Τεχνητή Νοημοσύνη - 1η Εργασία
Έκθεση Σχεδιαστικών Επιλογών

Ονοματεπώνυμο: Μπαρακλιλής Ιωάννης
ΑΕΜ: 3685
email: imparakl@csd.auth.gr
26 Μαρτίου 2021

Ζητούμενο α)

Ζητείται η μοντελοποίηση και υλοποίηση του προβλήματος που δίνεται στην εκφώνηση. Αυτό γίνεται μέσω της κλάσης Puzzle (που περιέχεται στα αρχεία Puzzle.h και Puzzle.cpp).

Η κλάση αυτή μοντελοποιεί την οντότητα της κατάστασης ενός προβλήματος πλακιδίων και έχει:

- Μία μεταβλητή που περιέχει τις θέσεις του κάθε πλακιδίου (του κάθε συμβόλου - αριθμού) ως στατικό πίνακα ακεραίων (όπου σε κάθε κελί υπάρχει ένα διαφορετικό σύμβολο) με WIDTH στήλες και HEIGHT γραμμές με WIDTH και HEIGHT να είναι σταθερές που ορίζονται στο πάνω μέρος του αρχείου Puzzle.h. Οι τιμές της επιστρέφονται ως “δισδιάστατο” vector με την μέθοδο getTiles και μπορούν να οριστούν με την χρήση της setTileValues.
- Μεταβλητές που αποθηκεύουν τις “συντεταγμένες” (γραμμή και στήλη) του κενού πλακιδίου για γρήγορη διαχείριση του.
- Δείκτη σε αντικείμενο Puzzle (“γονέα”) από το οποίο παρήχθησε η εκάστοτε κατάσταση και μία συμβολοσειρά που περιγράφει την μεταβολή (επιστρέφεται απο την μέθοδο getParentAction) που έγινε στην "γονική" κατάσταση ώστε να παραχθεί (η αρχική κατάσταση έχει ως συμβολοσειρά το “Root"), ώστε εφόσον λυθεί το πρόβλημα να μπορεί να γίνει “οπισθοδρόμηση” στην αρχική κατάσταση και να βρεθούν τα βήματα που απαιτούνται για την επίλυση του.
- Μεταβλητή που αποθηκεύει ευρετική τιμή που ορίζεται από αντίστοιχη μέθοδο-setter και επιστρέφεται από αντίστοιχη μέθοδο-getter (get/setHeuristicEvaluation αντίστοιχα) ώστε η τιμή αυτή να μπορεί να ορίζεται εξωτερικά, όπου είναι γνωστή η τελική κατάσταση, με την βοήθεια κάποιας άλλης συνάρτησης.
- Τις μεθόδους moveUp, moveDown, moveLeft και moveRight που δημιουργούν και επιστρέφουν, εφόσον είναι έγκυρες, τις καταστάσεις που προκύπτουν αν εφαρμοστεί η κατάλληλη “μετακίνηση” του κενού πλακιδίου προς τα πάνω, κάτω, αριστερά ή δεξιά αντίστοιχα. Με “μετακίνηση” του κενού πλακιδίου εννοείται η μετακίνηση ενός γειτονικού πλακιδίου προς την κενή θέση έτσι ουσιαστικά μετακινώντας το κενό πλακίδιο. Π.χ. η μετακίνηση του κενού πλακιδίου προς τα πάνω αντιστοιχεί με πραγματική μετακίνηση του πλακιδίου που βρίσκεται πάνω από το κενό, προς τα κάτω.
- Την μέθοδο expandAll που εφαρμόζει όλες τις πιθανές “μετακινήσεις” κενού πλακιδίου και επιστρέφει πίνακα με δείκτες προς εκείνες που προκύπτουν από έγκυρες μετακινήσεις.
- Υπερφορτωμένο τελεστή “<<” που τυπώνει στην δοθείσα έξοδο τα περιεχόμενα του πίνακα ανα γραμμή (δηλαδή την αναπαράσταση της κατάστασης). Η πρώτη γραμμή αναφέρεται ως Row 0.
- Την μέθοδο pathToRoot όπου επιστρέφει το “μονοπάτι” από την αρχική κατάσταση (πιο παλιός “πρόγονος” τρέχουσας) μεχρι την τρέχουσα. Χρησιμοποιείται στην τελική κατάσταση για να βρεθεί το μονοπάτι της λύσης.

