

**Τεχνητή Νοημοσύνη**  
**Εαρινό Εξάμηνο 2021**  
**Διδάσκων: Α. Λύκας**

- Ομάδες **δύο ή τριών** (κατά προτίμηση) **φοιτητών**.
- Φοιτητές που έχουν βαθμολογηθεί σε προηγούμενα έτη δεν δικαιούνται επανεξέτασης.
- Γλώσσα προγραμματισμού: **C ή Java** (όχι Python)
- Δεκτές για εξέταση γίνονται μόνο ασκήσεις που είναι ολοκληρωμένες, δηλ. τα προγράμματα μεταγλωττίζονται και εκτελούνται στους υπολογιστές του Τμήματος π.χ. opti3060ws03. Μην υποβάλετε κώδικα Java που απαιτεί το περιβάλλον eclipse για να μεταγλωττιστεί και να εκτελεστεί.
- **Δήλωση ομάδων:** e-mail στον διδάσκοντα με (ΑΜ, ονοματεπώνυμο) μελών της ομάδας μέχρι **25 Απριλίου 2021** (αυστηρή προθεσμία).
- Η δήλωση συνεπάγεται υποχρέωση υποβολής και εξέτασης των εργασιών.
- **Προθεσμία υποβολής εργασιών: 16 Μαΐου 2021** (αυστηρή προθεσμία).
- Η **υποβολή** θα γίνει με χρήση της εντολής **turnin** ως εξής:  
turnin assignment@myy602 <your\_filename>
- Θα δημιουργήσετε δύο καταλόγους, έναν για κάθε άσκηση. Κάθε κατάλογος θα περιλαμβάνει τον πηγαίο, τον εκτελέσιμο κώδικα της άσκησης και αρχείο κειμένου (pdf) (για την άσκηση 1). Φυσικά θα πρέπει να υπάρχει πληροφορία για τα ονόματα και τα ΑΜ των μελών της ομάδας.
- Μην υποβάλετε συμπιεσμένα αρχεία .rar

### **Εργαστηριακή Άσκηση 1 (Πρόβλημα Αναζήτησης) (20%)**

Να κατασκευάσετε πρόγραμμα για την επίλυση του ακόλουθου προβλήματος **αναζήτησης**:

Θεωρούμε μία λίστα  $N$  θέσεων στην οποία έχουν τοποθετηθεί οι φυσικοί αριθμοί από 1 έως  $N$  (μία φορά ο καθένας) (πχ. [1,3,2,4,5], [2,1,4,5,3] κλπ για  $N=5$ ). Επομένως **ο χώρος καταστάσεων αντιστοιχεί στις μεταθέσεις των  $N$  φυσικών αριθμών (από 1 έως  $N$ )**. Η **τελική κατάσταση είναι η κατάσταση [1,2,...,N]** (δηλ. οι αριθμοί είναι τοποθετημένοι σε αύξουσα σειρά).

Η αρχική κατάσταση καθορίζεται από τον χρήστη στην αρχή της εκτέλεσης του προγράμματος.

Το  $N$  ορίζεται μέσα στο πρόγραμμα ή δίνεται από τον χρήστη στην αρχή της εκτέλεσης του προγράμματος.

Οι **τελεστές μετάβασης** (δηλ. οι επιτρεπτές ενέργειες σε κάθε κατάσταση) ορίζονται ως εξής:

Έστω  $a=[a_1, \dots, a_N]$  μια κατάσταση. Η εφαρμογή του τελεστή  $T(k)$  (όπου  $2 \leq k \leq N$ ) στην κατάσταση  $a$  οδηγεί σε μια κατάσταση  $b=[b_1, \dots, b_N]$  που προκύπτει ως εξής: η  $a$  χωρίζεται σε δύο τμήματα:  $a_L=[a_1, \dots, a_k]$  (που περιλαμβάνει τα  $k$  πρώτα στοιχεία της  $a$ ) και  $a_R=[a_{k+1}, \dots, a_N]$  (που περιλαμβάνει τα υπόλοιπα στοιχεία της  $a$ ). Αν  $k=N$  τότε  $a_L=a$  και  $a_R=[]$  (κενή λίστα). Έστω  $g_{a_L}=[a_k, \dots, a_1]$  η λίστα που προκύπτει αντιστρέφοντας τη σειρά τοποθέτησης των στοιχείων της  $a_L$ . Η κατάσταση  $b=[b_1, \dots, b_N]$  ορίζεται από τη συνένωση της  $g_{a_L}$  και της  $a_R$ .

Για παράδειγμα: η εφαρμογή του τελεστή  $T(2)$  στην κατάσταση  $a=[3,5,4,1,2]$  οδηγεί στην κατάσταση  $b=[5,3,4,1,2]$ , η εφαρμογή του τελεστή  $T(4)$  στην κατάσταση  $a=[3,5,4,1,2]$  οδηγεί στην κατάσταση  $b=[1,4,5,3,2]$  και η εφαρμογή του τελεστή  $T(5)$  στην κατάσταση  $a=[3,5,4,1,2]$  οδηγεί στην κατάσταση  $b=[2,1,4,5,3]$ . Προσέξτε ότι ο τελεστής  $T(1)$  δίνει  $b=a$ , γι' αυτό δεν επιτρέπεται αφού δεν οδηγεί σε διαφορετική κατάσταση. **Όλες οι μεταβάσεις έχουν κόστος 1.**

Δοθείσης μια οποιαδήποτε αρχικής κατάστασης (ΑΚ) που καθορίζει ο χρήστης, θέλουμε το πρόγραμμα να βρίσκει την βέλτιστη ακολουθία ενεργειών για να φτάσουμε στην τελική κατάσταση ( $TK=[1,2,\dots,N]$ ).

Για το παραπάνω πρόβλημα να υλοποιήσετε:

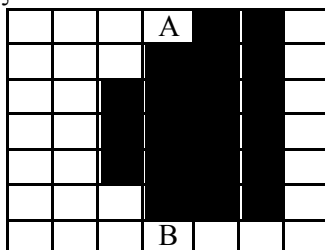
- i) αναζήτηση ομοίομορφου κόστους (UCS) (10%)
- ii) αναζήτηση  $A^*$  χρησιμοποιώντας όσο το δυνατόν καλύτερη **αποδεκτή ευρετική** συνάρτηση  $h(n)$ . Θα πρέπει να εξηγήσετε γραπτώς (σε έγγραφο κειμένου report.pdf) γιατί η συνάρτηση  $h(n)$  που σκεφτήκατε είναι αποδεκτή. (10%)

Να εξετάσετε διάφορες τιμές του  $N$  καθώς και διαφορετικές  $AK$ . Για κάθε  $N$  και  $AK$  να εφαρμόζετε τόσο την μέθοδο UCS όσο και την μέθοδο  $A^*$  για να μπορείτε να συγκρίνετε τις μεθόδους. Πιο συγκεκριμένα, το πρόγραμμα πρέπει να τυπώνει: α) το μονοπάτι, β) το κόστος του μονοπατιού αυτού, και γ) τον αριθμό των επεκτάσεων που έγιναν. Να αναφέρετε στο κείμενο (report.pdf) τα συμπεράσματά σας σχετικά με την αποδοτικότητα της  $A^*$  σε σχέση με τη UCS.

### Εργαστηριακή Άσκηση 2 (Κατασκευή Παιγνίου) (10%)

Να κατασκευάσετε πρόγραμμα το οποίο θα παίζει ενάντια σε κάποιο χρήστη το εξής παίγνιο δύο παικτών που παίζουν εναλλάξ.

