ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΑΚΑΔΗΜΑΙΚΟ ΕΤΟΣ 2012 – 2013 Χειμερινό εξαμηνο

Τεχνητή Νοημοσύνη

1η Ασκηση

7Ο ΕΞΑΜΗΝΟ

ΦΟΙΤΗΤΗΣ:

ΚΟΓΙΑΣ ΜΑΡΙΟΣ-ΕΥΑΓΓΕΛΟΣ

Το πρόβλημα

Το πρόβλημα που θέλουμε να επιλύσουμε στην συγκεκριμένη εργασία είναι ένα πρόβλημα αναζήτησης. Σε ένα επίπεδο χώρο με εμπόδια βρίσκονται 2 ρομπότ. Το ένα κινείται ελεύθερα ένα κουτάκι τη φορά, ενώ το δεύτερο, αυτό που κατευθύνουμε εμείς, κινείται 3 κουτάκια τη φορά και στόχο έχει να συναντήσει το πρώτο ρομπότ.

Για την αναζήτηση χρησιμοποιούμε τον αλγόριθμο A* με 2 διαφορετικές ευριστικές και θα συγκρίνουμε τελικά τα αποτελέσματα.

Τα ρομποτ κινούντα εναλλάξ μπροστά, πίσω, αριστέρα και δεξιά όπου μπορούν. Σε κάθε βήμα τρέχουμε τον αλγόριθμο A* και επιλέγουμε εκείνη την κατεύθυνση η ο οποία θα μας φέρει πιο κοντά στο ρομπότ στόχο και κινούμε το δικό μας ρομπότ κατά 3 κουτάκια.

Η Υλοποίηση

Για την λύση του συγκεκριμένου προβλήματος χρησιμοποιήθηκε η γλώσσα java. Δημιουργήθηκαν οι εξής κλάσεις:

-Main.java : η κεντρική κλάση του προγράμματος στην οποία εκτελείται ο αλγόριθμος A* και οδηγούνται τα ρομποτς.

-Robot.java : η κλάση των ρομποτ. Υλοποιούνται βασικές μέθοδοι κίνησης και διατηρούνται πληροφορίες σχετικά με τη θέση του ρομπότ.

-Node.java : η κλάση των κόμβων που χρησιμοποιούμε στην αναζήτηση με τον A*. Κάθε κόμβος αντιστοιχεί σε ένα κουτί του grid στο οποίο γίνεται η κίνηση.

-canNotReach.java : μια κλάση που επεκτείνει τις εξαιρέσεις.

-VisualRep.java : κλάση που υλοποιεί μεθόδους για το animation της κίνησης στο χώρο.

*Υλοποίηση του A**

Συγκεκριμένα για τον A* χρησιμοποιήθηκε η κλάση Node.java όπως αναφέρθηκε. Η κλάση αυτή περιέχει αρκετές από τις ιδιότητες του struct που δίνεται στην εκφώνηση της άσκησης, ενώ υλοποιεί και το interface Comparable για λόγους που θα δούμε παρακάτω. Περιέχει τις εξής ιδιότητες:

```
Node father; // δείκτης στο προηγούμενο κουτί
int cost_so_far; // κόστος για να φτάσουμε εδώ
int left_to_reach; // υπολοιπόμενο κόστος σύμφωνα με ευριστική
int overall_cost; // άθροισμα 2 προηγουμένων
int posX; // θέση στον X άξονα
int posY; // θέση στον Y άξονα

int depth; // βάθος στο δέντρο αναζήτησης
```

και τις εξής μεθόδους:

```
public Node(int x, int y, Node father) // ο constructor
```

```
public int compareTo(Node n) // από το interface Comparable
```

Σε κάθε βήμα λοιπόν κλωνοποιούμε το ρομπότ το οποίο κινούμε και δημιουργούμε ένα βοηθητικό ρομπότ το οποίο θα κάνει την αναζήτηση και θα εντοπίσει το ρομπότ στόχο ώστε τελικά κα κινήσουμε κατάλληλα το αρχικό ρομπότ.

Το ρομπότ αυτό σε κάθε κουτί στο οποίο βρίσκεται ελέγχει σε ποια γειτονικά κουτιά μπορεί να πάει και τα προσθέτει στο μέτωπο. Έπειτα επιλέγει αυτό με το μικρότερο overall_cost, πηγαίνει σε αυτό και επαναλαμβάνει τη διαδικασία.

Σε κάθε κουτί που μπορεί να φτάσει το ρομπότ αντιστοιχεί και ένα αντικείμενο της κλάσης Node. Τα αντικείμενα αυτά προστίθενται σε ένα priority queue ώστε ανά πάσα στιγμή να είναι ταξινομημένα σε αύξουσα σειρά overall_cost. Η priority queue είναι έτοιμη δομή της java που χρειάζεται όμως τα αντικείμενα στην ουρά να υλοποιούν το interface comparable. Γι'αυτό και υλοποιείται το συγκεκριμένο interface από την κλάση Node.

