ΤΗΜΜΥ Α.Π.Θ.

Ακ. Έτος 2020 – 21

Δομές Δεδομένων

Εργασία Java

Λαβύρινθος

Ο Θησέας και ο Μινώταυρος

Μέρος 3^{ο}

Παναγιώτης Σιταρίδης

ΑΕΜ: 10249

Καπλάνης Βασίλειος

ΑΕΜ: 10262

Περιγραφή Παιχνιδιού

Το παιχνίδι αποτελείται από:

- 1 ταμπλό
- 2 παίκτες
- Εφόδια
- Πλακίδια
- Τοίχους

Η εξέλιξη του παιχνιδιού λαμβάνει χώρα στο ταμπλό, το οποίο διαιρείται σε πλακίδια και περιλαμβάνει τοίχους που το καθιστούν λαβύρινθο. Τα εφόδια κατανέμονται με τυχαίο τρόπο σε ορισμένα πλακίδια του λαβυρίνθου. Οι δύο παίκτες που συμμετέχουν είναι ο Θησέας και ο Μινώταυρος, οι οποίοι καθίστανται αντίπαλοι και κινούνται τυχαία. Σκοπός του Θησέα είναι να μαζέψει όλα τα εφόδια χωρίς να πιαστεί από τον Μινώταυρο, ενώ ο δεύτερος προσπαθεί να το αποτρέψει. Το παιχνίδι τερματίζει εάν κερδίσει κάποιος από τους δύο παίκτες ή εάν παρέλθει ένας αρκετά μεγάλος αριθμός κινήσεων, όπου τότε κηρύσσεται ισοπαλία.

Περιγραφή Αλγορίθμου

Αρχικά να αναφερθεί πως η λύση μας για το τρίτο παραδοτέο παρέμεινε σχεδόν αναλλοίωτη με του δευτέρου, με εξαίρεση την κλάση Player, στην οποία η συνάρτηση **int [ ] move(int id, int pid, int dice, Board board)** έγινε όμοια με αυτή της ενδεικτικής λύσης του πρώτου παραδοτέου αλλά απέκτησε ένα επιπλέον όρισμα, το board , τύπου **Board.** Σκοπός της προσθήκης αυτής είναι να προσδιορίζεται σε ποιο ταμπλό εκτελείται η κίνηση, ώστε να μην υπάρχει αλλαγή του βασικού ταμπλό στις προσομοιώσεις των κινήσεων.

Η βιβλιοθήκη που χρησιμοποιήθηκε για τη συγγραφή όλου του κώδικα είναι η java.util.ArrayList.

Η δημόσια κλάση **Node** αντιπροσωπεύει έναν κόμβο του δέντρου. Περιλαμβάνει τις εξείς μεταβλητές: (α) **parent** τύπου Node ως πατέρα του κόμβου, (β) η λίστα **children,** στην οποία εισάγονται τα παιδιά του κόμβου, (γ) **nodeDepth** ως το επίπεδο του δέντρου που βρίσκεται ο κόμβος,(δ) ο πίνακας **nodeMove** τύπου integer που περιέχει πληροφορίες για την θέση και την κίνηση του παίκτη(συντεταγμένες x, y και ζάρι),(ε) **nodeBoard** τύπου Board ως το εικονικό ταμπλό στο οποίο εκτελείται η κίνηση που αντιπροσωπεύει ο κόμβος και (στ) **nodeEvaluation** ως η αξιολόγηση της κίνησης του κόμβου. Αρχικά δημιουργούνται οι **constructors** της κλάσης, που αρχικοποιούν τις τιμές της, είτε σε μηδέν(πρώτος constructor), είτε ανάλογα με τα ορίσματα που δέχονται(δεύτερος constructor). Ακολουθούν οι **getters** και οι **setters** των μεταβλητών που επιστρέφουν και εναποθέτουν τιμές στις μεταβλητές αντίστοιχα.

Η δημόσια κλάση **MinMaxPlayer** κληρονομεί την κλάση Player και αντιπροσωπεύει έναν έξυπνο παίκτη. Στο εσωτερικό της ορίζεται η μεταβλητή **path** ως λίστα της οποίας τα στοιχεία είναι πίνακες με πληροφορίες για την κίνηση του παίκτη(id νέας θέσης, εάν πήρε εφόδιο, απόσταση από το πιο κοντινό εφόδιο , απόσταση απο Μινώταυρο). Επίσης ορίζονται και οι μεταβλητές **up_num, down_num, right_num** και **left_num** που δείχνουν πόσες φορές επιλέγει ο Θησέας την κάθε κατεύθυνση στις κινήσεις του. Το αρχικό της μέγεθος είναι μηδέν και μετά από κάθε γύρο προστίθεται ένας πίνακας με νέα στοιχεία. Οι **constructors** της κλάσης αρχικοποιούν τις τιμές της είτε

σε μηδέν(1ος constructor), είτε ανάλογα με τα ορίσματα που δέχονται(2ος constructor). Στο εσωτερικό τους καλείται ο όρος super που αφορά τις μεταβλητές που έχουν κληρονομηθεί από την κλάση **Player**.

