Πανεπιστήμιο Πειραιώς
Τμήμα Πληροφορικής
Έτος: 2023 - 2024

Μάθημα:
«ΕΥΦΥΕΙΣ ΠΡΑΚΤΟΡΕΣ»
Εργασία: Υπολογιστική Εργασία Μαθήματος
Εξάμηνο: 8ο

<u>Ομάδα εργασίας:</u>
Αιμιλιανός Κουρπάς-Δανάς Π20100,

Αναστάσιος Μελαχροινούδης Π20124

Ημερομηνία παράδοσης : 13.07.2024

Περιεχομενα

Εκφώνηση Εργασίας

3 Ανάπτυξη generic planner

Περιγραφή

Ένας γεννήτορας σχεδίων (plan generation, planner) παράγει μια ακολουθία ενεργειών ώστε, όταν εκτελεστούν οι ενέργειες ο κόσμος του πράκτορα να βρεθεί από μια αρχική σε μια τελική κατάσταση.
Ο planner θεωρείται generic αν δεν αλλάζουμε σε τίποτα τον κώδικα του για την λύση διαφορετικών προβλημάτων.

Κάθε πρόβλημα που προσπαθούμε να επιλύσουμε διαφοροποιείται από τα άλλα μόνο από την διαφορετική περιγραφή του αρχικού, του τελικού, αλλά και οποιουδήποτε ενδιάμεσου στιγμιοτύπου του κόσμου. Δηλαδή για κάθε πρόβλημα έχουμε μεν τον ίδιο τρόπο αναπαράστασης των καταστάσεων του κόσμου, αλλά (πιθανώς) διαφορετική αναπαράσταση από την αναπράσταση σε άλλα προβλήματα.

Επίσης για κάθε πρόβλημα διαθέτουμε ένα διαφορετικό σετ ενεργειών που μπορούμε να εκτελέσουμε.

1. Αναζητείστε κώδικα ενός generic planner στο διαδίκτυο, ή αναπτύξτε τον δικό σας κώδικα.
2. Τρέξτε τον planner για 2 διαφορετικά προβλήματα : το Blocks World και το Water jug

https://en.wikipedia.org/wiki/Blocks_world

https://en.wikipedia.org/wiki/Water_pouring_puzzle

Τεκμηρίωση

Η τεκμηρίωση της εφαρμογής θα περιλαμβάνει τα εξής:
1. Περιγραφή του προβλήματος

2. Περιγραφή της θεωρητικής βάσης της εφαρμογής, συμπεριλαμβανομένων των δομών δεδομένων και αναπαράστασης γνώσης που υιοθετήθηκαν, των αλγορίθμων και μεθοδολογιών που χρησιμοποιήθηκαν, καθώς και των προσαρμογών και μεταβολών που έγιναν στα παραπάνω προκειμένου να είναι δυνατή η εφαρμογή τους στο συγκεκριμένο πρόβλημα

3. Περιγραφή σημαντικών σχεδιαστικών αποφάσεων και στοιχείων υλοποίησης

4. Ολοκληρωμένη περιγραφή μίας παραδειγματικής εκτέλεσης και των αποτελεσμάτων της τόσο για το Blocks World όσο και για το Water jug.

5. Αναλυτική περιγραφή της διαδικασίας εγκατάστασης

6. Περιγραφή πρόσθετων δυνατοτήτων της εφαρμογής, εάν υπάρχουν

8. Αναλυτική περιγραφή ανοικτών θεμάτων, ανεπίλυτων προβλημάτων και πιθανοτήτων εμφάνισης σφαλμάτων κατά την εκτέλεση

Είναι σημαντικό να υπάρχουν αναλυτικά και επεξηγημένα screenshots από την εκτέλεση της εφαρμογής.

Η εφαρμογή θα διαθέτει άμεσα εκτελέσιμο πρόγραμμα που δεν θα απαιτεί να κατέβουν διάφορα περιβάλλοντα για να εκτελεστεί.

