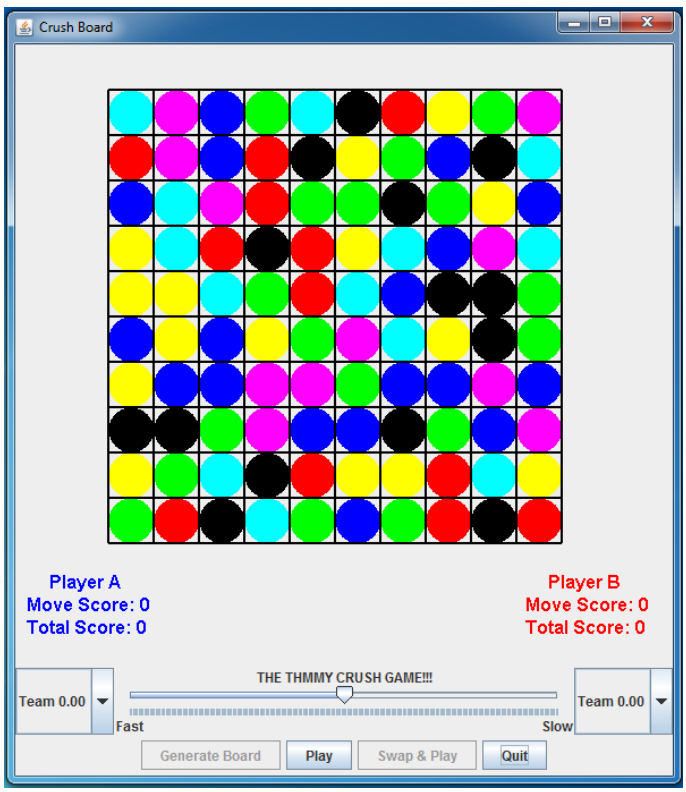
Εργαστήριο Επεξεργασίας Πληροφορίας και Υπολογισμών
Τομέας Ηλεκτρονικής και Υπολογιστών **5ο Εξάμηνο**
ΤΗΜΜΥ Α.Π.Θ **2016-2017**
Δευτέρα 28 / 11 / 2016

ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ

DS – CandyCrush – Part 2

Heuristic Algorithm (0.75 βαθμοί)

Στο δεύτερο παραδοτέο καλείστε να υλοποιήσετε μια συνάρτηση αξιολόγησης των διαθέσιμων κινήσεων που έχει ο παίκτης σε κάθε γύρο παιχνιδιού και την αντίστοιχη συνάρτηση επιλογής της καλύτερης κίνησης. Σκοπός σας είναι να δημιουργήσετε μια όσο το δυνατόν πιο ολοκληρωμένη συνάρτηση αξιολόγησης, που να λαμβάνει υπόψη της τα διαθέσιμα δεδομένα (μέχρι και) εκείνη τη στιγμή και να τα αξιολογεί κατάλληλα εφαρμόζοντας διάφορα κριτήρια.



Εικόνα 1: Το περιβάλλον του παιχνιδιού Candy-Crush για το μάθημα των Δομών Δεδομένων 2013-2014.

Ο νέος κώδικας της πλατφόρμας περιέχει την υλοποίηση για τον Random Player, η οποία θα είναι ίδια για όλους.

ΠΡΟΣΟΧΗ!!!
ΜΗΝ ΑΝΤΙΚΑΤΑΣΤΗΣΕΤΕ ΤΗΝ ΥΛΟΠΟΙΗΣΗ ΤΟΥ RANDOM PLAYER ΜΕ ΑΥΤΗ ΠΟΥ ΥΛΟΠΟΙΗΣΑΤΕ ΣΤΟ ΠΡΟΗΓΟΥΜΕΝΟ ΠΑΡΑΔΟΤΕΟ. ΘΕΛΟΥΜΕ ΟΛΟΙ ΝΑ ΕΧΟΥΝ ΤΗΝ ΙΔΙΑ ΠΛΑΤΦΟΡΜΑ.

Χρήσιμες Μεταβλητές και Συναρτήσεις

Για την υλοποίηση των παραπάνω συναρτήσεων πολύ πιθανόν να χρειαστεί να χρησιμοποιήσετε κάποιες μεταβλητές / συναρτήσεις που υπάρχουν ήδη υλοποιημένες στην πλατφόρμα.

