Εργαστηριακή Άσκηση 1
Αναφορά Υλοποίησης

Χρήστος Χριστοδούλου 5392

Δημήτρης Τσακίρης 5371

Λαλούτσος Νίκος 5266

Παρακάτω παρουσιάζεται συνοπτικά η υλοποιήση που κάναμε για την επίλυση του 8-παζλ με UCS και A* σε Java.

Τα βασικά στοιχεία του κώδικα είναι τα όμοια και για τα δύο προγράμματα.

UCS

Κλάση EightPuzzleSolver:

- Αρχικοποιεί ένα σύνολο settled το οποίο θα περιέχει τους κόμβους που θα έχουν εξεταστεί.
- Δημιουργεί ένα Priority Queue για να ταξινομούνται οι κόμβοι με βάση την απόσταση από την Αρχική Κατάσταση.
- Ορίζει την Τελική Κατάσταση.

- Αρχικοποιεί τις μεταβλητές που θα χρησιμοποιηθούν για τις κινήσεις του κενού κελιού.
- Δημιουργεί μία λίστα optimalpath στην οποία αποθηκεύει την βέλτιστη διαδρομή μεταξύ Α.Κ.-Τ.Κ.
- Αρχικοποιεί τον μετρητή extentionscount ο οποίος μετρά τη βέλτιστη διαδρομή μεταξύ ΑΚ-ΤΚ

Κλάση Node:

- State:Κατάσταση του puzzle.
- emptyX,emptyY:Είναι οι συντεταγμένες της κενής θέσης ,την οποία κινούμε στο board.
- Parent:Ο γονέας του κόμβου στο βέλτιστο μονοπάτι.

Κύρια Μέθοδος UCS:

1) Δημιουργούμε έναν κόμβο ο οποίος αντιστοιχεί στην αρχική κατάσταση και τον προσθέτουμε στην ουρά προτεραιότητας.
2) Node current:Τον καλούμε μέχρι την εξάντληση ολων των κόμβων ή μέχρι την εύρεση στόχου.Σε κάθε επανάληψη αφαιρώ τον κόμβο με το μικρότερο κόστος από την ουρά προτεραιότητας.
3) Ελέγχουμε αν ο τρεχών κόμβος είναι Τελική Κατάσταση.Εάν ναι ο αλγόριθμος τερματίζεται.
4) Για κάθε γειτονική κατάσταση του τρεχόντος κόμβου ,υπολογίζουμε το κόστος και δημιουργούμε έναν νέο κόμβο.
5) Αν ο νέος κόμβος δεν έχει ελεγχθεί ακόμα ή αν έχει μικρότερο κόστος από τον προη

6) Κάθε φορά που επεκτείνω έναν κόμβο ενημερώνω την αποστασή του από την Αρχική Κατάσταση.

A*

Κλάση EightPuzzleSolverAstr

Τα βασικά της στοιχεία είναι τα ίδια,έχοντας προσθέσει δύο νέες λειτουργίες.

- getTotalCost:Υπολογίζει το συνολικό κόστος κάθε κόμβου λαμβάνοντας υπόψιν το πραγματικό κόστος μέχρι εκείνο το σημείο και την ευρετική εκτίμηση του υπόλοιπου κόστους.
- calculateHeuristic:Υπολογίζει την ευρετική συνάρτηση για μια δεδομένη κατάσταση του παζλ.

Η βασική διαφορά πλέον είναι ότι κάθε κόμβος πλεον έχει το κόστος μέχρι τον προηγούμενο κόμβο ακριβώς όπως και με την UCS αλλά + της ευρετικής απόστασης από την Τελική Κατάσταση.

Η ευρετική συνάρτηση που χρησιμοποιούμε υπολογίζει την ελάχιστη απόσταση που πρέπει να διανύσει ένας κόμβος για να φτάσει στον στόχο.