Η κλάση Main έχει μια ιδιότητα, το nodeTable το οποίο είναι ένας πίνακας με references σε Nodes. Κάθε φορά που βλέπουμε ότι το ρομπότ μπορεί να κινηθεί σε ένα κουτί εκτελείται η μέθοδος της Main

```
public boolean shouldMove(int x, int y,int cost)
```

Η μέθοδος αυτή ελέγχει αν έχει νόημα να μπει αυτός ο κόμβος στην ουρά. Αν δηλαδή είτε αυτή είναι η πρώτη που συναντάμε το συγκεκριμένο κόμβο είτε τον έχουμε συναντήσει στο παρελθόν αλλά με μεγαλύτερο κόστος προς το στόχο. Τότε έχει νόημα ο συγκεκριμένος κόμβος να μπει στη λίστα.

Μόλις εντοπιστεί ο κόμβος στόχος η εκτέλεση σταματάει και διατρέχουμε το δέντρο της αναζήτησης αντίστροφα μέχρι να φτάσουμε σε βάθος 3, δηλαδή 3 βήματα από την αρχική μας θέση. Αυτή θα είναι και η νέα θέση του ρομπότ μας.

Στη συνέχεια, το ρομπότ στόχος κινείται ελεύθερα κατά μια θέση και η διαδικασία επαναλαμβάνεται.

Οι ευριστικές

Ζητούμενο από την άσκηση είναι να υλοποιήσουμε και να συγκρίνουμε 2 διαφορετικές ευριστικές, μια υπερεκτιμήτρια και μια υποεκτμήτρια.

Σαν υποεκτιμήτρια επιλέχθηκε η απόσταση Manhattan. Δεδομένου ότι στο χώρο υπάρχουν εμπόδια στην καλύτερη περίπτωση τα ρομποτ απέχουν απόσταση ίση με την Manhattan distance, αφού με μεγάλη πιθανότητα η κίνηση θα πρέπει να είναι τέτοια ώστε να αποφευχθούν τα εμπόδια.

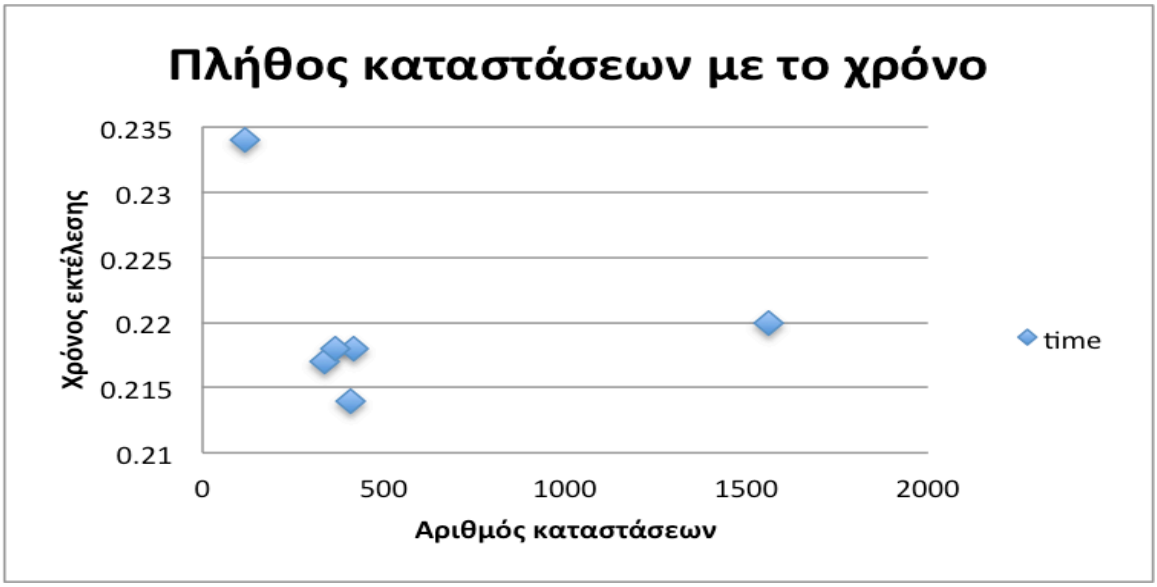
Σαν υπερεκτιμήτρια επιλέχθηκε το τετράγωνο της απόστασης Manhattan. Η συγκεκριμένη επιλογή έγινε ώστε να πληρούνται τα κριτήρια της υπερεκτιμήτριας θεωρώντας ότι η διάταξη των εμποδίων είναι τέτοια ώστε η πραγματική απόσταση να μην είναι μεγαλύτερη από το τετράγωνο της απόστασης manhatan.

