Τεχνητή Νοημοσύνη

Αναφορά Πρώτης Προγραμματιστικής Εργασίας

Ράπτης Παναγιώτης (ΑΜ : 2014030146)
Πεσλής Κωνσταντίνος (ΑΜ : 2014030074)

Σκοπός της Άσκησης

Σκοπός της συγκεκριμένης προγραμματιστικής άσκησης, είναι τόσο στο τεχνικό κομμάτι, η εξοικείωση με την γλώσσα προγραμματισμού **Python** και τα διάφορα εργαλεία τα οποία έχουμε στην διάθεση μας καθώς προγραμματίζουμε σε αυτήν, όσο και στην θεωρητική σκοπιά, η εξοικείωση με την υλοποίηση αλγορίθμων αναζήτησης και η εκπαίδευση στην διαδικασία δημιουργίας καλών ευρετικών συναρτήσεων για την επίλυση προβλημάτων.

Σημεία Ενδιαφέροντος

Τα κύρια σημεία που χρειάζεται κανείς να προσέξει σε αυτήν την εργασία, είναι οι υλοποιήσεις των αλγορίθμων αναζήτησης και η κύρια ευρετική συνάρτηση η οποία έχει φτιαχτεί αποκλειστικά για αυτό το πρόβλημα και προσπαθεί να μας οδηγήσει σε έξυπνες κινήσεις, οι οποίες ταυτόχρονα αποφεύγουν τα εμπόδια, αλλά διατηρούν και μια πορεία προς τον τελικό στόχο.

Η κύρια δυσκολία της άσκησης, βρισκόταν στην θεωρητική ανάλυση του προβλήματος και την δημιουργία μιας τέτοιας ευρετικής, η οποία εκτός από κινήσεις που θεωρούνται καλές σε ένα πρόβλημα αναζήτησης βέλτιστου μονοπατιού από έναν κόμβο σε έναν άλλο, να μας οδηγεί σε κινήσεις "έξυπνες", σε σχέση με τα εμπόδια που ίσως βρούμε μπροστά μας, χρησιμοποιώντας όσο καλύτερα γίνεται τις πληροφορίες που έχουμε στην διάθεση μας.

Κύρια προβλήματα

Δεν υπάρχει κάποιο κομμάτι της άσκησης που να μην πραγματοποιήθηκε, ή να μην είναι εκτελέσιμο. Ο κώδικας έχει παραμετροποιηθεί ως έναν βαθμό, ο τρόπος με τον οποίο έχει γραφεί όμως, επιβάλλει τον χειροκίνητο ορισμό μέσα απ΄την main.py του σεναρίου με το οποίο θα τρέξει, και μέσα απ΄τις αντίστοιχες Astar.py, IDAstar.py, την επιλογή της ευρετικής που θα χρησιμοποιηθεί.

Εκτέλεση του Παραδοτέου

Το παραδοτέο πρόγραμμα δέχεται 4 εισόδους, έτσι ώστε να μην αναλωνόμαστε σε χρονοβόρες περιπτώσεις όταν αυτό δεν είναι απαραίτητο και να μπορούμε να δούμε συγκεκριμένα αποτελέσματα γρηγορότερα. Ο τρόπος με τον οποίο μπορούμε να τρέξουμε το αρχείο είναι είτε δίνοντας τα ορίσματα στο Run Configurations ενός IDE, όπως το PyCharm ή σε κάποιο terminal απ΄τον φάκελο του παραδοτέου, μέ την εντολή : python main.py <Scenario Number> <Heuristic Number> <Minimum Weight> <Maximum Weight>.
Οι επιλογές των ευρετικών είναι :

- 1 για την ευκλείδια απόσταση
- 2 για την απόσταση Manhattan
- 3 για την δική μας ευρετική

Για παράδειγμα, για να τρέξουμε το αρχείο για το σενάριο 3, την δική μας ευρετική και για βάρη απο 0 έως 6, η αντίστοιχη εντολή είναι η : python main.py 3 3 0 6

