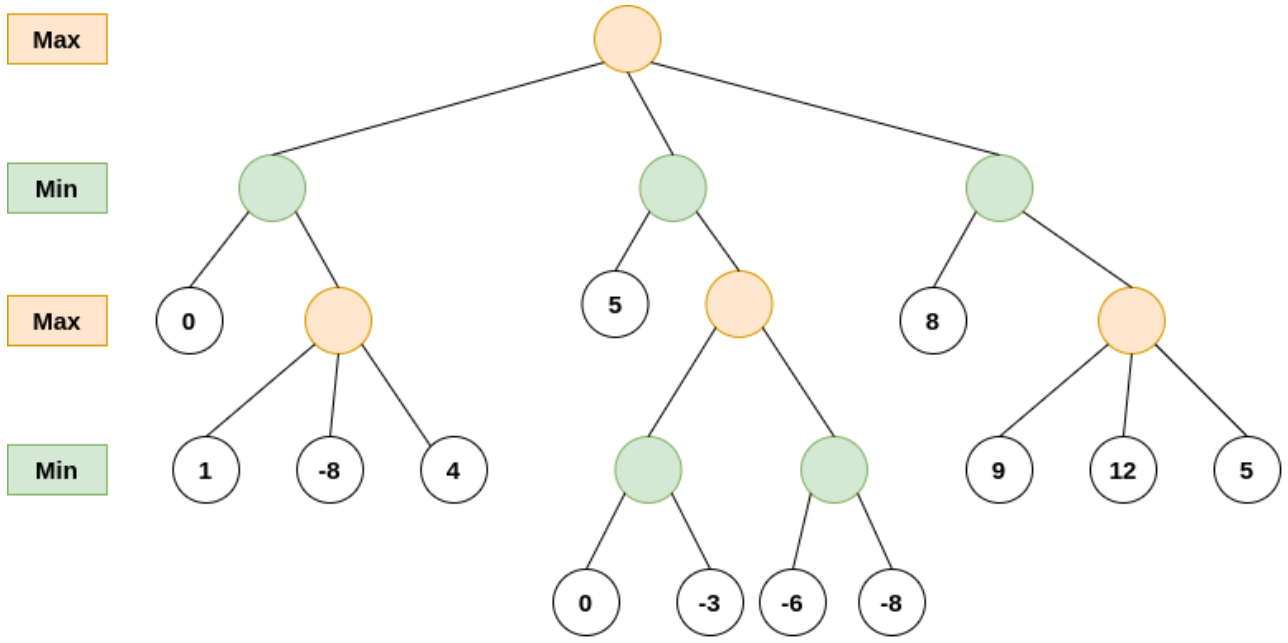
HW2 - MinMax Algorithm

Ημερομηνία παράδοσης: Κυριακή 26 Μαρτίου 2023

Σε αυτή την εργασία θα γράψετε ένα πρόγραμμα που υλοποιεί τον αλγόριθμο MinMax και την παραλλαγή του alpha-beta pruning. Ο αλγόριθμος MinMax εφαρμόζεται σε παιχνίδια στρατηγικής όπως το σκάκι, ντάμα, othello κ.α. Για την επίλυση του αλγορίθμου, το πρόγραμμα σας κατασκευάζει ένα δέντρο το οποίο έχει τα εξής χαρακτηριστικά:

- Ο βαθμός του δεν είναι εκ των προτέρων γνωστός.
- Οι εσωτερικοί κόμβοι δεν έχουν τιμές και έχουν ένα ή περισσότερα παιδιά.
- Με βάση τον αλγόριθμο MinMax οι εσωτερικοί κόμβοι επιλέγουν:
    - είτε τη μέγιστη τιμή από τις τιμές των παιδιών τους, λειτουργώντας ως maximizers.
    - είτε την ελάχιστη τιμή από τις τιμές των παιδιών τους, λειτουργώντας ως minimizers.
- Τα φύλλα είναι οι μόνοι κόμβοι που έχουν αρχικές τιμές. Τα φύλλα εκφράζουν καταστάσεις για τις οποίες μπορούμε να δώσουμε μία τιμή στο πρόβλημα με βάση κριτήρια που έχουμε ορίσει για το παιχνίδι.