Κλάση *CrushUtilities*:

- **Στατικές Μεταβλητές της κλάσης *CrushUtilities***

  Αριθμός Γραμμών και Στηλών:
  NUMBER_OF_ROWS = 20;
  NUMBER_OF_COLUMNS = 10;
  NUMBER_OF_PLAYABLE_ROWS = 10;

  Για να διασχείστε το ταμπλό που έχει διαστάσεις 10x10 χρησιμοποιήστε την NUMBER_OF_ROWS και την NUMBER_OF_PLAYABLE_ROWS.

  Κωδικός Κατευθύνσεων:
  LEFT = 0 (Αριστερά)
  DOWN = 1 (Κάτω)
  RIGHT = 2 (Δεξιά)
  UP = 3 (Πάνω)

  Κωδικοί Χρωμάτων:
  RED = 0 (Κόκκινο)
  GREEN = 1 (Πράσινο)
  BLUE = 2 (Μπλε)
  YELLOW = 3 (Κίτρινο)
  BLACK = 4 (Μαύρο)
  PURPLE = 5 (Μωβ)
  CYAN = 6 (Κυανό)

  Κωδικοί Παικτών:
  BLUE_PLAYER = 7 (Μπλε Παίκτης)
  RED_PLAYER = 8 (Κόκκινος Παίκτης)

  Όριο για το Σκορ (αν θέλετε μεγαλύτερα ή μικρότερα παιχνίδια):
  SCORE_LIMIT = 300

- **Στατικές Συναρτήσεις της κλάσης *CrushUtilities***

  board boardAfterFirstMove (Board board, int[] move): η συνάρτηση αυτή επιστρέφει την κατάσταση του ταμπλό *board* ακρίβως μετά την κίνηση *move* και πριν να σκάσουν τα πλακίδια που συμμετέχουν σε κάποια ν-αδα. Αυτό σημαίνει οτι μπορείτε να

ελέγξετε πόσα πλακάκια θα σκάσουν, **ανεξαρτήτου χρώματος**, ώστε να αξιολογήσετε κατάλληλα την κίνηση move.

Κλάση *Tile*:

- **Συναρτήσεις της κλάσης *Tile***

  int getX(): η συνάρτηση αυτή επιστρέφει την θέση του πλακιδίου στον άξονα x'x.

  int getY(): η συνάρτηση αυτή επιστρέφει την θέση του πλακιδίου στον άξονα y'y.

  int getColor(): η συνάρτηση αυτή επιστρέφει το χρώμα του ζαχαρωτού που βρίσκεται στο πλακίδιο. Συγκεκριμένα έχουμε τα εξής χρώματα:

  - **0 → Red**
  - **1 → Green**
  - **2 → Blue**
  - **3 → Yellow**
  - **4 → Black**
  - **5 → Purple**
  - **6 → Cyan**

Κλάση *Board*:

- **Συναρτήσεις της κλάσης *Board***

  int getRows(): η συνάρτηση αυτή επιστρέφει τον αριθμό των γραμμών του ταμπλό.

  int getCols(): η συνάρτηση αυτή επιστρέφει τον αριθμό των στηλών του ταμπλό

  Για να διασχείστε το ταμπλό που έχει διαστάσεις 10x10 χρησιμοποιήστε την NUMBER_OF_ROWS και την NUMBER_OF_PLAYABLE_ROWS της κλάσης CrushUtilities.

  Tile giveTileAt (int x, int y): η συνάρτηση αυτή επιστρέφει το πλακίδιο που βρίσκεται στην θέση (x,y) του ταμπλό.

  void showBoard(): η συνάρτηση αυτή εκτυπώνει στην κονσόλα το ταμπλό του παιχνιδιού με τη μορφή πίνακα αριθμών, όπου ο κάθε αριθμός συμβολίζει το χρώμα του ζαχαρωτού που περιέχεται σε κάθε πλακίδιο όπως φαίνεται στην οθόνη.