Ανάλυση Αλγορίθμων

Εισαγωγή

Στο πρώτο μέρος της εργασίας, υλοποιήσαμε 2 αλγορίθμους αναζήτησης, τον Weighted A* και τον IDA*. Η υλοποίηση τους βασίστηκε στις συναρτήσεις που δόθηκαν, αλλά και στην δημιουργία νέων υπεύθυνων για την ομαλή επικοινωνία των Nodes ή Primitives, και την γενικότερη λειτουργικότητα του κάθε αλγορίθμου. Αρχικά, αξίζει να αναφερθεί πως και οι 2 αλγόριθμοι εκτελούνται με αναδρομικές κλήσεις ώστε να αποφύγουμε κοστοβόρες(είτε σε χρόνο είτε σε χώρο) επαναλήψεις και έχουν υποβληθεί σε tests βασισμένα τόσο σε παραδείγματα που αναλύθηκαν σε διαλέξεις, όσο και στο ίδιο το πρόβλημα το οποίο καλούμαστε να λύσουμε. Περαιτέρω ανάλυση των αλγορίθμων και του τρόπου υλοποίησης του θα γίνει και στην συνέχεια.

Ο αλγόριθμος **WEIGHTED A***

Για την υλοποίηση αυτού του αλγορίθμου, χρειάστηκε να κατασκευάσουμε την συνάρτηση αξιολόγησής του, η οποία γνωρίζουμε απ΄την θεωρία πως είναι της μορφής :

$$f(node) = g(node) + w * h(node)$$

όπου $g(node)$ είναι το πραγματικό κόστος του εκάστοτε κόμβου, δηλαδή η απόστασή του απ΄τον αρχικό, το w είναι το βάρος που χρησιμοποιούμε το οποίο ορίζεται παραμετρικά, ενώ το $h(node)$ είναι η ευρετική συνάρτηση, η οποία μας δίνει το εκτιμώμενο κόστος απ΄τον κόμβο που βρισκόμαστε μέχρι τον τελικό.

Αρχικά, δημιουργήσαμε μια λίστα στην οποία διατηρούμε τους πιθανούς κόμβους προς επίσκεψη, δηλαδή το αντίστοιχο fringe.Η ιδιαιτερότητα της υλοποίησής μας, βασίζεται στην ταξινόμηση της λίστας του fringe, με βάση τις τιμές της $f(n)$ για κάθε κόμβο, ώστε να

επιλέγουμε με ευκολία τον κόμβο με την μικρότερη συνάρτηση αξιολόγησης κάθε φορά.
Η διαδικασία αυτή επαναλαμβάνεται αναδρομικά, εώς ότου βρεθεί ο τελικός κόμβος, όπου και επιστρέφουμε τις κατάλληλες τιμές μαζί με το βέλτιστο μονοπάτι προς τον στόχο.

Ο αλγόριθμος **IDA***

Η υλοποίηση του IDA* αλγορίθμου, είναι αντίστοιχη αυτής του weighted A*, με μόνη διαφορά την χρήση ενός ορίου αναζήτησης.

Πιο συγκεκριμένα, αρχικά θέτουμε το όριο αυτό ίσο με την τιμή της συνάρτησης f του αρχικού κόμβου και ελέγχουμε τους κόμβους που υπάρχουν στο fringe, των οποίων η τιμή της συνάρτησης f είναι μικρότερη του ορίου αυτού. Αν δεν υπάρχει κατάλληλος κόμβος προς επίσκεψη που να πληρεί την παραπάνω προϋπόθεση, ανανεώνουμε το limit με την μικρότερη τιμή f, απ΄τους διαθέσιμους κόμβους που έχουμε στο fringe.
Τέλος, επαναλαμβάνουμε αυτήν την διαδικασία αναδρομικά, μέχρι να βρεθεί ο τελικός κόμβος, όπως και στον αλγόριθμο weighted A*, με την διαφορά ότι κάθε φορά που ανανεώνεται το όριο αναζήτησης, η αναζήτηση ξεκινάει απ΄τον αρχικό κόμβο.

Ευρετικές Συναρτήσεις και Πειραματισμοί

Ευρετικές Συναρτήσεις που υλοποιήθηκαν

Το παραδοτέο αρχείο περιέχει 3 διαθέσιμες ευρετικές συναρτήσεις, μια βασισμένη στην ευκλείδια απόσταση, μία στην απόσταση Manhattan και μία δικής μας κατασκευής.