Ας υποθέσουμε ότι έχουμε το παρακάτω δέντρο που εκφράζει τον αλγόριθμο Min-Max για ένα υποθετικό παιχνίδι στρατηγικής. Παρατηρήστε ότι όλα τα φύλλα δεν έχουν το ίδιο βάθος, ούτε όλοι οι κόμβοι έχουν το ίδιο βαθμό.



Το περιεχόμενο του παραπάνω δέντρου μπορεί να περιγραφεί σε μορφή JSON ως εξής:

Περιγραφή JSON

```
{
  "type": "max",
  "children": [
    {
      "type": "min",
      "children": [
        {
          "type": "leaf",
          "value": 0
        },
```

```
      {
        "type": "max",
        "children": [
          {
            "type": "leaf",
            "value": 1
          },
          {
            "type": "leaf",
            "value": -8
          },
          {
            "type": "leaf",
            "value": 4
          }
        ]
      }
    ]
  },
  {
    "type": "min",
    "children": [
      {
        "type": "leaf",
        "value": 5
      },
      {
        "type": "max",
        "children": [
          {
            "type": "min",
            "children": [
              {
                "type": "leaf",
                "value": 0
              },
              {
                "type": "leaf",
                "value": -3
              }
            ]
          },
          {
            "type": "min",
            "children": [
              {
                "type": "leaf",
                "value": -6
              },
              {
                "type": "leaf",
                "value": -8
              }
            ]
          }
        ]
      }
    ]
  },
  {
    "type": "min",
    "children": [
      {
        "type": "leaf",
        "value": 8
      },
      {
        "type": "max",
        "children": [
          {
            "type": "leaf",
            "value": -9
```

```
            },
            {
              "type": "leaf",
              "value": 12
            },
            {
              "type": "leaf",
              "value": 5
            }
          ]
        }
      ]
    }
  ]
}
```

Διαβάζοντας τα παραπάνω αλφαριθμητικά σε μορφή JSON θέλουμε το πρόγραμμα σας να κατασκευάζει το δέντρο του παραπάνω σχήματος όπως αυτό προσδιορίζεται με τη βοήθεια των κλάσεων που περιγράφονται παρακάτω. Όλες οι κλάσεις που ακολουθούν θα βρίσκονται στο πακέτο **`ce326.hw2`**. Τα ονόματα των κλάσεων θα επιλεγούν από εσάς.

Η βασική κλάση του κόμβου του δέντρου

Η βασική κλάση του κόμβου περιέχει ως μοναδικό `private` πεδίο, έναν αριθμό κινητής υποδιαστολής που αντιπροσωπεύει την τιμή του κόμβου.

Η κλάση περιέχει δύο κατασκευαστές α) το *default* κατασκευαστή και β) ένα κατασκευαστή που λαμβάνει ως παράμετρο έναν αριθμό κινητής υποδιαστολής, μέσω της οποίας αρχικοποιεί το μοναδικό πεδίο του κόμβου.

Η συγκεκριμένη κλάση μπορεί να περιγράψει ένα φύλλο του δέντρου, αλλά όχι έναν εσωτερικό κόμβο.

Η κλάση του εσωτερικού κόμβου του δέντρου

Η κλάση του εσωτερικού κόμβου είναι απόγονος της κλάσης του κόμβου:

- Ένα πίνακα που αποθηκεύονται τα παιδιά του κόμβου. Σκεφτείτε προσεκτικά τον τύπο του πίνακα που θα πρέπει να επιλέξετε ως πεδίο.
- Έναν αριθμό κινητής υποδιαστολής που αντιπροσωπεύει την παράμετρο **alpha** στον αλγόριθμο alpha-beta pruning.
- Έναν αριθμό κινητής υποδιαστολής που αντιπροσωπεύει την παράμετρο **beta** στον αλγόριθμο alpha-beta pruning.

Η κλάση περιέχει δύο κατασκευαστές:

- Το *default* κατασκευαστή χωρίς ορίσματα και
- Ένα κατασκευαστή που λαμβάνει ως μοναδική παράμετρο ένα πίνακα με τα παιδιά του κόμβου.

Οι αρχικές τιμές για τα πεδία **alpha** και **beta** είναι τιμές που τείνουν στο **-∞** και **+∞** αντίστοιχα. Δείτε σχετικά τις σταθερές **`java.lang.Double.MIX_VALUE`** και **`java.lang.Double.MAX_VALUE`**.

