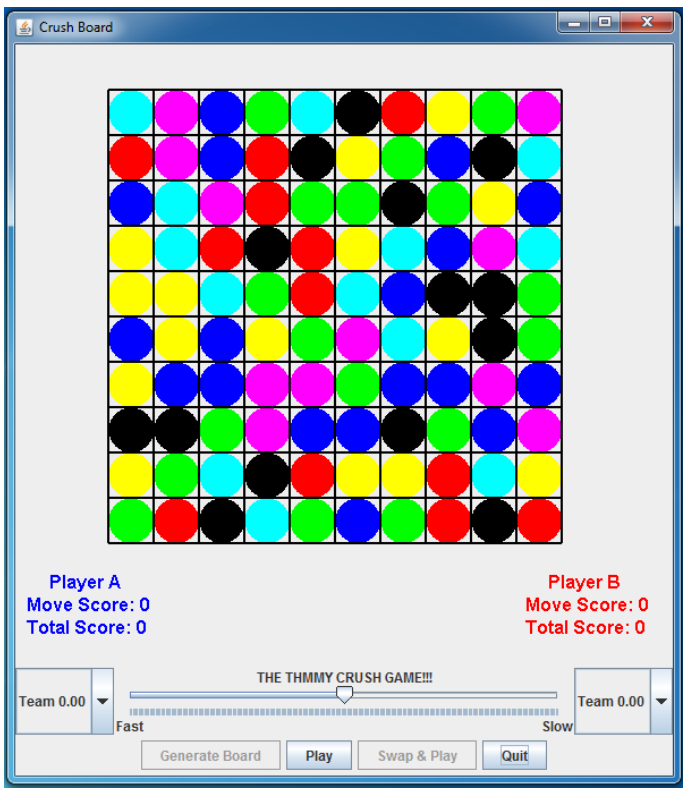
Εργαστήριο Επεξεργασίας Πληροφορίας και Υπολογισμών
Τομέας Ηλεκτρονικής και Υπολογιστών **5ο Εξάμηνο**
ΤΗΜΜΥ Α.Π.Θ **2015-2016**
Πέμπτη 19 / 12 / 2016

ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ

DS – Candy Crush – Part 3

MinMax Algorithm (0,5 βαθμοί)

Στο τρίτο παραδοτέο καλείστε να υλοποιήσετε τον αλγόριθμο MinMax για την βελτιστοποίηση του παίκτη σας. Σκοπός σας είναι να δημιουργήσετε ένα δέντρο βάθους 2 (τουλάχιστον), το όποιο σε συνδυασμό με την συνάρτηση αξιολόγησης που είχατε δημιουργήσει στην δεύτερη εργασία, θα αξιολογεί τις διαθέσιμές κινήσεις σε βάθος χρόνου και θα επιλέγει την καλύτερη δυνατή για τον επόμενο γύρο.



Εικόνα 1: Το περιβάλλον του παιχνιδιού DS-Candy Crush.

Ο νέος κώδικας της πλατφόρμας περιέχει την random υλοποίηση για τον ένα παίκτη καθώς και μια τυπική evaluation υλοποίηση για τον Heuristic παίκτη, η οποία θα είναι ίδια για όλους. Στον MinMax Player θα χρησιμοποιήσετε την δική σας υλοποίηση της evaluation function που είχατε φτιάξει για το δεύτερο παραδοτέο.