Παραδοτέα

1. Η εφαρμογή σε εκτελέσιμη μορφή (π.χ. άν είναι σε Unity πρέπει να έχει project και να έχει γίνει build)

2. Πηγαίος κώδικας για το σύνολο της εφαρμογής

3. Τεκμηρίωση του planner και των αναπαραστάσεων των καταστάσεων και των ενεργειών για τα 2 προβλήματα

4. Τα αποτελέσματα του planner για τα 2 προβλήματα

5. video + powerpoint

ΣΗΜΕΙΩΣΗ :

Η επιλογή γλωσσών και πλατφόρμας υλοποποίησης είναι ελεύθερη.

Περιγραφή Εργασίας

Ανάπτυξη Generic Planner

Η κατασκευή ενός γεννήτορα σχεδίων, γνωστού και ως planner, έχει ως σκοπό την παραγωγή μιας ακολουθίας ενεργειών που, όταν εκτελεστούν, θα μεταφέρουν τον κόσμο του πράκτορα από μια αρχική σε μια τελική κατάσταση. Η διαδικασία αυτή περιλαμβάνει την ανάλυση της αρχικής κατάστασης, τον προσδιορισμό των διαθέσιμων ενεργειών και τον σχεδιασμό μιας σειράς βημάτων που θα οδηγήσουν στην επιθυμητή κατάσταση.

Τι είναι Generic Planner;

Ο generic planner είναι ένας τύπος σχεδιαστικού αλγορίθμου που χαρακτηρίζεται από την ικανότητά του να επιλύει διάφορα προβλήματα χωρίς να απαιτούνται τροποποιήσεις στον κώδικα του. Αυτό σημαίνει ότι μπορεί να εφαρμοστεί σε διαφορετικά πλαίσια και σενάρια απλά με την παροχή των κατάλληλων δεδομένων εισόδου.

Πλεονεκτήματα και Μειονεκτήματα

Πλεονεκτήματα:

1) Ευελιξία: Μπορεί να χρησιμοποιηθεί για την επίλυση διαφορετικών προβλημάτων χωρίς αλλαγές στον κώδικα.
2) Επεκτασιμότητα: Μπορεί να διαχειριστεί αυξανόμενη πολυπλοκότητα των προβλημάτων.
3) Συντήρηση: Ο κώδικας είναι πιο εύκολος στη συντήρηση και την αναβάθμιση λόγω της γενικότητάς του.

Μειονεκτήματα:

1) Απόδοση: Η γενικότητα μπορεί να οδηγήσει σε χαμηλότερη απόδοση σε σχέση με εξειδικευμένους planners που είναι βελτιστοποιημένοι για συγκεκριμένα προβλήματα.
2) Πολυπλοκότητα Εφαρμογής: Η υλοποίηση ενός πραγματικά generic planner μπορεί να είναι πιο περίπλοκη και να απαιτεί περισσότερο χρόνο και προσπάθεια.

Περιγραφή προβλήματος Block world:

Ο κόσμος των μπλοκ (Block World) είναι ένας από τους πιο γνωστούς τομείς σχεδιασμού στην τεχνητή νοημοσύνη. Αυτός ο τομέας αναπαριστάται συνήθως με μια συλλογή από ξύλινα μπλοκ διαφόρων σχημάτων και χρωμάτων που βρίσκονται τοποθετημένα πάνω σε ένα τραπέζι. Ο στόχος είναι η δημιουργία μίας ή περισσότερων κάθετων στίβων από μπλοκ σύμφωνα με συγκεκριμένους κανόνες.

Βασικοί Κανόνες και Περιορισμοί

1) . Μετακίνηση Μπλοκ:

   - Μόνο ένα μπλοκ μπορεί να μετακινηθεί κάθε φορά.

   - Ένα μπλοκ μπορεί είτε να τοποθετηθεί στο τραπέζι είτε πάνω σε ένα άλλο μπλοκ.

2) Περιορισμοί Μετακίνησης:

   - Μπλοκ που βρίσκονται κάτω από άλλα μπλοκ δεν μπορούν να μετακινηθούν μέχρι να αφαιρεθούν τα μπλοκ που βρίσκονται από πάνω τους.

   - Ορισμένα είδη μπλοκ δεν επιτρέπεται να έχουν άλλα μπλοκ τοποθετημένα πάνω τους.

