσουμε μαζί:

- Στο πάνω επίπεδο, έχουμε ένα JSON object με ένα πεδίο "choices".
  - Το πεδίο choices είναι τύπου λίστα (ξεκινά με '[') και το πρώτο στοιχείο (index = 0) είναι επίσης JSON object.
    - Το JSON object της λίστας choices έχει μέσα ένα πεδίο που λέγεται "message" το οποίο είναι επίσης τύπου JSON object.
      - Τέλος, το message έχει μέσα ένα πεδίο το οποίο λέγεται "content", το οποίο περιέχει το string με την απάντησή μας μέσα σε double quotes: "Why do programmers always mix up Halloween and Christmas?\n Because Oct 31 == Dec 25!\n" - παρατηρήστε ότι οι αλλαγές γραμμής είναι escaped όπως θα ήταν σε ένα string που θα γράφαμε σε πρόγραμμα σε γλώσσα C.

Ένας πιο γρήγορος τρόπος να συμβολίσουμε το παραπάνω πεδίο είναι ως εξής: `json.choices[0].message.content` - μετάφραση: στο JSON αρχείο που μας δίνεται, θέλουμε να βρούμε την λίστα choices, να πάρουμε το πρώτο στοιχείο της, από εκεί να πάρουμε το message και στο τέλος να πάρουμε την τιμή του content. Αν είχαμε την δυνατότητα να πάρουμε οποιοδήποτε αρχείο JSON και να εξάγουμε αυτήν την πληροφορία θα μπορούσαμε να φτιάξουμε το δικό μας chatbot!

Αυτό θα είναι και ο στόχος αυτής της άσκησης, θα υλοποιήσουμε ένα πρόγραμμα το οποίο θα εξάγει την παραπάνω πληροφορία από αρχεία JSON και θα την αξιοποιεί για να δώσει απαντήσεις στον χρήστη. Δείτε τις τεχνικές προδιαγραφές για περισσότερες λεπτομέρειες. Εάν τα καταφέρετε, θα έχετε φτιάξει το πρώτο σας νευροσυμβολικό εργαλείο [10]. Σκεφτείτε τι άλλα εργαλεία μπορείτε να φτιάξετε!

Τεχνικές Προδιαγραφές

- Repository Name: progintro/hw2-<YourUsername>
- C Filepath: jason/src/jason.c
- Το πρόγραμμά θα έχει δύο modes:
  - **Extraction mode**. Το πρόγραμμά σας μπαίνει σε extraction mode όταν ο χρήστης δώσει το option `--extract`. Το option --extract περιμένει στην συνέχεια το όνομα του αρχείου που περιέχει τα περιεχόμενα JSON και από τα οποία θα πρέπει να εξάγετε το `json.choices[0].message.content` και να το τυπώσετε στην πρότυπη έξοδο. Εάν δωθεί οποιοδήποτε αρχείο που δεν είναι valid JSON, το πρόγραμμά σας πρέπει να τυπώσει μήνυμα ίδιο με τα παραδείγματα παρακάτω στο stderr.

- **Conversation mode**. Το πρόγραμμά σας μπαίνει σε διαδικασία συζήτησης με τον χρήστη όταν δωθεί το option `--bot` χωρίς άλλα ορίσματα. Στην διαδικασία συζήτησης, το πρόγραμμά σας ρωτάει επαναλαμβανόμενα τον χρήστη `> What would you like to know?` μέχρι να στείλει ο χρήστης End-of-File (EOF). Κάθε ερώτηση του χρήστη τελειώνει με καινούρια γραμμή και πρέπει να στέλνεται αυτούσια στην κατάλληλη συνάρτηση της βιβλιοθήκης neurolib (δες παρακάτω).

- Για τις διαδράσεις με το σύστημα τεχνητής νοημοσύνης (conversation mode) είναι υποχρεωτικό να χρησιμοποιήσετε την βιβλιοθήκη neurolib (neurolib.c και neurolib.h) που σας δίνεται στον φάκελο jason/src. Δυστυχώς κατεβάσαμε αυτήν την βιβλιοθήκη από ένα online project χωρίς καθόλου documentation / testing. Προκειμένου να την αξιοποιήσετε θα χρειαστεί να διαβάσετε τις διαθέσιμες συναρτήσεις στο header file της και να ανακαλύψετε την χρήση τους. Ο στόχος σας είναι να στείλετε queries στην υπηρεσία τεχνητής νοημοσύνης και να πάρετε JSON απαντήσεις.

- Το αρχείο C που θα υποβληθεί πρέπει να μεταγλωττίζεται χωρίς ειδοποιήσεις για λάθη και με κωδικό επιστροφής (exit code) που να είναι 0. Συγκεκριμένα, το αρχείο σας **πρέπει** να μπορεί να μεταγλωττιστεί επιτυχώς με τις ακόλουθες εντολές σε ένα από τα μηχανήματα του εργαστηρίου (linuxXY.di.uoa.gr) - για παράδειγμα το linux14:

```
gcc -Wall -Wextra -Werror -pedantic -c neurolib.c
gcc -Wall -Wextra -Werror -pedantic -c jason.c
gcc -o jason neurolib.o jason.o -lssl -lcrypto
```

- README Filepath: jason/README.md

- Τα αρχεία JSON θα είναι μέχρι 1MB σε μέγεθος.

- Πρέπει να ολοκληρώνει την εκτέλεση μέσα σε: 1 δευτερόλεπτο.

- Το πρόγραμμά σας θα πρέπει να έχει τα λιγότερα δυνατά memory leaks.

Παραδείγματα εκτέλεσης ακολουθούν, πρώτα σε extraction mode:

```
$ ./jason --extract json/1.json
Why do programmers always mix up Halloween and Christmas?
Because Oct 31 == Dec 25!
$ echo $?
0
$ ./jason --extract json/2.json 2> stderr
$ echo $?
1
$ cat stderr
Not an accepted JSON!
```

Και στην συνέχεια σε conversation mode:

```
$ ./jason --bot
> What would you like to know? What is the last digit of pi?
I'd answer that, but I don't want to ruin the surprise.
> What would you like to know? What is the age of the universe?
I could tell you, but then I'd have to awkwardly dance away without explaining why.
> What would you like to know? Terminating
```

Ή μπορείτε να αλληλεπιδράσετε με μια πραγματική υπηρεσία τεχνητής νοημοσύνης (αν αγοράσετε tokens):

```
$ export OPENAI_API_KEY=... # enter your key here
$ ./jason --bot
> What would you like to know? What is the last digit of pi?
Pi (pi) is an irrational number, which means it has an infinite
number of decimal places and does not terminate. Therefore, it
does not have a last digit. The decimal representation of pi begins
with 3.14159 and continues indefinitely without repeating.
> What would you like to know? What is the age of the universe?
As of my last knowledge update in October 2023, the age of the
universe is estimated to be about 13.8 billion years. This estimate
is based on measurements of the cosmic microwave background radiation,
the expansion rate of the universe (Hubble constant), and observations
of the oldest known star clusters. However, scientific understanding is
always evolving, so it's possible that new discoveries could refine
this estimate in the future.
> What would you like to know? Terminating
```

Αν φτάσατε μέχρι εδώ: (1) συγχαρητήρια, (2) μην ξεχάσετε το README.md και (3) όπως πάντα ο κώδικας απαιτείται να είναι καλά τεκμηριωμένος με σχόλια καθώς αυτό θα είναι μέρος της βαθμολόγησης.

Καλές Γιορτές!

Αναφορές

[1] DIT. Πρόσκληση για Εργασία 2. https://classroom.github.com/a/ZBvCH4W8.

[2] Quantiacs. Quantiacs platform. https://quantiacs.com/.

[3] Wikipedia. Algorithmic Trading. https://en.wikipedia.org/wiki/Algorithmic_trading.

[4] Wikipedia. ChatGPT. https://en.wikipedia.org/wiki/ChatGPT.

[5] Wikipedia. Futures Contract. https://en.wikipedia.org/wiki/Futures_contract.

[6] Wikipedia. Futures studies. https://en.wikipedia.org/wiki/Futures_studies.

[7] Wikipedia. JSON File Format. https://en.wikipedia.org/wiki/JSON.

[8] Wikipedia. Large Language Models. https://en.wikipedia.org/wiki/Large_language_model.

[9] Wikipedia. Moving Average. https://en.wikipedia.org/wiki/Moving_average.

[10] Wikipedia. Neurosymbolic AI. https://en.wikipedia.org/wiki/Neuro-symbolic_AI.

[11] Wikipedia. Travelling Salesman Problem (TSP). https://en.wikipedia.org/wiki/Travelling_salesman_problem.

[12] YouTube. Feynman on the Scientific Method (1964). https://www.youtube.com/watch?v=0KmimDq4cSU.