ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ

&

ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Τεχνητή Νοημοσύνη

Θέμα 1

7ο Εξάμηνο

Ροή Α

Αθανασίου Νικόλαος

03112074

Σταυρακάκης Δημήτριος

03112017

Ερώτηση 1

Στο πρόβλημα ασχοληθήκαμε με την εύρεση λύσης σε χώρο καταστάσεων. Συγκεκριμένα υλοποιήθηκε ο αλγόριθμος A* ώστε δύο ρομπότ να βρουν το δρόμο προς τον τελικο στόχο περνώντας από προκαθορισμένα σημεία του χάρτη και επιστρέφοντας στην αρχική τους θέση τελικά.

Τα ρομπότ έχουν πλήρη γνώση του περοβάλλοντος τους και καθώς αυτό δεν αλλάζει αλλά είναι στατικό η κίνηση του καθενός μπορεί να σκιαγραφηθεί εξ'αρχής.

Για την επίλυση του προβλήματος χρησιμοποιήθηκε ο αλγόριθμος A* ως μια παραλλαγή του Djiikstra με την προσθήκη της Manhattan για τον υπολογισμό των ευριστικών αποστάσεων. Η εκτίμηση αποστάσεων είναι χρήσιμη ώστε με λιγότερες αναζητήσεις να οδηγηθούμε σε βέλτιστη λύση δηλαδή ώστε να γίνει η ελάχιστη η διάσχιση του χώρου καταστάσεων.

Είναι φανερό ότι, αφού το ένα ρομπότ γνωρίζει για την κίνηση του άλλου,το πρόβλημα λύνεται τρέχοντας τον A* ανεξάρτητα μία φορά για κάθε ρομπότ δίνοντας ως παράμετρο στην συνάρτση του A* τις θέσεις από τις οποίες πρέπει να περάσει το καθένα πτιν φτάσουν στον τελικό στόχο. Έπειτα μπορούμε να αποφασίσουμε να βάλουμε το ένα από τα δύο να περιμένει (π.χ. το A ή το B ανάλογα με την περίσταση) δηλαδή για μία χρονική στιγμή να μείνει ακίνητο αν προκύψει να βρίσκονται μια συγκεκριμένη χρονική στιγμή στην ίδια θέση δηλαδή έχουμε μία σύγκρουση (collision).Συνεχίζοντας το ρομπότ που θα φτάσει δεύτερο στο στόχο μπορεί να μείνει σε κουτάκι που είναι γειτονικό του στόχου, αφού αυτό ορίζεται από την εκφώνηση αλλά σίγουρα αυτή η θέση θα είναι μέρος της διαδρομής προς τον τελικό στόχο.

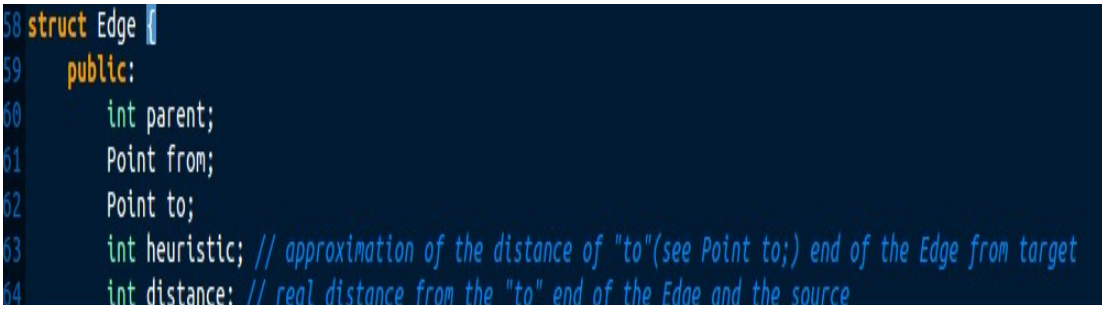
Για την υλοποίηση επιλέχθηκε η γλώσσα C++ λόγω του ότι προσφέρει χρήσιμες έτοιμες βιβλιοθήκες δομών δεδομένων που βοηθάνε στην επίλυση του προβλήματος. Συγκεκριμένα χρησιμοποιήθηκαν οι δομές set, vector, list για να είναι ευκολότερη η ταξινόμηση και διατήρηση καταστάσεων. Επιπλέον, το πρόγραμμα χωρίστηκε σε 3 αρχεία: robot.cpp astar.cpp visualizer.cpp και τα αντίσοιχα header files για το καθένα. Το αρχείο robot διαβάζει τα δεδομένα και υλοποιεί τις απαραίτητες κλήσεις συναρτήσεων και αρχικοποιήσεις για την εύρεση της λύσης στο πρόβλημα. Η aStar υλοποιεί την αναζήτηση σε χώρο καταστάσεων με χρήση του αλγορίθμου A*. Το αρχείο visualizer αναλαμβάνει την οπτικοποίηση του αποτελέσματος σε εικόνα μορφής PNG. Για την μεταγλώττιση και τη δημιουργία εκτελέσιμου επισυνάπτεται εύχρηστο Makefile.