Στόχος

Ο στόχος είναι να οργανωθούν τα μπλοκ έτσι ώστε να σχηματίζουν μία ή περισσότερες κάθετες στοίβες σύμφωνα με συγκεκριμένους κανόνες και περιορισμούς που ορίζει το πρόβλημα. Αυτό μπορεί να περιλαμβάνει την επίτευξη μιας συγκεκριμένης διάταξης ή την κατασκευή μιας στοίβας με καθορισμένο ύψος.

Αντιμετώπιση με Τεχνητή Νοημοσύνη

Η απλότητα και η δομή αυτού του προβλήματος το καθιστούν ιδανικό για την εφαρμογή κλασικών προσεγγίσεων στην τεχνητή νοημοσύνη. Το πρόβλημα διαμορφώνεται ως ένα σύνολο αφηρημένων συμβόλων και κανόνων που μπορούν να αναπαρασταθούν λογικά. Ένας αλγόριθμος σχεδιασμού μπορεί να χρησιμοποιηθεί για να βρει την κατάλληλη ακολουθία ενεργειών που θα οδηγήσουν στη λύση του προβλήματος.

Περιγραφή Προβλήματος Water Jug

Το πρόβλημα με τις jug νερού (water jug problem) είναι μια κατηγορία γρίφων που περιλαμβάνει τη διαχείριση jug με γνωστές ακέραιες χωρητικότητες (σε λίτρα ή γαλόνια). Το πρόβλημα είναι επίσης γνωστό ως παζλ μέτρησης ή παζλ απόφραξης και έχει ως στόχο τον καθορισμό της κατάλληλης σειράς ενεργειών για την επίτευξη ενός προκαθορισμένου όγκου νερού σε μία ή περισσότερες jug.

Βασικοί Κανόνες και Περιορισμοί

1) Δεδομένα:

   - Υπάρχει μια πεπερασμένη συλλογή από jug νερού με γνωστές ακέραιες χωρητικότητες.

   - Κάθε jug αρχικά περιέχει έναν γνωστό ακέραιο όγκο υγρού, ο οποίος δεν είναι απαραίτητα ίσος με την πλήρη χωρητικότητα της jug.

2) Ενέργειες:

   - Μπορείς να γεμίσεις μια jug μέχρι την πλήρη χωρητικότητά της.

   - Μπορείς να αδειάσεις μια jug εντελώς.

   - Μπορείς να μεταφέρεις νερό από μία jug σε μία άλλη μέχρι είτε να αδειάσει η πρώτη jug είτε να γεμίσει η δεύτερη.

3) Στόχος:

- Ο στόχος καθορίζεται ως ένας συγκεκριμένος όγκος υγρού που πρέπει να υπάρχει σε μία ή περισσότερες jug. Το πρόβλημα ρωτάει πόσα βήματα απαιτούνται για να φτάσει σε αυτήν την κατάσταση στόχου.

Λύση με Αλγορίθμους Τεχνητής Νοημοσύνης

Η λύση αυτού του προβλήματος μπορεί να επιτευχθεί μέσω της εφαρμογής αλγορίθμων αναζήτησης και σχεδιασμού. Η προσέγγιση αυτή περιλαμβάνει τη διαμόρφωση του προβλήματος ως ένα σύνολο καταστάσεων και ενεργειών, και την αναζήτηση της ακολουθίας βημάτων που θα οδηγήσουν από την αρχική στην τελική κατάσταση.

Πλεονεκτήματα και Μειονεκτήματα

Πλεονεκτήματα:

- Εκπαιδευτική Αξία: Το πρόβλημα των jug νερού είναι ένα εξαιρετικό εργαλείο για τη διδασκαλία βασικών αρχών της λογικής, της αναζήτησης και του προγραμματισμού.

- Απλότητα:*Η απλότητα των κανόνων το καθιστά κατανοητό και προσβάσιμο για μαθητές και αρχάριους στην τεχνητή νοημοσύνη.

Μειονεκτήματα:

- Περιορισμένη Πρακτική Εφαρμογή: Αν και είναι χρήσιμο για εκπαίδευση, η πρακτική εφαρμογή του προβλήματος είναι περιορισμένη σε πραγματικούς κόσμους.

- Αυξανόμενη Πολυπλοκότητα: Η πολυπλοκότητα μπορεί να αυξηθεί δραματικά με την προσθήκη περισσότερων jug ή διαφορετικών χωρητικοτήτων.