Επιπλέον, περιέχει κατ' ελάχιστον τις παρακάτω *public* μεθόδους:

- **`void setChildrenSize(int size)`**: Δηλώνει το μέγεθος του πίνακα που αποθηκεύονται τα παιδιά του κόμβου.
- **`int getChildrenSize()`**: Επιστρέφει το μέγεθος του πίνακα που αποθηκεύονται τα παιδιά του κόμβου.
- **`void insertChild(int pos, Node X)`**: θέτει τον κόμβο X στη θέση pos του πίνακα στον οποίο αποθηκεύονται τα παιδιά του τρέχοντος κόμβου.

- **`Node getChild(int pos)`:** Επιστρέφει τον κόμβο στη θέση pos του πίνακα στον οποίο αποθηκεύονται τα παιδιά του τρέχοντος κόμβου.

Η κλάση του maximizer κόμβου

Πρόκειται για έναν εσωτερικό κόμβο ο οποίος έχει την επιπλέον ιδιότητα να επιλέγει τη μέγιστη τιμή από τις τιμές των παιδιών του.

Η κλάση του minimizer κόμβου

Πρόκειται για ένα εσωτερικό κόμβο ο οποίος έχει την επιπλέον ιδιότητα να επιλέγει την ελάχιστη τιμή από τις τιμές των παιδιών του.

Η κλάση του δέντρου

Γράψτε την κλάση του δέντρου η οποία έχει ως μοναδικό *private* πεδίο των κόμβο της ρίζας του. Η κλάση έχει τους εξής κατασκευαστές:

- Ένα κατασκευαστή που λαμβάνει ως παράμετρο ένα String σε μορφή JSON και δημιουργεί το δέντρο. Ο κατασκευαστής εξετάζει αν πρόκειται για JSON πληροφορία. Εάν η πληροφορία δεν είναι σε μορφή JSON παράγει ένα `java.util.IllegalArgumentException`.
- Ένα κατασκευαστή που λαμβάνει ως παράμετρο ένα αντικείμενο τύπου `java.io.File`. Ανοίγει το αρχείο που δίνεται ως παράμετρος και διαβάζει ένα αλφαριθμητικό σε μορφή JSON από το οποίο κατασκευάζει το δέντρο. Εάν το αρχείο δεν υπάρχει ή δεν έχει δικαίωμα να το ανοίξει για διάβασμα παράγει `java.io.FileNotFoundException`. Εάν το περιεχόμενο του αρχείου δεν είναι σε μορφή JSON παράγει ένα `java.util.IllegalArgumentException`.

Η κλάση έχει τις εξής επιπλέον μεθόδους:

- **`double minMax()`:** Επιλύει τον αλγόριθμο MinMax για το δέντρο και επιστρέφει την τιμή που λαμβάνει η ρίζα του.
- **`int size()`:** Επιστρέφει τον αριθμό των κόμβων του δέντρου.
- **`ArrayList<Integer> optimalPath()`:** Επιστρέφει το μονοπάτι των κόμβων από τη ρίζα προς τα φύλλα που οδηγεί στη βέλτιστη επιλογή με βάση τον αλγόριθμο MinMax. Η πρώτη θέση της λίστας (θέση 0) αντιστοιχεί στον αριθμό παιδιού του κόμβου της ρίζας πάνω στο μονοπάτι, η δεύτερη θέση (θέση 1) στον αριθμό παιδιού του παιδιού της ρίζα πάνω στο μονοπάτι κ.ο.κ. Με τον όρο “αριθμός παιδιού”, αναφέρεται η αρίθμηση των παιδιών κάθε κόμβου, εάν ξεκινήσουμε να τα αριθμούμε από το αριστερότερο προς το δεξιότερο ξεκινώντας από το μηδέν (0).
- **`String toString()`:** Επιστρέφει ένα αλφαριθμητικό σε μορφή JSON. Εάν έχουν υπολογιστεί οι τιμές των εσωτερικών κόμβων, αυτές απεικονίζονται μέσω της επιπλέον παραμέτρου **`value`** (**`“value” = “X”`**, όπου X η τιμή του κόμβου).
- **`String toDOTString()`:** Επιστρέφει ένα αλφαριθμητικό κατάλληλο να διαβαστεί από το πρόγραμμα dot της σουίτας graphviz, εφόσον αυτό είναι αποθηκευμένο σε ένα αρχείο.
- **`void toFile(File file)`:** Δημιουργεί το αρχείο file για γράψιμο. Εάν το αρχείο υπάρχει παράγει ένα **`java.io.IOException`**. Διαφορετικά γράφει το περιεχόμενο της συνάρτησης **`toString`**.
- **`void toDotFile(File file)`:** Δημιουργεί το αρχείο file για γράψιμο. Εάν το αρχείο υπάρχει παράγει ένα **`java.io.IOException`**. Διαφορετικά γράφει το περιεχόμενο της συνάρτησης **`toDotString`**.