Οι δύο πρώτες υλοποιήθηκαν αρκετά απλά, χρησιμοποιώντας τις συντεταγμένες του κάθε κόμβου και τους τύπους των αντίστοιχων αποστάσεων, γρήγορα όμως διαπιστώσαμε πως δεν είναι ιδανικές για την αντιμετώπιση εμποδίων που μπορεί να βρεθούν μεταξύ του κόμβου που βρισκόμαστε και του τελικού στόχου.
Μετά απο διάφορους πειραματισμούς, παρατηρήσαμε πως ενώ η απόσταση Manhattan βελτιώνει την επίδοση σε σχέση με την Ευκλείδια σε σενάρια που περιέχουν δυσκολότερα εμπόδια προς αποφυγή, δεν πετυχαίνει επιδόσεις οι οποίες κρίνονται βέλτιστες, τόσο χρόνικα, όσο και σε σχέση με τον συνολικό αριθμό κόμβων που επισκέπτεται προτού βρεί τον στόχο.
Έτσι, κρίθηκε απαραίτητη η δημιουργία μιας τρίτης ευρετικής συνάρτησης, η οποία να μπορεί να ανταποκριθεί σε υψηλότερες προσδοκίες.

Δημιουργία νέας Ευρετικής Συνάρτησης

Για την δημιουργία μιας νέας ευρετικής συνάρτησης, η οποία να είναι ειδικά σχεδιασμένη για το συγκεκριμένο πρόβλημα που μας δώθηκε, χρειάστηκε να σκεφτούμε εκτός της λογικής του προγραμματιστικού μέρους και περισσότερο με βάση πιο "έξυπνες", βασισμένες στην γεωμετρία και τις αποστάσεις, λύσεις.

Η αρχική σκέψη περιλαμβάνει τον εντοπισμό του πλησιέστερου εμποδίου και την εκμετάλλευση του γεγονότος ότι γνωρίζουμε το κέντρο και τις διαστάσεις του και συνεπώς τις συντεταγμένες των άκρων του. Έτσι, καταλήξαμε στον διαχωρισμό δύο βασικών περιπτώσεων :

- Την περίπτωση που βρισκόμαστε σε υψηλότερο σημείο απ΄τον στόχο, το οποίο παρουσιάζεται εδώ :

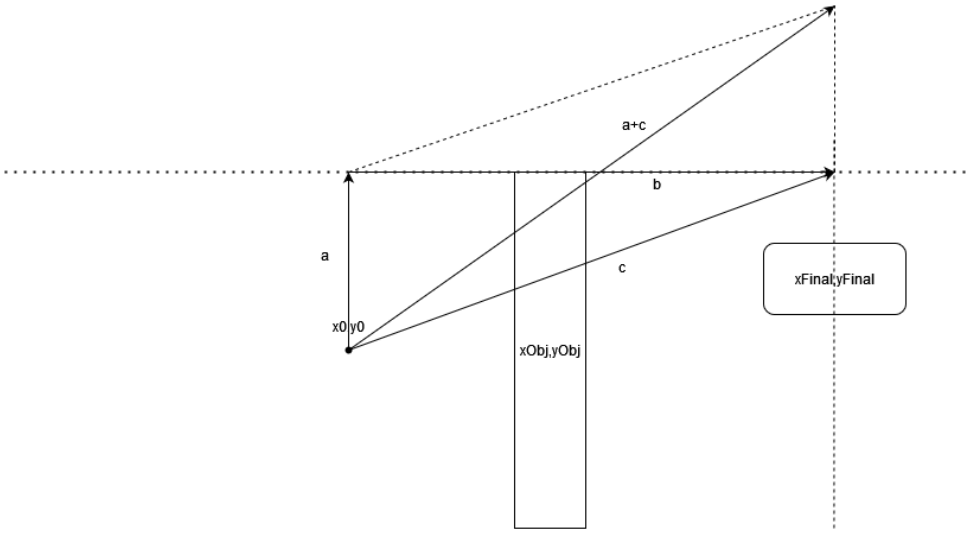

Figure 1: Current Node is Below Goal

- Την περίπτωση που βρισκόμαστε σε χαμηλότερο σημείο απ΄τον στόχο, το οποίο παρουσιάζεται εδώ :