Περιγραφή Θεωρητικής Βάσης Εφαρμογής

Ένας σχεδιαστής (planner) δημιουργεί μια ακολουθία ενεργειών έτσι ώστε, όταν αυτές οι ενέργειες εκτελούνται, ο κόσμος του πράκτορα μεταβαίνει από την αρχική στην τελική κατάσταση. Ο σχεδιαστής θεωρείται γενικός όταν μπορεί να λύσει διαφορετικά προβλήματα χωρίς να απαιτείται καμία αλλαγή στον κώδικά του.

Για τη δημιουργία της ακολουθίας ενεργειών, ο σχεδιαστής χρησιμοποιεί έναν αλγόριθμο αναζήτησης για να κατασκευάσει ένα δέντρο αναζήτησης. Η θεωρητική βάση της εφαρμογής έχει στόχο την επίλυση δύο διαφορετικών προβλημάτων: το Blocks World και το Water Jug. Για την επίλυση αυτών των προβλημάτων, χρησιμοποιήσαμε τον αλγόριθμο A* (A* algorithm). Ο A* είναι ένας αλγόριθμος διασταύρωσης γραφήματος και αναζήτησης διαδρομής, ο οποίος χρησιμοποιείται συχνά σε πολλούς τομείς της επιστήμης των υπολογιστών λόγω της πληρότητάς του, της βελτιστοποίησης και της βέλτιστης απόδοσής του.

Όπως ο αλγόριθμος Dijkstra, ο A* λειτουργεί δημιουργώντας ένα δέντρο διαδρομής χαμηλότερου κόστους από τον κόμβο έναρξης έως τον κόμβο προορισμού. Αυτό που κάνει τον A* διαφορετικό και συχνά προτιμότερο είναι η χρήση της συνάρτησης f(n), η οποία δίνει μια εκτίμηση του συνολικού κόστους μιας διαδρομής που περιλαμβάνει τον συγκεκριμένο κόμβο.

Επιπλέον, για την επίλυση του προβλήματος Blocks World χρησιμοποιήσαμε και τον αλγόριθμο Best First Search. Ουσιαστικά, η χρήση του είναι ως **εξής**:

1. Δημιουργεί δύο κενές λίστες: OPEN και CLOSED.

2. Ξεκινά από τον αρχικό κόμβο (N) και τον τοποθετεί στη λίστα OPEN.

3. Επαναλαμβάνει τα παρακάτω βήματα έως ότου επιτευχθεί ο κόμβος GOAL:

- **Αν** η λίστα OPEN είναι κενή, τότε τερματίζει και επιστρέφει "False".

- Επιλέγει τον πρώτο κόμβο (N) από τη λίστα OPEN και τον μετακινεί στη λίστα CLOSED, καταγράφοντας επίσης τις πληροφορίες του γονικού κόμβου.

- **Αν** ο N είναι ο κόμβος GOAL, μετακινεί τον κόμβο στη λίστα CLOSED και επιστρέφει "True". Η λύση μπορεί να βρεθεί ακολουθώντας το μονοπάτι.

- **Αν** ο N δεν είναι ο κόμβος GOAL, αναπτύσσει τον κόμβο N για να δημιουργήσει τους "άμεσους" επόμενους κόμβους που συνδέονται με τον κόμβο N και προσθέτει όλους αυτούς στη λίστα OPEN, αναδιατάσσοντας τους κόμβους σε αύξουσα σειρά σύμφωνα με τη συνάρτηση αξιολόγησης f(n).

Αυτός ο αλγόριθμος διασχίζει πρώτα τη συντομότερη διαδρομή. Η χρονική πολυπλοκότητά του είναι O(n * log n).

Η σχεδίαση της εφαρμογής έγινε με γνώμονα τη λύση των προβλημάτων στη γλώσσα Python, αφού πρώτα πραγματοποιήθηκε αναλυτική μελέτη στο διαδίκτυο για τα προβλήματα Water Jug και Blocks World. Κατά τη μελέτη, έγινε κατανοητό ότι η επίλυση των προβλημάτων μπορεί να επιτευχθεί με τη χρήση τόσο του αλγορίθμου Best First Search όσο και του A* algorithm. Επιπλέον, η γλώσσα προγραμματισμού Python διευκολύνει τη διαδικασία χάρη στις πολλές βιβλιοθήκες που μπορούν να χρησιμοποιηθούν.

