ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ
ΤΜΗΜΑ ΕΦΑΡΜΟΣΜΕΝΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ

6ο εξάμηνο

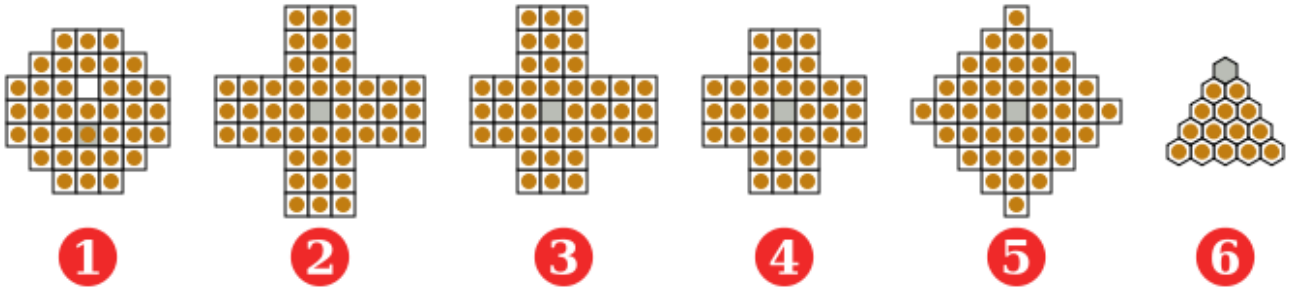
Διδάσκων: Γιάννης Ρεφανίδης

1η Εργασία
18/3/2021

Peg Solitaire

Εισαγωγικά:

Το παιχνίδι Peg Solitaire[1] είναι ένα παιχνίδι τύπου ντάμας για έναν παίκτη, που παίζεται πάνω σε έναν πίνακα με οπές και μετακινούμενους πασσάλους. Μια οπή μπορεί να είναι κενή ή να φιλοξενεί έναν πάσσαλο. Αρχικά υπάρχει μόνο μια κενή οπή (στην εικόνα που ακολουθεί φαίνονται διάφορες παραλλαγές του αρχικού πίνακα του παιχνιδιού).



Υπάρχει μία μοναδική κίνηση που μπορεί να κάνει ο παίκτης: Εάν υπάρχουν δύο πάσσαλοι και κενή οπή σε τρεις γειτονικές θέσεις στην ίδια ευθεία, με την οπή στο ένα άκρο, ο πάσσαλος που βρίσκεται στο άλλο άκρο μετακινείται στην οπή ενώ ο πάσσαλος στην ενδιάμεση θέση αφαιρείται από τον πίνακα του παιχνιδιού.

Στόχος του παιχνιδιού είναι να αφαιρεθούν από τον πίνακα όλοι οι πάσσαλοι πλην ενός.

Στην εργασία αυτή καλείστε να γράψετε ένα πρόγραμμα (σε όποια γλώσσα προγραμματισμού επιθυμείτε) που να λύνει προβλήματα τύπου Peg Solitaire, χρησιμοποιώντας δύο διαφορετικούς αλγορίθμους αναζήτησης. Το πρόγραμμά σας θα διαβάζει το κάθε πρόβλημα από αρχείο και θα γράφει τη λύση σε αρχείο, όπως εξηγείται παρακάτω.

---

[1] http://en.wikipedia.org/wiki/Peg_solitaire. Υπάρχουν διαθέσιμες πολλές online εκδοχές του παιχνιδιού στο διαδίκτυο.

Θέμα 1ο : Αναζήτηση πρώτα σε βάθος

Κατασκευάστε ένα πρόγραμμα που να λύνει προβλήματα peg solitaire με τον αλγόριθμο πρώτα σε βάθος (depth-first search). Το πρόγραμμα θα δέχεται ως παραμέτρους τη μέθοδο επίλυσης, το όνομα του αρχείου περιγραφής του προβλήματος και το όνομα του αρχείου στο οποίο θα γραφεί η λύση. Για παράδειγμα, εάν το όνομα του προγράμματός σας είναι `pegsol.exe` (μπορείτε φυσικά να το ονομάσετε όπως αλλιώς θέλετε), θέλετε να χρησιμοποιήσετε αναζήτηση πρώτα σε βάθος, το αρχείο εισόδου είναι το `input.txt` (ή ό,τι άλλο όνομα εσείς επιθυμείτε) και θέλετε η λύση να γραφεί στο αρχείο `solution.txt` (ή ό,τι άλλο όνομα εσείς επιθυμείτε), θα πρέπει να καλέσετε το πρόγραμμά σας με την εντολή:

```
pegsol.exe depth input.txt  solution.txt
```

Το πρόγραμμά σας θα υποστηρίζει μόνο προβλήματα με ορθογώνια πλέγματα θέσεων (όπως τα παραδείγματα 1 έως 5 της προηγούμενης σελίδας). Δεν θα υποστηρίζει προβλήματα με μη-ορθογώνια πλέγματα θέσεων, όπως είναι το εξαγωνικό πλέγμα θέσεων της εικόνας 6 της προηγούμενης σελίδας.