Κλάση ArrayList

Για την υλοποίηση αυτής της εργασίας, θα χρειαστείτε οπωσδήποτε την κλάση ArrayList του πακέτου java.util. Η κλάση αυτή αναπαριστά έναν δυναμικό πίνακα, του οποίου το μέγεθος δεν είναι καθορισμένο από την αρχή, αλλά στον οποίο μπορείτε να προσθαφαιρείτε στοιχεία

(αντικείμενα) κατά βούληση. Μπορείτε να δηλώσετε τι τύπου αντικείμενα θα περιέχονται. Για παράδειγμα, μια ArrayList με αντικείμενα τύπου String θα αρχικοποιηθεί ως εξής:

ArrayList<String> thisArrayList = new ArrayList<String>();

Οι μέθοδοι που θα χρειαστείτε είναι οι εξής:

- `void add(Object o):` Προσθέτει το αντικείμενο που δίνεται ως όρισμα στην ArrayList.
- `Object get(int pos):` Επιστρέφει το αντικείμενο που βρίσκεται στην θέση που δώσαμε ως όρισμα. Αν η ArrayList έχει αρχικοποιηθεί για να παίρνει αντικείμενα ενός συγκεκριμένου τύπου, τότε επιστρέφει τον συγκεκριμένο τύπο και όχι ένα απλό Object.
- `int size():` Επιστρέφει το πλήθος των αντικειμένων που είναι αποθηκευμένα στην ArrayList.

Αξιολόγηση Διαθέσιμων Κινήσεων

Το ζητούμενο είναι να υλοποιήσετε την συνάρτηση **double moveEvaluation (int[] move, Board board)** στην κλάση *Player.HeuristicPlayer,* που υπολογίζει το πόσο αξιόλογη θα είναι η κίνηση σας, αν επιλέξετε την συγκεκριμένη κίνηση move από την λίστα διαθέσιμων κινήσεων, στην παρούσα κατάσταση του ταμπλό board.

Η αξιολόγηση των κινήσεων μπορεί να γίνει με διάφορα κριτήρια. Κάποιες **ιδέες** είναι:

1) Ο αριθμός των ζαχαρωτών που θα απομακρυνθούν μετά την κίνηση σας από το ταμπλό (3, 4, 5 ή και παραπάνω). Υπάρχει περίπτωση η κίνηση που θα επιλέξετε να οδηγήσει σε ταυτόχρονη απομάκρυνση ζαχαρωτών και στον κάθετο αλλά και στον οριζόντιο άξονα (σκεφτείτε το).
   Ένας πολύ γενικός αλγόριθμος για την συνάρτηση αυτή θα μπορούσε να έχει τα εξής βήματα:
- Για κάθε κίνηση move που θέλω να αξιολογήσω:
  - Δίνω την κίνηση move και το board σαν ορίσματα στην boardAfterFirstMove
  - Το αποτέλεσμα της boardAfterFirstMove είναι ένα αντικείμενο της κλάσης Board, έστω με όνομα aBoard. Δίνω το αντικείμενο αυτό σαν όρισμα σε μια συνάρτηση, έστω η calculateTilesThatWillCrush(), η οποία σκανάρει το ταμπλό και μου επιστρέφει τον αριθμό τον πλακιδίων που θα σκάσουν απο την/τις ν-αδα/ν-αδες που θα δημιουργηθουν, ανεξαρτήτως χρώματος.
  - Το αποτέλεσμα της calculateTilesThatWillCrush() θα είναι η άμεση ανταμοιβή απο την κίνση μου

~~2) Η κατάσταση του ταμπλό μετά την κίνηση οδηγεί σε νέα απομάκρυνση ζαχαρωτών (chain moves).~~ **(Μην ασχοληθείτε με τις chain-moves για την δεύτερη εργασία. Αν το κάνατε δεν χρειάζεται να το σβήσετε. Απλά περιγράψτε στην αναφορά το τι ακριβώς κάνατε.)**

3) Η απόσταση των ζαχαρωτών που απομακρύνθηκαν από το κάτω μέρος της οθόνης (όπου y=0).
4) Αν η κίνηση που επιλέξατε οδηγεί σε οριζόντια ή κάθετη απομάκρυνση ζαχαρωτών από το ταμπλό.

5) Πόσα ζαχαρωτά ιδίου χρώματος (μεταξύ τους, όχι του χρώματος των ζαχαρωτών που πρόκειται να απομακρυνθούν) βρίσκονται σε μικρή απόσταση από αυτά που πρόκειται απομακρυνθούν.

