Αρχές Γλωσσών Προγραμματισμού

Άσκηση 3
Santorini

Ημερομηνία ανάρτησης: 13/1/2021
Ημερομηνία παράδοσης: 20/2/2021

Εισαγωγή

Το κύριο ζητούμενο της άσκησης είναι η υλοποίηση του “Santorini”.

Το Santorini είναι ένα επιτραπέζιο παιχνίδι που παίζεται σε ένα ταμπλό 5 επί 5. Για την άσκηση, θα θεωρήσουμε πως υπάρχουν δυο παίκτες, αν και το παιχνίδι κανονικά μπορεί να έχει και περισσότερους. Έστω για απλότητα πως αυτοί είναι ο μπλε και ο κόκκινος και πως ο μπλε παίζει πρώτος.

Αρχικά ο μπλε τοποθετεί δύο πιόνια όπου θέλει στο ταμπλό. Το ίδιο κάνει αμέσως μετά ο κόκκινος. Στην συνέχεια εναλλάξ μετακινούν ένα από τα πιόνια τους σε ένα από τα οχτώ γειτονικά τετράγωνα και προσθέτουν έναν “όροφο” σε ένα τετράγωνο που είναι γειτονικό σε αυτό που κατέληξε το πιόνι.

Σε κάθε τετράγωνο επιτρέπεται να υπάρχει ένα το πολύ πιόνι ανά πάσα στιγμή. Τα πιόνια μπορούν να “κατέβουν” όσους ορόφους θέλουν σε μία μετακίνηση, αλλά μπορούν να “ανέβουν” το πολύ έναν όροφο.
Δηλαδή, μπορούν να μεταβούν από ένα τετράγωνο που έχει 2 ορόφους σε κάποιο που έχει 0, αλλά όχι το αντίστροφο. Δεν υπάρχει ανάλογος περιορισμός για το που θα χτίσει κάποιο πιόνι.

Οταν κάποιος παίκτης χτίζει πάνω σε ένα τετράγωνο με τρεις ορόφους, το τετράγωνο αυτό θεωρείται ανενεργό για την συνέχεια του παιχνιδιού, υπό την έννοια πως δεν μπορεί πλέον κανένα πιόνι να μεταβεί ή να χτίσει σε αυτό.
Ένας παίκτης κερδίζει εάν μετακινήσει ένα πιόνι του σε ένα τετράγωνο με τρεις ορόφους. Αν κάποιος παίκτης αδυνατεί να μετακινήσει κάποιο πιόνι του και να χτίσει όροφο, χάνει.

Εσωτερική αναπαράσταση – Οπτικοποίηση

Για να βοηθηθείτε στην υλοποίηση και αποσφαλμάτωση του προγράμματός σας, έχει αναπτυχθεί το τμήμα του κώδικα που θα οπτικοποιεί το παιχνίδι. Θα μπορείτε δηλαδή, εφ'όσον έχετε μία στοιχειωδώς λειτουργική έκδοση των ζητούμενων να παίξετε κανονικά το παιχνίδι σε γραφικό περιβάλλον.

Για να μπορεί να λειτουργήσει σωστά η οπτικοποίηση βέβαια, θα πρέπει το πρόγραμμα σας να έχει την δυνατότητα να πληροφορεί για την κατάσταση του παιχνιδιού με βάση ένα απλό πρωτόκολλο επικοινωνίας που θα οριστεί παρακάτω.

Ζητούμενος τύπος δεδομένων

Καλείστε να ορίσετε τον τύπο δεδομένων **Game**, ο οποίος θα περιέχει τις πληροφορίες που θα κρίνετε εσείς οτι είναι απαραίτητες για την αναπαράσταση της κατάστασης του παιχνιδιού ανά πάσα στιγμή. Φυσικά θα πρέπει να μπορείτε μέσω αυτού να παράξετε τα ζητούμενα δεδομένα για την οπτικοποίηση.