Τα περιεχόμενα του αρχείου input.txt για το παράδειγμα της εικόνας 1 της προηγούμενης σελίδας θα είναι τα εξής:

```
7 7
0 0 1 1 1 0 0
0 1 1 1 1 1 0
1 1 1 2 1 1 1
1 1 1 1 1 1 1
1 1 1 1 1 1 1
0 1 1 1 1 1 0
0 0 1 1 1 0 0
```

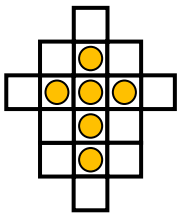
Στην πρώτη γραμμή οι δύο αριθμοί (έστω N και M) δηλώνουν το πλήθος των γραμμών και των στηλών του ταμπλό του παιχνιδιού. Στη συνέχεια ακολουθούν N γραμμές με M στήλες έκαστη, με αριθμούς που ερμηνεύονται ως εξής:

0: Η θέση αυτή δεν είναι έγκυρη (εκτός ταμπλό)

1: Η θέση αυτή είναι κατειλημμένη από ένα πάσσαλο

2: Η θέση αυτή είναι διαθέσιμη (ελεύθερη από πασσάλους)

Παρόμοια, η περιγραφή του παρακάτω πιο απλού προβλήματος:



είναι η εξής:

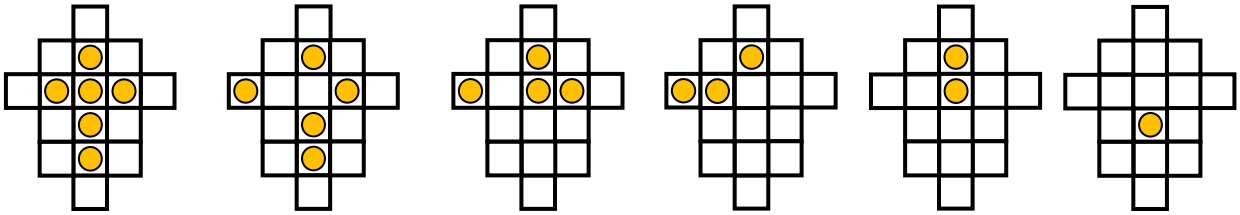
```
6 5
0 0 2 0 0
0 2 1 2 0
2 1 1 1 2
0 2 1 2 0
0 2 1 2 0
0 0 2 0 0
```

Σε σχέση με το αρχείο εξόδου solution.txt, το περιεχόμενό του θα πρέπει να είναι της παρακάτω μορφής (ακολουθεί ως παράδειγμα η λύση του τελευταίου προβλήματος):

```
5
3 3 3 1
5 3 3 3
3 4 3 2
3 1 3 3
2 3 4 3
```

Στην πρώτη γραμμή αναγράφεται το πλήθος των κινήσεων (πέντε στην προκειμένη περίπτωση). Κάθε μία από τις επόμενες γραμμές περιγράφει μια κίνηση. Οι δύο πρώτοι αριθμοί κάθε σειράς δίνουν τη θέση (X,Y)

του πασσάλου που θα μετακινηθεί και οι επόμενοι δύο αριθμοί (X', Y') δίνουν τη νέα θέση του μετακινούμενου πασσάλου, με (X', Y') = (X±2, Y) ή (X', Y') = (X, Y±2). Θεωρούμε ότι οι αρίθμηση των γραμμών X γίνεται από πάνω προς τα κάτω, με την πρώτη γραμμή να έχει τον αριθμό 1, ενώ η αρίθμηση των στηλών γίνεται από αριστερά προς τα δεξιά. Η λύση που δόθηκε παραπάνω αντιστοιχεί στην παρακάτω ακολουθία καταστάσεων:



Το πρόγραμμά σας μπορεί να τυπώνει περιορισμένης έκτασης μηνύματα στην οθόνη, όπως π.χ. το χρόνο που χρειάστηκε για να λύσει το πρόβλημα, ενδεχόμενα μηνύματα λάθους (π.χ. αδυναμία επίλυσης του προβλήματος μέσα σε συγκεκριμένα χρονικά όρια, π.χ. 5 mins κλπ).

Δώστε ιδιαίτερη προσοχή στον τρόπο κλήσης του προγράμματός σας, καθώς και στη μορφή των αρχείων εισόδου και εξόδου, σύμφωνα με όσα περιγράφηκαν παραπάνω, ώστε το πρόγραμμά (συμπεριλαμβανομένων των λύσεων που αυτό παράγει) να μπορεί να **ελεγχθεί αυτόματα**.

Προσοχή: Δεν χρειάζεται να ασχοληθείτε με την αποφυγή ατέρμονων βρόχων, μιας και η ίδια κατάσταση δεν μπορεί να εμφανιστεί δύο φορές σε ένα μονοπάτι (έλεγχος για βρόγχους).

