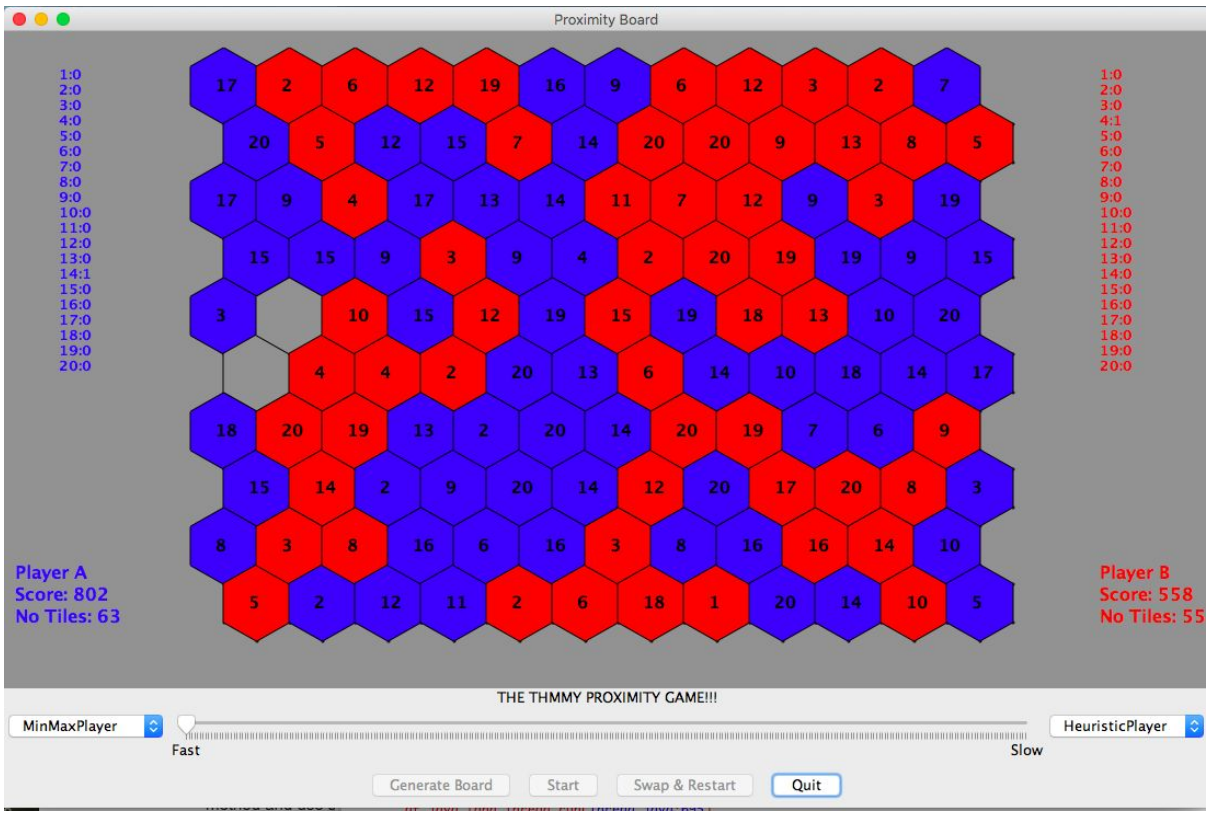
Εργαστήριο Επεξεργασίας Πληροφορίας και Υπολογισμών

Τομέας Ηλεκτρονικής και Υπολογιστών

Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών Α.Π.Θ



DS Proximity

Task 3 Report

Mamalakis Evangelos | 8191
Evangelou Alexandros | 8309

Το πρόβλημα

Ζητείται η δημιουργία ενός λειτουργικού παίκτη του ds-proximity ο οποίος θα χρησιμοποιεί ένα MinMax αλγόριθμο για να επιλέξει την επόμενη κίνηση του. Ο παίκτης θα λαμβάνει υπόψιν του τα δεδομένα που έχει διαθέσιμα την κάθε στιγμή καθώς και το μελλοντικό αποτέλεσμα που θα έχει η κάθε κίνηση του με βάση τον heuristic αλγόριθμο της προηγούμενης άσκησης.