Η συνάρτηση που θα φτιάξετε θα πρέπει να επιστρέφει μια τιμή double μέσα σε κάποια λογικά όρια (πχ 0 έως 100), η οποία θα αντιστοιχεί στην αξιολόγηση της κίνησης που εξετάζετε. Οι μικρότεροι αριθμοί αντιστοιχούν σε καταστάσεις που η κίνηση δεν σας αποφέρει πολλούς βαθμούς, ενώ οι μεγαλύτεροι αριθμοί αντιστοιχούν σε καταστάσεις με μεγαλύτερο κέρδος για εσάς (περισσότερα από 3 ζαχαρωτά, πολλαπλές κινήσεις κτλ). Όσο καλύτερη κρίνεται η κίνηση σας στο ταμπλό, τόσο μεγαλύτερος πρέπει να είναι ο αριθμός που επιστρέφει η συνάρτηση αξιολόγησης.

Για να μην είναι μεροληπτική η αξιολόγηση, ιδανικά θα πρέπει να φροντίσετε να λαμβάνετε υπόψη σας και την θέση που μπορεί να βρίσκεται ο αντίπαλος μετά από εσάς (πχ του έχετε αφήσει πολλά όμοια ζαχαρωτά το ένα κοντά στο άλλο). Αυτό δεν είναι υποχρεωτικό σε αυτή την εργασία.

Η υλοποίηση αυτής της μεθόδου είναι ελεύθερη αλλά μπορείτε να βασιστείτε στις γενικές κατευθύνσεις που σας δίνονται παραπάνω. **Όμως, ακριβώς επειδή η υλοποίηση είναι ελεύθερη, θα πρέπει να δώσετε ιδιαίτερη βαρύτητα στον σχολιασμό του κώδικα, όσο και στην λεπτομερή ανάλυση του σκεπτικού που ακολουθήσατε στην γραπτή αναφορά που θα παραδώσετε.**

Μερικές χρήσιμες συναρτήσεις θα μπορούσαν να είναι:

1. **int deletedCandies():** Η συνάρτηση αυτή υπολογίζει τον αριθμό των ζαχαρωτών που θα απομακρύνει η κίνηση που αξιολογείται.
2. **int heightOfMove():** Η συνάρτηση αυτή υπολογίζει το ύψος που βρίσκονται τα ζαχαρωτά που θα απομακρυνθούν. Λαμβάνεται σαν την απόσταση του πιο κοντινού ζαχαρωτού στο κάτω μέρος του ταμπλο.
3. **boolean vertOrHor():** Η συνάρτηση αυτή επιστρέφει αν η κίνηση απομάκρυνει ζαχαρωτά στον οριζόντιο ή στον κάθετο άξονα.
4. **int[] sameColorInProximity(int widht, int height):** Η συνάρτηση αυτή επιστρέφει τον αριθμό των ζαχαρωτών όλων των χρωμάτων που βρίσκονται σε μικρή απόσταση από το σημείο όπου θα πραγματοποιηθεί η κίνηση των ζαχαρωτών. Παίρνει σαν είσοδο το μήκος και το ύψος γύρω από το σημείο που έγινε η κίνηση και επιστρέφει έναν πίνακα με τιμές.

   Η διάσταση του πίνακα ορίζεται από τα χρώματα όπως δίνονται παραπάνω και η τιμή από τον αριθμό όμοιων ζαχαρωτών για κάθε χρώμα που υπάρχουν στην περιοχή που οριοθετήθηκε. Για παράδειγμα **sameColorInProximity** [0] = 10 σημαίνει οτι υπάρχουν στην περιοχή που οριοθετήσαμε 10 ζαχαρωτά χρώματος κόκκινου.

ΠΡΟΣΟΧΗ!!!

Θα πρέπει να είστε πολύ προσεκτικοί με τα όρια του ταμπλό (πίνακα). Κατά την διάρκεια της αναζήτησης και της αξιολόγησης υπάρχει μεγάλη πιθανότητα να βγείτε εκτός ορίων, αν δεν είστε προσεκτικοί.

Επιλογή Διαθέσιμης Κίνησης

Η συνάρτηση που καλείστε να υλοποιήσετε για την επιλογή της τελικής σας κίνησης είναι η `int findBestMoveIndex(ArrayList<int[]> availableMoves, Board board)` στην κλάση *Player.HeuristicPlayer.* Ο αλγόριθμος υλοποίησης είναι ο εξής:

Για κάθε διαθέσιμη κίνηση που έχετε στην λίστα availableMoves:

I. Αξιολόγηση κάθε διαθέσιμης κίνησης με χρήση της συνάρτησης moveEvaluation (int[] move, Board board).

II. Επιλογή της καλύτερης κίνησης με βάση την αξιολόγηση που κάνατε.

Επιστροφή του δείκτη της καλυτερης κίνησης από την λίστα availableMoves. (Για παράδειγμα αν η καλύτερη κίνηση είναι η τρίτη, θα πρέπει να επιστρέψετε το 2).

