ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

DEPARTMENT OF INFORMATICS

Ευφυείς Πράκτορες

Απαλλακτική Εργασία 2025

ΜΑΡΙΟΣ ΚΥΡΟΓΛΟΥ Π21080

ΙΩΑΝΝΑ ΤΑΓΑΡΑ Π21161

Περιεχόμενα

Τεχνική Αναφορά 3

1. Περιγραφή Συστήματος 3
2. Στόχος και Επίλυση 3
3. Αρχιτεκτονική Πρακτόρων 5
4. Τεχνολογίες που χρησιμοποιήθηκαν 9
5. Δυσκολίες που Αντιμετωπίσαμε 9
6. Παραδείγματα Εκτέλεσης 9
7. Βιβλιογραφία-Πηγές 12

Τεχνική Αναφορά

1. Περιγραφή Συστήματος

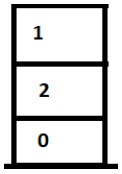
Το σύστημα υλοποιεί έναν ευφυή πράκτορα που αναλαμβάνει την κατανόηση φυσικής γλώσσας και την επίλυση προβλημάτων τύπου blocks (στοίβες μπλοκ) και jug (δοχεία/λίτρα). Ο χρήστης εισάγει μια περιγραφή σε φυσική γλώσσα και το σύστημα ταξινομεί το πρόβλημα, ενεργοποιώντας τον κατάλληλο υπο-πράκτορα για επίλυση. Η αναγνώριση γίνεται μέσω συνδυασμού λέξεων-κλειδιών (fuzzy matching) και LLM (τοπικό μοντέλο Mistral μέσω Ollama).

2. Στόχος και Επίλυση

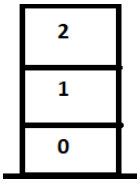
Το σύστημα λύνει δύο τύπους προβλημάτων:
- Blocks: Μετακίνηση μπλοκ μεταξύ στοιβών ώστε να επιτευχθεί η επιθυμητή διάταξη.
- Jug: Μέτρηση συγκεκριμένης ποσότητας νερού χρησιμοποιώντας δοχεία με διαφορετικές χωρητικότητες.
Κάθε πρόβλημα λύνεται με αλγοριθμική προσέγγιση (A* για blocks και BFS για jug).

Για το Blocks World πρόβλημα , ο στόχος είναι να φτιαχτεί μία ταξινομημένη στοίβα από μπλοκ. Μόνο ένα μπλοκ μπορεί να μετακινηθεί κάθε φορά: μπορεί είτε να τοποθετηθεί στο τραπέζι είτε να τοποθετηθεί πάνω σε ένα άλλο μπλοκ. Εξαιτίας αυτού, κάποιο μπλοκ που μια δεδομένη στιγμή είναι κάτω από ένα άλλο μπλοκ δεν μπορούν να μετακινηθεί. Ο στόχος είναι να βρεθεί το σύστημα από μια αρχική κατάσταση στην κατάσταση στόχου που είναι η ταξινόμηση της.

Παράδειγμα Blocks World προβλήματος με 3 στοιβάδες και 3 μπλοκς όπου:

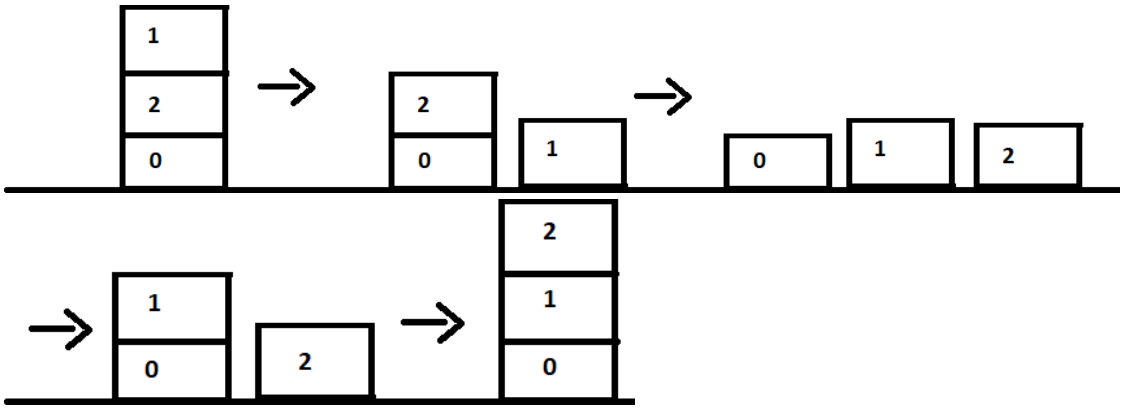


Αρχική κατάσταση :



