Τεχνητή Νοημοσύνη  
Εργασία 1

Γραμμένος Θεόδωρος - Α.Ε.Μ.:3294

Στην παρακάτω έκθεση θα αναλυθεί η υλοποίησή μου της πρώτης εργασίας στην Τεχνητή Νοημοσύνη.

Ενότητα 1: Κλάσεις

Η υλοποίηση της άσκησης βασίζεται σε 2 κλάσεις: την PuzzleBoard και την Orchestrator.

PuzzleBoard

Η κλάση PuzzleBoard περιέχει πληροφορίες για την τρέχουσα κατάσταση του ταμπλό. Παρέχονται 2 κατασκευαστές:

* Ένας ο οποίος δέχεται δύο παραμέτρους int με τις διαστάσεις του ταμπλό και μία παράμετρο bool. Όταν η παράμετρος είναι true δίνονται στο ταμπλό τυχαίες τιμές, ενώ όταν είναι false δεν γίνεται κάποια αρχικοποίηση στις τιμές.
* Ένας δεύτερος ο οποίος δέχεται πάλι δύο παραμέτρους με τις διαστάσεις του ταμπλό και ως τρίτο όρισμα δέχεται έναν πίνακα με int μεγέθους ίσο με το μήκος επί το πλάτος του ταμπλό. Ο πίνακας αυτός χρησιμοποιείται για να αρχικοποιηθεί το ταμπλό. Οι τιμές από την αρχή στο τέλος του πίνακα τοποθετούνται στο ταμπλό από τα αριστερά προς τα δεξιά και από πάνω προς τα κάτω.
* Επίσης έχει υλοποιηθεί ο κατασκευαστής αντιγράφου και καταστροφέας.

Τα περιεχόμενα του ταμπλό αποθηκεύονται σε έναν δισδιάστατο πίνακα που δεσμεύεται δυναμικά ανάλογα με τις διαστάσεις του ταμπλό. Τα περιεχόμενα του ταμπλό αποτελούν αριθμούς από το 1 έως το γραμμές επί στήλες. Το “κενό κομμάτι” αναπαριστάται με τον αριθμό γραμμές επί στήλες. Επίσης μέσα στην κλάση περιέχονται:

* Ένα μέλος emptyPiece που περιέχει ένα pair αριθμών, με τον πρώτο να δείχνει τη γραμμή στην οποία βρίσκεται το “κενό κομμάτι” και το δεύτερο τη στήλη.
* Ένα μέλος parent, δείκτη σε ένα άλλο PuzzleBoard ο οποίος δείχνει στο PuzzleBoard το οποίο τροποποίηθηκε για να προκύψει το τρέχον PuzzleBoard. Για το ταμπλό το οποίο δημιουργείται στην αρχή ο δείκτης έχει τιμή NULL.
* Ένα μέλος lastMove τύπου moveTypes που δείχνει την μετακίνηση που έγινε στο κενό πλακίδιο στο parent ταμπλό για να προκύψει το τρέχον ταμπλό.

Επίσης παρέχονται οι ακόλουθες μέθοδοι:

* Η μέθοδος showBoard επιστρέφει ένα string με την αναπαράσταση του τρέχοντος PuzzleBoard.
* Η μέθοδος randomBoard δέχεται τις διαστάσεις ενός ταμπλό και επιστρέφει ένα ταμπλό με τυχαίες τιμες. Αυτό το κάνει δημιουργώντας 2 πίνακες: έναν με αριθμούς από το 1 έως το γραμμές\*στήλες και έναν με ζευγάρια που αναπαριστούν όλες τις θέσεις του ταμπλό. Οι δύο πίνακες ανακατεύονται και έπειτα προσπελαύνονται σειριακά. Στη θέση που δείχνει ο ένας πίνακας τοποθετείται το νούμερο από τον άλλο πίνακα.
* Η συνάρτηση move δέχεται μία παράμετρο τύπου moveTypes και επιστρέφει ένα PuzzleBoard που είναι το τρέχον ταμπλό με εφαρμοσμένη την κίνηση που δίνεται ως παράμετρος. Αν η κίνηση δεν είναι έγκυρη(π.χ. το κενό κομμάτι βρίσκεται στην πρώτη σειρά και πάει να μετακινηθεί επάνω) επιστρέφει NULL. Το αντικείμενο που επιστρέφει είναι ανεξάρτητο από το αρχικό.