Η οπτικοποίηση έγινε χρησιμοποιώντας τη C βιβλιοθήκη γραφικών cairo. Η απεικόνιση γίνεται στην εικόνα robots.png η οποία φαίνεται σε ένα από τα παρακάτω ζητούμενα για δεδομένο testcase.

Εκτός της οπτικής αναπαράστασης το πρόγραμμα επίσης εμφανίζει βοηθητικά μηνύματα σχετικά με το τι προσθέτουμε στην ουρά κάθε φορά ,με τον αν βρίσκουμε εμπόδιο και με τον προκύπτει σύγκρουση ανάμεσα στα 2 ρομπότ. (για ευκολία στην εμφανιση αποτελεσματων αυτα τα μηνύματα βρίσκονται σα σχόλια στο αρχείο astar.cpp)

Ερώτηση 2

Χρησιμοποιήσαμε την εξής δομή δεδομένων στην C++ για την επίλυση του προβλήματος:



Η κλάση αυτή αντιπροσωπεύει τη μετάβαση από μία θέση σε μία άλλη.

Ειδικότερα εμπεριέχει τη θέση προέλευσης(from) ,τη θέση προορισμού(to), την ευριστική εκτίμηση μέχρι το στόχο(heuristic), καθώς και την πραγματική απόσταση που έχουμε διανύσει από την πηγή(distance).

Ερώτηση 3

Η υλοποίηση του αλγορίθμου A* γίνεται μέσω των συναρτήσεων:

- **aStar**

Είναι η βασική συνάρτηση υλοποίησης του αλγορίθμου καθώς ξεκινά από αυτήν βάζοντας σε μιά ουρά τις συνδέσεις στο χάρτη τις οποίες μπορεί να χρησιμοποιήσει.

Στη συνέχεια ο αλγόριθμος χρησιμοποιεί την ουρά σαν ουρά προτεραιότητας ταξινομώντας τις υποψήφιες ακμές με βάση μια συνάρτηση σύμφωνα με την εκτιμώμενη απόσταση από τον στόχο. Η aStar αφορά μόνο μία πηγή και ένα στόχο, άρα καλείται μια φορά για καθε ρομπότ και για κάθε ενδιάμεση θέση που αυτό πρέπει να περάσει.

Η συνάρτηση επίσης περιέχει τη λογική πιθανών μεταβάσεων από κουτάκι σε κουτάκι του χάρτη.

- **enqueue**
  Η συνάρτηση enqueue χρησιμοποιείται βοηθητικά εντός της

συνάρτησης aStar για να προσθέσει μία νέα πιθανή επιλογή στην ουρά προτεραιότητας.

Όλες οι συναρτήσεις είναι κατασκευασμένες να μην χρησιμοποιούν global μεταβλητές και χωρίς παρενέργειες(side effects) ώστε να είναι επαναχρησιμοποιήσιμες.