Ζητούμενες συναρτήσεις

Για να είναι πιό ξεκάθαρες οι περιγραφές των συναρτήσεων, θα οριστούν εδώ κάποια συνώνυμα τύπων, τα οποία εννοείται πως μπορείτε να εκμεταλλευτείτε στην υλοποίηση σας, χωρίς αυτό να είναι απαραίτητο βέβαια.

```
type RowPos = Int
type ColumnPos = Int
type Position = (RowPos, ColumnPos)

type PlayerPositions = (Position, Position)
type BluePlayerPositions = PlayerPositions
type RedPlayerPositions = PlayerPositions

type Height = Int
type Building = (Height, Position)
type BuildingsList = [Building]

type Turn = Char
type Depth = Int
```

Μοντελοποίηση παιχνιδιού

1. **initializeGame** ::
BluePlayerPositions → RedPlayerPositions → Game (20%)

Στην συνάρτηση αυτή θα δίνονται οι αρχικές θέσεις που θα έχουν τα πιόνια των παικτών στην μορφή ((Int, Int), (Int, Int)). Θα πρέπει να γίνεται κατάλληλη αρχικοποίηση του τύπου δεδομένου **Game** που έχετε ορίσει και να επιστρέφεται το δημιουργημένο αντικείμενο. Οι θέσεις είναι στην μορφή (row, col). Η θέση (0, 0) αναφέρεται στο πάνω αριστερά τετράγωνο και η (4, 4) στο κάτω δεξιά.

2. **tryMove** :: Game → (Position, Position, Position) → Game (40%)

Εδώ το πρώτο **Position** είναι αυτό στο οποίο βρίσκεται το πιόνι το οποίο πρόκειται να μετακινηθεί. Το δεύτερο αναφέρεται στο τετράγωνο στο οποίο θα καταλήξει το πιόνι. Τέλος, η τρίτη θέση είναι αυτή στην οποία θα χτιστεί όροφος. Η συνάρτηση θα πρέπει να εφαρμόζει την κίνηση που ορίζεται αν και μόνο αν η κίνηση αυτή είναι "νόμιμη" με βάση τους κανόνες του παιχνιδιού. Τελικά επιστρέφεται το **Game** που προκύπτει. Αν η κίνηση είναι "παράνομη", επιστρέφεται το **Game** που δόθηκε ως είσοδος.

Νόμιμη είναι μία κίνηση όταν:

- Κινείται ένα από τα πιόνια του παίκτη που παίζει σε αυτόν τον γύρο.
- Το τετράγωνο στο οποίο κινείται το πιόνι θα πρέπει να είναι **υποχρεωτικά** ένα από τα οχτώ γειτονικά του αρχικού. Δεν μπορεί δηλαδή να μείνει στάσιμο. Επίσης, δεν θα πρέπει να καταλαμβάνεται από άλλο πιόνι ή να είναι ανενεργό.
- Οι παραπάνω περιορισμοί ισχύουν και για το τρίτο **Position**, σε σχέση με το δεύτερο.
- Το πιόνι που κινείται δεν ανεβαίνει πάνω από έναν όροφο. Δηλαδή, αν το αρχικό τετράγωνο είχε "υψος" x, το τετράγωνο στο οποίο θα καταλήξει πρέπει να έχει ύψος το πολύ $x + 1$.

Υπενθυμίζεται πως κάποιος παίκτης μπορεί να κερδίσει ανεβάζοντας ένα από τα πιόνια του σε κτήριο με τρεις ορόφους. Επίσης, κερδίζει εάν με την κίνησή του "εγκλωβίσει" τον αντίπαλο, υπό την έννοια πως εκείνος δεν θα έχει καμμία νόμιμη κίνηση στην συνέχεια. Θα πρέπει να ελέγχετε σε κάθε κίνηση αν κάποιος παίκτης έχει κερδίσει και να ενημερώνετε την δομή σας κατάλληλα.