- Η δημόσια μέθοδος **double getMoveValue(double NearSupplies, double NearOpponent)** χρησιμοποιείται για τον υπολογισμό της τιμής της κίνησης χρησιμοποιώντας μια μαθηματική σχέση. Δέχεται ως ορίσματα την απόσταση από το κοντινότερο εφόδιο και από τον Μινώταυρο .
- Η δημόσια μέθοδος **double evaluate(int currentPos, int dice, int MinId)** υπολογίζει και επιστρέφει την αξιολόγηση μιας κίνησης χρησιμοποιώντας την συνάρτηση **getMoveValue.** Δέχεται ως ορίσματα το id της θέσης του Θησέα , τον αριθμό του ζαριού και το id της θέσης του Μινώταυρου. Ξεκινάει ελέγχοντας το id του παίκτη και αξιολογεί πώς η κίνησή του επηρεάζει τον Θησέα. Έπειτα ελέγχει το νούμερο του ζαριού, στη συνέχεια ελέγχει αν βρίσκεται κάποιο εφόδιο ή ο αντίπαλος στην κατεύθυνση που πρόκειται να κινηθεί και εναποθέτει στις μεταβλητές OpponentDistance και NearSupply τις αντίστοιχες τιμές. Ειδικότερα εξετάζεται η περίπτωση όπου ένα εφόδιο βρίσκεται ανάμεσα στον Θησέα και στον Μινώταυρο, στην οποία ο Θησέας επιλέγει να μην πάρει το εφόδιο και διακινδυνεύσει να χάσει. Αυτή η επιλογή προϋποθέτει το συγκεκριμένο εφόδιο να μην είναι το τελευταίο που έχει απομείνει, διότι τότε αν επιλέξει να το πάρει θα κερδίσει. Σε αντίθεση με το δεύτερο παραδοτέο η συνάρτηση ελέγχει για αντικείμενα που βρίσκονται τόσο προς αλλά και αντίθετα της φοράς της κίνησης. Η διαδικασία της αξιολόγησης είναι ίδια και για τους 2 παίκτες, με τη μόνη διαφορά ότι ο Μινώταυρος δεν λαμβάνει υπ' όψιν του τα εφόδια που τον περιβάλλουν, οπότε η μεταβλητή NearSupply παραμένει για αυτόν πάντα μηδέν.

- Η δημόσια μέθοδος **void createMySubtree(int currentPos, int opponentCurrentPos, Node root,int depth)** χρησιμοποιείται για τη δημιουργία του δέντρου. Δημιουργεί το πρώτο επίπεδο του και έπειτα καλεί την συνάρτηση **createopponentSubtree** για τη συμπλήρωση του δεύτερου. Ως ορίσματα δέχεται την θέση του Θησέα και του Μινώταυρου, τη ρίζα του δέντρου και το βάθος του επιπέδου που φτιάχνει. Αναλυτικότερα, ελέγχει τις πιθανές κινήσεις του Θησέα και για κάθε μια δημιουργεί και αρχικοποιεί ένα νέο αντικείμενο τύπου **Node.** Ως μεταβλητή parent δέχεται το root, ως **Nodedepth** το depth και ως **Nodeboard** το ταμπλό του root. Παράλληλα κάθε Node που αντιστοιχεί σε μία κίνηση του Θησέα μπαίνει στο arrayList **children** του root. Στη συνέχεια κάθε κίνηση αξιολογείται και η τιμή της μπαίνει στην μεταβλητή nodeEvaluation του Node που αντιστοιχεί. Στο εικονικό board του Node ποσομοιάζεται η κίνηση ώστε όταν κληθεί η **createopponentSubtree** για τη συμπλήρωση του δεύτερου επιπέδου να ληφθεί ως δεδομένο η νέα εικονική θέση του Θησέα.