Ως ευρετική συνάρτηση, υλοποιεί την μέθοδο tilesOutOfPlace που επιστρέφει τον αριθμό των πλακιδίων εκτός θέσης σε σχέση με το αντικείμενο του ορίσματος και χρησιμοποιείται ως ευρετική συνάρτηση στις περιπτώσεις που η χρήση της είναι απαραίτητη.

Ως συνάρτηση κατακερματισμού, υλοποιεί την μέθοδο getHashValue που επιστρέφει μία αντιπροσωπευτική τιμή για το αντικείμενο για το οποίο καλείται. Η συνάρτηση κατακερματισμού ορίζεται ως: Έχει αρχική τιμή κατακερματισμού το 127 (πρώτος αριθμός), για κάθε στοιχείο του πίνακα με τις τιμές των πλακιδίων (διάσχιση από αριστερά προς τα δεξιά και από πάνω προς τα κάτω) θέτουμε ως νέα τιμή κατακερματισμού το γινόμενο της προηγούμενης τιμής με 11 (πρώτος αριθμός) που προστίθεται η τιμή του πλακιδίου και το άθροισμα αυτό περιορίζεται (με τον τελεστή %) στο εύρος του unsigned long.

Τέλος, υλοποιεί και άλλες βοηθητικές μεθόδους, τις swapTiles, getLengthToRoot και τους τελεστές “=” και “==” των οποίων οι λειτουργίες θεωρούνται τετριμμένες.

Παράλληλα με την Puzzle, στο αρχείο Puzzle.h ορίζονται και:

- Η κλάση PuzzleHashFunction, που που υλοποιεί τον τελεστή () που επιστρέφει την τιμή συνάρτησης κατακερματισμού για κάποιο αντικείμενο στο οποίο αντιστοιχεί ο δείκτης ορίσματος. Χρησιμοποιείται ως παράμετρος τύπου στην κλάση unordered_set ώστε εκείνη να μπορεί να αποθηκεύει δείκτες σε και όχι τα ίδια τα αντικείμενα Puzzle, ενώ παράλληλα να έχει πρόσβαση στην συνάρτηση κατακερματισμού του αντικειμένου και είναι απαραίτητο για την λειτουργία της κλάσης unordered_set.
- Η κλάση PuzzlePtrEquals, υλοποιεί τον τελεστή () που αποτελεί τον τελεστή ισότητας για δύο αντικείμενα Puzzle μέσω δεικτών τους. Χρησιμοποιείται ως παράμετρος τύπου στην κλάση unordered_set ώστε εκείνη να μπορεί να αποθηκεύει δείκτες σε και όχι τα ίδια τα αντικείμενα Puzzle, ενώ παράλληλα να έχει την δυνατότητα να ελέγχει την ισότητα των αντικειμένων που μας ενδιαφέρουν (και αντιστοιχούν στους δείκτες) και είναι απαραίτητο για την λειτουργία της κλάσης unordered_set με τον τρόπο που μας ενδιαφέρει (να ελέγχει την ύπαρξη και να εξασφαλίζει μοναδικότητα των αντικειμένων Puzzle που αντιστοιχούν σε δείκτες που αποθηκεύει).
- Η κλάση PuzzleGreaterHeuristicValueComparator, υλοποιεί τον τελεστή () που αποτελεί συγκριτή των αποθηκευμένων ευρετικών τιμών δύο καταστάσεων που αντιστοιχούν στους δείκτες των παραμέτρων.Χρησιμοποιείται ως παράμετρος τύπου στην κλάση priority_queue ώστε εκείνη να μπορεί να αποθηκεύει δείκτες σε και όχι τα ίδια τα αντικείμενα Puzzle, ενώ παράλληλα να έχει την δυνατότητα να μπορεί να συγκρίνει τις ευρετικές τιμές των αντικειμένων που μας ενδιαφέρουν (και αντιστοιχούν στους δείκτες) και είναι απαραίτητο για την λειτουργία της κλάσης priority_queue με τον τρόπο που μας ενδιαφέρει (να δίνει με την χρήση της κατάλληλης μεθόδου σε μικρό χρόνο της κατάσταση εκείνη με την μικρότερη ευρετική τιμή).

Ζητούμενο β)

Ζητείται να υλοποιηθούν οι αλγόριθμοι Depth First Search, Breadth First Search, Best First Search και A* για να λυθεί το δοθέν πρόβλημα. Αυτό γίνεται στο αρχείο main.cpp με τον ορισμό των συναρτήσεων DFS, BFS, BestFS και A_Star αντίστοιχα όπου ο ορισμός της αρχικής και τελικής κατάστασης, η κλήση των παραπάνω συναρτήσεων για επίλυση του προβλήματος και η εμφάνιση αποτελεσμάτων εκτελούνται απο την συνάρτηση main.

Στην συνάρτηση DFS:

1. Ορίζεται το μέτωπο αναζήτησης όπου αρχικά είναι κενό, ως στοίβα (LIFO).
2. Ορίζεται το κλειστό σύνολο ως unordered_set όπου ως ορίσματα τύπων (εκτός δείκτη σε Puzzle) δίνονται η PuzzleHashFunction για χρήση ως συνάρτηση κατακερματισμού και η PuzzlePtrEquals για χρήση ως συνάρτηση ισότητας, απο την δομή.
3. Αρχικοποιούνται οι μετρητές στατιστικών.
4. Εισάγεται η αρχική κατάσταση στο μέτωπο αναζήτησης.
5. Όσο το μέτωπο αναζήτησης δεν είναι κενό (όσο υπάρχουν καταστάσεις που μπορούν να εξερευνηθούν ακόμα) εκτελείται ο βρόχος των επομένων βημάτων:
6. Ενημερώνεται, εφόσον χρειάζεται, η μέγιστη τιμή μνήμης (συνολικός αριθμός καταστάσεων που υπάρχουν την τρέχουσα χρονική στιγμή στο κλειστό σύνολο και μέτωπο αναζήτησης).
7. Λαμβάνεται το πρώτο στοιχείο από το μέτωπο αναζήτησης ως τρέχον και αφαιρείται από το μέτωπο.
8. Αν το στοιχείο αυτό ανήκει στο κλειστό σύνολο, το διαγράφω και η επανάληψη προχωράει στον επόμενο βρόχο.
9. Ενημερώνεται ο αριθμός καταστάσεων που εξερευνήθηκαν.
10. Αν η κατάσταση του τρέχοντος στοιχείου ισούται (μέσω εφαρμογής του τελεστή ==) με την τελική, αποδεσμεύω την μη απαιτούμενη μνήμη (του μετώπου αναζήτησης και των υπολοίπων του μονοπατιού λύσης καταστάσεων) και επιστρέφεται το μονοπάτι της λύσης.
11. Σε διαφορετική από τις παραπάνω περιπτώσεις, ευρίσκονται και εισάγονται στην αρχή (επιτυγχάνεται μέσω χρήσης στοίβας) του μετώπου αναζήτησης οι παράγωγες του τρέχοντος καταστάσεις και η κατάσταση του τρέχοντος αυτού στοιχείου εισάγεται στο κλειστό σύνολο.
12. Αν ο παραπάνω βρόχος των βημάτων 5 - 11 δεν τερματιστεί με συνάρτηση επιστροφής, αλλά λόγω κενού μετώπου αναζήτησης αυτό σημαίνει ότι δεν υπάρχει κατάσταση που ισούται με την τελική και επομένως δεν υπάρχει λύση και επιστρέφεται κενό μονοπάτι λύσης αφού αποδεσμευτεί η μνήμη που χρησιμοποιούν το κλειστό σύνολο και μέτωπο αναζήτησης.

Στην συνάντηση BFS εκτελούνται παρόμοια βήματα με εκείνα του DFS οπότε απλά θα επισημανθούν οι διαφορές με τα βήματα του DFS:

- Ως δομή του μετώπου αναζήτησης χρησιμοποιείται, αντί για στοίβα, ουρά FIFO. Έτσι στο βήμα εισαγωγής των παραγώγων καταστάσεων αυτές εισάγονται στο τέλος αντί της αρχής του μετώπου αναζήτησης.
- Επίσης, αλλάζει και η μέθοδος πρόσβασης των καταστάσεων στο μέτωπο αναζήτησης: front αντί για top.