Περιγραφή Σημαντικών Σχεδιαστικών Αποφάσεων και Στοιχείων Υλοποίησης

Όταν το πρόγραμμά μας ξεκινά, εκτελείται η κύρια συνάρτηση (main). Αυτή καλεί τη συνάρτηση `showMenu()` από το αρχείο `menu.py`, η οποία εμφανίζει το μενού επιλογών. Ο χρήστης έχει τις εξής δυνατότητες: να επιλέξει 1 για το πρόβλημα των κανάτων (jugs problem), 2 για το πρόβλημα του Block World ή Q για να τερματίσει το πρόγραμμα.

Λύση του Προβλήματος των Κανατών (Jugs Problem)

Επιλέγοντας 1, ο χρήστης εισάγει τη χωρητικότητα των κανάτων και τον επιθυμητό όγκο νερού. Αφού ελέγξουμε ότι οι τιμές είναι έγκυρες και όχι μηδενικές, περνάμε τα δεδομένα στη συνάρτηση `bfs()` από το αρχείο `SearchAlgorithms.py`. Εκεί υλοποιείται ο αλγόριθμος Αναζήτησης κατά Πλάτος (Breadth First Search).

Η Αναζήτηση κατά Πλάτος είναι ένας αλγόριθμος που χρησιμοποιείται για τη διάσχιση ή την αναζήτηση σε δομές δεδομένων όπως δέντρα ή γράφους. Ξεκινά από τη ρίζα ή από έναν αυθαίρετο κόμβο και εξερευνά πρώτα τους γειτονικούς κόμβους, πριν προχωρήσει στους γείτονες του επόμενου επιπέδου. Η αναζήτηση αυτή

υλοποιείται με τη χρήση μιας ουράς, όπου εισάγονται διαδοχικά οι κόμβοι κάθε επιπέδου.

Η διαδικασία περιλαμβάνει δύο κύριες φάσεις:

1. Αρχική Φάση: Η ουρά είναι αρχικά κενή και ο κόμβος της ρίζας προστίθεται σε αυτήν.

2. Γενική Φάση: Όσο η ουρά δεν είναι κενή, εξάγεται και επεξεργάζεται ο πρώτος κόμβος. Οι γειτονικοί κόμβοι του εισάγονται στη συνέχεια στην ουρά. Αν ο κόμβος είναι φύλλο, δεν εισάγεται τίποτα.

Λύση του Προβλήματος Block World

Με την επιλογή 2, ο χρήστης εισάγει τον αριθμό των μπλοκ. Δημιουργούμε τυχαίες αρχικές και τελικές καταστάσεις για τα μπλοκ, διασφαλίζοντας ότι αυτές δεν είναι ίδιες. Στη συνέχεια, καλούμε τη συνάρτηση `BlocksWorldH2`, η οποία υπολογίζει τον αριθμό των κινήσεων που απαιτούνται για να τοποθετηθούν τα μπλοκ στη σωστή θέση. Στη συνέχεια, καλούμε τη συνάρτηση `a_star()` από το αρχείο `SearchAlgorithms.py`, όπου υλοποιείται ο αλγόριθμος A*.

Ο αλγόριθμος A* χρησιμοποιείται για την εύρεση της συντομότερης διαδρομής από μία καθορισμένη αρχή σε έναν καθορισμένο στόχο. Αυτός ο αλγόριθμος διαφέρει από άλλους, καθώς εστιάζει αποκλειστικά στη συντομότερη διαδρομή προς έναν συγκεκριμένο στόχο, χρησιμοποιώντας ευρετικές μεθόδους για την κατεύθυνση της αναζήτησης.

Τέλος, εκτυπώνουμε τις καταστάσεις που προκύπτουν μέχρι την επίτευξη του τελικού στόχου.