Συγκεκριμένα, αυτή η ευρετική συνάρτηση υπολογίζει το άθροισμα των αποστάσεων στον άξονα x και y για κάθε

κελί του παζλ από την τρέχουσα θέση του κενού κελιού μέχρι τη θέση που θα έπρεπε να βρίσκεται στην τελική κατάσταση. Αυτό σημαίνει ότι υπολογίζουμε την απόσταση του κάθε κελιού από τη θέση του στην τελική κατάσταση, και συνολικά προσθέτουμε όλες αυτές τις αποστάσεις.

Στην ουσία, αυτή η ευρετική συνάρτηση μας βοηθάει να επιλέξουμε τον επόμενο καλύτερο κόμβο που θα εξετάσουμε, προσδιορίζοντας ποιος κόμβος φαίνεται πιο κοντά στον στόχο μας.Έτσι,έχουμε ομοιόμορφη εκτίμηση της απόστασης για την Τελική Κατάσταση για κάθε Τρέχουσα Κατάσταση.

Εκτέλεση αλγορίθμων.

Ως πειράματα έχουμε επιλέξει τα παρακάτω σενάρια Αρχικής Κατάστασης,τα οποία υπάρχουν στον κώδικα ως σχόλια για να βάλετε όποιο από τα 5 εσείς θελετε.

Για την UCS:

```
C:\Users\lalou\OneDrive\Υπολογιστ
Cost: 5
[6, 0, 4]
[1, 7, 3]
[8, 5, 2]
------
[6, 7, 4]
[1, 0, 3]
[8, 5, 2]
------
[6, 7, 4]
[1, 5, 3]
[8, 0, 2]
------
[6, 7, 4]
[0, 5, 3]
[8, 1, 2]
------
[6, 0, 4]
[7, 5, 3]
[8, 1, 2]
------
[6, 5, 4]
[7, 0, 3]
[8, 1, 2]
------
Number of node extensions: 715
```

```
Cost: 2
[6, 5, 4]
[0, 1, 3]
[8, 7, 2]
------
[6, 5, 4]
[7, 1, 3]
[8, 0, 2]
------
[6, 5, 4]
[7, 0, 3]
[8, 1, 2]
------
Number of node extensions: 13
```

```
Cost: 6
[6, 5, 0]
[7, 1, 3]
[8, 4, 2]
------
[6, 5, 3]
[7, 1, 0]
[8, 4, 2]
------
[6, 5, 3]
[7, 1, 4]
[8, 0, 2]
------
[6, 5, 3]
[7, 0, 4]
[8, 1, 2]
------
[6, 5, 3]
[7, 4, 0]
[8, 1, 2]
------
[6, 5, 0]
[7, 4, 3]
[8, 1, 2]
------
[6, 5, 4]
[7, 0, 3]
[8, 1, 2]
------
Number of node extensions: 1309
```

```
C:\Users\lalou\OneDrive\Υπολογιστ
Cost: 14
[4, 0, 8]
[1, 7, 6]
[3, 5, 2]
------
[4, 6, 8]
[1, 7, 0]
[3, 5, 2]
------
[4, 6, 8]
[1, 7, 5]
[3, 0, 2]
------
[4, 6, 8]
[1, 7, 5]
[0, 3, 2]
------
[4, 6, 8]
[1, 0, 5]
[7, 3, 2]
------
[4, 6, 0]
[1, 8, 5]
[7, 3, 2]
------
[4, 6, 5]
[1, 8, 0]
[7, 3, 2]
------
[4, 6, 5]
[1, 8, 3]
[7, 0, 2]
------
[4, 6, 5]
[0, 8, 3]
[7, 1, 2]
------
[4, 6, 5]
[7, 8, 3]
```

```
[4, 6, 5]
[7, 0, 3]
[8, 1, 2]
------
[0, 6, 5]
[7, 4, 3]
[8, 1, 2]
------
[6, 0, 5]
[7, 4, 3]
[8, 1, 2]
------
[6, 5, 0]
[7, 4, 3]
[8, 1, 2]
------
[6, 5, 4]
[7, 0, 3]
[8, 1, 2]
------
Number of node extensions: 223060
```

```
C:\Users\lalou\OneDrive\Υπολογιστή
Cost: 8
[6, 5, 4]
[3, 2, 7]
[8, 1, 0]
------
[6, 5, 4]
[3, 0, 7]
[8, 1, 2]
------
[6, 5, 4]
[3, 7, 0]
[8, 1, 2]
------
[6, 0, 4]
[3, 7, 5]
[8, 1, 2]
------
[6, 3, 4]
[0, 7, 5]
[8, 1, 2]
------
[6, 3, 4]
[7, 0, 5]
[8, 1, 2]
------
[6, 3, 4]
[7, 5, 0]
[8, 1, 2]
------
[6, 0, 4]
[7, 5, 3]
[8, 1, 2]
------
[6, 5, 4]
[7, 0, 3]
[8, 1, 2]
------
Number of node extensions: 5844
```