3. **screenshotGame** :: Game →
(Bool, Turn, BluePlayerPositions, RedPlayerPositions, BuildingsList) (10%)

Η συγκεκριμένη συνάρτηση απαιτείται για να λειτουργεί η οπτικοποίηση αλλά και για να αξιολογηθεί η ορθότητα της υλοποίησής σας.
Εξάγονται από το παιχνίδι και επιστρέφονται οι θέσεις των πιονιών, μία λίστα με τις περιγραφές των κτηρίων, καθώς και το αναγνωριστικό του παίκτη που έχει σειρά να παίξει. Αν παίζει ο μπλε τότε επιστρέφεται 'B', αλλιώς 'R'.
Για κάθε κτήριο (τετράγωνο με τουλάχιστον έναν όροφο), θα πρέπει να υπάρχει στην λίστα μία δυάδα (Int, (Int, Int)). Το πρώτο στοιχείο της δυάδας αντιπροσωπεύει το "ύψος" του κτηρίου, δηλαδή το πλήθος των ορόφων του. Αν ένα τετράγωνο είναι ανενεργό, το ύψος του είναι 4. Το δεύτερο στοιχείο είναι η θέση του τετραγώνου στο οποίο αναφερόμαστε.
Αν το παιχνίδι έχει τελειώσει, η τιμή Bool θα είναι True και το αναγνωριστικό το οποίο επιστρέφεται θα αναφέρεται στον παίκτη ο οποίος κέρδισε.

4. **undoMove** :: Game → Game (20%)

Η συνάρτηση αυτή θα πρέπει να αναιρεί τις αλλαγές που έγιναν από την τελευταία κίνηση που εφαρμόσθηκε (είτε μέσω της **tryMove** είτε μέσω της **redoMove**) στο **Game** που δίνεται ως είσοδος. Θα πρέπει να μπορεί να καλείται πολλές φορές διαδοχικά, ακόμα και μέχρι να φτάσουμε στην αρχική κατάσταση που προκύπτει από την **initializeGame**. Αν στην είσοδο δεν υπάρχει κάποια κίνηση να αναιρέσουμε, απλά επιστρέφουμε την είσοδο.

5. **redoMove** :: Game → Game (10%)

Αυτή η συνάρτηση θα πρέπει να μπορεί να αναιρεί τα αποτελέσματα της πιό πρόσφατης **undoMove** που έχει εκτελεστεί (και δεν έχει γίνει redo) πάνω στο **Game** εισόδου. Θα πρέπει να μπορεί να εκτελείται όσες φορές έχει εκτελεστεί και η **undoMove**. Αν όμως μετά από την πιό πρόσφατη κλήση της **undoMove**, έχει **εφαρμοστεί** (και όχι αγνοηθεί λόγω μη νομιμότητας) κάποια κίνηση μέσω της **tryMove**, τότε η **redoMove** δεν απαιτείται να κάνει κάτι.

Στην ουσία οι δύο παραπάνω συναρτήσεις υλοποιούν τις λειτουργίες των control + Z και control + Y. Έτσι θα μπορείτε να τις χρησιμοποιείτε και στο γραφικό περιβάλλον, απλά χωρίς το control, για τεχνικούς λόγους. Εδώ ίσως φανούν χρήσιμα τα "Zippers", που περιγράφονται εδώ.

Υποστήριξη παίκτη τεχνητής νοημοσύνης (Bonus 20%)

Στον κώδικα που σας δίνεται παρέχεται και η δυνατότητα να παίξει κανείς εναντίον του υπολογιστή. Έχει υλοποιηθεί δηλαδή μία έκδοση του αλγόριθμου Minimax, την οποία καλείστε να υποβοηθήσετε. Τα κομμάτια που χρειάζεται να υλοποιήσετε εσείς φαίνονται παρακάτω.