Ερώτηση 4

Ως admissible heuristic χρησιμοποιήθηκε η συνάρτηση απόστασης Manhattan. Ο λόγος που είναι admissible είναι διότι ένα robot χρειάζετα τόσα βήματα όσα δείχνει η απόστασης Manhattan για να φτάσει στο στόχο στην καλύτερη περίπτωση, αν δεν συναντήσει εμπόδια. Διαφορετικά χρειάζεται περισσότερα βήματα. Ως non-admissible heuristic χρησιμοποιήθηκε η συνάρτηση $x^2 + y^2$. Αυτή η συνάρτηση είναι non-admissible heuristic. Ο λόγος είναι ότι υπερεκτιμά την απόσταση προς το στόχο. Για παράδειγμα, αν δεν υπάρχουν καθόλου εμπόδια ανάμεσα στη θέση που βρίσκεται το robot και στο στόχο, τότε ο υπερεκτιμητής θα δώσει ως αποτέλεσμα μεγαλύτερο απ’ ό,τι η πραγματική απόσταση που θα διένυε η βέλτιστη λύση. Γι’ αυτό το λόγο, ενδέχεται ο υπερεκτιμητής να οδηγηθεί σε υποβέλτιστα αποτελέσματα, αν και σε λιγότερο πλήθος βημάτων.Για την non admissible υπάρχει η συνάρτηση της στο astar.h και βρίσκεται σε σχόλια στο κύριο πρόγραμμα.Αυτή που εκτελείται ως έχει ο κώδικας είναι η admissible.

Ερώτηση 5

Το κομμάτι του κώδικα που χρησιμοποιείται για την αποφυγή συγκρούσεων είναι το εξής (πχ για το Robot B, αντίστοιχα για το Robot A):

```
BPath = aStar(B,prev,obstacle,mapSize);

   prev=BPath.begin()->from;

   for ( list< Edge >::iterator it = BPath.begin(); it != BPath.end(); ++it ) {

         if (timeA > timeB){

                  helper=temp.front();

                  temp.pop_front();

                  if (helper.x==it->from.x && helper.y==it->from.y &&
helper.timer==timeB){ //collision,stay where you were mr B

                           printf("Collision Happens, so MR. B waits here for a moment\n");

                           printf( "(%i, %i) at moment:%i\n", prev.x + 1, prev.y + 1,timeB );

                           --it;//to remain at the same element

                  }

                  else{
```

```
                printf( "(%i, %i) at moment:%i\n", it->from.x + 1, it->from.y + 1,timeB );
                prev=it->from;
                }
        }
        else{
                printf( "(%i, %i) at moment:%i\n", it->from.x + 1, it->from.y + 1,timeB );
                prev=it->from;
        }
        timeB++;
        if (it->from == B)
                break;
  }
}
```

Δηλαδή αν επιλέξουμε ως αρχική την διαδρομή του A και ο B πάει να συμπέσει για κάποια χρονική στιγμή με τον A, του λέμε να περιμένει 1 χρονική στιγμή ώστε να φύγει ο A απο κει και μετά να συνεχίσει κανονικά τη διαδρομή του ο B.

Αλγοριθμική περιγραφή

1.Βρίσκουμε το PathA

2.Αποθηκεύουμε όλα τα στοιχεία του σε μία λίστα

3.Βρίσκουμε το PathB και αποθηκέυουμε και αυτού τα στοιχεία στη λίστα.

4.Τσεκάρουμε ποιο θέλει πιο πολύ χρόνο να φτάσει στο target με βέλτιστη διαδρομή και το παίρνουμε.

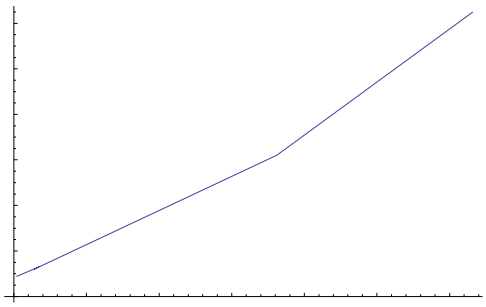
5.Παίρνουμε το άλλο path και βλέπουμε αν υπάρχει στιγμή που να συμπίπτουν. Αν ναι λεμε στο robot του 2ου αυτού path να περιμένει μια στιγμή και μετά να συνεχίζει.