Λειτουργία της Πλατφόρμας

- Μέσα από το περιβάλλον του Eclipse πάτε στον φάκελο lib του project. Δεξί κλικ στο αρχείο crush.jar και πατάτε run -> As a Java application.
- Πατάτε το κουμπί που λέει “Generate Board” για να δημιουργηθεί ένα καινούργιο στιγμιότυπο του ταμπλό.
- Στην συνέχεια επιλέγεται τους παίκτες που θα αναμετρηθούν χρησιμοποιώντας τα drop-down boxes στα δύο άκρα της οθόνης. Οι επιλογές είναι 2 για αυτή τη φάση της εργασίας: “Random Player” (που είναι η δική μας υλοποίηση) και “Heuristic Player” που θα είναι ο παίκτης που θα παράγεται από την κλάση που καλείστε να υλοποιήσετε για την δεύτερη εργασία.
- Στη συνέχεια πατάτε το κουμπί “Play” για να ξεκινήσει το παιχνίδι.
- Στο τέλος του γύρου μπορείτε να επιλέξετε να πατήσετε το κουμπί «Swap and Play” που αλλάζει θέση στους δύο παίκτες (ο Μπλε παίκτης γίνεται Κόκκινος και το αντίστροφο) και ξεκινάει νέο παιχνίδι. Αυτό γίνεται καθώς για να είναι δίκαιο το αποτέλεσμα ενός ζεύγους παιχνιδιών ανάμεσα σε δύο παίκτες, καθώς θα πρέπει και οι δύο να εκκινήσουν το παιχνίδι παίζοντας πρώτοι.
- Το σκορ που εμφανίζεται στην οθόνη εξάγεται και σαν αρχείο μέσα στο φάκελο του Project σας.

Οδηγίες

Τα προγράμματα θα πρέπει να υλοποιηθούν σε Java, με πλήρη τεκμηρίωση του κώδικα. Το πρόγραμμά σας πρέπει να περιέχει επικεφαλίδα σε μορφή σχολίων με τα στοιχεία σας (ονοματεπώνυμα, ΑΕΜ, τηλέφωνα και ηλεκτρονικές διευθύνσεις). Επίσης, πριν από κάθε κλάση ή μέθοδο θα υπάρχει επικεφαλίδα σε μορφή σχολίων με σύντομη περιγραφή της λειτουργικότητας του κώδικα. Στην περίπτωση των μεθόδων, πρέπει να περιγράφονται και οι μεταβλητές τους.

Είναι δική σας ευθύνη η απόδειξη καλής λειτουργίας του προγράμματος.

Παραδοτέα για κάθε μέρος της εργασίας

1. Ηλεκτρονική αναφορά που θα περιέχει: εξώφυλλο, περιγραφή του προβλήματος, του αλγορίθμου και των διαδικασιών που υλοποιήσατε και τυχόν ανάλυσή τους. Σε καμία περίπτωση να μην αντιγράφεται ολόκληρος ο κώδικας μέσα στην αναφορά (εννοείται ότι εξαιρούνται τμήματα κώδικα τα οποία έχουν ως στόχο τη διευκρίνιση του αλγορίθμου)
Προσοχή: Ορθογραφικά και συντακτικά λάθη πληρώνονται.

2. Ένα αρχείο σε μορφή .zip με όνομα "AEM1_AEM2_PartB.zip", το οποίο θα περιέχει **όλο το project σας** στον eclipse καθώς και το αρχείο της γραπτής αναφοράς σε pdf (**αυστηρά**). Το αρχείο .zip θα γίνεται upload στο site του μαθήματος ***στην ενότητα των ομαδικών εργασιών και μόνο***. Τα ονόματα των αρχείων πρέπει να είναι με **λατινικούς χαρακτήρες**.

Προθεσμία υποβολής

Κώδικας και αναφορά Τετάρτη 21 Δεκεμβρίου, 23:59 (ηλεκτρονικά)
Δε θα υπάρξει καμία παρέκκλιση από την παραπάνω προθεσμία.