Σημείωση: Στην άσκηση αυτή δεν σας ζητείται να υλοποιήσετε την αναζήτηση πρώτα σε πλάτος, μιας και αυτή δεν προσφέρει κάποιο πλεονέκτημα σε σχέση με την αναζήτηση πρώτα σε βάθος, αφού όλες οι λύσεις έχουν το ίδιο πλήθος κινήσεων (ίσο με το πλήθος των πασσάλων μείον ένα).

Θέμα 2ο : Αλγόριθμος πρώτα στο καλύτερο

Τροποποιείστε το πρόγραμμά σας ώστε να υποστηρίζει και τον αλγόριθμο αναζήτησης πρώτα στο καλύτερο (best first search). Σχεδιάστε μια ευρετική συνάρτηση για το συγκεκριμένο πρόβλημα. Η συνάρτησή σας θα πρέπει να λαμβάνει υπόψη διάφορους παράγοντες, όπως πόσοι πάσσαλοι απομένουν και τι χώρο καλύπτουν (π.χ., το πλάτος και το ύψος της ορθογώνιας περιοχής που καλύπτει όλους τους πασσάλους). Με δεδομένο ότι πάντα γνωρίζουμε πόσα βήματα απέχουμε από τη λύση (όσοι οι πάσσαλοι μείον ένα), το ερώτημα που καλείται να απαντήσει η ευρετική συνάρτηση είναι κατά πόσο υπάρχει λύση ξεκινώντας από την τρέχουσα κατάσταση. Προφανώς όσο λιγότεροι πάσσαλοι απομένουν και όσο μικρότερη είναι η περιοχή που αυτοί καλύπτουν, τόσο πιο πιθανό είναι να υπάρχει λύση και να βρισκόμαστε κοντά σε αυτή (λάβετε υπόψη το εμβαδό ή την περίμετρο της ελάχιστης ορθογώνιας περιοχής που περικλείει τα πασσάλοι – εναλλακτικά μπορείτε να υπολογίσετε το άθροισμα των αποστάσεων Manhattan κάθε πασσάλου από όλους τους υπόλοιπους και να προσθέσετε αυτά τα αθροίσματα).

Για την επιλογή του αλγορίθμου πρώτα στο καλύτερο το πρόγραμμά σας θα δέχεται ως είσοδο την λέξη `best`:

```
pegsol.exe best input.txt  solution.txt
```

Σημείωση: Στην άσκηση αυτή δεν σας ζητείται να υλοποιήσετε την αναζήτηση A*, μιας και αυτή δεν προσφέρει κάποιο πλεονέκτημα σε σχέση με την αναζήτηση πρώτα στο καλύτερο, αφού όλες οι λύσεις έχουν το ίδιο πλήθος κινήσεων (ίσο με το πλήθος των πασσάλων μείον ένα), ενώ πάντα γνωρίζουμε πόσο απέχουμε από τη λύση.

Θέμα 3ο : Πειραματική αξιολόγηση των αλγορίθμων

Δοκιμάστε τους δύο αλγορίθμους αναζήτησης που υλοποιήσατε σε διάφορα προβλήματα (συμπεριλαμβανομένων προβλημάτων διαφορετικών μεγεθών) που θα κατασκευάσετε μόνοι σας. Αναπτύξτε ένα πρόγραμμα επαλήθευσης των λύσεων σας. Συγκρίνετε τους χρόνους επίλυσης των δύο αλγορίθμων. Εκθέστε τα συμπεράσματά σας όσον αφορά (συγκριτικά) το χρόνο που απαιτείται από κάθε αλγόριθμο για να λύσει τα κάθε πρόβλημα. Υπάρχει σημαντικό κέρδος με τη χρήση της ευρετικής συνάρτησης;

Κριτήρια αξιολόγησης:

Αναζήτηση πρώτα σε βάθος	30
Αναζήτηση πρώτα στο καλύτερο	30
Υπολογιστική μελέτη	20
Γενική εικόνα (ευανάγνωστος κώδικας, σχολιασμός, ανάπτυξη προγράμματος ελέγχου λύσεων, κλπ.)	20
ΣΥΝΟΛΟ	**100**

Ο συνολικός βαθμός θα διαιρεθεί δια 100, ώστε να προκύψει ο τελικός βαθμός της εργασίας (με μέγιστη τιμή το 1).

Οδηγίες υποβολής: Η εργασία θα πρέπει να υποβληθεί μέσω του Classroom. Θα υποβάλλετε ένα έγγραφο κειμένου με περιγραφή των πειραμάτων σας και των αποτελεσμάτων σας, καθώς και τα αρχεία κώδικα και προβλημάτων που υλοποιήσατε/χρησιμοποιήσατε. Ο κώδικας θα πρέπει να είναι ευανάγνωστος και σχολιασμένος.