Παράδειγμα τελικής κατάστασης :

Βήματα:



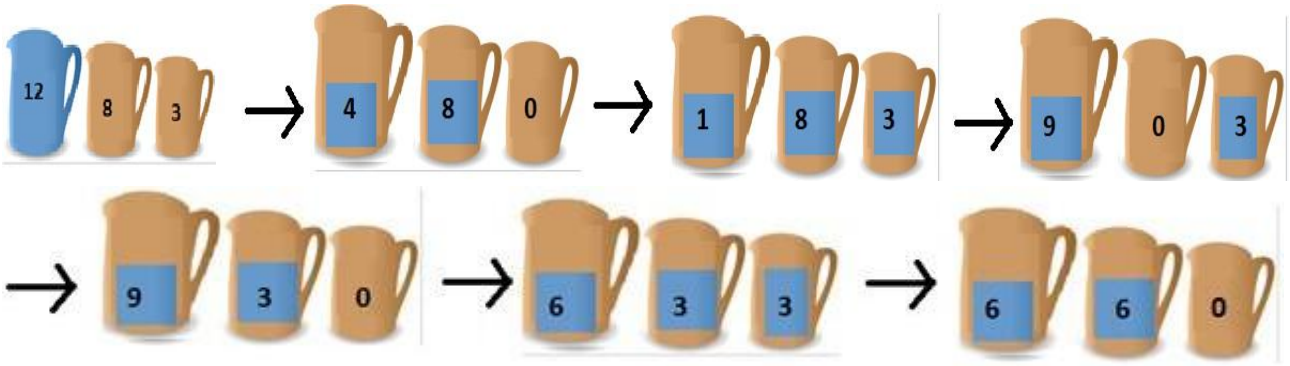
Για το Water Jug πρόβλημα , κάθε κανάτα περιέχει έναν γνωστό ακέραιο όγκο υγρού, όχι απαραίτητα ίσο με την χωρητικότητά του. Οι γρίφοι αυτού του τύπου ρωτούν πόσα βήματα γίνονται , όταν ρίχνουμε νερό από τη μία κανάτα στην άλλη (έως ότου η μία κανάτα αδειάσει ή η άλλη γεμίσει) για να φτάσει σε μια κατάσταση στόχου, που καθορίζεται σε όρους όγκου υγρού που πρέπει να υπάρχει σε κάποια κανάτα.

Παράδειγμα Water Jug προβλήματος με 3 κανάτες , όγκου νερού 12 λίτρα νερού η πρώτη , 8 λίτρα νερού η δεύτερη και 3 λίτρα νερού η τρίτη.

Αρχική Κατάσταση (να είναι γεμάτη μόνο η πρώτη κανάτα):

Τελική Κατάσταση (νάναι με 6 λίτρα η πρώτη και 6 λίτρα η δεύτερη):

Διαδικασία από αρχική σε τελική κατάσταση



3. Αρχιτεκτονική Πρακτόρων

Το σύστημα βασίζεται σε modular σχεδίαση πρακτόρων:
- **ClassifierAgent**: Αναλύει την είσοδο φυσικής γλώσσας και αποφασίζει τον τύπο προβλήματος.

Η κλάση ClassifierAgent είναι υπεύθυνη για την **κατανόηση της περιγραφής του χρήστη** σε φυσική γλώσσα και την **ταξινόμηση** του προβλήματος σε μία από τρεις κατηγορίες: blocks (προβλήματα με μπλοκ και στοίβες), jug (προβλήματα με δοχεία και λίτρα), ή unknown (αν δεν μπορεί να αποφασίσει). Η ταξινόμηση αυτή είναι σημαντική, καθώς καθορίζει ποιος υπο-πράκτορας θα αναλάβει να επιλύσει το πρόβλημα στη συνέχεια.