6. **possibleMoves** :: Game → [(Position, Position, Position)] (Bonus 10 %)

Η συνάρτηση αυτή θα επιστρέφει όλες τις νόμιμες κινήσεις που μπορεί να κάνει ο παίκτης που έχει σειρά στο παιχνίδι εισόδου. Αυτή η συνάρτηση καλείται από τον αλγόριθμο Minimax ούτως ώστε να μπορέσει να "δημιουργήσει" το δένδρο αναζήτησης. Η πολυπλοκότητα της συνάρτησης επηρεάζει αυτή του Minimax, οπότε προσπαθήστε να την υλοποιήσετε όσο πιο αποδοτικά μπορείτε.

7. **evaluateState** :: Turn → Game → Int (Bonus 10%)

Αυτή θα είναι μία ευρετική συνάρτηση η οποία θα αξιολογεί σε πόσο καλή θέση βρίσκεται ο παίκτης που ορίζεται από την είσοδο, δηλαδή πόσο πιθανό είναι να κερδίσει από εδώ και πέρα. Αν για παράδειγμα ζητηθεί να αξιολογήσουμε την κατάσταση για τον μπλε παίκτη και αυτός έχει ένα πιόνι το οποίο μπορεί σε μία κίνηση να ανέβει σε κτήριο με τρεις ορόφους, τότε θα πρέπει να επιστραφεί μία αρκετά υψηλή τιμή, αφού στην ουσία έχει κερδίσει ήδη.
Όπως καταλαβαίνετε αυτή θα είναι η λογική βάσει της οποίας θα λειτουργεί ο AI παίκτης. Αν και ο σχεδιασμός ευρετικής συνάρτησης επιδέχεται φαντασίας, ενδεικτικά θα μπορούσατε να λαμβάνετε υπ'όψιν το ύψος στο οποίο βρίσκονται τα πιόνια του παίκτη, πόσα και πόσο ψηλά κτήρια έχει κοντά του, τις αντίστοιχες πληροφορίες για τον αντίπαλο και ούτω καθεξής.
Σκοπός εδώ είναι να βρεθεί μία ισορροπία μεταξύ ακρίβειας και ταχύτητας υπολογισμού. Θεωρητικά θα μπορούσε δηλαδή κανείς μέσα στην συνάρτηση να κάνει εξαντλητική αναζήτηση για όλα τα πιθανά σενάρια. Αυτό όμως είναι προφανές πως δεν είναι δυνατό να υπολογιστεί σε λογικά χρονικά πλαίσια. Από την άλλη, μία πολύ απλή και γρήγορη ευρετική συνάρτηση ίσως να μην προσεγγίζει ικανοποιητικά την πιθανότητα να κερδίσει κάποιος.
Για να πάρει κανείς το πλήρες bonus του ερωτήματος, θα πρέπει η ευρετική να μην είναι πολύ βαριά, δηλαδή να τρέχει ο Minimax σε 1-2 λεπτά το πολύ για βάθος 3. Επίσης, θα πρέπει η συνάρτηση να μην είναι υπερβολικά απλή. Εάν χρησιμοποιήσετε για παράδειγμα τις πληροφορίες που ανέφερα με όχι τετριμμένο τρόπο, είστε καλυμμένοι.

Πως να παίξετε το παιχνίδι

Για να παίξετε το παιχνίδι, θα πρέπει πρώτα να μεταγλωττίσετε το πρόγραμμα με την εντολή “ghc visualization”. Είναι απαραίτητη η βιβλιοθήκη gloss, την οποία μπορείτε να εγκαταστήσετε μέσω του cabal. Το cabal περιέχεται στο haskell-platform. Για την εγκατάσταση του gloss εκτελείτε απλά “cabal install gloss”.