6. Έτσι ως το τελος.

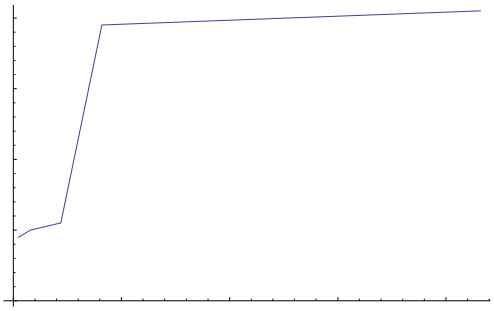
Ερώτηση 6

Παρατίθενται τα στατιστικά αποτελέσματα της εκτέλεσης.Τα testcases στα οποία βασιστήκαμε έχουν επισυναφθεί στο zip αρχείο της άσκησης.Και τα δύο διαγράμματα είναι “Χρόνος εκτέλεσης συναρτήσει του αριθμού των κόμβων”.

Admissible heuristic



Non-admissible heuristic



Και στις δύο περιπτώσεις ο χρόνος εκτέλεσης είναι αμελητέος για τις μικρές δοκιμαστικές περιπτώσεις και πολύ μικρός για τις μεγαλύτερες. Για την μέτρηση χρησιμοποιήθηκε η κλήση time του UNIX.

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».

Ερώτηση 7

Παρακάτω φαίνεται η έξοδος του προγράμματος για το testcase της εκφώνησης:

```
dimstav23@Spam ~/Desktop/Artificial Intelligence/correct/ex1 $ ./robot <input.txt
A* algorithm completed in 37 steps.
A* algorithm completed in 25 steps.
A* algorithm completed in 58 steps.
A* algorithm completed in 9 steps.
A* algorithm completed in 23 steps.
A* algorithm completed in 25 steps.
A* algorithm completed in 58 steps.
A* algorithm completed in 9 steps.
Robot A path:
(13, 8) at moment:1
(13, 7) at moment:2
(13, 6) at moment:3
(13, 5) at moment:4
(13, 4) at moment:5
(12, 4) at moment:6
(11, 4) at moment:7
(10, 4) at moment:8
(9, 4) at moment:9
(8, 4) at moment:10
(7, 4) at moment:11
(6, 4) at moment:12
(5, 4) at moment:13
(5, 5) at moment:14
(5, 6) at moment:15
(5, 7) at moment:16
(5, 8) at moment:17
(5, 9) at moment:18
(5, 10) at moment:19
(5, 11) at moment:20
(5, 12) at moment:21
(5, 13) at moment:22
(6, 13) at moment:23
(7, 13) at moment:24
(7, 12) at moment:25
(7, 11) at moment:26
(7, 10) at moment:27
```

```
(8, 10) at moment:28
(9, 10) at moment:29
(10, 10) at moment:30
(10, 9) at moment:31
(11, 9) at moment:32
(11, 8) at moment:33
(12, 8) at moment:34
(13, 8) at moment:35
(14, 8) at moment:36
(15, 8) at moment:37
(16, 8) at moment:38
(17, 8) at moment:39
(17, 9) at moment:40
(16, 9) at moment:41
(15, 9) at moment:42
(14, 9) at moment:43
(13, 9) at moment:44
A* algorithm completed in 23 steps.
Robot B path:
(2, 8) at moment:1
(2, 7) at moment:2
(2, 6) at moment:3
(2, 5) at moment:4
(2, 4) at moment:5
(3, 4) at moment:6
(4, 4) at moment:7
(5, 4) at moment:8
A* algorithm completed in 25 steps.
(5, 5) at moment:9
(5, 6) at moment:10
(5, 7) at moment:11
(5, 8) at moment:12
(5, 9) at moment:13
(5, 10) at moment:14
(5, 11) at moment:15
(5, 12) at moment:16
(5, 13) at moment:17
(6, 13) at moment:18
(7, 13) at moment:19
A* algorithm completed in 58 steps.
```

```
(7, 12) at moment:20
(7, 11) at moment:21
(7, 10) at moment:22
(8, 10) at moment:23
(9, 10) at moment:24
(10, 10) at moment:25
(10, 9) at moment:26
(11, 9) at moment:27
(11, 8) at moment:28
(12, 8) at moment:29
(13, 8) at moment:30
(14, 8) at moment:31
(15, 8) at moment:32
(16, 8) at moment:33
(16, 8) at moment:34
(16, 8) at moment:35
(16, 8) at moment:36
(16, 8) at moment:37
(16, 8) at moment:38
(16, 8) at moment:39
A* algorithm completed in 47 steps.
(16, 9) at moment:40
(16, 10) at moment:41
(15, 10) at moment:42
(14, 10) at moment:43
(13, 10) at moment:44
(12, 10) at moment:45
(11, 10) at moment:46
(10, 10) at moment:47
(9, 10) at moment:48
(8, 10) at moment:49
(8, 9) at moment:50
(7, 9) at moment:51
(6, 9) at moment:52
(5, 9) at moment:53
(4, 9) at moment:54
(3, 9) at moment:55
(2, 9) at moment:56
```