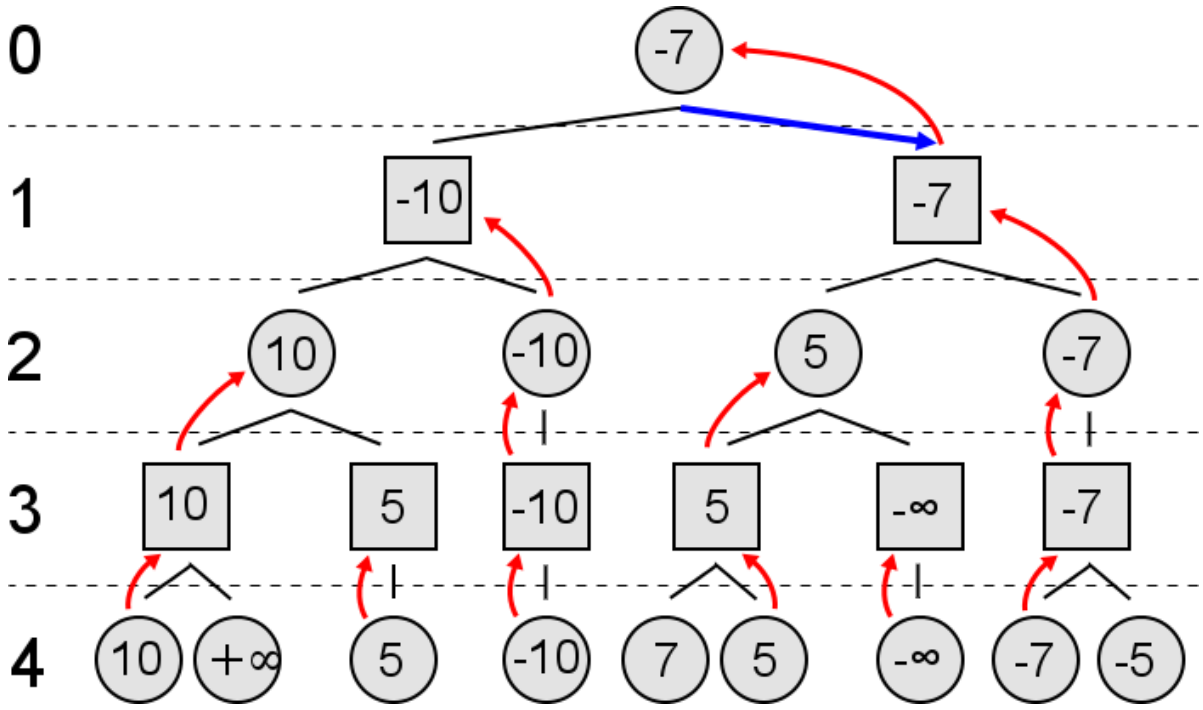
ΠΡΟΣΟΧΗ!!!
ΜΗΝ ΑΝΤΙΚΑΤΑΣΤΗΣΕΤΕ ΤΙΣ ΥΛΟΠΟΙΗΣΕΙΣ ΑΥΤΕΣ ΜΕ ΑΥΤΕΣ ΠΟΥ ΥΛΟΠΟΙΗΣΑΤΕ ΣΤΑ ΠΡΟΗΓΟΥΜΕΝΑ ΠΑΡΑΔΟΤΕΑ. ΘΕΛΟΥΜΕ ΟΛΟΙ ΝΑ ΕΧΟΥΝ ΤΗΝ ΙΔΙΑ ΠΛΑΤΦΟΡΜΑ.

Αλγόριθμος MinMax – AB Pruning

Όπως σε πολλά παιχνίδια, έτσι και στο δικό μας παιχνίδι, για τον υπολογισμό της βέλτιστης κίνησης οι υπολογιστές δεν μπορούν να αναπτύξουν όλους τους δυνατούς συνδυασμούς των δικών μας κινήσεων και των κινήσεων του αντιπάλου σε εύλογο χρονικό διάστημα. Εξαιτίας αυτού του γεγονότος, το δένδρο των πιθανών συνδυασμών αναπτύσσεται μέχρι ενός βάθους (π.χ. για βάθος 2 έχω όλες τις κινήσεις μου και όλες τις πιθανές απαντήσεις του αντιπάλου). Σε κάθε φύλλο του δένδρου (στο τέλος της απάντησης του αντιπάλου) εξετάζω τη συνάρτηση αξιολόγησης που υλοποιήθηκε στη δεύτερη εργασία. Σε περίπτωση που ένας κόμβος του δένδρου είναι τελικός (το παιχνίδι τελειώνει), μπορούμε να υπολογίσουμε το νικητή και να σταματήσουμε εκεί.

Το δένδρο που θα φτιαχτεί θα έχει τη δομή του Σχήματος 2, δίνοντας σε κάθε **φύλλο** του το αποτέλεσμα της συνάρτησης αξιολόγησης. Η δική μας νίκη συμβολίζεται με +∞, ενώ η νίκη του αντιπάλου με -∞. Με τετράγωνα εμφανίζονται οι κινήσεις μας, ενώ με κύκλους οι κινήσεις του αντιπάλου. Για τον αντίπαλο θεωρούμε ότι θα παίξει τη χειρότερη για εμάς κίνηση. Για αυτό επιλέγουμε το **ελάχιστο (Minimum)** της συνάρτησης αξιολόγησης (γραμμές 1 και 3). Εμείς φυσικά θα παίξουμε την κίνηση που **μεγιστοποιεί (Maximum)** τη συνάρτηση αξιολόγησης (γραμμές 0 και 2), λαμβάνοντας ως δεδομένο ότι ο αντίπαλος θα παίξει την καλύτερη κίνηση για αυτόν. Στο σχήμα, λοιπόν, θα επιλέξουμε τη δεξιά κίνηση γιατί μεγιστοποιεί τη συνάρτηση αξιολόγησης, ασχέτως με το τι θα επιλέξει να παίξει ο αντίπαλος.

Το βάθος του δέντρου μας δείχνει πόσες κινήσεις μπροστά κοιτάμε, και κάθε επίπεδο του δείχνει ακριβώς ποια από τις μελλοντικές κινήσεις εξετάζουμε.



Σχήμα 2: Ο αλγόριθμος MinMax για βάθος 4.