Ο πράκτορας αυτός αρχικά **προεπεξεργάζεται το κείμενο** που του δίνει ο χρήστης. Συγκεκριμένα, αφαιρεί τόνους και διακριτικά από τα γράμματα, και ομαλοποιεί το ελληνικό τελικό "ς" σε "σ", ώστε να βελτιωθεί η αξιοπιστία στην αναγνώριση λέξεων. Αυτή η κανονικοποίηση επιτρέπει στο σύστημα να αναγνωρίζει λέξεις ακόμα και αν έχουν διαφορετικές γραφές ή ορθογραφικές παραλλαγές.

Αφού καθαριστεί το κείμενο, ο ClassifierAgent εφαρμόζει μια **λογική fuzzy matching**: ελέγχει αν κάποια από τις λέξεις που περιέχει το prompt μοιάζει (με βαθμό ομοιότητας >70%) με λέξεις-κλειδιά που έχουν οριστεί για κάθε είδος προβλήματος. Για παράδειγμα, λέξεις όπως "δοχείο", "νερό", "λίτρα", "jug", "bucket" οδηγούν στο συμπέρασμα ότι πρόκειται για πρόβλημα τύπου jug. Αντίστοιχα, λέξεις όπως "στοίβα", "μπλοκ", "block", "πύργος" οδηγούν στην αναγνώριση blocks. Αν εντοπιστεί κάποιο ταίριασμα, η συνάρτηση επιστρέφει αμέσως τον τύπο προβλήματος.

Αν όμως καμία λέξη δεν ταιριάζει με τις λέξεις-κλειδιά, τότε ο ClassifierAgent **κάνει fallback σε ένα LLM (Large Language Model)** που τρέχει τοπικά μέσω του Ollama, χρησιμοποιώντας το μοντέλο mistral. Στέλνει ένα ειδικά σχεδιασμένο prompt που ζητά από το LLM να διαβάσει την περιγραφή και να απαντήσει αποκλειστικά με μία από τις λέξεις "blocks", "jug" ή "unknown", χωρίς καμία άλλη εξήγηση. Η απάντηση του μοντέλου επεξεργάζεται και επιστρέφεται ως τελική κατηγοριοποίηση του προβλήματος.

Με αυτή τη συνδυαστική προσέγγιση (fuzzy λογική + LLM), ο ClassifierAgent παρέχει **ευελιξία, ακρίβεια και ανθεκτικότητα** στην κατανόηση φυσικής γλώσσας, αξιοποιώντας τόσο απλούς κανόνες όσο και τεχνητή νοημοσύνη. Η χρήση του τοπικού LLM επιτρέπει την offline λειτουργία και διατηρεί χαμηλό κόστος χωρίς εξωτερικά APIs.

Αυτός ο πράκτορας καλείται από τον main.py για κάθε είσοδο του χρήστη και είναι υπεύθυνος για τη σωστή ενεργοποίηση των υπόλοιπων πρακτόρων επίλυσης (BlockSolverAgent ή JugSolverAgent).

- **BlockSolverAgent**: Υλοποιεί λύση για προβλήματα τύπου blocks με A* αναζήτηση.

Η κλάση BlockSolverAgent υλοποιεί έναν ευφυή πράκτορα που αναλαμβάνει την επίλυση του **blocks world**, όπου ο στόχος είναι η μετακίνηση μπλοκ μεταξύ στοίβων ώστε να δημιουργηθεί μία συγκεκριμένη τελική διάταξη. Ο πράκτορας λαμβάνει ως είσοδο μια περιγραφή του προβλήματος σε φυσική γλώσσα και την επεξεργάζεται για να κατασκευάσει την αρχική και την τελική κατάσταση του προβλήματος.