Στην συνέχεια, χρειάζεται απλά να τρέξετε το εκτελέσιμο (./visualization). Αρχικά κάνετε 4 διαδοχικά κλικ που ορίζουν που θα πάνε τα πιόνια (την είσοδο της **initializeGame** δηλαδή). Έπειτα κάθε τριάδα από κλικ ορίζει και μία κίνηση που δίνεται στην **tryMove**. Πατώντας τα z, y μπορείτε να κάνετε undo και redo αντίστοιχα, εφ’όσον έχετε υλοποιήσει τις αντίστοιχες συναρτήσεις.
Πατώντας το a μπορείτε να εναλλάσετε το παιχνίδι μεταξύ ανθρώπου εναντίον ανθρώπου και ανθρώπου εναντίον μηχανής. Μην απογοητευτείτε εάν ο AI παίκτης δεν τα πάει τόσο καλά, καθώς -τουλάχιστον με την υλοποίηση του Minimax που παρέχεται- το βάθος αναζήτησης δεν μπορεί να είναι πολύ μεγάλο. Συγκεκριμένα, η default τιμή που έχει δοθεί είναι 3. Με το + μπορείτε να αυξήσετε το βάθος και με το – να το μειώσετε.

Διαγωνισμός καλύτερης ευρετικής

Μετά την βαθμολόγηση των εργασιών, θα γίνει (για όσους το επιθυμούν) ένας διαγωνισμός στον οποίο στην ουσία θα συγκρίνονται οι συναρτήσεις **evaluateState**.
Συγκεκριμένα, ένας “αγώνας” θα γίνεται ως εξής. Τις αρχικές θέσεις που θα έχουν τα πιόνια θα τις επιλέγουν οι ομάδες. Η κίνηση των παικτών θα ορίζεται από τον αλγόριθμο Minimax με βάση την ευρετική συνάρτηση της κάθε υποβολής. Θα παιχτούν από δύο παιχνίδια (με διαφορετικό πρώτο παίκτη κάθε φορά) για βάθος 2 και 3. Η υποβολή με τις περισσότερες νίκες κερδίζει. Σε περίπτωση ισοπαλίας θα γίνει κλήρωση για το ποιά ομάδα θα παίξει πρώτη και θα παιχτεί ακόμη ένα παιχνίδι με βάθος 3 ή 4 αν οι χρόνοι είναι λογικοί.
Η γενικότερη δομή του διαγωνισμού θα ορίστει βάσει του πλήθους των συμμετέχοντων.

1η θέση : Bonus 10%
2η θέση : Bonus 8%
3η θέση : Bonus 5%

Μερικά παραδείγματα εκτέλεσης

*StudentCode> game1 = initializeGame ((0, 0), (1, 1)) ((2, 2),(3, 3))

*StudentCode> screenshotGame game1
(False,'B',((0,0),(1,1)),((2,2),(3,3)),[])

--νόμιμη κίνηση
*StudentCode> game2 = tryMove game1 ((0, 0), (1, 0), (2, 0))

--αλλαγή κατάστασης
*StudentCode> screenshotGame game2
(False,'R',((1,0),(1,1)),((2,2),(3,3)),[(1,(2,0))])

--παράνομη κίνηση καθώς πάει να μετακινηθεί πάνω σε πιόνι (στο (1, 1))
*StudentCode> game3 = tryMove game2 ((1, 0), (1, 1), (1, 2))

--καμία αλλαγή στην κατάσταση
*StudentCode> screenshotGame game3
(False,'R',((1,0),(1,1)),((2,2),(3,3)),[(1,(2,0))])

*StudentCode> game4 = undoMove game3

*StudentCode> screenshotGame game4
(False,'B',((0,0),(1,1)),((2,2),(3,3)),[])

