**Τεχνητή Νοημοσύνη, Διάσχιση της γέφυρας**

Καράμπελας Γεώργιος 3180072

Ραυτόπουλος Μάριος 3180163

Χριστουλάκης Αντώνιος 3180207

**Έχουμε τρεις κλάσεις State, SpaceSearcher, CrossBridge.**

Στην κλάση State φτιάχνουμε αντικείμενα όπου για να αναπαραστήσουμε το πρόβλημα μας το κάθε αντικείμενο έχει δύο ArrayList<State> ένα για την αρχική πλευρά κι ένα για τελική πλευρά της γέφυρα, μια λογική μεταβλητή **torch (true/false)** για αρχή και τέλος ανάλογα με το που βρίσκεται η λάμπα. Κρατάμε τον συνολικό χρόνο στη μεταβλητή **totalTime** και το **score** όπου αξιολογούμε την κάθε κατάσταση ανάλογα.

Έχουμε τρεις constructors, έναν default και ένα όπου έχουμε ως ορίσματα τα: αρχική και τελική πλευρά, συνολικό χρόνο και σκορ.

Έχουμε **setters/getters** για όλα τα ορίσματα μας.

Την **equals** που ελέγχει αν είναι ίδιο το τρέχον αντικείμενο με κάποιο άλλο.

Η **hashCode** φτιάχνει ένα μοναδικό ακέραιο που χρησιμοποιείται για να ξεχωρίζουν τα διαφορετικά αντικείμενα.

Η **compareTo** την οποία την χρησιμοποιούμε για να κάνουμε σύγκριση μεταξύ 2 αντικειμένων. Η toString η οποία μας επιστρέφει το String του αντικειμένου.

Οι μέθοδοι **equals**, **hashCode**, **compareTo** και **toString** είναι **@Override** για να μπορούν να χρησιμοποιηθούν πιο εύκολα στην υλοποίηση μας.

Η **getChildren** ελέγχει σε πια πλευρά είναι η λάμπα και μετακινούμαστε ανάλογα και επιστρέφει όλα τα παιδία της τρέχουσας κατάστασης σε ένα ArrayList.

H **moveStartToEnd** δέχεται το ArrayList στο οποίο προσθέτει τα παιδία της τρέχουσας κατάστασης, για να πάρουμε όλα τα παιδία της τρέχουσας κατάστασης χρησιμοποιούμε δυο εμφωλευμένα Loop, ώστε να πάρουμε τις πιθανές διατάξεις, μέσα σε κάθε επανάληψη φτιάχνουμε ένα αντίγραφο της κατάστασης σε ένα προσωρινό αντικείμενο τύπου State και θέτουμε ως πατέρα τον τρέχων, κάνομε τις απαραίτητες αλλαγές στο προσωρινό αντικείμενο για να μεταφέρουμε 2 άτομα στην τελική πλευρά, αξιολογούμε την κατάσταση με την heuristic(την οποία εξηγούμε παρακάτω) και το προσθέτουμε στο ArrayList με τα παιδία.

Η **moveEndToStart** κάνει την ίδια διαδικασία με την **moveStartToEnd** με τη διαφορά πώς το κάνουμε για ένα άτομο άρα μας αρκεί ένα Loop για το ArrayList της τελικής πλευράς.

Στις μεθόδους **moveStartToEnd & moveEndToStart** ελέγχουμε αν είναι άδεια η αντίστοιχη πλευρά, αν είναι δεν κάνουν τίποτα και επιστρέφουν στο πρόγραμμα.

Στην **heuristic** για να αξιολογήσουμε μία κατάσταση αφού κάνουμε τις αλλαγές στις move... δηλαδή αξιολογούμε το τι αλλαγή έγινε κι όχι τι θα γίνει. Εφόσον η λάμπα είναι στην αρχική κατάσταση (torch == true) τότε θα αξιολογήσουμε όλες τις καταστάσεις που οδηγούν στο να είναι η λάμπα true και θα διαλέξει την καλύτερη κατάσταση δηλαδή προσθέτουμε το ελάχιστο στοιχείο της αρχικής πλευράς. Αντίστοιχα αν η λάμπα είναι false αξιολογούμε όλες τις καταστάσεις και διαλέγουμε την καλύτερη, για να γίνει αυτό θα πρέπει να τηρούνται οι εξής συνθήκες: αν ο αρχικός μας πίνακας είναι κενός σημαίνει πως φτάσαμε σε τελική κατάσταση οπότε η heuristic τερματίζει. Αν έχει μείνει ένα στοιχείο στην αρχική κατάσταση όπου σε αυτή την περίπτωση προσθέτουμε στο score τον ελάχιστον χρόνο από την τελική πλευρά. Αλλιώς αν είναι παραπάνω από ένα άτομα ελέγχουμε αν τα 2 πιο γρήγορα άτομα είναι σε διαφορετικές πλευρές και προσθέτουμε στο score τον χρόνο του πιο αργού αλλιώς του πιο γρήγορο.

Η **isTerminal** ελέγχει αν έχουμε φτάσει σε τελική κατάσταση ή όχι για το τρέχων αντικείμενο.