Κλάση Αποθήκευσης Διαθέσιμων Κινήσεων

Για την υλοποίηση των διαθέσιμων κινήσεων προτείνεται η δημιουργία μιας δικής σας κλάσης με το όνομα NodeAEM1AEM2[1]. Η κλάση αυτή θα έχει ως μεταβλητές:

1. **Node parent**: Ο κόμβος πατέρας του κόμβου που δημιουργήσατε.
2. **ArrayList<Node> children**: Ο πίνακας που περιλαμβάνει τα παιδιά του κόμβου που δημιουργήσατε.
3. **int nodeDepth**: το βάθος του κόμβου στο δέντρο του MinMax Αλγορίθμου.
4. **int[] nodeMove**: Τον κίνηση που αντιπροσωπεύει το Node, σαν πίνακας ακεραίων που περιλαμβάνει το x, το y και την κατεύθυνση. (όπως σας δίνονται από την λίστα με τις διαθέσιμες κινήσεις).
5. **Board nodeBoard**: το ταμπλό του παιχνιδιού για αυτό τον κόμβο-κίνηση.
6. **double nodeEvaluation**: Την τιμή της συνάρτησης αξιολόγησης που έχετε δημιουργήσει για αυτή τη κίνηση.

Οι συναρτήσεις της κλάσης που θα πρέπει να υλοποιηθούν είναι:

1. **Node():** Ένας ή περισσότεροι constructors για την κλάση σας, με διαφορετικά ορίσματα.
2. Κατάλληλες συναρτήσεις ***get*** και ***set***.
3. **double evaluate():** η συνάρτηση αυτή θα υπολογίζει το πόσο αξιόλογη είναι η συγκεκριμένη κίνηση (μπορείτε να χρησιμοποιήσετε αυτή που υλοποιήσατε στην δεύτερη εργασία ή μια βελτιωμένη έκδοση της).

Οι συναρτήσεις που είχατε στην προηγούμενη υλοποίηση του δεύτερου παραδοτέου για την evaluate() θα αποτελέσουν την βάση σας και για αυτή την εργασία, με όσες αλλαγές χρειάζονται για να συμπεριληφθούν και οι νέες μεταβλητές.

Στο package node βρίσκεται μια πρότυπη κλάση με το όνομα Node. Θα πρέπει να την μετονομάσετε (δεξί κλικ → Refactor → Rename) σύμφωνα με την εκφώνηση και να υλοποιήσετε τις κατάλληλες συναρτήσεις.

Χρήσιμες Μεταβλητές και Συναρτήσεις

Για την κλάση CrushUtilities:

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

  Αριθμός Γραμμών και Στηλών:
  ```
  NUMBER_OF_ROWS = 500;
  NUMBER_OF_PLAYABLE_ROWS = 10;
  NUMBER_OF_COLUMNS = 10;
  ```

[1] Όπου AEM1 και AEM2 ο αριθμός μητρώου των δύο ατόμων κάθε ομάδας.

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;

Ταχύτητα του παιχνιδιού (αν θέλετε να το επιταχύνετε ή να το επιβραδύνετε κι άλλο):
TIME_STEP = 100;

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

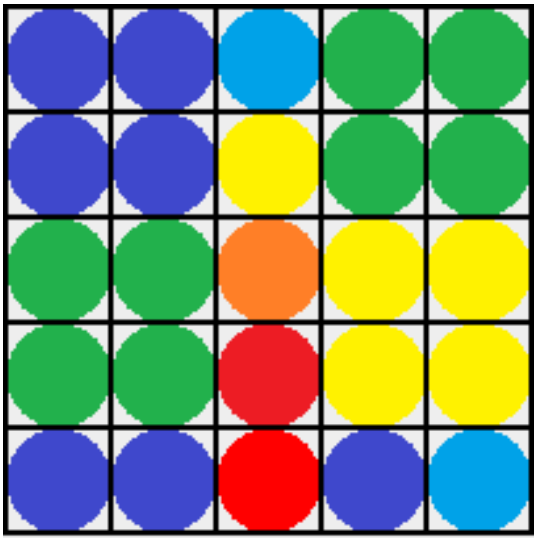
  `Board cloneBoard(Board board):` Επιστρέφει ένα ακριβές αντίγραφο του ταμπλό. Η χρήση της συνάρτησης αυτής κρίνεται απαραίτητη για να μπορέσετε να δείτε τις επόμενες κινήσεις σε βάθος χρόνου.

  `int[] calculateNextMove (int[] move):` Επιστρέφει την κίνηση που θα πρέπει να κάνετε στο ταμπλό. Είναι ίδια με αυτή που χρησιμοποιήσατε στην προηγούμενη εργασία, απλά μεταφέρθηκε από την κλάση Player στην κλάση CrushUtilities.

  `ArrayList<int[]> getAvailableMoves (Board board):` Επιστρέφει μια λίστα με τις διαθέσιμες κινήσεις για τον παίκτη που πρόκειται να παίξει στην παρούσα κατάσταση του ταμπλό.

- **Τέσσερεις συναρτήσεις για να δείτε στο μέλλον**

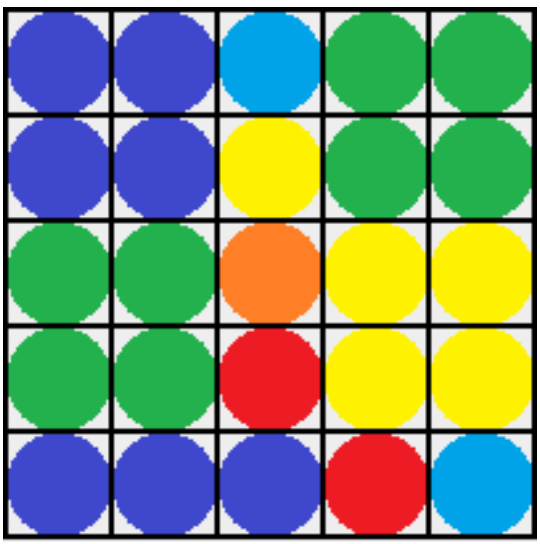
Έστω πως την χρονική στιγμή t_o έχουμε ένα ταμπλό διαστάσεων 5*5 όπως αυτό παρουσιάζεται στην Εικόνα 2.



Εικόνα 2 Ταμπλό την χρονική στιγμή το

Σας δίνονται τρεις συναρτήσεις μέσα στην κλάση CrushUtilities οι οποίες σας επιτρέπουν να δείτε πως θα είναι το ταμπλό σε κάποια στιγμή στο μέλλον. Αυτές είναι οι εξής:

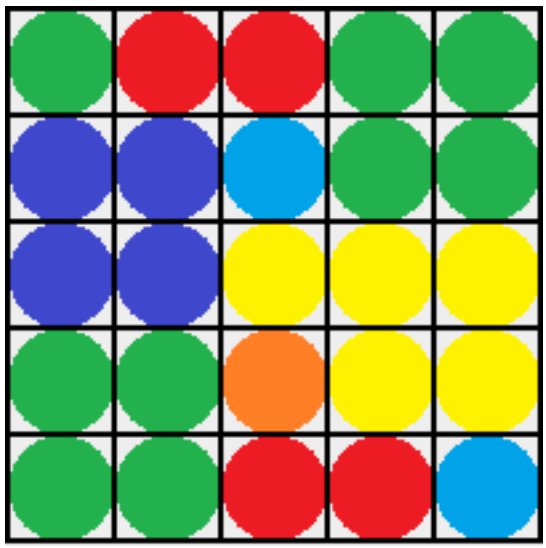
- `Board boardAfterFirstMove (Board board, int[] move):` Η συνάρτηση αυτή παίρνει ως ορίσματα το ταμπλό της χρονικής στιγμής t_o board, την κίνηση move (της μορφής int[] τριών θέσεων που περιλαμβάνει [x, y, κατεύθυνση]) που θέλουμε να εκτελέσουμε και επιστρέφει ένα αντίγραφο του ταμπλό με αλλαγμένες τις θέσεις των ζαχαρωτών που ορίστηκαν από την move. Στην περίπτωσή μας το ταμπλό που θα επιστρέφονταν θα ήταν αυτό που φαίνεται στην Εικόνα 3 (θεωρούμε ότι επιλέξαμε να κάνουμε την 3άδα με τα μπλε ζαχαρωτά στην κάτω γραμμή).



Εικόνα 3: Το ταμπλό στην χρονική στιγμή t_1 μετά την αναδιάταξη των ζαχαρωτών. Επιστρέφεται καλώντας την συνάρτηση boardAfterFirstMove()

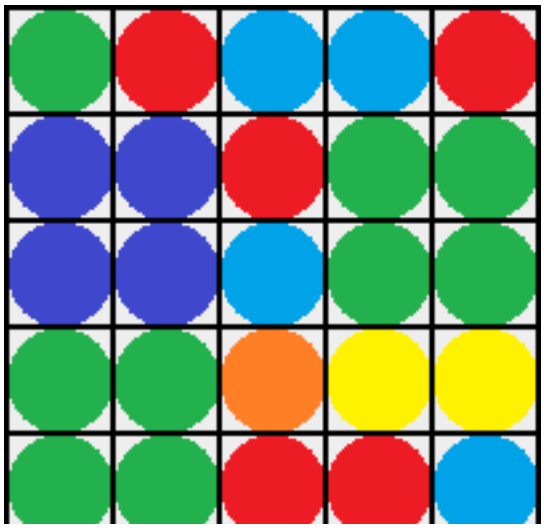
- `Board boardAfterFirstCrush(Board board, int[] move):` Η συνάρτηση αυτή παίρνει ως ορίσματα το ταμπλό της χρονικής στιγμής t_o board, την κίνηση move που θέλουμε να εκτελέσουμε (ομοίως με επάνω) και επιστρέφει ένα αντίγραφο του

ταμπλό με σπασμένα τα ζαχαρωτά που συμμετείχαν σε 3άδα η οποία δημιουργήθηκε λόγω της κίνηση move. Στο παράδειγμά μας το ταμπλό που θα επιστρέφονταν θα ήταν αυτό που φαίνεται στην Εικόνα 4 (θεωρούμε ότι επιλέξαμε να κάνουμε την 3άδα με τα μπλε ζαχαρωτά στην κάτω γραμμή, και η τριάδα αυτή έσπασε).