Το παίγνιο παίζεται σε πίνακα με  $M \times N$  θέσεις. Κάθε παίκτης έχει ένα πόνι το οποίο καταλαμβάνει ένα τετράγωνο στον πίνακα. Στο παρακάτω σχήμα έχουμε έναν πίνακα  $7 \times 7$  όπου το πόνι A (παίκτης MAX) καταλαμβάνει το μεσαίο τετράγωνο της πρώτης σειράς, ενώ το πόνι B (παίκτης MIN) καταλαμβάνει το μεσαίο τετράγωνο της τελευταίας σειράς.



**Τα μαύρα τετράγωνα** είναι αυτά τα οποία έχουν ήδη **επισκεφτεί ή διατρέξει** τα πόνια και δεν μπορούν τα πόνια να ξαναπεράσουν από αυτά.

Κάθε παίκτης μπορεί να κινήσει το πόνι του, υπό προϋποθέσεις, όπως μία βασίλισσα στο σκάκι: οριζόντια, κατακόρυφα και διαγώνια. Μπορείτε να βάλετε όριο στην μετακίνηση (π.χ. μέχρι δύο θέσεις).

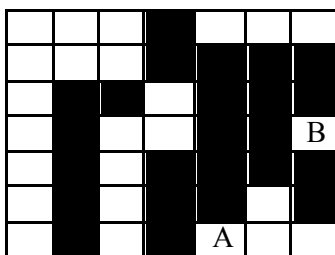
Οι προϋποθέσεις για την κίνηση ενός πιονιού είναι οι εξής:

- Ένα πόνι μπορεί να διατρέξει μόνο λευκά τετράγωνα και να τοποθετηθεί μόνο σε λευκό τετράγωνο.
- Ένα πόνι δεν μπορεί να τοποθετηθεί ή να περάσει (κατά την κίνηση του) από το τετράγωνο στο οποίο είναι τοποθετημένο το πόνι του αντιπάλου.

**Όταν ένα πόνι κάνει μια κίνηση, το τετράγωνο από το οποίο ξεκινά και τα τετράγωνα τα οποία διατρέχει γίνονται μαύρα.**

**Χάνει ο παίκτης που δεν έχει καμία επιτρεπτή κίνηση όταν είναι η σειρά του να παίζει.**

Στο παρακάτω παράδειγμα το πόνι A έχει τρεις επιτρεπτές κινήσεις: (α) μία θέση διαγώνια (πάνω-δεξιά), (β) μία θέση δεξιά και (γ) δύο θέσεις δεξιά. Το πόνι B δεν έχει καμία επιτρεπτή κίνηση. Κατά συνέπεια, ο MAX (πόνι A) κερδίζει.



Θεωρείστε ότι 'παίκτης MAX' = πρόγραμμα (πόνι A), 'παίκτης MIN' = αντίπαλος (πόνι B) και ότι παίζει πρώτος ο MAX. Το μέγεθος  $M \times N$  του ταμπλό να είναι παράμετρος του προγράμματος (π.χ.  $M=N=4$ ).

Αφού πρώτα ορίσετε κατάλληλες τιμές για την αξία των τελικών καταστάσεων, να κατασκευάσετε το πρόγραμμα εκτέλεσης του παιγνίου στο οποίο ο **MAX** πρέπει να παίζει βέλτιστα εκτελώντας τον αλγόριθμο **MINIMAX** με ρίζα την τρέχουσα κατάσταση για να αποφασίσει για την κίνηση που θα κάνει κάθε φορά. (Η υλοποίηση του **MINIMAX** να γίνει με τη χρήση αναδρομής).

Στην αρχή του προγράμματος πρέπει να καθορίζονται οι θέσεις A και B, καθώς επίσης (προαιρετικά) και κάποιες θέσεις που θέλουμε εξαρχής να είναι μαύρες (για να μειώνονται οι δυνατές κινήσεις των πιονιών).

Στην αρχή του κώδικα του προγράμματος να αναφέρετε τυχόν υποθέσεις που κάνατε για τις επιτρεπτές κινήσεις των πιονιών.