Επεξήγηση Λειτουργίας

Το μενου μας :

```
aa@As-MacBook-Pro intelligent-agents % python3 main.py

Welcome!
Please select a problem to solve:
1 - Water Jug Problem
2 - Block World
Exit - Exit the program
Your choice>
```

1.Water Jug Problem (Water Pouring Puzzle)

```
aa@As-MacBook-Pro intelligent-agents % python3 main.py

Welcome!
Please select a problem to solve:
1 - Water Jug Problem
2 - Block World
Exit - Exit the program
Your choice> 1
Please enter the capacity of Jug 1: 5
Please enter the capacity of Jug 2: 6
Which jug should hold the target amount? (Enter 1 or 2): 1
[✓] Solution found for the Water Pouring Puzzle!
[(0, 0), (5, 0), (0, 6), (5, 6), (0, 5), (5, 1)]
```

Ο χρηστης εχει επιλεξει την επιλογη 1 και εχει βαλει την χωριτικότητα στα Jugs.

Κανόνες

1) **Γέμισμα της Κανάτας 1:**

   (jug_1_capacity, y)

   Γεμίζουμε την Κανάτα 1 μέχρι τη μέγιστη χωρητικότητά της, ενώ η Κανάτα 2 παραμένει στην ίδια κατάσταση.

2) **Γέμισμα της Κανάτας 2:**

   (x, jug_2_capacity)

   Γεμίζουμε την Κανάτα 2 μέχρι τη μέγιστη χωρητικότητά της, ενώ η Κανάτα 1 παραμένει στην ίδια κατάσταση.

3) **Άδειασμα της Κανάτας 1:**

   (0, y)

   Αδειάζουμε πλήρως την Κανάτα 1, ενώ η Κανάτα 2 παραμένει στην ίδια κατάσταση.

4) **Άδειασμα της Κανάτας 2:**

   (x,0)

   Αδειάζουμε πλήρως την Κανάτα 2, ενώ η Κανάτα 1 παραμένει στην ίδια κατάσταση.

5) **Μεταφορά από την Κανάτα 2 στην Κανάτα 1:**

(min(x + y, jug_1_capacity), max(0, x + y - jug_1_capacity))

Μεταφέρουμε νερό από την Κανάτα 2 στην Κανάτα 1 μέχρι η Κανάτα 1 να γεμίσει ή η Κανάτα 2 να αδειάσει. Η νέα ποσότητα νερού στην Κανάτα 1 είναι η ελάχιστη μεταξύ του συνολικού νερού και της χωρητικότητάς της. Η νέα ποσότητα νερού στην Κανάτα 2 είναι η υπόλοιπη ποσότητα μετά το γέμισμα της Κανάτας 1.

6) **Μεταφορά από την Κανάτα 1 στην Κανάτα 2:**

(max(0, x + y - jug_2_capacity), min(x + y, jug_2_capacity))

Μεταφέρουμε νερό από την Κανάτα 1 στην Κανάτα 2 μέχρι η Κανάτα 2 να γεμίσει ή η Κανάτα 1 να αδειάσει. Η νέα ποσότητα νερού στην Κανάτα 2 είναι η ελάχιστη μεταξύ του συνολικού νερού και της χωρητικότητάς της. Η νέα ποσότητα νερού στην Κανάτα 1 είναι η υπόλοιπη ποσότητα μετά το γέμισμα της Κανάτας 2.

Αποσπασμα του κωδικα με τους κανόνες:

```
next_states = [
    (jug_1_capacity, y),  # Fill Jug 1
    (x, jug_2_capacity),  # Fill Jug 2
    (0, y),  # Empty Jug 1
    (x, 0),  # Empty Jug 2
    (min(x + y, jug_1_capacity), max(0, x + y - jug_1_capacity)),  # Pour Jug 2 into Jug 1
    (max(0, x + y - jug_2_capacity), min(x + y, jug_2_capacity))  # Pour Jug 1 into Jug 2
]

for state in next_states:
    if state not in visited:
        front.append(state)
        visited.add(state)
```