Και σε περίπτωση collision:

```
dimstav23@Spam ~/Desktop/Artificial Intelligence/correct/ex1 $ g++ -c robot.cpp -o robot.o
dimstav23@Spam ~/Desktop/Artificial Intelligence/correct/ex1 $ g++ -o robot robot.o astar.o
dimstav23@Spam ~/Desktop/Artificial Intelligence/correct/ex1 $ ./robot <test1.txt
A* algorithm completed in 17 steps.
A* algorithm completed in 8 steps.
A* algorithm completed in 22 steps.
A* algorithm completed in 11 steps.
A* algorithm completed in 8 steps.
A* algorithm completed in 14 steps.
Robot B path:
(4, 1) at moment:1
(3, 1) at moment:2
(3, 2) at moment:3
(3, 3) at moment:4
(3, 4) at moment:5
(3, 5) at moment:6
(3, 6) at moment:7
(3, 5) at moment:8
(4, 5) at moment:9
(5, 5) at moment:10
(6, 5) at moment:11
(7, 5) at moment:12
(6, 5) at moment:13
(5, 5) at moment:14
(4, 5) at moment:15
(3, 5) at moment:16
(3, 4) at moment:17
(3, 3) at moment:18
(3, 2) at moment:19
(3, 1) at moment:20
(4, 1) at moment:21
(5, 1) at moment:22
A* algorithm completed in 17 steps.
Robot A path:
(1, 2) at moment:1
(1, 3) at moment:2
(1, 4) at moment:3
(1, 5) at moment:4
(1, 6) at moment:5
(2, 6) at moment:6
```

```
Collision Happens, so MR. A waits here for a moment
(2, 6) at moment:7
(3, 6) at moment:8
A* algorithm completed in 8 steps.
(3, 5) at moment:9
(4, 5) at moment:10
(5, 5) at moment:11
(6, 5) at moment:12
A* algorithm completed in 20 steps.
(5, 5) at moment:13
(4, 5) at moment:14
(3, 5) at moment:15
(3, 4) at moment:16
(3, 3) at moment:17
(3, 2) at moment:18
(3, 1) at moment:19
(2, 1) at moment:20
(1, 1) at moment:21

dimstav23@Spam ~/Desktop/Artificial Intelligence/correct/ex1 $
```

Ερώτηση 8

Παρακάτω φαίνονται τα βήματα το κάθε robot για ένα από τα input files, όπως παράγονται από το πρόγραμμα σε μορφή εικόνας: