1η Εργασία στο Μάθημα της Τεχνητής Νοημοσύνης

Ονοματεπώνυμο: Σωτήριος-Λουκάς Καμπύλης

ΑΕΜ: 3805

Ένα μεγάλο μέρος του κώδικα, που παράχθηκε για την υλοποίηση του προβλήματος της εργασίας, βασίστηκε πάνω στον κώδικα (Maze) του καθηγητή. Οι αλγόριθμοι:1)Depth First Search (DFS), 2)Breadth First Search (BFS) και 3)Best First Search (Best FS) βασίζονται πάνω στον κώδικα που διδαχθήκαμε στην θεωρία και είδαμε σε εφαρμογή στο εργαστήριο ενώ η υλοποίηση του αλγορίθμου 4)A* βασίστηκε μόνο από το θεωρητικό μέρος του μαθήματος. Μάλιστα, μέσα στον κώδικα υπάρχουν αρκετά σχόλια που περιγράφουν στο περίπου τι κάνει η κάθε μέθοδος (μόνο για τις μεθόδους που είναι δυσνόητες).

Υλοποίηση Αλγορίθμων:

1. DFS: Έχει ως μέτωπο αναζήτησης μια στοίβα (stack). Το κλειστό σύνολο υλοποιήθηκε με την δομή vector, οπότε πραγματοποιείται σειριακή αναζήτηση. Αρχικά, τοποθετώ στο μέτωπο αναζήτησης την αρχική μου κατάσταση. Με μια επαναληπτική δομή (και όσο δεν είναι άδειο το μέτωπο αναζήτησης) εφαρμόζω τον αλγόριθμο. Παίρνω την πρώτη κατάσταση από το μέτωπο αναζήτησης και ελέγχω αν η τρέχουσα κατάσταση υπάρχει μέσα στο κλειστό σύνολο. Μετά, ελέγχω αν η κατάσταση είναι τερματική. α)Αν είναι τερματική: την επιστρέφω ως λύση. β)Αν δεν είναι τερματική: Βάζω την τρέχουσα κατάσταση στο κλειστό σύνολο και δημιουργώ τα παιδιά του (μέσα από την μέθοδο expand). Τέλος, με μια επαναληπτική δομή προσθέτω τα παιδιά στο μέτωπο αναζήτησης (εφόσον δεν ανήκουν στο κλειστό σύνολο). Μάλιστα, ο συγκεκριμένος αλγόριθμος αν δεν μπορέσει να λύσει το πρόβλημα θα επιστρέψει null (αλλιώς επιστρέφει την τερματική κατάσταση).
2. BFS: Ο συγκεκριμένος αλγόριθμος ακολουθεί την ίδια σχεδιαστική υλοποίηση με τον DFS μόνο που έχει ως μέτωπο αναζήτησης μια

ουρά (queue). Άρα, τα παιδιά θα βγουν πρώτα από το μέτωπο αναζήτησης.

3. Best FS: Για τον συγκεκριμένο αλγόριθμο χρησιμοποιήθηκε ως μέτωπο αναζήτησης μια ουρά προτεραιότητας (priority queue). Επίσης, για το κλειστό σύνολο χρησιμοποιήθηκε η δομή unordered map (ταχύτερη από την δομή vector).Μάλιστα, η ευριστική συνάρτηση που χρησιμοποιήθηκε στον αλγόριθμο βρίσκει με την απόσταση Manhattan την απόσταση μιας κατάστασης από μια άλλη. Η μέθοδος heuristic χρησιμοποιείται μέσα στην κλάση myComparator, η οποία ταξινομεί μέσα από την μέθοδο heuristic τις τιμές της κάθε κατάστασης.
4. A*: Ο συγκεκριμένος αλγόριθμος ακολουθεί σχεδιαστική υλοποίηση μέσα από την κατανόηση της θεωρίας του μαθήματος (χρησιμοποιεί δική της ευριστική μέθοδο-την aStarHeuristic, η οποία βρίσκει την απόσταση Manhattan και έχει ίδια λογική με την heuristic). Έχει ως μέτωπο αναζήτησης την δομή priority queue και ως κλειστό σύνολο την δομή vector.

Μοντελοποίηση προβλήματος:

Η εργασία έχει υλοποιηθεί στο περιβάλλον του codeblocks.Δεν έχει κάποιο error άρα η διασύνδεση των αρχείων πρέπει να πραγματοποιείται με επιτυχία. Το πρόγραμμα βρίσκει λύση και στους 4 αλγορίθμους (στο συγκεκριμένο μοτίβο με τα πλακίδια που φαίνεται στην εκφώνηση της εργασίας) <<σέρνοντας>> πάνω, κάτω, αριστερά και δεξιά τα πλακίδια έτσι ώστε να καταλήξει στην τελική μορφή που δίνεται στην εκφώνηση της εργασίας. Μάλιστα, φαίνεται κάθε βήμα που πραγματοποιείται στο cmd.

Στατιστικά επίλυσης:

1. DFS:

```
DFS:depth=33330,Mem:59575,Examined:33630
Process returned 0 (0x0)   execution time : 140.034 s
Press any key to continue.
```

2. BFS:

```
BFS:depth=10,Mem:1064,Examined:665
Process returned 0 (0x0)   execution time : 5.267 s
Press any key to continue.
```

3. Best FS:

```
BestFS:depth=10,Mem:21,Examined: 15
Process returned 0 (0x0)   execution time : 1.663 s
Press any key to continue.
```

4. A*:

```
A*:depth=10,Mem:463,Examined:32
Process returned 0 (0x0)   execution time : 1.627 s
Press any key to continue.
```