- Η δημόσια μέθοδος **void createopponentSubtree (int currentPos, int opponentCurrentPos, Node parent,int depth, double parentEval)** χρησιμοποιείται για τη συμπλήρωση του δευτέρου επιπέδου του δέντρου. Ως ορίσματα δέχεται την θέση του Θησέα και του Μινώταυρου, το βάθος του επιπέδου που φτιάχνει, τον πατέρα κάθε κλάδου του δεύτερου επιπέδου και την αξιολόγηση του πατέρα αυτού. Το δεύτερο επίπεδο αφορά τις κινήσεις του Μινώταυρου. Για κάθε κίνηση του Μινώταυρου ορίζονται και αρχικοποιούνται αντικείμενα τύπου Node με παρόμοιο τρόπο με αυτόν της createMySubtree. Κάθε κόμβος του επιπέδου έχει

NodeEvaluation ίσο με την αξιολόγηση του Θησέα μείον την αξιολόγηση της κίνησης που του αντιστοιχεί.

- Η δημόσια μέθοδος **int chooseMinMaxMove(Node root)** επιλέγει και επιστρέφει την καλύτερη κίνηση για τον Θησέα, λαμβάνοντας υπ' όψιν και την κίνηση του Μινώταυρου. Αρχικά ορίζουμε δύο μεταβλητές min και max. Με εμφολευμένους βρόγχους for, όπου ο πρώτος αφορά την προσπέλαση των κόμβων του επιπέδου 1 και ο δεύτερος τους κόμβους του επιπέδου 2 βρίσκουμε την ελάχιστη από τις αξιολογήσεις του δευτέρου επιπέδου ανά κλάδο και την τοποθετούμε στην μεταβλητή NodeEvaluation του πατέρα τους στο πρώτο επίπεδο. Έπειτα, από τις αξιολογήσεις όλων των παιδιών του root επιλέγουμε την μεγαλύτερη και πιο συμφέρουσα για τον Θησέα. Για να καλύψουμε το ενδεχόμενο όπου οι μέγιστες αξιολογήσεις είναι παραπάνω από μία, δημιουργούμε ένα arrayList **bestMoves** και τοποθετούμε μέσα τις ζαριές από τις κινήσεις που έχουν τη μέγιστη αξιολόγηση. Τελος μέσω της συνάρτησης Random( ) επιλέγεται και επιστρέφεται τυχαία ένα από τα στοιχεία της BestMoves.

- Η δημόσια μέθοδος **int getNextMove(int currentPos, int opponentCurrentPos)** καθίσταται υπεύθυνη για την κίνηση του Θησέα. Δημιουργεί ένα Node που αποτελεί τη ρίζα του δέντρου το οποίο ως όρισμα nodeBoard δέχεται το βασικό ταμπλό του παιχνιδιού. Καλεί την createMySubtree για να δημιουργήσει το δέντρο, και ανανεώνει την μεταβλητή path. Ορίζεται ένας πίνακας **p** στον οποίο αποθηκεύονται οι πληροφορίες της κίνησης που πραγματοποιεί ο Θησέας σε κάθε γύρο. Συγκεκριμένα, για το 1ο στοιχείο του καλούμε την συνάρτηση chooseMinMaxMove( ). Για τις

αποστάσεις από το κοντινότερο εφόδιο και τον Μινώταυρο ελέγχονται μόνο τα πλακίδια στην διεύθυνση της κίνησης. Έπειτα ,αυξάνεται κάποια από τις μεταβλητές **up_num, down_num, right_num** και **left_num** ανάλογα με την κίνηση που επιλέχθηκε, ελέγχεται αν ο Θησέας πήρε κάποιο εφόδιο και ο p εισάγεται στη λίστα path.Τέλος ο παίκτης κινείται με την συνάρτηση **move** και επιστρέφεται ο πίνακας που προκύπτει από την κλήση της.

- Η δημόσια μέθοδος **void statistics(boolean x)** χωρίζεται σε δύο σκέλη. Το πρώτο σκέλος αφορά την εκτύπωση στοιχείων για τις κινήσεις του Θησέα σε κάθε γύρο, λαμβάνοντας δεδομένα από την λίστα path. Κατά το δεύτερο μέρος ελέγχεται με το όρισμα της συνάρτησης αν το παιχνίδι έχει φτάσει στο τέλος του, ανακοινώνει τη λήξη του και προβάλλει τις μεταβλητές **up_num, down_num, right_num** και **left_num** με κάποια σχετικά μηνύματα.

Η δημόσια κλάση **Game** είναι πανομοιότυπη με αυτή της λύσης του δευτέρου παραδοτέου. Ωστόσο ορίστηκε ένας πίνακας MMplayer τύπου MinMaxPlayer στον οποίο τοποθετείται το νέο αντικείμενο που αντιπροσωπεύει τον Θησέα. Κατά τη διάρκεια του γύρου διαχωρίζουμε την διαδικασία της κίνησης και την ανανέωση των συντετεγμένων για τον Θησέα και τον Μινόταυρο. Τέλος γίνεται κλήση της **statistics** τόσο στο πέρας του γύρου όσο και του παιχνιδιού.