Για την A*:

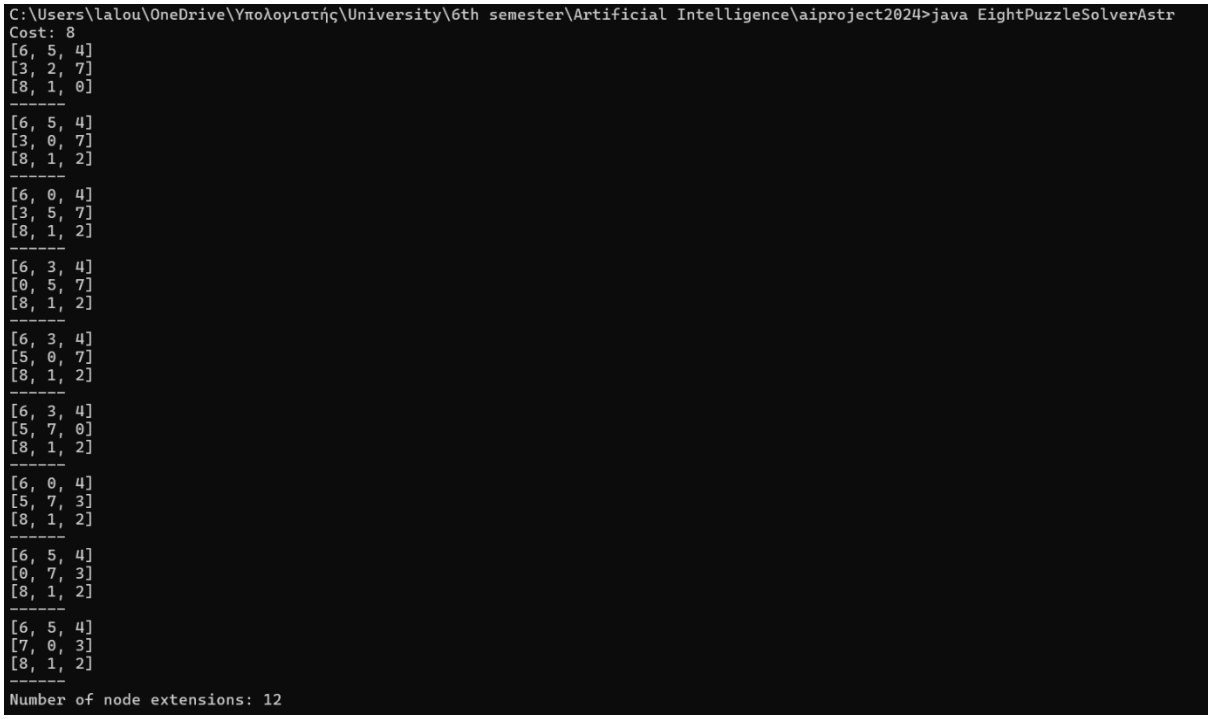
```
C:\Users\lalou\OneDrive\Υπολογιστής\University\6th semester\Artificial Intelligence\aiproject2024>java EightPuzzleSolverAstr
Cost: 16
[4, 0, 8]
[1, 7, 6]
[3, 5, 2]
------
[4, 7, 8]
[1, 0, 6]
[3, 5, 2]
------
[4, 7, 0]
[1, 8, 6]
[3, 5, 2]
------
[4, 7, 6]
[1, 8, 0]
[3, 5, 2]
------
[4, 7, 6]
[1, 8, 5]
[3, 0, 2]
------
[4, 7, 6]
[0, 8, 5]
[3, 1, 2]
------
[4, 7, 6]
[3, 8, 5]
[0, 1, 2]
------
[4, 7, 6]
[3, 0, 5]
[8, 1, 2]
------
[4, 7, 6]
[0, 3, 5]
[8, 1, 2]
------
```

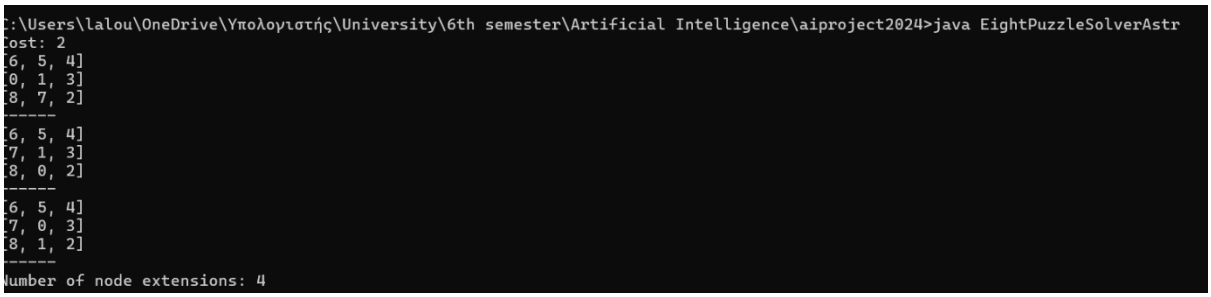

```
C:\Users\lalou\OneDrive\Υπολογιστής\University\6th semester\Artificial Intelligence\aiproject2024>java EightPuzzleSolverAstr
Cost: 16
[4, 0, 8]
[1, 7, 6]
[3, 5, 2]
------
[4, 7, 8]
[1, 0, 6]
[3, 5, 2]
------
[4, 7, 0]
[1, 8, 6]
[3, 5, 2]
------
[4, 7, 6]
[1, 8, 0]
[3, 5, 2]
------
[4, 7, 6]
[1, 8, 5]
[3, 0, 2]
------
[4, 7, 6]
[0, 8, 5]
[3, 1, 2]
------
[4, 7, 6]
[3, 8, 5]
[0, 1, 2]
------
[4, 7, 6]
[3, 0, 5]
[8, 1, 2]
------
[4, 7, 6]
[0, 3, 5]
[8, 1, 2]
------
```

```
[4, 7, 6]
[0, 3, 5]
[8, 1, 2]
------
[4, 0, 6]
[7, 3, 5]
[8, 1, 2]
------
[4, 6, 0]
[7, 3, 5]
[8, 1, 2]
------
[4, 6, 3]
[7, 0, 5]
[8, 1, 2]
------
[0, 6, 3]
[7, 4, 5]
[8, 1, 2]
------
[6, 0, 3]
[7, 4, 5]
[8, 1, 2]
------
[6, 5, 3]
[7, 4, 0]
[8, 1, 2]
------
[6, 5, 0]
[7, 4, 3]
[8, 1, 2]
------
[6, 5, 4]
[7, 0, 3]
[8, 1, 2]
------
Number of node extensions: 2342

C:\Users\lalou\OneDrive\Υπολογιστής\University\6th semester\Artificial Intelligence\aiproject2024>
```

```
C:\Users\lalou\OneDrive\Υπολογιστής\University\6th semester\Artificial Intelligence\aiproject2024>java EightPuzzleSolverAstr
Cost: 6
[6, 5, 0]
[7, 1, 3]
[8, 4, 2]
------
[6, 5, 3]
[7, 1, 0]
[8, 4, 2]
------
[6, 5, 3]
[7, 1, 4]
[8, 0, 2]
------
[6, 5, 3]
[7, 0, 4]
[8, 1, 2]
------
[6, 5, 3]
[7, 4, 0]
[8, 1, 2]
------
[6, 5, 0]
[7, 4, 3]
[8, 1, 2]
------
[6, 5, 4]
[7, 0, 3]
[8, 1, 2]
------
Number of node extensions: 12
```