Εικόνα 4: Το ταμπλό μετά το σπάσιμο τον τριάδων που δημιουργήθηκαν από την κίνηση move. Επιστρέφεται από την `boardAfterFirstCrush()`

- `Board boardAfterFullMove(Board board, int[] move)`: Η συνάρτηση αυτή παίρνει ως ορίσματα το ταμπλό της χρονικής στιγμής t_o board, την κίνηση move που θέλουμε να εκτελέσουμε (ομοίως με πάνω) και επιστρέφει ένα αντίγραφο του ταμπλό με σπασμένα τα ζαχαρωτά που συμμετείχαν σε 3άδα η οποία δημιουργήθηκε λόγω της κίνηση move, καθώς και σπασμένα όλα τα πιθανά chain moves. Στην ουσία επιστρέφει τα ταμπλό στην κατάσταση που θα το βρει ο αντίπαλος όταν πάει να παίξει την δική του κίνηση. Στο παράδειγμά μας το ταμπλό που θα επιστρέφονταν θα ήταν αυτό που φαίνεται στην Εικόνα 5 (θεωρούμε ότι επιλέξαμε να κάνουμε την 3άδα με τα μπλε ζαχαρωτά στην κάτω γραμμή, η τριάδα αυτή έσπασε, και στην συνέχεια έσπασε και η chain τριάδα που δημιουργήθηκε στα κίτρινα).



Εικόνα 5: Το ταμπλό μετά το σπάσιμο τον τριάδων που δημιουργήθηκαν από την κίνηση move και το σπάσιμο όλων των πιθανών chain moves. Επιστρέφεται από την `boardAfterFullMove()`

- `Board boardAfterDeletingNples(Board board)`: Η συνάρτηση αυτή παίρνει ως όρισμα το ταμπλό σε κάποια χρονική στιγμή **t** και επιστρέφει ένα αντίγραφό του με διαγραμμένες τις ήδη υπάρχουσες ν-άδες που τυχόν υπάρχουν. Τοποθετεί στην πάνω γραμμή του ταμπλό ζαχαρωτά με χρώμα -1 (άχρωμα).

Όλες τις παραπάνω συναρτήσεις μπορείτε να τις καλέσετε χρησιμοποιώντας ένα επιπλέον όρισμα. Για παράδειγμα μπορείτε να καλέσετε την clone ως εξής:

- `CrushUtilities.cloneBoard(Board board, int numOfRowsToKeep)`: Σε αυτή την περίπτωση το αντικείμενο που θα σας επιστραφεί θα είναι της κλάσης Board και θα είναι αντίγραφο του αρχικού board που όμως θα περιλαμβάνει μόνο όσες γραμμές ορίζονται στο όρισμα `numOfRowsToKeep`. Αν δηλαδή το `Board A` έχει μέγεθος 1000 γραμμών η `CrushUtilities.cloneBoard(A,100)` θα σας επιστρέψει ένα αντίγραφο των πρώτων 60 γραμμών του A.

Μπορείτε να καλέσετε όλες τις συναρτήσεις που περιγράφονται σε αυτήν την παράγραφο με αυτό τον τρόπο, δηλαδή θέτοντας ως τελευταίο όρισμα το πλήθος των γραμμών που θέλετε να κρατήσετε. Θα σας φανούν χρήσιμες στην περίπτωση που θέλετε να επιταχύνετε τους υπολογισμούς του MinMax παίκτη σας ή αν θέλετε να αυξήσετε το βάθος του δέντρου που θα υλοποίησετε.

Αξιολόγηση Κατάστασης Ταμπλό

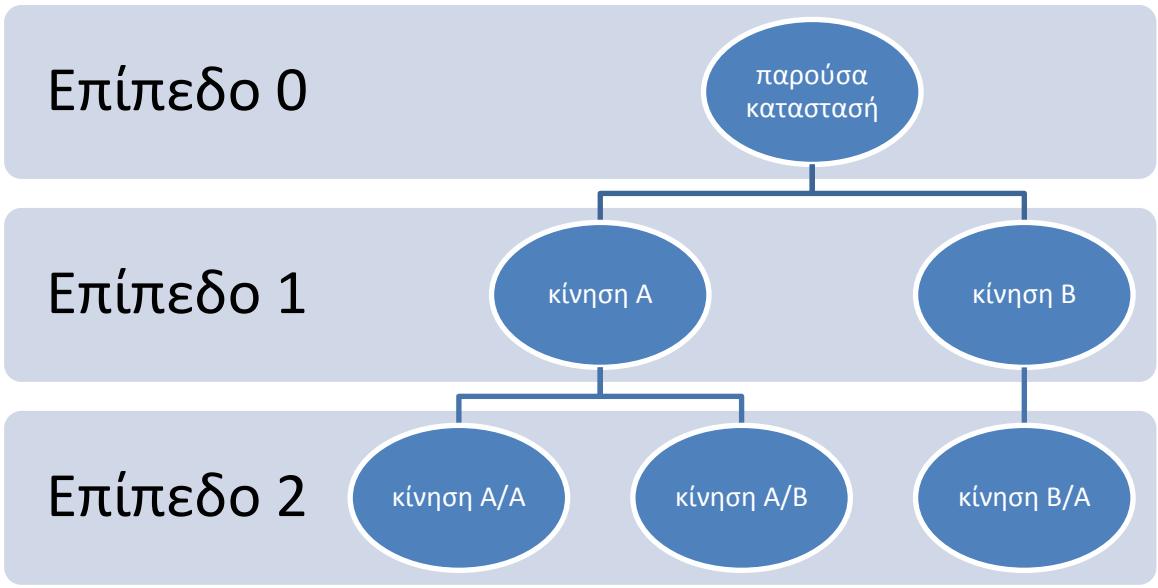
Για να χρησιμοποιήσετε την αξιολόγηση που είχατε κάνει στην προηγούμενη εργασία θα πρέπει να κάνετε κάποιες (μικρές) αλλαγές. Οι περισσότερες είναι αποτέλεσμα του κώδικα που γράφτηκε στην πλατφόρμα για να σας βοηθήσει. Πιο συγκεκριμένα:

1. Στην παρούσα εργασία σας δίνουμε βοηθητικές συναρτήσεις ώστε να μπορέσετε να υπολογίσετε και την ανταμοιβή του παίκτη σας από τα chain moves. Αν στην $2^{η}$ εργασία για να υπολογίσετε το πόσα πλακίδια σπάνε φτιάξατε συνάρτηση που διατρέχει **όλο** το ταμπλό για πιθανές τριάδες, τότε μπορείτε να χρησιμοποιήσετε την ίδια συνάρτηση, σε συνδυασμό με τις βοηθητικές συναρτήσεις που σας δίνουμε, για τον υπολογισμό των chain moves. Διαφορετικά θα πρέπει να την δημιουργήσετε.

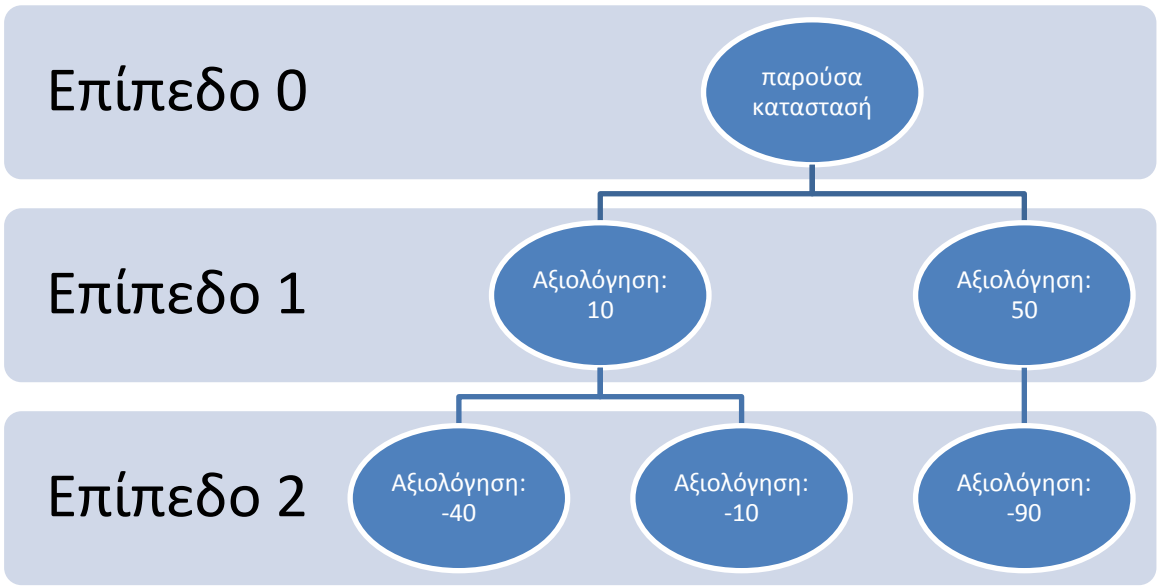
2. Επειδή η αξιολόγηση στο $2^{ο}$ επίπεδο (που είναι το ελάχιστο που ζητείται να υλοποιήσετε) γίνεται για την κατάσταση του ταμπλό για κίνηση του αντιπάλου σας, θα πρέπει να την αξιολογήσετε ως το αρνητικό της τιμής αξιολόγησης σας (πχ αν η evaluate σας για την κίνηση του αντιπάλου σας δίνει 50 τότε η πραγματική της τιμή είναι -50). Σε περίπτωση που κάνετε δέντρο μεγαλύτερου βάθους, θα πρέπει να το κάνετε αυτό κάθε φορά που αξιολογείτε την κατάσταση του ταμπλό για κίνηση του αντιπάλου.

3. Ο αλγόριθμος MinMax στην συγκεκριμένη περίπτωση έχει μια ιδιαιτερότητα. Επειδή έχουμε διάφορες καταστάσεις ταμπλό που το τι παίζει ο κάθε παίκτης επηρεάζει την βαθμολογία της κίνησης του επόμενου παίκτη, θα πρέπει να αξιολογείται ΚΑΘΕ επίπεδο καθώς κατεβαίνετε και να λαμβάνετε υπόψη σας την επιμέρους βαθμολογία την αξιολόγηση του κατώτερου επιπέδου. Όταν η αξιολόγηση φτάσει στο τέλος, δηλαδή έχετε αξιολογήσει το τελευταίο επίπεδο, τότε κάνετε το MinMax αλγόριθμο κανονικά, μη λαμβάνοντας υπόψη τις αξιολογήσεις των ενδιάμεσων επιπέδων. Αυτό γίνεται γιατί τα έχετε συμπεριλάβει στην αξιολόγηση του τελικού επιπέδου.