Η κλάση του βέλτιστου δέντρου

Γράψτε την κλάση του βέλτιστου δέντρου η οποία είναι απόγονος της κλάσης του δέντρου. Η κλάση υλοποιεί τον αλγόριθμο MinMax με τη βελτιστοποίηση alpha-beta pruning. Εφόσον εκτελεστεί ο αλγόριθμός MinMax διαφοροποιούνται οι παρακάτω συναρτήσεις:

- **`double minMax()`:** Επιλύει τον αλγόριθμο MinMax με τη βελτιστοποίηση alpha-beta pruning για το δέντρο και επιστρέφει την τιμή που λαμβάνει η ρίζα του.
- **`ArrayList<Integer> optimalPath()`:** Επιστρέφει το μονοπάτι των κόμβων από τη ρίζα προς τα φύλλα που οδηγεί στη βέλτιστη επιλογή με βάση τον αλγόριθμο minMax. Η πρώτη θέση της λίστας (θέση 0) αντιστοιχεί στο παιδί του κόμβου της ρίζας πάνω στο μονοπάτι, η δεύτερη θέση (θέση 1) στο παιδί του παιδιού της ρίζα πάνω στο μονοπάτι κ.ο.κ.
- **`double prunedNodes()`:** Επιστρέφει τον αριθμό των κόμβων που δεν υπολογίστηκαν (έχουν γίνει pruned).
- **`String toString()`:** Επιστρέφει ένα αλφαριθμητικό σε μορφή JSON. Εφόσον έχουν υπολογιστεί οι τιμές των εσωτερικών κόμβων, αυτές απεικονίζονται μέσω της επιπλέον παραμέτρου "**`value`**". Εάν ένας κόμβος έχει αγνοηθεί κατά τους υπολογισμούς του αλγορίθμου MinMax προστίθεται η παράμετρος "**`pruned`**" **`= true`**. Αν ο κόμβος δεν έχει αγνοηθεί η παράμετρος **`pruned`** δεν εμφανίζεται.
- **`String toDOTString()`:** Επιστρέφει ένα αλφαριθμητικό κατάλληλο να διαβαστεί από το πρόγραμμα dot, εφόσον αυτό είναι αποθηκευμένο σε ένα αρχείο. Εφόσον έχουν υπολογιστεί οι τιμές των εσωτερικών κόμβων, αυτές απεικονίζονται στο εσωτερικό των κόμβων. Οι κόμβοι που έχουν αγνοηθεί κατά τους υπολογισμούς του αλγορίθμου MinMax, εμφανίζονται σε κόκκινο πλαίσιο (περίγραμμα) αντί για μαύρο (στο αρχείο dot προστίθεται ως πληροφορία στον κόμβο το εξής: **`[color ="red"]`**.
- **`void toFile(File file)`:** Δημιουργεί το αρχείο file για γράψιμο. Εάν το αρχείο υπάρχει παράγει ένα **`java.io.IOException`**. Διαφορετικά γράφει το περιεχόμενο της συνάρτησης **`toString`**.
- **`void toDotFile(File file)`:** Δημιουργεί το αρχείο file για γράψιμο. Εάν το αρχείο υπάρχει παράγει ένα **`java.io.IOException`**. Διαφορετικά γράφει το περιεχόμενο της συνάρτησης **`toDotString`**.