Συμπεράσματα

Τα αποτελέσματά μας αποδεικνύουν ότι για μικρό ή μεσαίο κόστος, ο αλγόριθμος A* απαιτεί λιγότερη εξερεύνηση κόμβων και, κατά συνέπεια, λιγότερο χρόνο για να φτάσει

στην τελική κατάσταση. Επιπλέον, παρατηρήσαμε ότι τα κόστη του UCS και του A* συμπίπτουν σε αυτές τις περιπτώσεις. Ωστόσο, για μεγάλο κόστος, η διαφορά γίνεται ακόμη πιο εμφανής καθώς ο UCS απαιτεί πολύ περισσότερο χρόνο εκτέλεσης. Παρόλα αυτά, παρατηρούμε ότι για μεγάλο κόστος, ο UCS είναι πιο αποδοτικός από τον A*, αν και αυτό συνεπάγεται μεγαλύτερο χρόνο εκτέλεσης. Σημειώνουμε επίσης ότι η διαφορά στο κόστος παραμένει σταθερή σε 2, ακόμη και για μεγαλύτερα κόστη.

Ο A* μπορεί να είναι πιο αποδοτικός σε περιπτώσεις με μικρό ή μεσαίο κόστος, καθώς μειώνει τον αριθμό των κόμβων που πρέπει να εξερευνήσει πριν βρει τη λύση. Αντίθετα, ο αλγόριθμος UCS εξερευνά τους κόμβους με βάση το πραγματικό τους κόστος, χωρίς να λαμβάνει υπόψη την πρόβλεψη του τελικού κόστους. Αυτό μπορεί να οδηγήσει σε περισσότερη εξερεύνηση και, συνεπώς, μεγαλύτερο χρόνο εκτέλεσης, ιδίως σε περιπτώσεις μεγάλου κόστους.

Στην περίπτωση του UCS, ο λόγος που εμφανίζεται πιο αποδοτικός σε μεγάλο κόστος αλλά πιο αργός, μπορεί να οφείλεται στον τρόπο λειτουργίας του αλγορίθμου. Ο UCS εξερευνά τους κόμβους με βάση το πραγματικό κόστος τους, επιλέγοντας πάντα τον κόμβο με το χαμηλότερο συνολικό κόστος μέχρι στιγμής. Σε προβλήματα με μεγάλο κόστος, ο UCS ενδέχεται να εξερευνά πρώτα κόμβους που απαιτούν σημαντικά χαμηλότερο κόστος, αλλά που βρίσκονται σε μεγάλη απόσταση από την αρχική κατάσταση. Αυτό μπορεί να οδηγήσει σε μεγαλύτερη αναζήτηση και, συνεπώς, σε αυξημένο χρόνο εκτέλεσης, καθώς ο UCS είναι υποχρεωμένος να εξερευνήσει όλους τους κόμβους με αύξουσα σειρά κόστους. Αντίθετα, ο αλγόριθμος A* χρησιμοποιεί μια ευρετική συνάρτηση για να εστιάσει την αναζήτηση σε κατευθυνόμενο τρόπο προς τον στόχο, με αποτέλεσμα να εξερευνά λιγότερους κόμβους και να ολοκληρώνει τον υπολογισμό του με μικρότερο χρόνο, όταν υπάρχει κατάλληλη ευρετική συνάρτηση που εκτιμά σωστά το κόστος της διαδρομής.