Ευριστική Συνάρτηση

Η ευριστική συνάρτηση που χρησιμοποιείται βασίζεται στην απόσταση Manhattan. Για κάθε πλακίδιο υπολογίζεται η απόσταση Manhattan και στο τέλος το άθροισμά τους. Όσο μικρότερο είναι το άθροισμα τόσο πιο κοντά στη σωστή θέση βρίσκονται περισσότερα πλακίδια. Όταν το άθροισμα είναι 0 όλα τα πλακίδια βρίσκονται στη σωστή θέση. Η συνάρτηση υλοποιείται από τις μεθόδους pieceManhattanDistance, rangeManhattanDistance και totalManhattanDistance. Σε όλες τις συναρτήσεις η αρίθμηση των θέσεων του ταμπλό ξεκινάει από το 0 και αυξάνεται από αριστερά στα δεξιά, από πάνω προς τα κάτω, σε αντίθεση με τα κομμάτια των οποίων η αρίθμηση ξεκινά από το 1.

* Η συνάρτηση pieceManhattanDistance δέχεται τη θέση ενός κομματιού και το νούμερό του και υπολογίζει την απόσταση Manhattan από τη σωστή θέση.
* Η συνάρτηση rangeManhattanDIstance υπολογίζει το άθροισμα των αποστάσεων Manhattan για ένα μέρος του πίνακα, στις θέσεις που δίνονται.
* Η συνάρτηση totalManhattanDistance υπολογίζει το συνολικό άθροισμα Manhattan για όλες τις θέσεις του ταμπλό. Αν το μέγεθος του ταμπλό είναι μεγάλο, το ταμπλό χωρίζεται σε τμήματα ανάλογα με το σύστημα που τρέχει το πρόγραμμα, και εκτελείται η συνάρτηση rangeManhattanDistance για τα επιμέρους τμήματα σε ξεχωριστά threads. Αν το ταμπλό είναι μικρό, τότε εκτελείται σειριακά η pieceManhattanDistance για όλα τα κομμάτια.

Στους αλγορίθμους DFS και BFS χρησιμοποιείται η συνάρτηση totalManhattanDIstance μόνο για να διαπιστωθεί αν μια κατάσταση είναι τερματική. Στον A\* στην τιμή προστίθεται το βάθος της κατάστασης.

Στο πλαίσιο της κλάσης PuzzleBoard έχουν υλοποιηθεί τρία βοηθητικά struct τα οποία απαιτούνται για να χρησιμοποιηθούν οι δομές της STL:

* Το BoardHash υπερφορτώνει τον τελεστή () και δεχόμενο ως όρισμα ένα δείκτη σε PuzzleBoard υπολογίζει το hash του. Δύο αντικείμενα PuzzleBoard είναι ίσα, και άρα πρέπει να έχουν το ίδιο hash αν τα ίδια πλακίδια(με τους ίδιους αριθμούς) βρίσκονται στις ίδιες θέσεις. Συνεπώς για να δημιουργηθεί το hash λαμβάνονται υπόψιν αυτοί οι παράγοντες.
* Το BoardEquality συγκρίνει δύο BoardBoard εξετάζοντας τις ίδιες παραμέτρους με το PuzzleHash
* Το BoardReverseComperator δέχεται 2 δείκτες σε ζευγάρια PuzzleBoard με intκαι επιστρέφει true εάν το int του πρώτου ζευγαριού είναι μεγαλύτερο από το int του δεύτερου.

Orchestrator

Η κλάση Orchestrator είναι υπεύθυνη για την διαχείριση των PuzzleBoard και την υλοποίηση των αλγορίθμων αναζήτησης. Με τη δημιουργία ενός αντικειμένου Orchestrator δημιουργείται ένα αντικείμενο τύπου PuzzleBoard με τις δοσμένες διαστάσεις. Αν δεν δοθεί πίνακας με αρχικές τιμές το ταμπλό αρχικοποιείται τυχαία. Το ταμπλό που αρχικοποιείται αποτελεί την αρχική κατάσταση για την οποία θα δώσουν τα αποτελέσματά τους οι αλγόριθμοι αναζήτησης. Οι αλγόριθμοι αναζήτησης δεν μεταβάλλουν την αρχική κατάσταση και επιστρέφουν μόνο μία συμβολοσειρά με την ακολουθία κινήσεων που απαιτούνται για την επίλυση του παζλ, σύμφωνα με τη λύση που βρήκαν. Εάν δεν βρεθεί λύση επιστρέφουν “no solution found”.

Στην κλάση Orchestrator περιλαμβάνεται και η συνάρτηση buildSolution. Η συνάρτηση αυτή δέχεται ένα PuzzleBoard και σχηματίζει ένα string με την ακολουθία των κινήσεων για να εφαρμοστεί αυτή η λύση, εξετάζοντας τους δείκτες parent και το πεδίο lastMove που έχουν τα αντικείμενα PuzzleBoard.

