ΑΝΑΦΟΡΑ 1Ης ΕΡΓΑΣΙΑΣ ΤΕΧΝΗΤΗΣ ΝΟΗΜΟΣΥΝΗΣ

Μέλη: Δημήτρης Λιαρόπουλος

Γεώργιος- Στέφανος Μεϊδάνης (3170107)

Παναγιώτης Κυρμπάτσος

Η εργασία μας έχει σαν θέμα την διάσχιση της γέφυρας, και επιλέξαμε να την υλοποιήσουμε με java. Περιέχει 4 αρχεία κλάσεων, BridgeCross, Solver, State, MinPQ

Το BridgeCross είναι το αρχείο που έχει την main. Αφού ολοκληρωθεί η αποθήκευση των παραμέτρων σε μεταβλητές και μια δυναμική λίστα ταξινομούμε την λίστα και αρχικοποιούμε τα δεδομένα(initial State και Solver). Τέλος η απάντηση στο πρόβλημα συκγρίνεται με τον δωθέντα χρόνο και αν δεν είναι εκτος ορίων παρουσιάζουμε την διαδρομή του αλγορίθμου που χρησιμοποιήσαμε από την λύση προς την ρίζα.

Κλάση State και αρχιτεκτονική

H Κλάση State χρησιμοποιείται για την αναπαράσταση καταστάσεων του προβλήματος και για την δημιουργία και αξιολόγηση των πιθανών παραγόμενων καταστάσεων. Κάθε στιγμιότυπό της περιέχει 2 δυναμικές λίστες στις οποίες αποθηκεύονται ο αριθμός των δευτερολέπτων που χρειάζεται κάθε ξεχωριστό πρόσωπο για να διασχίσει την γέφυρα, ανάλογα με την πλευρά στην οποία βρίσκεται. Συμπεριλαμβάνονται μεταβλητές για τον χρόνο που έχει περάσει (σε δευτερόλεπτα), την μεριά που βρίσκεται η λάμπα, για την αξία ή προτεραιότητα της συγκεκριμένης κατάστασης( το πόσο γρήγορα θα εξεταστεί) και για την ταυτότητα του ‘πατέρα’ κάθε στιγμιότυπου, δηλαδή την αμέσως προηγούμενη κατάσταση.

Η State είναι η βάση της αρχιτεκτονικής μας. Το πρώτο της στιγμιότυπο δημιουργείται με μια ταξινομημένη ArrayList(χρόνοι μελών) και μια άδεια arrayList. Με βάση της ευρετικής μας συνάρτησης κάθε κατάσταση αξιολογείται, και με βάση αυτής της αξιολόγησης συγκρίνονται για το ποια θα εξεταστεί πρώτη. Αναλυτικότερα, αναθέτουμε σε κάθε στιγμιότυπο το άθροισμα του περασμένου χρόνου, και του μεγαλύτερου χρόνου που θα κάνει κάποιο άτομο να περάσει την γέφυρα από την αφετηρία (μέθοδος getDist()). Η Τιμή ανατίθεται στην δημιουργία της κάθε πιθανής κατάστασης.

Η getChildren() είναι η μέθοδος που χρησιμοποιείται για την παραγωγή των επόμενων πιθανών καταστάσεων. Πρακτικά απαντάει στο ερώτημα ποιος ή ποιο ζευγάρι πρέπει να περάσει την γέφυρα κάθε φορά, επιστρέφοντας μια λίστα με τις νέες καταστάσεις. Όταν η λάμπα είναι στην δεξιά πλευρά, αρχικά γεμίζουμε μια λίστα με τα πιθανά ζευγάρια που μπορούν να περάσουν. Έπειτα για κάθε πιθανό ζευγάρι δημιουργείται ένα καινούριο στιγμιότυπο κατάστασης, και αν δεν υπάρχει κάποιο λάθος στο πέρασμα του ζευγαριού τότε, η κανούρια κατάσταση αξιολογείται από την ευρετική συνάρτηση getDist() και προστίθεται στην λίστα children. Εναλλακτικά, όταν η λάμπα είναι στα αριστερά, δημιουργείται ένα παιδί για κάθε μέλος της οικογένειας που έχει περάσει, με μια παρόμοια διαδικασία.

Οι 2 προηγούμενες βασικές μέθοδοι συνοδεύονται από getters, setters, print, isTerminal και compare μεθόδους οι οποίες βοηθούν στην λειτουργικότητα της κλάσης. Για τον έλεγχο των περασμάτων και την αποφυγή λαθών υλοποιούνται και οι fromAtoB() και fromBtoA().

Κλάση Solver- Μέθοδοι Τεχνητής Νοημοσύνης

Στην κλάση Solver περιέχονται μια λίστα από καταστάσεις και οι μέθοδοι για την αναζήτηση της τελικής κατάστασης. Δέχονται σαν παράμετρο την αρχική κατάσταση, και επιστρέφουν την πρώτη τερματική κατάσταση που βρίσκουν. Ειδικότερα, έχουμε χρησιμοποιήσει BFS με closed set και Α\*. Με τον BFS ελέγχουμε αν κάθε μια κατάσταση είναι τερματική και αν δεν είναι την βάζουμε στο closedSet, παράγουμε τα παιδιά της και τα προσθέτουμε στο τέλος της λίστας. Για τον A\* χρησιμοποιούμε μια ουρά προτεραιότητας(MinPQ) στην οποία οι καταστάσεις συγκρινονται με βαση το σκορ της ευρετικής μας. Η κατάσταση που έχει το μικρότερο σκορ αξιολογείται πρώτη, και τα παιδιά της αξιολογούνται και εισάγονται στην ουρά (εφόσον δεν έχουν καταχωρηθεί στο κλειστό σετ).