*StudentCode> game5 = redoMove game4

*StudentCode> screenshotGame game5
(False,'R',((1,0),(1,1)),((2,2),(3,3)),[(1,(2,0))])

```
*StudentCode> possibleMoves game5
[((2,2),(1,2),(0,1)),((2,2),(1,2),(0,2)),((2,2),(1,2),(0,3)),((2,2),(1,2),(1,3)),((2,2),(1
,2),(2,1)),((2,2),(1,2),(2,2)),((2,2),(1,2),(2,3)),((2,2),(1,3),(0,2)),((2,2),(1,3),(0,3))
,((2,2),(1,3),(0,4)),((2,2),(1,3),(1,2)),((2,2),(1,3),(1,4)),((2,2),(1,3),(2,2)),((2,2),(1
,3),(2,3)),((2,2),(1,3),(2,4)),((2,2),(2,1),(1,2)),((2,2),(2,1),(2,0)),((2,2),(2,1),(2,2))
,((2,2),(2,1),(3,0)),((2,2),(2,1),(3,1)),((2,2),(2,1),(3,2)),((2,2),(2,3),(1,2)),((2,2),(2
,3),(1,3)),((2,2),(2,3),(1,4)),((2,2),(2,3),(2,2)),((2,2),(2,3),(2,4)),((2,2),(2,3),(3,2))
,((2,2),(2,3),(3,4)),((2,2),(3,1),(2,0)),((2,2),(3,1),(2,1)),((2,2),(3,1),(2,2)),((2,2),(3
,1),(3,0)),((2,2),(3,1),(3,2)),((2,2),(3,1),(4,0)),((2,2),(3,1),(4,1)),((2,2),(3,1),(4,2))
,((2,2),(3,2),(2,1)),((2,2),(3,2),(2,2)),((2,2),(3,2),(2,3)),((2,2),(3,2),(3,1)),((2,2),(3
,2),(4,1)),((2,2),(3,2),(4,2)),((2,2),(3,2),(4,3)),((3,3),(2,3),(1,2)),((3,3),(2,3),(1,3))
,((3,3),(2,3),(1,4)),((3,3),(2,3),(2,4)),((3,3),(2,3),(3,2)),((3,3),(2,3),(3,3)),((3,3),(2
,3),(3,4)),((3,3),(2,4),(1,3)),((3,3),(2,4),(1,4)),((3,3),(2,4),(2,3)),((3,3),(2,4),(3,3))
,((3,3),(2,4),(3,4)),((3,3),(3,2),(2,1)),((3,3),(3,2),(2,3)),((3,3),(3,2),(3,1)),((3,3),(3
,2),(3,3)),((3,3),(3,2),(4,1)),((3,3),(3,2),(4,2)),((3,3),(3,2),(4,3)),((3,3),(3,4),(2,3))
,((3,3),(3,4),(2,4)),((3,3),(3,4),(3,3)),((3,3),(3,4),(4,3)),((3,3),(3,4),(4,4)),((3,3),(4
,2),(3,1)),((3,3),(4,2),(3,2)),((3,3),(4,2),(3,3)),((3,3),(4,2),(4,1)),((3,3),(4,2),(4,3))
,((3,3),(4,3),(3,2)),((3,3),(4,3),(3,3)),((3,3),(4,3),(3,4)),((3,3),(4,3),(4,2)),((3,3),(4
,3),(4,4)),((3,3),(4,4),(3,3)),((3,3),(4,4),(3,4)),((3,3),(4,4),(4,3))]

*StudentCode> evaluateState 'R' game5
-3
```

Οδηγίες παράδοσης

Στα αρχεία που σας δίνονται υπάρχει ένα αρχείο StudentCode.hs. Σε αυτό το αρχείο θα συμπληρώσετε τα ζητούμενα της άσκησης. Μην αλλάξετε το όνομα του αρχείου κατά την διάρκεια της εκπόνησης της άσκησης, καθώς δεν θα λειτουργεί η οπτικοποίηση σε αυτήν την περίπτωση. Αυτό ακριβώς το αρχείο θα υποβάλλετε στο eclass.

Τυχόν απορίες

Για οποιαδήποτε πρόβλημα ή απορία, γενικότερα αλλά και ειδικά όσον αφορά την οπτικοποίηση, μην διστάσετε να επικοινωνήσετε μέσω e-mail στο sdi1700069@di.uoa.gr.

Κωνσταντίνος Λάκης