**Τεχνητή Νοημοσύνη - Εργασία**

*Άσκηση 1*

Για την άσκηση έχουμε να εφαρμόσουμε τους αλγόριθμους UCS και A\* ώστε να ταξινομούμε τους κύβους με την σειρά, σε κάθε γραμμή. Η υλοποίηση έχει πραγματοποιηθεί σε java με συνολικά 7 κλάσεις.

***Κλάσεις και σημαντικές μέθοδοι***

Cube: Η Cube δημιουργεί ξεχωριστά τον κάθε κύβο με τις συντεταγμένες του και το νούμερο του.

* printCube(): Τυπώνει τον κύβο με τον αριθμό.
* copy(): Δημιουργεί ένα αντίγραφο του κύβου και το κάνει return.

CubeLine: Η CubeLine δημιουργεί και τοποθετεί τους κύβους με τυχαία νούμερα στην σειρά. Η κάθε σειρά θα έχει Κ κύβους.

* CubeLine(int start , int numOfCubesPerLine , int size): Αυτός είναι ο constructor, που δημιουργεί μια λίστα απο cubes και τα κάνει randomize ώστε να μπουν τυχαία στην σειρά.
* isInOrder(): Ελέγχει αν όλοι οι κύβοι είναι στην σειρά.
* moveCube(): Μετακινεί έναν κύβο σε μια συγκεκριμένη θέση.
* copy(): Δημιουργεί ένα αντίγραφο των σειρών.

CubeManager: Στην CubeManager γίνεται η διαχείρηση των κύβων, δηλαδή η μετακίνησή τους και η ανάθεση κόστων ανά κίνηση.

* moveCube(): Ελέγχει αν μπορούμε να μετακινήσουμε έναν κύβο. Αν μπορούμε, το μετακινούμε.
* calculateAndIncreaseCost(): Υπολογίζει το κόστος κάθε κίνησης και το προσθέτει στο ολικό κόστος μετακινήσεων του παιχνιδιού.
* calculationOfMove(): Εδώ θέτουμε τα κόστη της κάθε κίνησης. Πχ, αν μετακινήσουμε έναν κύβο στο ίδιο επίπεδο, τοτε το κόστος από την άσκηση θα είναι 0.75.
* printCubeLine(): Τυπώνει όλα τα cubes στην σειρά.

CubeMatrix: H κλάση CubeMatrix δημιουργεί το σύνολο σειρών των κύβων, δηλαδή τον πίνακα από κύβους.

* CubeMatrix(): Είναι ο constructor και δημιουργεί όλες τις σειρές.
* isMoveable(): Βλέπει αν ένας κύβος είναι μετακινήσιμος. Υπάρχουν πολλοί μέθοδοι που βλέπουν αν ένας κύβος είναι ελεύθερος να μετακινηθεί, αλλά αυτή είναι η βασική ιδέα.
* findSmallestMovableCube(): Βρίσκει τον μικρότερο αριθμό σε κύβο που μπορεί να μετακινηθεί.
* getAboveCube(): Παίρνει τον κύβο ακριβώς πάνω από τον current κύβο.
* getBelowCube(): Παίρνει τον κύβο ακριβώς από κάτω από τον current κύβο.
* isInOrder(): Τσεκάρει αν όλα βρίσκονται στην σωστή θέση.
* copy(): Δημιουργεί ένα αντίγραφο του CubeMatrix.
* getFinalPositionOfCube(): Μετά την μετακίνηση ή την αλλαγή, παίρνει την τελική θέση του κύβου από το matrix.
* getManhatamDistanceFromFinalPosition(): Υπολογίζει και παίρνει την απόσταση Manhatam από το position που βρίσκεται ο κύβος με το position που θα έπρεπε να βρίσκεται στην σωστή του θέση ο κύβος.

Node: Δημιουργεί το δέντρο που χρησιμοποιούν οι αλγόριθμοι UCS και A\*. Δημιουργεί παιδιά έως ότου φτάσει σε μια τελική κατάσταση.

* getDeepestChildren(): Παίρνει τα παιδιά που βρίσκονται στο μεγάλυτερο βάθος.
* cleanTree(): Καθαρίζει όλα τα nodes και κρατάει τα τελευταία ώστε να φτάσει η μνήμη.

Botaki20: Υλοποιεί τους αλγόριθμους UCS και A\*, καθώς και την ευρετική συνάρτηση υπολογίζοντας την καλύτερη κίνηση.

* calculateCostOfMove(): Υπολογίζει το κόστος κάθε κίνησης κύβου.
* calculateAllPossibleMovesForCube(): Υπολογίζει όλες τις πιθανές κινήσεις για έναν κύβο.
* expandNode(): Παίρνει έναν κόμβο-πατέρα, τον επεκτείνει και δίνει τα παιδιά του.
* calculateHeuristicCost(): Υπολογίζει μέσω ευρετικής συνάρτησης το κόστος.
* UCS(): Υλοποιεί τον UCS αλγόριθμο.
* AStar(): Υλοποιεί τον Α\* αλγόριθμο.

Table: Στο table πραγματοποιειται το παιχνίδι. Χρησιμοποιεί όλες τις προηγούμενες κλάσεις ώστε να συνθέσει το τελικό παιχνίδι.

Το UML διάγραμμα έιναι το εξής:



UCS αλγόριθμος:

A\* αλγόριθμος:

Διαφορά μεταξύ UCS και A\*: