ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Τεχνητή Νοημοσύνη

1η Εργασία

Διάσχιση της γέφυρας

Ονοματεπώνυμο: Αναστασία Κανάτα 3180065

Αναστασία-Αντωνία Γρήγου 3180037

Διάσχιση της γέφυρας

Στο πρόβλημα διάσχισης της γέφυρας, με το οποίο αποφασίσαμε να ασχοληθούμε δίνεται η δυνατότητα στον χρήστη να ορίσει πόσα άτομα θα συμμετέχουν στο παιχνίδι, ποιος θα είναι ο χρόνος διάσχισης του καθενός και πόση θα είναι η διάρκεια ζωής της λάμπας.

Το πρόγραμμα που δημιουργήσαμε έχει την δυνατότητα να βρίσκει κάθε φορά την βέλτιστη λύση του προβλήματος κάνοντας αναζήτηση με τον αλγόριθμο A* με κλειστό σύνολο. Αυτός ο αλγόριθμος ευρετικής αναζήτησης χρησιμοποιεί μια ευρετική συνάρτηση. Ο τρόπος εύρεσης της ευρετικής έγινε ύστερα από αφαίρεση περιορισμών και πιο συγκεκριμένα δεν υπάρχει ο περιορισμός στο πλήθος των ατόμων που μπορούν να διασχίσουν την γέφυρα. (Κανονικά μπορούν να διασχίσουν μόνο 2 άτομα την γέφυρα κάθε φορά, ενώ με την ευρετική μπορούν να την διασχίσουν όλοι)

Το πρόγραμμα αποτελείται από 3 αρχεία Java τα οποία είναι:

- State.java
- SpaceSearcher.java
- Main.java

State.java

Ο κώδικα της κλάσης State αποτελείται από μεθόδους. Αρχικά, κάνουμε αρχικοποίηση των στιγμιότυπων State δημιουργώντας 2 πίνακες που αντιπροσωπεύουν την right (αρχή) και την left (τέλος) όχθη της γέφυρας και υπάρχουν getters, setters που θα χρησιμοποιηθούν παρακάτω στον κώδικα. Έτσι έχουμε getLeft και getRight που επιστρέφουν την αριστερή ή δεξιά πλευρά του στιγμιότυπου αντίστοιχα, getFather που επιστρέφει τον πατέρα του στιγμιότυπου και τα αντίστοιχα setters.

Στη συνέχεια, υλοποιούμε την συνάρτηση getChildren η οποία δημιουργεί τα παιδία (στιγμιότυπα) της κατάστασης που βρίσκεται κάθε φορά . Πιο συγκεκριμένα, μέσα στο arraylist children προστίθενται όλα τα πιθανά ζευγάρια που πρόκειται να περάσουν στην απέναντι πλευρά ανάλογα με το που είναι ο φακός (torch = true φακός δεξιά, torch = false φακός αριστερά).

Έπειτα, με τις συναρτήσεις StartToEnd και EndToStart γίνονται οι κινήσεις. Δηλαδή, με την StartToEnd οι 2 παίχτες που έχουν επιλεχτεί μετακινούνται από την αρχή στο τέλος και αντίστοιχα με την EndToStart ο ένας παίχτης που έχει επιλεχτεί επιστρέφει στην αρχή. Μόλις μετακινηθεί κάποιος παίχτης από την θέση του τότε σε αυτήν μπαίνει η τιμή 0.

Επίσης, η συνάρτηση isTerminal ελέγχει αν η κατάσταση που έχει δοθεί σαν όρισμα είναι η τελική και αυτό το καταφέρνει ελέγχοντας αν όλοι οι παίχτες έχουν φύγει από την αρχή, αν δηλαδή ο πίνακας right έχει σε όλες τις θέσεις του τιμή 0.

Για να γίνει βρεθεί η βέλτιστη λύση με την χρήση του αλγόριθμου A* θα πρέπει να υπολογίζεται και η ευρετική συνάρτηση, η οποία όπως αναφέρθηκε και παραπάνω δεν υπάρχει περιορισμός στο πλήθος των ατόμων που μπορούν να διασχίσουν την γέφυρα. Έτσι, ανάλογα με το που είναι ο φακός υπολογίζει το κόστος της ευρετικής (costh) ώστε να βρεθεί στην τελική κατάσταση. Για παράδειγμα, αν ο φακός βρίσκεται στα δεξιά το κόστος της ευρετικής είναι ο μεγαλύτερος χρόνος διάσχισης των παιχτών που βρίσκονται στα δεξιά. Αντίθετα, αν ο φακός βρίσκεται αριστερά επιλέγει από τα αριστερά τον παίχτη με το μικρότερο χρόνο ώστε να μεταφέρει τον φακό στα δεξιά και από εκεί επιλέγει τον παίχτη με το μεγαλύτερο χρόνο διάσχισης και αθροίζει τα δύο αυτά ποσά.