Αρχικά, ο πράκτορας εφαρμόζει προκαταρκτική επεξεργασία στο κείμενο του χρήστη μέσω των συναρτήσεων strip_accents και normalize_final_sigma, αφαιρώντας τόνους από ελληνικά γράμματα και μετατρέποντας τελικά "ς" σε "σ" αντίστοιχα, ώστε να βελτιωθεί η ακρίβεια στην αναγνώριση των λέξεων και αριθμών. Στη συνέχεια, με τη συνάρτηση parse_blocks_input, χρησιμοποιείται κανονική έκφραση (regex) για να εντοπιστούν όλοι οι αριθμοί που περιλαμβάνονται στην περιγραφή. Έπειτα, η συνάρτηση επιχειρεί να εντοπίσει λέξεις όπως "στοίβα", "stack" ή "pile" για να προσδιορίσει πόσες στοίβες πρέπει να περιλαμβάνει η αρχική κατάσταση, καθώς και να διαχωρίσει τον αριθμό των στοίβων από τους αριθμούς που αντιστοιχούν στα μπλοκ. Τελικά, επιστρέφει δύο λίστες: την αρχική κατάσταση των στοίβων (με όλα τα μπλοκ στην πρώτη στοίβα) και την αυστηρή τελική κατάσταση (με τα μπλοκ πάντα ταξινομημένα στην ίδια στοίβα).

Για την επίλυση του προβλήματος χρησιμοποιείται ο αλγόριθμος αναζήτησης **A***, με την κλάση Node να αναπαριστά κάθε κόμβο στο δέντρο αναζήτησης. Κάθε κόμβος περιλαμβάνει την τρέχουσα κατάσταση, τον στόχο, τον γονικό κόμβο, το κόστος διαδρομής (depth) και μία ευρετική συνάρτηση που μετράει πόσα μπλοκ είναι εκτός θέσης. Η κλάση Node περιλαμβάνει επίσης μεθόδους για την παραγωγή των διαδόχων καταστάσεων, καθώς και τη μέθοδο traceback, η οποία διασχίζει αναδρομικά το μονοπάτι από τη λύση πίσω στη ρίζα και εκτυπώνει όλα τα βήματα που ακολούθησε το σύστημα.

Η αναζήτηση γίνεται με την κλάση PriorityQueue, η οποία υλοποιεί ουρά προτεραιότητας. Οι κόμβοι προστίθενται με προτεραιότητα ανάλογη του συνολικού κόστους $f(n) = g(n) + h(n)$, όπου $g(n)$ είναι το κόστος της διαδρομής και $h(n)$ η εκτίμηση της απόστασης από τον στόχο.

Τέλος, η συνάρτηση solve ενορχηστρώνει όλη τη διαδικασία. Καλεί την parse_blocks_input για να δημιουργήσει τα δεδομένα του προβλήματος, αρχικοποιεί τον κόμβο ρίζας, και ξεκινά τον αλγόριθμο αναζήτησης. Όταν βρεθεί λύση, εκτυπώνεται το πλήρες μονοπάτι από την αρχική στην τελική κατάσταση. Αν δεν βρεθεί λύση, εμφανίζεται κατάλληλο μήνυμα.

- **JugSolverAgent**: Υλοποιεί λύση για το water preblem jug με BFS.

Η κλάση JugSolverAgent είναι ένας πράκτορας που επιλύει το γνωστό πρόβλημα μέτρησης νερού με τη χρήση δοχείων διαφορετικής χωρητικότητας (γνωστό και ως **Water Jug Problem**). Ο χρήστης εισάγει μια περιγραφή σε φυσική γλώσσα, π.χ. "Έχω δοχεία 8, 5 και 3 λίτρων. Θέλω τελικά 4 λίτρα", και ο πράκτορας αναλαμβάνει να κατανοήσει τα δεδομένα, να εντοπίσει τις χωρητικότητες των δοχείων και τον επιθυμητό στόχο, και στη συνέχεια να υπολογίσει τα βήματα που απαιτούνται ώστε να μετρηθεί ακριβώς η ζητούμενη ποσότητα νερού.

Αρχικά, με τη βοήθεια της μεθόδου strip_accents, το κείμενο καθαρίζεται από τόνους και άλλα διακριτικά σημεία ώστε η επεξεργασία φυσικής γλώσσας να είναι πιο αξιόπιστη. Στη συνέχεια, η βασική συνάρτηση extract_jug_data χρησιμοποιεί κανονικές εκφράσεις (regex) για να εντοπίσει όλους τους αριθμούς που αναφέρονται στην περιγραφή. Αυτοί οι αριθμοί πιθανόν να περιλαμβάνουν τόσο τις χωρητικότητες των δοχείων όσο και τον τελικό στόχο.