Αποτελέσματα

Για να ελέγξουμε τον αλγόριθμο δημιουργήσαμε 6 διαφορετικά μεταξύ τους testcases μεταβλητής δυσκολίας, από έναν άδειο χώρο χωρίς καθόλου εμπόδια, μέχρι ένα testcase λαβύρινθο. Για τα 6 αυτά testcases τρέξαμε το πρόγραμμα τόσο χρησιμοποιώντας την υποεκτιμήτρια όσο και την υπερεκτιμήτρια. Τα testcases αυτά είχαν διαστάσεις 13x20. Τα αποτελέσματα είναι τα εξής:

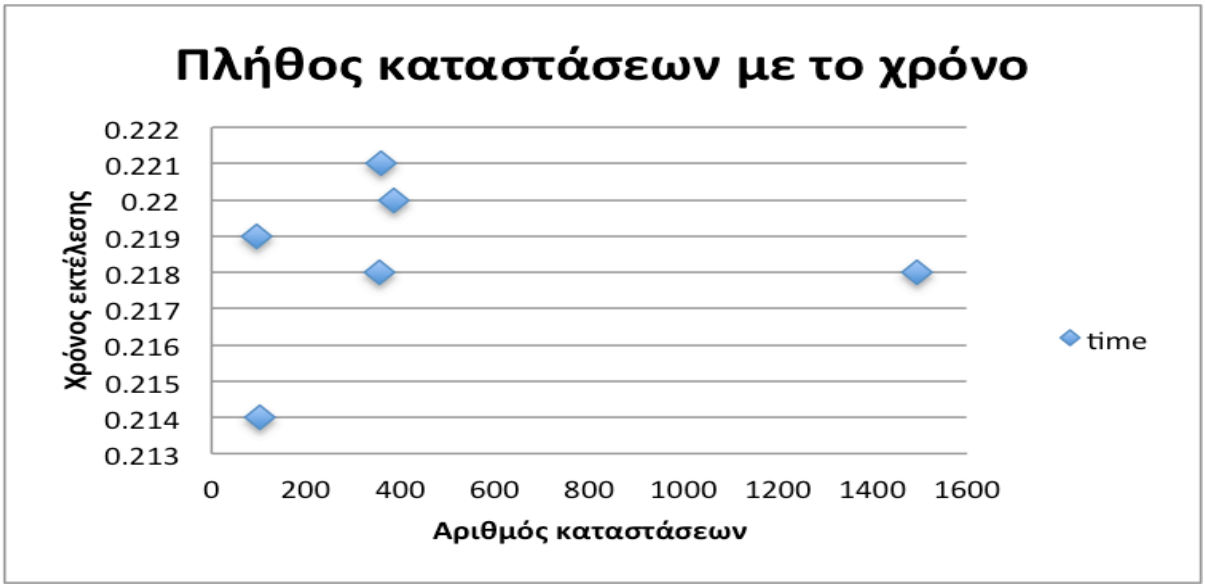
Για την υποεκτιμήτρια

testcase	time	states
1	0.217	337
2	0.218	416
3	0.214	407
4	0.22	1560
5	0.234	116
6	0.218	365



Αντίστοιχα για την υπερεκτιμήτρια έχουμε

testcase	time	states
1	0.214	102
2	0.218	355
3	0.22	387
4	0.218	1495
5	0.219	95
6	0.221	360



Παρατηρούμε λοιπόν ότι παρόλο που αλλάζουμε testcase ο χρόνος παραμένει σταθερός στην ίδια τάξη μεγέθους και στις δύο εκτιμήτριες. Αυτό οφείλεται στο γεγονός ότι τα testcases που χρησιμοποιήσαμε ήταν αρκετά μικρά.

Ωστόσο, ο αριθμός καταστάσεων, ή ο αριθμός των κόμβων των δέντρων αναζήτησης ποικιλει ανάλογα το testcase. Χαρακτηριστικό όμως είναι ότι η χρήση της υπερεκτιμήτριας ανοίγει λιγότερες καταστάσεις συγκρητικά με την υποεκτιμήτρια.

Με τη χρήση της υπερεκτιμήτριας (ύψωση στο τετράγωνο) δίνεται περισσότερο βάρος στο ευριστικό κριτήριο παρά στην απόσταση που έχει διανυθεί ως τη συγκεκριμένη χρονική στιγμή. Άρα, δεδομένου ότι το κριτήριο είναι αποδεκτό, οι εκτελέσεις με τη χρήση της υποεκτιμήτριας ανοίγουν λιγότερες καταστάσεις αφού τα βήματα που γίνονται είναι πιο «σίγουρα».

Ένα στιγμιότυπο από το animation της κίνησης των 2 ρομπότ είναι το παρακάτω όπου με μαύρο χρώμα είναι τα εμπόδια, με κόκκινο χρώμα το ρομπότ στόχος και με μπλε χρώμα το ρομπότ που κινούμε εμείς.