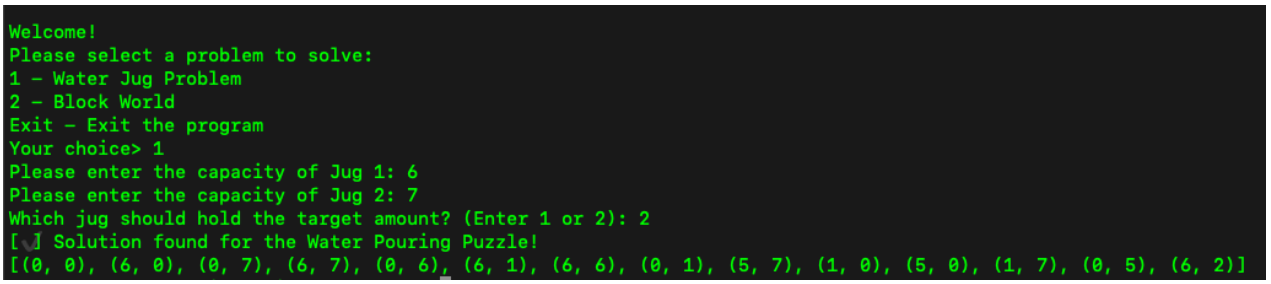
2.Block world

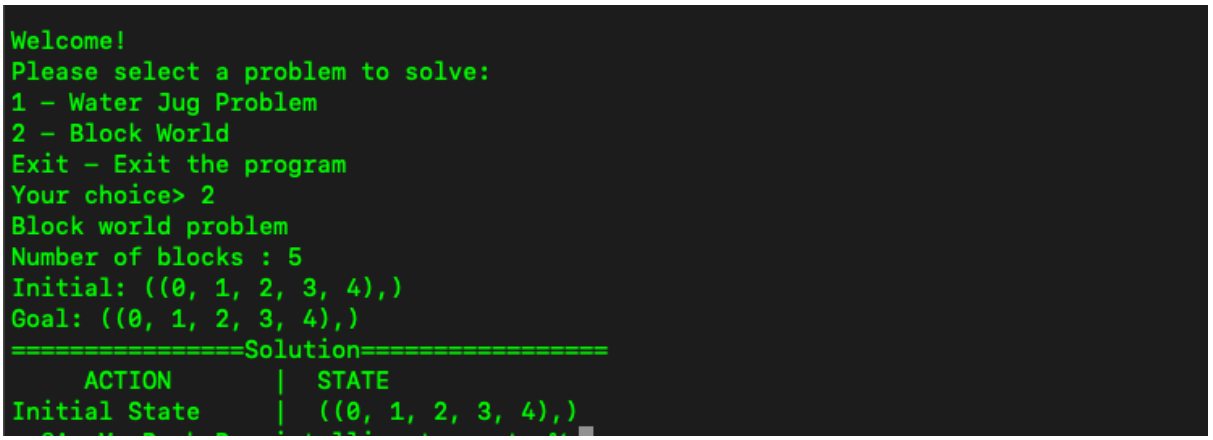
Ο χρηστης εχει επιλεξει την επιλογη 2 και εχει βαλει τον αριθμο των blocks.

```
Welcome!
Please select a problem to solve:
1 - Water Jug Problem
2 - Block World
Exit - Exit the program
Your choice> 2
Block world problem
Number of blocks : 4
Initial: ((0, 1, 2, 3),)
Goal: ((0, 1, 2, 3),)
================Solution================
     ACTION       |  STATE
Initial State     |  ((0, 1, 2, 3),)
```

Αρχικά, δημιουργούνται τυχαίες καταστάσεις για την αρχική και την τελική κατάσταση.
Η υλοποίηση πραγματοποιείται με τη χρήση των αλγορίθμων A* και BFS (best first search), οι οποίοι καταλήγουν στην τελική κατάσταση στόχου (goal).

Παραδείγματα από στιγμιότυπα οθόνης





Περιγραφή διαδικασίας εγκατάστασης

1. Κατεβάζουμε την python 3.10

https://www.python.org/downloads/

2.Βρισκουμε το terminal στο path που βρίσκεται το main.py , και γράφετε
`python3 main.py`

Βιβλιογραφία

1) IntelligentAgents(σημειώσεις από το μάθημα gunet2)
2) https://www.geeksforgeeks.org/water-jug-problem-using-bfs/
3) https://www.geeksforgeeks.org/a-search-algorithm/
4) https://en.wikipedia.org/wiki/Blocks_world#:~:text=The%20blocks%20world%20is%20one,more%20vertical%20stacks%20of%20blocks.