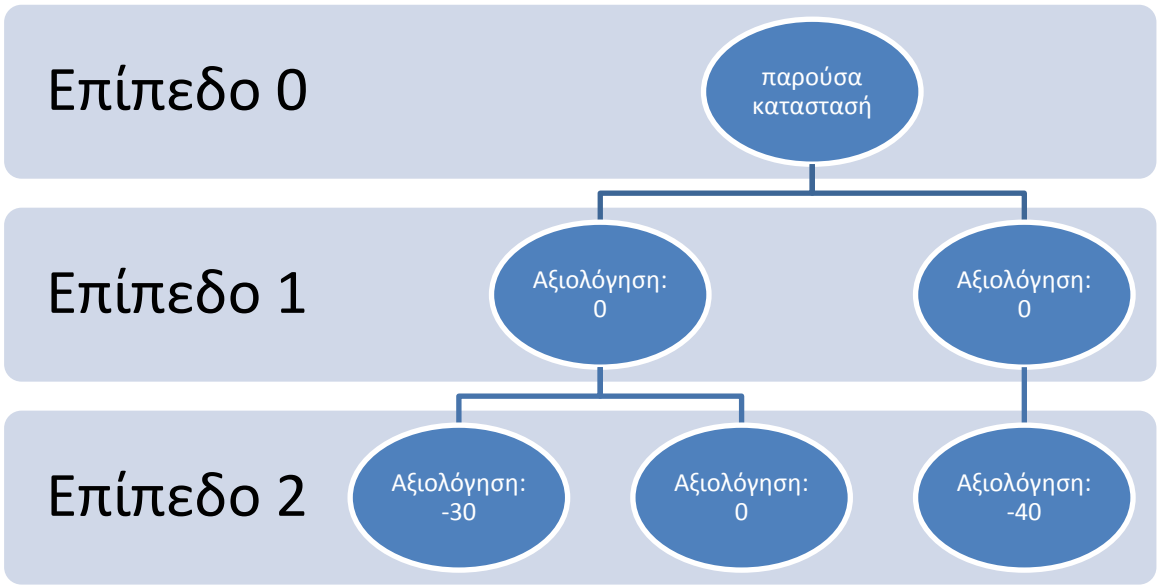
   Παράδειγμα: Έστω το δέντρο βάθους δύο που δίνεται παρακάτω.



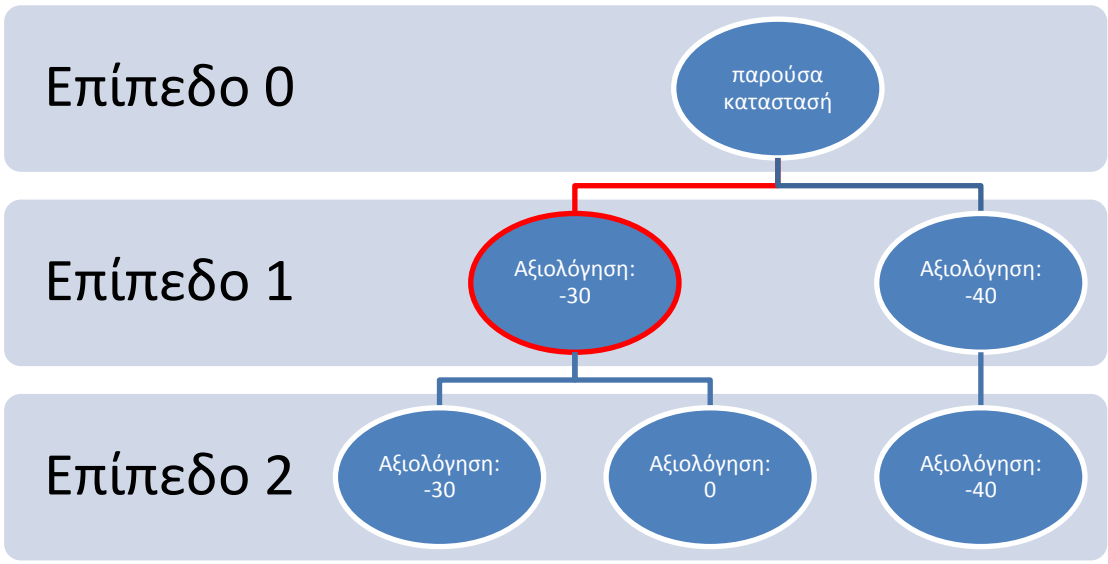
Στο επίπεδο 0, έχετε την κατάσταση του ταμπλό όπως είναι τώρα. Στο επίπεδο 1, έχετε την κατάσταση του ταμπλό όπως είναι αν μετακινήσετε τα ζαχαρωτά σύμφωνα με την διαθέσιμη κίνηση Α. Στο επίπεδο 2, έχετε την κατάσταση του ταμπλό αφού έχετε παίξει εσείς την κίνησή σας και ενώ ο αντίπαλος έχει επιλέξει την δική του διαθέσιμη κίνηση Α, και ουτω καθ'εξής.



Στην επόμενη φάση, θα πρέπει να δημιουργήσετε το δέντρο και να τρέξετε την συνάρτηση αξιολόγησης **για κάθε ένα από τα επίπεδα του δέντρου (και όχι μόνο για το τελευταίο)**. Αυτό γίνεται γιατί χρειάζεται να λάβετε υπόψη σας και την αξιολόγηση της δικής σας κίνησης που οδήγησε σε εκείνο το φύλλο του δέντρου.



Στην τελική κατάσταση, πριν ξεκινήσει η διαδικασία του MinMax αλγορίθμου, προσθέτουμε την τιμή της αξιολόγησης του ενδιάμεσου επιπέδου στην τιμή των φύλλων. Στην συνέχεια, μηδενίζουμε (ή απειρίζουμε) τις τιμές στο πρώτο επίπεδο, καθώς οι τελικές τιμές του κάθε κόμβου θα υπολογιστούν από τα φύλλα στον αλγόριθμο MinMax (βλέπε σχήμα που ακολουθεί).



Το αρχείο boardConfig.properties

Μέσα στο project του eclipse θα βρείτε το αρχείο boardConfig.properties το οποίο περιέχει 2 μεταβλητές.

- numOfRows : Ορίζει την τιμή του συνόλου των γραμμών του board.
- scoreLimit: Ορίζει το σύνολο των βαθμών που πρέπει να μαζέψει κάποιος παίκτης για να κερδίσει.

Είστε ελεύθεροι να αλλάξετε τις τιμές των μεταβλητών αυτών για να δείτε πως συμπεριφέρεται ο παίκτης σας.

Η κλάση gr.auth.ee.dsproject.crush.player.HeuristicPlayer

Μπορείτε να χρησιμοποιήστε αυτή την κλάση αν θέλετε να χρησιμοποιήσετε τον HeuristicPlayer που δημιουργήσατε στο δεύτερο παραδοτέο και να το συγκρίνετε με τον MinMax. Για να το κάνετε αυτό θα πρέπει να αντιγράψετε τα περιεχόμενα της κλάσης που δημιουργήσατε στο 2^{ο} παραδοτέο μέσα στην κλάση HeuristicPlayer του πακέτου gr.auth.ee.dsproject.crush.player.

Προσοχή στην εισαγωγή των πακέτων στην αρχή του κώδικα. Αν αντιγράψετε τις δηλώσεις import από την παλιά σας κλάση στην νέα, ο eclipse θα σας πει ότι έχετε λάθος.

Επίσης, αν στο προηγούμενο παραδοτέο χρησιμοποιήσατε την εντολή board.getRows()/2 για να σαρώσετε το ορατό τμήμα του board, θα πρέπει να το αντικαταστήσετε, όπως αναφέρεται και στην εκφώνηση, με την συνάρτηση board.getPRows() που επιστρέφει τις γραμμές που είναι ορατές στο ταμπλό.

Μετά τις απαραίτητες διορθώσεις, μπορείτε να χρησιμοποιήσετε τον παίκτη σας επιλέγοντάς τον από το μενού επιλογής του παιχνιδιού. Είναι ο παίκτης με το όνομα "Your Heuristic".

Αλγόριθμος Δημιουργίας Δέντρου για βάθος 2 κινήσεων

```
void getNextMove (ArrayList<int[]> availableMoves, Board board)
  Make a copy of the board as it is now using cloneBoard().
  Use this clone to create a new node which corresponds to the root of the tree.
  Call createMySubtree(root, 1)
  // The tree is now finished
  Call the chooseMinMaxMove(Node root)
  Choose the best move as input to calculateNextMove() function.

void createMySubtree(Node parent, int depth)
  Find the available moves of the board of the parent.
  For each available move on the board:
        Take the board after using the move using boardAfterMovingCandies().
        Create a new node as child of the parent node using new board state.
         Evaluate the node.
         Add the node as child of the parent node.
        Complete the tree branches by calling
             createOpponentSubtree(newNode, depth+1)


void createOpponentSubtree(Node parent, int depth)
  Create the new state of the board after a full move using boardAfterFullMove()
  Find the available moves of this new state of board.
  For each available move for the opponent's turn:
        Take the board after using the move using boardAfterMovingCandies().
        Create a new node as child of the parent node using new board state.
        Evaluate the new node as a negative evaluation of the state.
         Add the value of the parent node in the evaluation state of the node.
         Add the node as child of the parent node.


int chooseMinMaxMove(Node root)
  Implement a minmax algorithm to find the best available move.
  Return the index of the best available move.
```

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_PartC.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 να είναι με **λατινικούς χαρακτήρες**.

Προθεσμία υποβολής
Κώδικας και αναφορά Κυριακή 15 Ιανουαρίου, 23:59 (ηλεκτρονικά)
Δε θα υπάρξει καμία παρέκκλιση από την παραπάνω προθεσμία.