Στην συνάντηση BestFS εκτελούνται παρόμοια βήματα με εκείνα του DFS οπότε απλά θα επισημανθούν οι διαφορές με τα βήματα του DFS:

- Ως δομή του μετώπου αναζήτησης χρησιμοποιείται, αντί για στοίβα, ουρά προτεραιότητας όπου ως ορίσματα τύπων (εκτός δείκτη σε Puzzle) δίνονται η deque<Puzzle *> για χρήση ως δομή της ουράς και η PuzzleGreaterHeuristicValueComparator για χρήση ως συγκριτής στοιχείων (έτσι ώστε να υπάρχει στην αρχή της ουράς η κατάσταση με την μικρότερη ευρετική τιμή, δηλαδή εκείνη με το μικρότερο κόστος). Έτσι στο βήμα λήψης επόμενου στοιχείου για εξερεύνηση, θα ληφθεί εκείνο με την μικρότερη ευρετική τιμή.
- Επίσης, στο σημείο εισαγωγής παράγωγων καταστάσεων στο μέτωπο αναζήτησης τίθεται (μέσω της μεθόδου setHeuristicEvaluation) ως ευρετική τιμή κάθε μιας κατάστασης ο αριθμός των πλακιδίων εκτός θέσης ως προς την δοθείσα (απο όρισμα συνάρτησης) τελική κατάσταση. Η τιμή αυτή θα χρησιμοποιηθεί από την ουρά προτεραιότητας για να επιστρέφει γρήγορα την κατάσταση με το μικρότερο κόστος (μικρότερη ευρετική τιμή).

Στην συνάντηση A_Star εκτελούνται παρόμοια βήματα με εκείνα του BestFS οπότε απλά θα επισημανθούν οι διαφορές με τα βήματα του BestFS:

- Στο σημείο εισαγωγής παράγωγων καταστάσεων στο μέτωπο αναζήτησης τίθεται (μέσω της μεθόδου setHeuristicEvaluation) ως ευρετική τιμή κάθε μιας κατάστασης ο αριθμός των πλακιδίων εκτός θέσης ως προς την δοθείσα (απο όρισμα συνάρτησης) τελική κατάσταση σύν το μήκος του μονοπατιού (αριθμός κινήσεων (απόσταση) από την αρχική κατάσταση μέχρι να βρεθούμε στην τρέχουσα, μέσω της getLengthToRoot) από την αρχική κατάσταση. Η τιμή αυτή θα χρησιμοποιηθεί από την ουρά προτεραιότητας για να επιστρέφει γρήγορα την κατάσταση με το μικρότερο κόστος (μικρότερο άθροισμα ευρετικής τιμής και μονοπατιού από την αρχή).

Στην συνάρτηση main:

- Ορίζονται οι αρχικές και τελικές καταστάσεις, σύμφωνα με την εκφώνηση.
- Για τον αλγόριθμο Depth First Search:
    - Βρίσκεται η λύση στο πρόβλημα με κλήση της DFS, ενώ παράλληλα γίνεται χρονομέτρηση της.
    - Τυπώνονται τα στατιστικά εκτέλεσης της (χωρίς να τυπωθούν τα βήματα λύσης, η λύση γιατί είναι ιδιαίτερα μεγάλη σε πλήθος βημάτων).
    - Γίνεται αποδέσμευση μνήμης για τις καταστάσεις της λύσης με τον αλγόριθμο αυτό (με εξαίρεση της αρχικής που χρησιμοποιείται και σε επόμενους αλγορίθμους).
- Για τον αλγόριθμο Breadth First Search εκτελούνται όμοια βήματα, όμως τυπώνονται τα βήματα της λύσης.
- Για τον αλγόριθμο Best First Search εκτελούνται όμοια βήματα με του Depth First Search.
- Για τον αλγόριθμο A* εκτελούνται όμοια βήματα με του Breadth First Search.

Στατιστικά επίλυσης

Παρακάτω βρίσκονται τα στατιστικά εκτέλεσης του προγράμματος:

Αλγόριθμος	Βάθος μονοπατιού λύσης	Αριθμός καταστάσεων που εξερευνήθηκαν	Μέγιστος αριθμός καταστάσεων ταυτόχρονα στην μνημη	Χρόνος εκτέλεσης (σε milliseconds)
Depth First Search	26762	27018	59271	179.652
Breadth First Search	10	437	905	1.76793
Best First Search	104	999	1777	14.8194
A*	10	37	90	0.382302

Η συντομότερη λύση στο πρόβλημα (που δίνεται τους αλγορίθμους Breadth First Search και A*) έχει πλήθος βημάτων 10 και είναι η εξής:

1. Μετακίνηση 6 δεξιά,
2. Μετακίνηση 3 δεξιά,
3. Μετακίνηση 1 πάνω,
4. Μετακίνηση 4 αριστερά,
5. Μετακίνηση 2 αριστερά,
6. Μετακίνηση 6 κάτω,
7. Μετακίνηση 3 δεξιά,
8. Μετακίνηση 2 πάνω,
9. Μετακίνηση 5 πάνω και
10. Μετακίνηση 8 αριστερά.