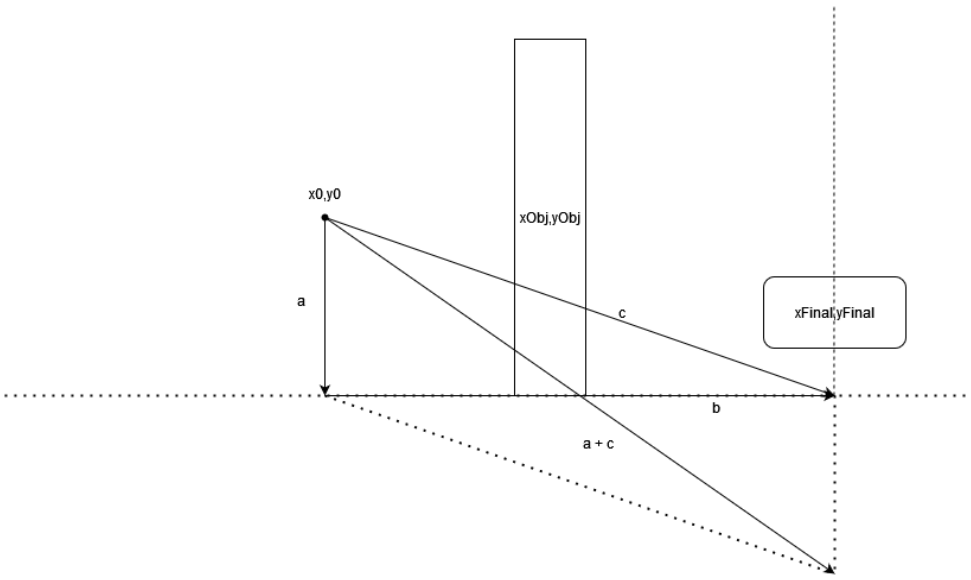

Figure 2: Current Node is Above Goal

Στα παραπάνω σχήματα, και στις 2 περιπτώσεις, ορίζουμε ένα ορθογώνιο τρίγωνο, με πλευρές τις a, b, c. Αυτές υπολογίζονται ως εξής:

- Το a είναι η απόσταση στον άξονα y του κόμβου στον οποίο βρισκόμαστε, από το κατάλληλο άκρο του εμποδίου, το οποίο και διαλέγουμε με βάση τον τελικό κόμβο.
- Το b είναι η απόσταση στον άξονα x του κόμβου στον οποίο βρισκόμαστε με τον τελικό κόμβο κατά μήκος του κατάλληλου άκρου του εμποδίου που μας ενδιαφέρει να ξεπεράσουμε.
- Το c είναι η υποτείνουσα του τριγώνου, ενώνοντας τις 2 άκρες που έχουν δημιουργηθεί ως τώρα.

Αξίζει να σημειωθεί επίσης, πως σε κάθε περίπτωση απ΄τις παραπάνω, ελέγχουμε αν είναι σκόπιμο να ακολουθήσουμε όντως το μονοπάτι που μας δίνει το αντίστοιχο τρίγωνο, ή θα πρέπει να αλλάξουμε την γωνία του εμποδίου προς την οποία θα κατευθυνθούμε, είτε επειδή δεν μπορούμε να ξεπεράσουμε το εμπόδιο από αυτήν την πλευρά λόγω περιορισμών του grid, είτε γιατί συμφέρει περισσότερο με βάση την θέση του στόχου.

Όπως παρατηρούμε και στα Figures 1,2, αξίζει να αναφερθεί πως το άθροισμα των διανυσμάτων $a + c$, μας δίνει το διάνυσμα που εμφανίζουμε, το οποίο μας δίνει μια καλή πρώτη άποψη για τον τρόπο με τον οποίο μπορούμε να αποφύγουμε το εμπόδιο, αφού μας οδηγεί στο σημείο που θέλουμε, πετυχαίνοντας τόσο αποφυγή του εμποδίου, όσο και την κίνηση προς τον στόχο.
Τέλος, μετά απο πειραματισμό με διάφορα μεγέθη και μια διαδικασία trial & error, καταλήξαμε στον τελικό τύπο της ευρετικής μας συνάρτησης, ο οποίος είναι διαφορετικός για τον weighted A* και τον IDA* και είναι:

- Για τον weighted A* έχουμε :

$$h(n) = \sqrt{a^2 + (c^2 + a^2)}$$

- Για τον IDA* έχουμε :

$$h(n) = 2\sqrt{a^2 + (c^2 + a^2)}$$

Ο διπλασιασμός στην ευρετική του IDA*, υλοποιήθηκε διότι με την απουσία του παράγοντα του βάρους στην ευρετική συνάρτηση, χρειαζόμασταν έναν τρόπο να "κατευθύνουμε" τον αλγόριθμο μας, ώστε να προσπελάσει πιο έξυπνα τους κόμβους δίνοντας μεγαλύτερη βάση στην ευρετική συνάρτηση απ΄ότι στην συνάρτηση κόστους.

Συμπεράσματα, Παρουσίαση και Συγκρίσεις Αποτελεσμάτων

Παρουσίαση αποτελεσμάτων

Τα αποτελέσματα μπορούν να βρεθούν στα αντίστοιχα **text files**, με τις κατάλληλες ονομασίες. Τα αρχεία αυτά έχουν δημιουργηθεί ήδη για μεγαλύτερη ευκολία και τα ονόματα τους έχουν την μορφή **output<Scenario Number>_<Heuristic Number>.txt**, οπότε στο αρχείο **output2_3.txt** για παράδειγμα, περιμένουμε να δούμε τα αποτελέσματα της δικής μας ευρετικής για το σενάριο 2. Κάθε νέο τρέξιμο του αρχείου διατηρεί τα ήδη παραγμένα **text files** και δημιουργεί(ή γράφει πάνω στο ήδη υπάρχον) **output.txt**.Για ευκολία στην αναφορά θα παρουσιάσουμε τα στοιχεία της ευκλείδιας απόστασης και της δικής μας ευρετικής συνάρτησης, όπως αναφέρεται στην εκφώνηση. Για παραπάνω πληροφορίες για την απόσταση **Manhattan**, ανατρέξτε στον σχολιασμό παρακάτω, ή στα αντίστοιχα **text** αρχεία. Παρουσιάζουμε λοιπόν τους αντίστοιχους πίνακες :
Για την Ευρετική Συνάρτηση που χρησιμοποιεί την Ευκλίδεια Απόσταση για τον αλγόριθμο **weighted A*** :

Scenario	Weight	Visited Nodes
1	0	278
1	1	50
1	2	23
2	0	1224
2	1	308
2	2	147
3	0	607
3	1	409
3	2	425

Table 1: Weighted A* with Euclidean Heuristic

Ενώ για τον IDA* έχουμε

Scenario	Visited Nodes
1	574
2	18134
3	NaN

Table 2: IDA* with Euclidean Heuristic

Με την ευκλείδια απόσταση ως ευρετική συνάρτηση, παρατηρούμε πως σε περιπτώσεις στις οποίες δεν έχουμε έυκολη πρόσβαση σε στόχο μέσω μιας ευθείας γραμμής, αλλά χρειάζεται να αποφύγουμε μια σειρά εμποδίων, τόσο ο Weighted A*, αλλά ακόμα περισσότερο ο IDA* δυσκολεύονται σε τεράστιο βαθμό να βρουν μονοπάτι προς τον στόχο, με αποκορύφωμα το τρίτο σενάριο, στο οποίο ο IDA* δεν μπορεί να τρέξει, αφού υπερβαίνει κατά πολύ τόσο το όριο αναδρομών που επιτρέπει η Python by default, όσο και το νέο όριο των 10.000 που θέτουμε εμείς. Συνεχίζοντας, ας παρατηρήσουμε την δική μας ευρετική συνάρτηση κάτω απ΄τις ίδιες συνθήκες :
Για τον αλγόριθμο weighted A* :

Scenario	Weight	Visited Nodes
1	0	278
1	1	77
1	2	24
2	0	1224
2	1	208
2	2	30
3	0	607
3	1	402
3	2	44

Table 3: Weighted A* with Custom Heuristic

Ενώ για τον IDA* έχουμε

Scenario	Visited Nodes
1	24
2	30
3	44

Table 4: IDA* with Custom Heuristic

Όπως διακρίνουμε απ΄τους παραπάνω πίνακες, η δική μας ευρετική συνάρτηση αν εξαιρέσουμε το βάρος 0 του Weighted A*, στο οποίο και ο αλγόριθμος εκφυλλίσσεται σε μία μορφή του αλγορίθμου Djikstra, παράγει πολύ καλύτερα αποτελέσματα, αφού καταφέρνει να βρει το μονοπάτι προς τον κόμβο στόχο, περιορίζοντας κατά πολύ τον αριθμό των κόμβων που επισκέπτεται.
Για παράδειγμα, στο σενάριο 3 με την χρήση του IDA*, ενώ η ευκλείδια απόσταση μας δίνει μια ευρετική που αναγκάζει το πρόγραμμα μας να τρέξει σε σημείο που ξεπερνάει τα όρια αναδρομών που υποστηρίζουμε, η ευρετική μας συνάρτηση καταφέρνει να επισκεφτεί μόνο 44 κόμβους, χάρη -βέβαια- στην υπερεκτίμηση του κόστους και την υπερκάλυψη της συνάρτησης κόστους $g(n)$, όπως αναφέραμε παραπάνω και μπορούμε να δούμε απ΄τα παραγόμενα αρχεία.