Υλοποίηση αλγορίθμων

Σε όλους τους αλγορίθμους για το κλειστό σύνολο χρησιμοποιείται η δομή unordered\_set της stl. Τόσο σε αυτή τη δομή, όσο και στις υπόλοιπες βοηθητικές δομές που χρησιμοποιούνται αποθηκεύονται δείκτες σε αντικείμενα PuzzleBoard, για εξοικονόμηση χώρου. Για αυτό το λόγο χρησιμοποιούνται τα structs που αναφέρονται παραπάνω, έτσι ώστε να μπορούν να γίνουν οι πράξεις που χρειάζονται οι δομές πάνω στα αντικείμενα που δείχνουν οι δείκτες, και όχι στην τιμή των δεικτών.

DFS

Στον αλγόριθμο DFS χρησιμοποιείται ως βοηθητική δομή μία στοίβα καθώς χρειάζεται να εξετάζονται πρώτα οι καταστάσεις οι οποίες ανακαλύφθηκαν πιο πρόσφατα. Ο αλγόριθμος όταν βρει μια λύση καλεί απευθείας τη συνάρτηση buildSolution και δεν αναζητάει για άλλες λύσεις. Όταν τερματίζει, επιτυχώς ή ανεπιτυχώς, καθαρίζει όλα τα στοιχεία τόσο του κλειστού συνόλου, όσο και του μετώπου αναζήτησης, εάν έχουν απομείνει.

BFS

Ο αλγόριθμος BFS μοιράζεται το μεγαλύτερο μέρος του κώδικά του με τον DFS. Η μόνη διαφορά είναι ότι χρειάζεται να εξετάζονται πρώτες οι καταστάσεις οι οποίες ανακαλύφθηκαν πιο παλιά. Για αυτό το λόγο αντί για στοίβα χρησιμοποιείται μία ουρά.

BestFS

Ο αλγόριθμος BestFS χρειάζεται κάθε φορά να αναδιατάσσει το μέτωπο αναζήτησης και να εντοπίζει την καλύτερη κατάσταση. Στη συγκεκριμένη περίπτωση η καλύτερη κατάσταση είναι αυτή με τη μικρότερη τιμή της ευριστικής συνάρτησης. Για αυτό το λόγο η βοηθητική δομή που χρησιμοποιούμε είναι ένα priority\_queue το οποίο περιέχει ζευγάρια PuzzleBoard και ενός int με τη βαθμολογία τους. Προκειμένου το priority queue να είναι minheap δίνεται ως custom συγκριτής το struct BoardReverseComperator.

A\*

Στον A\* πάλι χρησιμοποιείται ένα minheap ίδιο με αυτό του BestFS. Η διαφορά έγκειται ότι στη βαθμολογία του κάθε ταμπλό προστίθεται κάθε φορά το βάθος του. Έτσι οι λύσεις που βρίσκονται πιο βαθιά στο δέντρο έχουν μεγαλύτερη βαθμολογία και δεν προτιμώνται. Ταυτόχρονα για να υπολογιστεί το βάθος μιας κατάστασης αφαιρείται από την συνολική βαθμολογία της το totalManhattanDistance. Στη βαθμολογία των παιδιών αυτής της κατάστασης προστίθεται το βάθος συν ένα.

Ενότητα 2: Στατιστικά

Παρακάτω παρατίθενται στατιστικά σχετικά με τον χρόνο εκτέλεσης των διαφόρων αλγορίθμων ανάλογα με το εάν το παζλ που εξετάζουν έχει λύση ή όχι. Οι τιμές βασίζονται σε 100 μετρήσεις για παζλ χωρίς λύση και 100 για παζλ που έχουν λύση διαστάσεων 3x3. Ο κώδικας για την πραγματοποίηση των μετρήσεων βρίσκεται στο αρχείο main.cpp.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Αλγόριθμος | DFS | BFS | BestFS | A\* |
| Επιλύσιμο | 0.2881 | 0.5346 | 0.0013 | 0.01185 |
| Μη επιλύσιμο | 1.065 | 1.115 | 1.64 | 1.733 |

Φαίνεται ότι στα μη επιλύσιμα προβλήματα όλοι οι αλγόριθμοι κάνουν περίπου την ίδια ώρα, με τους BestFS και A\* να κάνουν περισσότερη ώρα από τους BFS και DFS, ίσως λόγω της πιο περίπλοκης δομής που χρησιμοποιούν. Στα επιλύσιμα προβλήματα όμως ο BFS είναι 2 φορές πιο αργός από τον DFS, ενώ ο A\*, ο οποίος βρίσκει τη συντομότερη λύση, είναι 50 φορές πιο γρήγορος από τον BFS. Ο BestFS είναι ο γρηγορότερος από όλους.