Για τον διαχωρισμό των δύο, ο πράκτορας προσπαθεί να εντοπίσει λέξεις-κλειδιά που συνήθως συνοδεύουν τον στόχο, όπως "θέλω", "ζητάω", "να έχω", "χρειάζομαι". Μόλις εντοπιστεί μία τέτοια λέξη, ο πράκτορας παίρνει τον επόμενο αριθμό ως τον τελικό στόχο (goal). Όλοι οι υπόλοιποι αριθμοί θεωρούνται οι χωρητικότητες των διαθέσιμων δοχείων. Αν δεν εντοπιστεί λέξη-κλειδί, θεωρεί ως στόχο τον τελευταίο αριθμό στο κείμενο και εξαιρεί αυτόν από τη λίστα των δοχείων.

Η μέθοδος solve εφαρμόζει **Breadth-First Search (BFS)** για την εύρεση της λύσης. Η αρχική κατάσταση ορίζεται ως tuple με όλα τα δοχεία κενά (π.χ. (0, 0, 0)). Η αναζήτηση πραγματοποιείται με ουρά (queue), και για κάθε κατάσταση εφαρμόζονται τρεις δυνατές

ενέργειες ανά δοχείο: γέμισμα, άδειασμα και μεταφορά νερού από το ένα δοχείο στο άλλο. Για να αποφευχθούν επαναλήψεις, κάθε νέα κατάσταση που παράγεται αποθηκεύεται σε ένα σύνολο επισκεπτών καταστάσεων (visited), ώστε να εξετάζεται μόνο μία φορά.

Η επίλυση ολοκληρώνεται όταν σε κάποια κατάσταση, ένα από τα δοχεία περιέχει ακριβώς την επιθυμητή ποσότητα νερού. Τότε το πρόγραμμα εκτυπώνει τα βήματα που οδήγησαν σε αυτή τη λύση, δείχνοντας στο χρήστη κάθε ενέργεια που έγινε (π.χ. "Γέμισμα δοχείου 2", "Μεταφορά από δοχείο 2 στο 3" κ.λπ.), μαζί με την κατάσταση των δοχείων μετά από κάθε βήμα.

Σε περίπτωση που εξαντληθούν όλες οι δυνατές καταστάσεις χωρίς να επιτευχθεί ο στόχος, ο πράκτορας ενημερώνει το χρήστη ότι δεν βρέθηκε λύση.

main.py

Κατά την εκτέλεση, εμφανίζεται ένα μήνυμα στον χρήστη που τον καλεί να πληκτρολογήσει την περιγραφή του προβλήματος. Ο χρήστης εισάγει ελεύθερο κείμενο, π.χ. "Έχω δοχεία 8, 5 και 3 λίτρων. Θέλω 4 λίτρα. Αυτό το κείμενο προωθείται στον πράκτορα ClassifierAgent, ο οποίος είναι υπεύθυνος να αναλύσει το περιεχόμενο και να αποφασίσει αν το πρόβλημα αφορά δοχεία (jug), στοίβες μπλοκ (blocks), ή είναι ασαφές (unknown).

Αφού ο ClassifierAgent επιστρέψει τον τύπο προβλήματος, το πρόγραμμα ενημερώνει το χρήστη για την αναγνώριση του LLM (ή του fuzzy ταξινομητή) και αμέσως ενεργοποιεί τον αντίστοιχο υπο-πράκτορα. Αν το πρόβλημα αναγνωριστεί ως blocks, δημιουργείται στιγμιότυπο του BlockSolverAgent και καλείται η μέθοδος solve() που εφαρμόζει A* αναζήτηση για να βρει λύση. Αντίστοιχα, αν το πρόβλημα αναγνωριστεί ως jug, ενεργοποιείται ο JugSolverAgent, ο οποίος εφαρμόζει BFS για την επίλυση.

Αν η ταξινόμηση αποτύχει, δηλαδή ο ClassifierAgent επιστρέψει unknown, τότε το main.py εμφανίζει ένα κατάλληλο μήνυμα λάθους στον χρήστη, ενημερώνοντάς τον ότι το σύστημα δεν μπόρεσε να καταλάβει τον τύπο του προβλήματος. Έτσι, το αρχείο main.py λειτουργεί ως ο **συνδετικός κρίκος** ανάμεσα στην είσοδο φυσικής γλώσσας και στους πράκτορες επίλυσης, ενσωματώνοντας όλα τα μέρη του συστήματος σε μια αρμονική και λειτουργική ροή.

4. Τεχνολογίες που χρησιμοποιήθηκαν

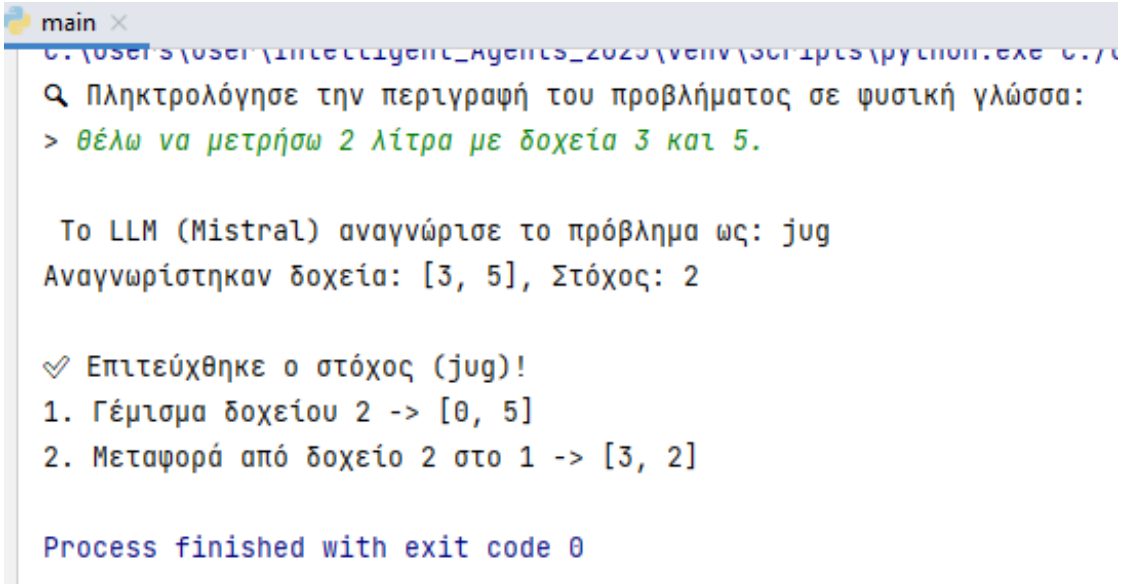
- Python 3
- Ollama (τοπικός LLM runner)
- Mistral 7B (τοπικό LLM)
- βιβλιοθήκες: requests, copy, re, difflib, collections
- A* και BFS για λύση προβλημάτων

5. Δυσκολίες που Αντιμετωπίσαμε

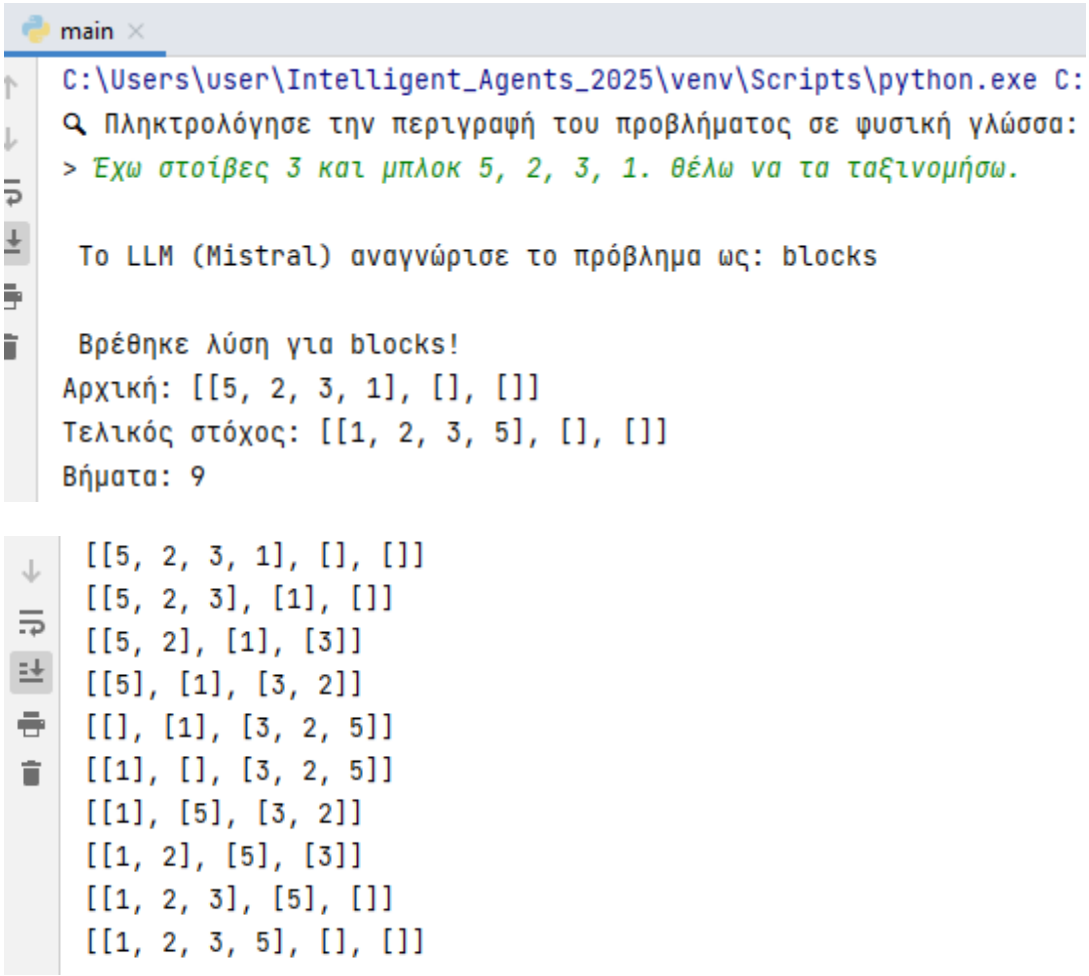
- Η αναγνώριση φυσικής γλώσσας από το LLM Mistral δεν ήταν πάντα αξιόπιστη.
- Προσθέσαμε λογική με fuzzy matching (difflib) ως πρώτη γραμμή ταξινόμησης.
- Χρειάστηκε ρύθμιση σωστών prompts προς το LLM για να απαντά μόνο 'blocks', 'jug', ή 'unknown'.
- Επίσης υπήρχε πρόκληση στο να διαχωρίζουμε τον στόχο (π.χ. 4 λίτρα) από τα δοχεία στην ανάλυση αριθμών.

6. Παραδείγματα Εκτέλεσης

Παράδειγμα 1



Παράδειγμα 2



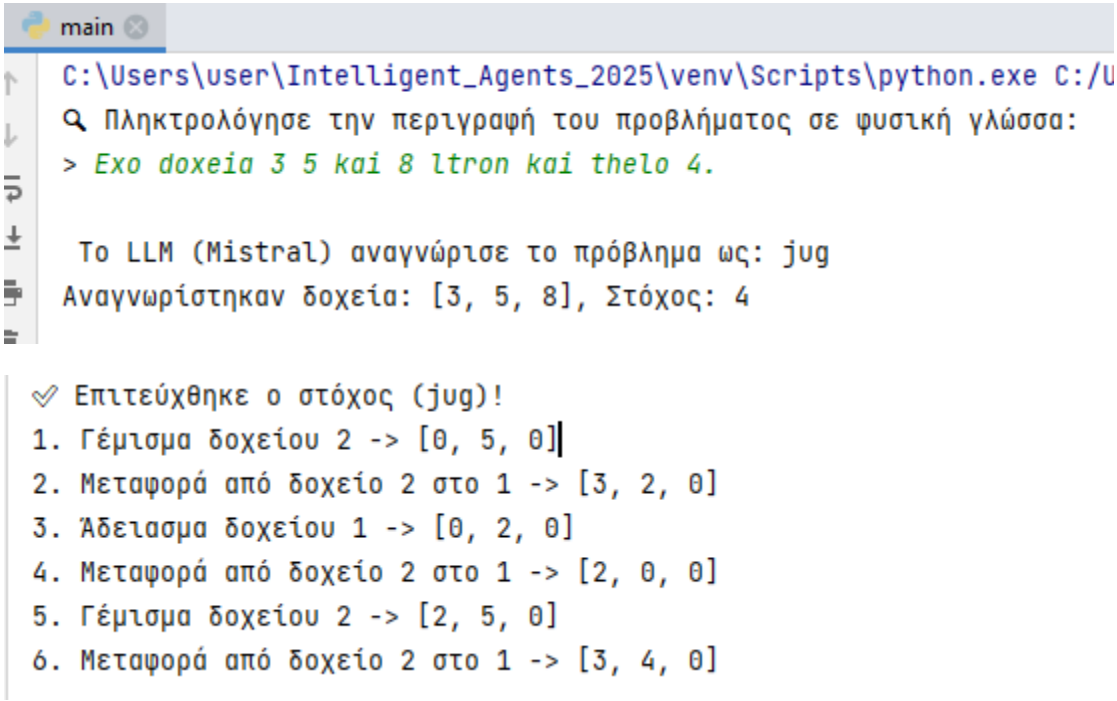
Παράδειγμα μη αναγνωρίσιμου input (unknown)

