

ΑΝΑΦΟΡΑ ΔΕΥΤΕΡΗΣ ΕΡΓΑΣΙΑΣ ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ 2019 – SNAKE
GAME
3/12/2018



ΑΝΑΦΟΡΑ ΔΕΥΤΕΡΗΣ ΕΡΓΑΣΙΑΣ ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ 2019 – SNAKE GAME
3/12/2018

Υπεύθυνοι Φοιτητές :

Χάρος Φίλης ΑΕΜ : 9449 email: charisfilis@ece.auth.gr
Γρηγόρης Παυλάκης ΑΕΜ : 9571 email: grigpavl@ece.auth.gr

ΤΟ ΠΡΟΒΛΗΜΑ

Το δεύτερο μέρος της εργασίας περιλαμβάνει την υλοποίηση της κλάσης *HeuristicPlayer*, η οποία αντιπροσωπεύει έναν παίκτη ο οποίος έχει τη δυνατότητα να θέτει ο ίδιος το ζάρι στην τιμή που τον συμφέρει ώστε να κάνει τη βέλτιστη κίνηση. Ο ορισμός της εκάστοτε βέλτιστης κίνησης γίνεται με ευριστικό τρόπο, και λαμβάνει υπόψη του την απόσταση που θα διανυθεί με τη συγκεκριμένη κίνηση, όπως και τους πόντους που θα κερδηθούν από τα μήλα.

ΥΛΟΠΟΙΗΣΗ

Κλάση HeuristicPlayer

Η κλάση *HeuristicPlayer* κληρονομεί την κλάση *Player*, οπότε μοιράζεται τις ίδιες μεταβλητές και μεθόδους με αυτήν. Προσθέτει μόνο μία επιπλέον μεταβλητή, την:

- ***ArrayList<Integer[]> path*** ~ ένας δυναμικός πίνακας (λίστα) ο οποίος κρατά το ιστορικό όλων των κινήσεων που έχει κάνει ο συγκεκριμένος παίκτης.

Η κλάση λόγω της σχέσης κληρονομικότητάς της με την *Player* έχει ίδιες ακριβώς βασικές μεθόδους, εκτός από τον constructor, ο οποίος καλεί τον constructor της γονικής κλάσης και αρχικοποιεί την *path*.

Εκτός από τις συναρτήσεις της *Player*, η κλάση παρέχει και τις ακόλουθες 3 συναρτήσεις που υλοποιούν τη λειτουργία της:

- ***public double evaluate(int currentPos, int dice)***
(αξιολογεί την κίνηση από τη θέση *currentPos* με ζάρι *dice* σύμφωνα με τη συνάρτηση στόχου)
- ***public int getnextMove(int currentPos)***
(επιλέγει από τις πιθανές κινήσεις για τη θέση *currentPos* τη βέλτιστη και την αποθηκεύει στην *path*)
- ***public void statistics(int currentRound)***
(εκτυπώνει στατιστικά κατάστασης του παίκτη)

ΑΝΑΦΟΡΑ ΔΕΥΤΕΡΗΣ ΕΡΓΑΣΙΑΣ ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ 2019 – SNAKE GAME
3/12/2018

Συνάρτηση αξιολόγησης *public double evaluate(int currentPos, int dice)*

Η συνάρτηση αξιολόγησης αναζητά στους πίνακες φιδιών, σκαλών και μήλων αν υπάρχει κάτι από αυτά στη θέση **currentPos + dice** που είναι η νέα θέση για την κίνηση αυτή. Υπολογίζει την απόσταση που διανύεται (είτε πατώντας σε αρχή σκάλας είτε σε κεφάλι φιδιού, λαμβάνοντας υπόψη της αν η σκάλα μπορεί να ακολουθηθεί) και τους πόντους που συγκεντρώνονται (αν πατήσει μήλο), τα οποία δεδομένα αποθηκεύει στις μεταβλητές **distance** και **score** αντίστοιχα. Τελικά επιστρέφει το σταθμισμένο άθροισμα των 2 αυτών τιμών, με βάρος 0,65 για την απόσταση και 0,35 για τους πόντους.

Συνάρτηση *public int getNextMove(int currentPos)*

Η συνάρτηση αυτή είναι υπεύθυνη για την επιλογή της καλύτερης κίνησης από όλες τις πιθανές. Χρησιμοποιεί έναν *hashmap* (*possibleMoves*) ο οποίος έχει ως κλειδιά του όλες τις ζαριές και ως τιμές τις αξιολογήσεις των κινήσεων από τη θέση **currentPos**, όπως αυτές δίνονται από τη συνάρτηση **evaluate()** για θέση **currentPos** και για το κάθε ζάρι. Αφού αξιολογηθούν όλες οι κινήσεις και βρεθεί η καταλληλότερη (αυτή που μεγιστοποιεί τη συνάρτηση αξιολόγησης) το ζάρι που χρειάζεται γι' αυτήν αποθηκεύεται στη μεταβλητή **neededDie**. Έπειτα εκτελείται η κίνηση με τη συνάρτηση *move* και αποθηκεύεται η νέα κατάσταση του παίκτη (νέα θέση κλπ.) στον πίνακα **statusAfterMove**. Από τον πίνακα αυτόν δημιουργείται ο *Integer[]* πίνακας **pathReg** ο οποίος αποθηκεύει με τη σειρά τα εξής δεδομένα: Ζάρι της βέλτιστης κίνησης, ολικό σκορ κίνησης, διανυθείσα απόσταση, αριθμός μήλων, αριθμός σκαλών και αριθμός φιδιών. Εν τέλει, ο πίνακας **pathReg** αποθηκεύεται στο **path** και η συνάρτηση επιστρέφει τη νέα θέση, η οποία είναι αποθηκευμένη στην πρώτη θέση (index 0) του **statusAfterMove**.

Συνάρτηση *public void statistics()*

Η συνάρτηση αυτή εκτυπώνει ορισμένες πληροφορίες κατάστασης του παίκτη, όπως αριθμό γύρου, ζαριά στο γύρο αυτό, αριθμός κεφαλιών φιδιού που πάτησε στο γύρο, αριθμός βάσεων σκάλας που πάτησε (επίσης στο γύρο), όπως και τον αριθμό των κόκκινων και μαύρων μήλων που τυχόν πάτησε ο παίκτης στο γύρο αυτόν.

Κλάση Game – Συνάρτηση *public Map<Integer, Integer> setTurns(ArrayList<Player> players)*

Η συνάρτηση αυτή αναλαμβάνει να καθορίσει τη σειρά με την οποία θα παίζουν οι παίκτες. Η αρχή λειτουργίας της στηρίζεται στο γεγονός ότι αν γίνει προσπάθεια εισαγωγής σε *hashmap* μιας τιμής σε ένα ήδη υπάρχον κλειδί, η τιμή του κλειδιού εκείνου θα αντικατασταθεί από τη νέα. Παράγεται ένας τυχαίος αριθμός από 1 έως 6 ο οποίος αποθηκεύεται στη μεταβλητή **die** και αντιστοιχίζεται στο εκάστοτε ID παίκτη της *ArrayList*, αν δεν υπάρχει ήδη κάποιος παίκτης με τέτοια ζαριά. Αν πάλι υπάρχει, το ζάρι ξαναρίζεται. Η τελική σειρά των παικτών καθορίζεται διαβάζοντας το παραγόμενο *hashmap* (με κλειδιά ζαριές και τιμές τα ID) από το μικρότερο έως το μεγαλύτερο ζάρι.

ΑΝΑΦΟΡΑ ΔΕΥΤΕΡΗΣ ΕΡΓΑΣΙΑΣ ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ 2019 – SNAKE
GAME
3/12/2018

Κλάση Game - Συνάρτηση main()

Στη συνάρτηση *main* γίνεται αρχικοποίηση ενός 10x20 ταμπλό με 3 φίδια, 3 σκάλες και 6 μήλα. Επίσης, δημιουργούνται δύο νέοι παίκτες (ένας απλός που παίζει με τυχαίο ζάρι και ένας HeuristicPlayer). Αποφασίζεται η σειρά των παικτών διαβάζοντας το map **turns** από τα χαμηλότερα προς τα υψηλότερα ζάρια και τα αντίστοιχα Ids αποθηκεύονται στον πίνακα **playerSequence**. Έπειτα αρχικοποιούνται οι καταστάσεις των παικτών και εισερχόμαστε στον κύριο βρόχο, ο οποίος τερματίζει όταν κάποιος φτάσει στο τέλος. Μέσα στο βρόχο οι παίκτες αυτοί παίζουν εναλλάξ (με τη σειρά που έχει καθοριστεί). Μόλις κάποιος από τους δύο παίκτες τερματίσει, εκτυπώνονται τα στατιστικά του ευριστικού παίκτη και υπολογίζεται μία γενική σταθμισμένη βαθμολογία για τον καθένα τους, η οποία ισούται με $(0,65 * \text{πόντους} + 0,35 * \text{απόλυτη απόσταση})$. Ο παίκτης με τη μεγαλύτερη σταθμισμένη βαθμολογία είναι και ο νικητής.