Τεχνητή Νοημοσύνη
Εαρινό Εξάμηνο 2022
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 **23 Απριλίου** 2022 (αυστηρή προθεσμία).
- Η δήλωση συνεπάγεται υποχρέωση υποβολής και εξέτασης των εργασιών.
- **Προθεσμία υποβολής εργασιών: 22 Μαΐου 2022** (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 (Αναζήτηση σε λαβύρινθο) (20%)
Να κατασκευάσετε πρόγραμμα αναζήτησης για την επίλυση του ακόλουθου προβλήματος **πλοήγησης ρομπότ σε λαβύρινθο**:
Ορισμός λαβύρινθου: Θεωρούμε ένα πίνακα με NxN κελιά, κάποια από τα οποία είναι ελεύθερα, ενώ τα υπόλοιπα περιέχουν εμπόδια και δεν μπορούμε να τα επισκεφθούμε. Κάθε κελί (x,y) χαρακτηρίζεται ως ελεύθερο ή όχι αποφασίζοντας ανεξάρτητα με πιθανότητα p (π.χ p=0.1). Τα N και p καθορίζονται στην αρχή του προγράμματος. Σε κάθε **ελεύθερο** κελί (x,y) αναθέτουμε ένα αριθμό (που ονομάζεται **val(x,y)**) ο οποίος επιλέγεται τυχαία από τους ακεραίους 1 έως 4.
Μετακίνηση ρομπότ: Εστω ότι το ροπμπότ βρίσκεται στο κελί (x,y). Τότε μπορεί κάθε φορά να μετακινείται **είτε οριζόντια είτε κατακόρυφα είτε διαγώνια** σε ένα **γειτονικό ελεύθερο** κελί (x',y'). Εστω Δ=|val(x,y)-val(x',y')|. Το κόστος της μετακίνησης είναι ίσο με Δ+1 για την ορίζοντια και τη κατακόρυφη μετακίνηση και ίσο με Δ+0.5 για την διαγώνια μετακίνηση.

Θέλουμε να βρούμε τη **διαδρομή** (εάν υπάρχει) που πρέπει να ακολουθήσει το ρομπότ ξεκινώντας από ένα αρχικό κελί (S) ώστε να φτάσει στο **πλησιέστερο (με βάση το κόστος διαδρομής)** από δύο συγκεκριμένα τελικά κελιά G1 και G2. Οι συντεταγμένες (x,y) των κελιών S, G1 και G2 δίνονται από τον χρήστη στην αρχή του προγράμματος.
Για το παραπάνω πρόβλημα να υλοποιήσετε:

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

Κάθε μέθοδος (UCS ή A*) θα πρέπει να εκτελείται **μία φορά** και να καταλήγει στην πλησιέστερη τελική κατάσταση G1 ή G2. Δεν επιτρέπεται η εκτέλεση του αλγορίθμου μία φορά με τελική κατάσταση την G1 και μία φορά με τελική κατάσταση την G2.
Για κάθε λαβύρινθο που θα ορίσετε, να εφαρμόζετε τόσο την μέθοδο UCS όσο και την μέθοδο A* για να μπορείτε να συγκρίνετε τις μεθόδους. Πιο συγκεκριμένα, για κάθε μέθοδο να τυπώνετε: α) τον λαβύρινθο, τις τιμές val(x,y) κάθε ελεύθερου κελιού, τα κελιά S, G1 και G2 καθώς και το μονοπάτι που βρήκατε, 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.

Να εξετάσετε διάφορες τιμές του N και του p.

Εργαστηριακή Ασκηση 2 (Κατασκευή Παιγνίου) (10%)
Να κατασκευάσετε πρόγραμμα το οποίο θα παίζει ενάντια σε κάποιο χρήστη ένα παίγνιο δύο παικτών **(connect-4)** που παίζουν εναλλάξ (για τον ορισμό του παιγνίου δείτε: http://www.papg.com/show?2XLU).

Το παίγνιο παίζεται σε πίνακα με M γραμμές και N στήλες. Ο πίνακας αρχικά είναι κενός. **Ο παίκτης MAX (πρόγραμμα) τοποθετεί 'X' στον πίνακα σύμφωνα με τους κανόνες του παιγνίου, ενώ ο παίκτης MIN (χρήστης) τοποθετεί 'O' σύμφωνα με τους κανόνες του παιγνίου**. Κερδίζει ο παίκτης που θα τοποθετήσει K συνεχόμενα 'X' (MAX) ή 'O' (MIN) στην ίδια σειρά, στήλη ή διαγώνιο. Τα M, N, K καθορίζονται στην αρχή του προγράμματος.

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. Δεν απαιτείται κλάδεμα α-β).

Το τυπικό παίγνιο ορίζεται για M=6, N=7, K=4. Ωστόσο μπορείτε να χρησιμοποιήσετε και μικρότερες τιμές για να μειώσετε την πολυπλοκότητα του προβλήματος.