Παράδειγμα 3

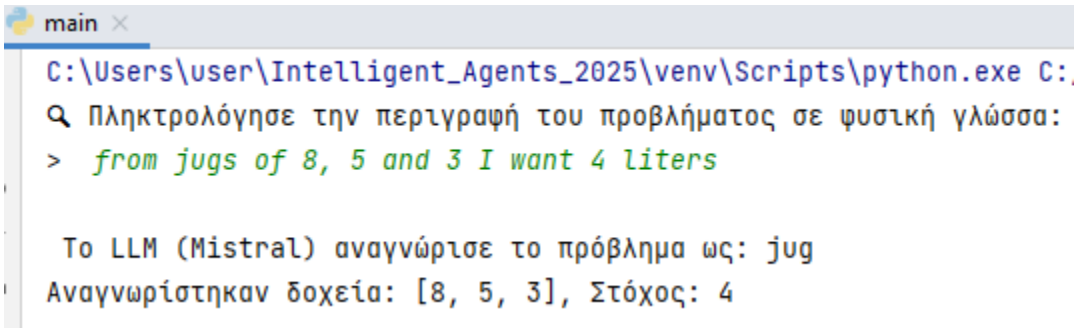
```
Πληκτρολόγησε την περιγραφή του προβλήματος σε φυσική γλώσσα:
> Έχω τρία αντικείμενα και θέλω να κάνω κάτι με αυτά

 Το LLM (Mistral) αναγνώρισε το πρόβλημα ως: blocks (for problems with blocks/puzzles)
Δεν κατάλαβα τον τύπο του προβλήματος.
```

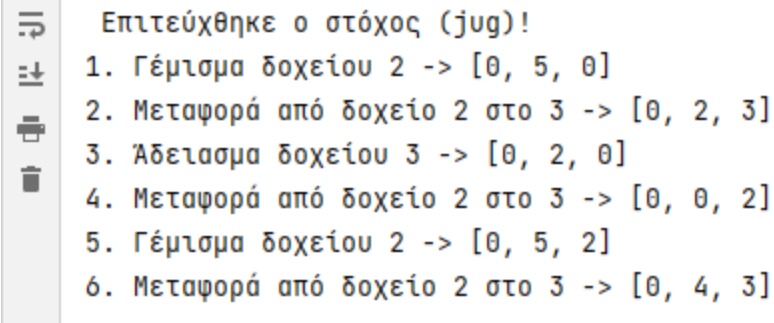
Παράδειγμα με λάθη ως Input (ή γραμμένα με διαφορετικό τρόπο)

Παράδειγμα 4



Παράδειγμα 5





7. Βιβλιογραφία-Πηγές

[1] https://en.wikipedia.org/wiki/Blocks_world

[2] https://www.futurelearn.com/info/courses/recreational-math/0/steps/43519

[3] https://ollama.com

[4] https://cw.fel.cvut.cz/b202/courses/b4b36zui/tasks/task1-2021

[5] https://www.geeksforgeeks.org/water-jug-problem-using-bfs/