Τέλος, με την findCost υπολογίζεται το κόστος της κάθε κίνησης συγκρίνοντας την τρέχουσα κατάσταση (currentState) με μία από τα παιδιά της (newState). Για παράδειγμα, εάν δύο παίχτες μετακινηθούν από τα δεξιά στα αριστερά τότε ως κόστος (costf) ορίζεται ο μεγαλύτερος χρόνος από τους δύο αυτούς παίχτες, ενώ αν ένας παίχτης πρέπει να μετακινηθεί από τα αριστερά στα δεξιά τότε το κόστος είναι ο μικρότερος χρόνος διάσχισης των παιχτών που βρίσκονται στην αριστερή όχθη.

SpaceSearcher.java

Στην SpaceSearcher έχουμε την υλοποίηση του αλγόριθμου αναζήτησης A* με κλειστό σύνολο.

Αρχικά, στον πίνακα states προσθέτουμε την αρχική κατάσταση (initialState) και όσο αυτός δεν είναι άδειος συνεχίζεται η διαδικασία της αναζήτησης. Ορίζουμε ως currentState την θέση 0 του πίνακα states. Αν η currentState είναι η τελική κατάσταση τοτε η αναζήτηση τερματίζεται και επιστρέφει αυτή την κατάσταση. Αλλιώς, αν η currentState δεν βρίσκεται στο κλειστό σύνολο τότε εξετάζεται ανοίγοντας τα παιδιά της. Πιο συγκεκριμένα, την προσθέτουμε στο κλειστό σύνολο και καλούμε την getChildren για να παράξουμε τα παιδιά της, τα οποία τα προσθέτουμε στις ενεργές καταστάσεις(activeStates). Ως ενεργές καταστάσεις θεωρούμε όλες τις καταστάσεις που δεν βρίσκονται στο κλειστό σύνολο. Παράλληλα, υπάρχει και ένας πίνακας activeCosts στον οποίο βρίσκονται τα κόστη των ενεργών καταστάσεων. Κάθε φορά που παράγονται παιδία, για κάθε ένα από αυτά υπολογίζεται το συνολικό κόστος το οποίο είναι το άθροισμα του κόστους της ευρετικής και του κόστους διάσχισης. Αυτά υπολογίζονται καλώντας τις συναρτήσεις heuristic και findCost αντίστοιχα. Για να επιλέξει μια νέα κατάσταση ως currentState διαλέγει από τις ενεργές καταστάσεις αυτή με το μικρότερο συνολικό κόστος. Αν για currentState δεν επιλεχτεί κάποιο από τα παιδία τότε δεν αλλάζει η τιμή του φακού και από το τελικό κόστος αφαιρείται το κόστος της τρέχουσας κατάστασης.

Τέλος, με την συνάρτηση getTotalTime επιστρέφεται το συνολικό κόστος.

Main.java

Στην Main ζητείται από τον χρήστη να εισάγει πόσοι θα είναι οι παίχτες του παιχνιδιού, τον χρόνο διάσχισης του καθενός και τον χρόνο ζωής της λάμπας. Στην συνέχεια, δημιουργείται η initialState στην οποία ο πίνακας right είναι αυτός που περιέχει τους χρόνους των παιχτών και ο left είναι ένας κενός πίνακας. Έπειτα καλείται ο αλγόριθμος αναζήτησης A*, ξεκινά η διαδικασία αναζήτησης και το αποτέλεσμά της αποθηκεύεται στον terminalState. Τέλος, αν ο βέλτιστος χρόνος ολοκλήρωσης του παιχνιδιού είναι μεγαλύτερος από τον χρόνο ζωής της λάμπας τότε εμφανίζει στον χρήστη μήνυμα ότι δεν μπόρεσε να βρει λύση, αλλιώς του εμφανίζει τον βέλτιστο χρόνο που βρήκε ως λύση.

Επιτυχημένες προσπάθειες

```
C:\Users\agrig\OneDrive\aueb\5ο εξάμηνο\Τεχνητή Νοημοσύνη\Εργασίες\3180065_3180037>java Main
How many are the people?
4
Give time for player 1
1
Give time for player 2
2
Give time for player 3
5
Give time for player 4
8
Give Time
16
Found solution in 15 minutes
```

```
C:\Users\agrig\OneDrive\aueb\5ο εξάμηνο\Τεχνητή Νοημοσύνη\Εργασίες\3180065_3180037>java Main
How many are the people?
5
Give time for player 1
1
Give time for player 2
3
Give time for player 3
6
Give time for player 4
8
Give time for player 5
12
Give Time
30
Found solution in 29 minutes
```

Αποτυχημένη προσπάθεια

```
C:\Users\agrig\OneDrive\aueb\5ο εξάμηνο\Τεχνητή Νοημοσύνη\Εργασίες\3180065_3180037>java Main
How many are the people?
5
Give time for player 1
1
Give time for player 2
3
Give time for player 3
6
Give time for player 4
8
Give time for player 5
12
Give Time
25
Could not find solution
```