Υλοποίηση της MinMaxPlayer.getNextMove()

Η συνάρτηση getNextMove() επιλέγει την επόμενη κίνηση χρησιμοποιώντας τη συνάρτηση chooseMinMaxMove(). Αρχικά φτιάχνει τον rootNode και στην συνέχεια τον περνάει στην chooseMinMaxMove η οποία επιστρέφει την κίνηση με τις καλύτερες προοπτικές.

Υλοποίηση της MinMaxPlayer.chooseMinMax()

Η συνάρτηση αυτή αναλαμβάνει να προσδιορίσει την καλύτερη κίνηση και να την επιστρέψει. Καλεί την συνάρτηση createSubtree() η οποία, αφού αναπτύξει ένα δέντρο MinMax μέχρι το μέγιστο βάθος, επιστρέφει την καλύτερη κίνηση για το αμέσως επόμενο επίπεδο(επίπεδο με βάθος 1 που είναι και το επίπεδο που μας ενδιαφέρει)

Υλοποίηση της MinMaxPlayer.createSubtree()

Η συνάρτηση createSubtree() υλοποιεί έναν αναδρομικό αλγόριθμο για να φτιάξει το δέντρο MinMax. Κάθε φορά που καλείται για ένα parent φτιάχνει όλα τα πιθανά child nodes και στη συνέχεια, αν δεν έχει φτάσει στο maxDepth καλεί ξανά τον εαυτό της για κάθε ένα από αυτά. Στο τέλος επιστρέφει το child node με τη μέγιστη αξιολόγηση για τον παίκτη που παίζει σε εκείνο το επίπεδο. Στην περίπτωση που καλέσει ξανά τον εαυτό της, αφαιρεί το συνολικό μέγιστο σκορ που επιστρέφεται από το σκορ του εξεταζόμενου παιδιού. Κάνουμε αφαίρεση και όχι πρόσθεση γιατί το επιστρεφόμενο σκορ αφορά πάντα κίνηση του αντιπάλου.

Βελτιστοποίηση της createSubtree()

Επειδή η createSubtree είναι αναδρομική με πολυπλοκότητα O(k^n) όπου k οι πιθανες κινήσεις και n το βάθος, έχει μπει ένα όριο στις πιθανές κινήσεις που εξετάζονται. Αφου υπολογιστή η heuristic αξιολόγηση για κάθε κίνηση, οι κινήσεις ταξινομούνται και επιλόγονται οι πρώτες N_BEST_NODES_TO_EXAMINE κινήσεις για να εξεταστούν σε μεγαλύτερο βάθος.

```
int limitNodesToExamine = Math.min(N_BEST_NODES_TO_EXAMINE,
nodesToExamine.size());
nodesToExamine = new ArrayList<Node81918309>(nodesToExamine.subList(0,
limitNodesToExamine));
```

Υλοποίηση της κλάσης Node81918309

Η κλάση αυτη αναλαμβάνει να αναπαραστήσει ένα κόμβο του δέντρου MinMax.

Υλοποίηση της Node81918309.Node81918309()

Ο Contstructor της Node81918309 έχει δύο εκδοχές. Η πρώτη χρησιμοποιείται για να φτιάξει έναν κόμβο root(που δεν έχει πατέρα), ενώ η δεύτερη για να φτιάξει ένα κόμβο παιδί κάτω από έναν άλλο κόμβο.

Στην δεύτερη περίπτωση γίνονται όλοι οι απαραίτητοι υπολογισμοί με βάση τον εκάστοτε parent για να πάρουν τιμές οι παράμετροι της κλάσης. Η παράμετρος nodePlayerId αποθηκεύει τον παίκτη που παίζει στον συγκεκριμένο κόμβο.

Υλοποίηση του Node81918309.evaluationComparator()

Ο Comparator αυτός χρησιμοποιείται για να γίνει ταξινόμηση των αντικειμένων node με αύξουσα σειρά και με βάση την παράμετρο EvaluationSum, δηλαδή το συνολικό σκορ από τον κόμβο μέχρι το τέλος του δέντρου.

Πηγές και χρήσιμοι σύνδεσμοι

http://jayisgames.com/review/proximity.php

https://en.wikipedia.org/wiki/Minimax

https://en.wikipedia.org/wiki/Alpha%E2%80%93beta_pruning