Το κυρίως πρόγραμμα

Το κυρίως πρόγραμμα βρίσκεται στην κλάση **`ce326.hw2.MinMax`**. Το πρόγραμμα σε επανάληψη εκτυπώνει το παρακάτω μενού επιλογών :

```
-i <filename>   :  insert tree from file
-j [<filename>] :  print tree in the specified filename using JSON format
-d [<filename>] :  print tree in the specified filename using DOT format
-c              :  calculate tree using min-max algorithm
-p              :  calculate tree using min-max and alpha-beta pruning optimization
-q              :  quit this program
```

Στη συνέχεια ο χρήστης εισάγει μία από τις παρακάτω επιλογές:

- **`-i <filepath>:`** Το πρόγραμμα διαβάζει το περιεχόμενο του αρχείου `<filepath>`. Εάν το αρχείο δεν υπάρχει εκτυπώνεται το μήνυμα "`Unable to find file <filepath>`". Εάν το αρχείο υπάρχει αλλά ο χρήστης δεν έχει επαρκή δικαιώματα για διάβασμα του αρχείου εκτυπώνεται το μήνυμα "`Unable to open file <filepath>`". Εάν το αρχείο ανοιχθεί για διάβασμα, αλλά το περιεχόμενο του δεν είναι σε μορφή JSON ή το περιεχόμενο του δεν είναι σύμφωνο με τις προδιαγραφές της εργασίας εκτυπώνει το μήνυμα "`Invalid format`". Τα μηνύματα ακολουθούνται από χαρακτήρα αλλαγής γραμμής.
- **`-c:`** Υπολογίζεται ο αλγόριθμος MinMax για το δέντρο. Εκτυπώνεται η τιμή επιστροφής της συνάρτησης **`optimalPath`** για το δέντρο. Μετά από κάθε ακέραιο εκτυπώνεται κόμμα και κενός χαρακτήρας και μετά το τελευταίο ακέραιο χαρακτήρας αλλαγής γραμμής.
- **`-p:`** Υπολογίζεται ο αλγόριθμος MinMax με τη βελτιστοποίηση alpha-beta pruning. Εκτυπώνεται αριστερή αγκύλη ο συνολικός αριθμός των κόμβων του δέντρου, κόμμα, ο αριθμός των κόμβων που δεν έχουν υπολογιστεί (pruned κόμβοι), δεξιά αγκύλη και κενό. Στη συνέχεια, εκτυπώνεται η τιμή επιστροφής της συνάρτησης **`optimalPath`** για το δέντρο. Μετά από κάθε ακέραιο εκτυπώνεται κόμμα και κενός χαρακτήρας και μετά το τελευταίο ακέραιο χαρακτήρας αλλαγής γραμμής.
- **`-j [<filepath>]:`** Εκτυπώνεται στο αρχείο `<filepath>` το περιεχόμενο του δέντρου σε μορφή JSON. Το αρχείο `<filepath>` είναι προαιρετικό. Εφόσον απουσιάζει, η εκτύπωση γίνεται στο stdout.

Εάν το αρχείο `<filepath>` υπάρχει στο σύστημα εκτυπώνεται το μήνυμα **"File already exists!"** ακολουθούμενο από χαρακτήρα αλλαγής γραμμής και το αρχείο δε μεταβάλλεται.

- **`-d [<filepath>]:`** Εκτυπώνεται στο αρχείο `<filepath>` περιεχόμενο του δέντρου σε μορφή κατάλληλη για είσοδο στο πρόγραμμα dot της σουίτας graphviz. Εάν το αρχείο `<filepath>` υπάρχει στο σύστημα εκτυπώνεται το μήνυμα "File already exists!" και το αρχείο δε μεταβάλλεται.
- **`-q:`** η επανάληψη σταματά και το πρόγραμμα τερματίζει.
- Οποιαδήποτε άλλη επιλογή, επαναλαμβάνει την εκτύπωση του μενού.

Οδηγίες Αποστολής

Η αποστολή της εργασίας θα γίνει μέσω της πλατφόρμας autolab. Για την υποβολή ακολουθήστε τα εξής βήματα.

- **Υποβολή κώδικα Java:** Συμπιέστε το περιεχόμενο του καταλόγου **`hw2`** μέσα στον οποίο βρίσκονται όλα τα αρχεία με κατάληξη **`.java`** της εργασίας, σε μορφή zip. Το αρχείο που προκύπτει πρέπει να έχει όνομα **`hw2.zip`**.
- Συνδέεστε στο autolab και επιλέγετε το μάθημα **`ECE326_2023 (S23)`** και από αυτό την εργασία **`HW3`**.
- Για να υποβάλετε την εργασία σας κάνετε click στην επιλογή "I affirm that I have compiled with this course academic integrity policy…" και πατάτε submit. Στη συνέχεια επιλέγετε το αρχείο **`hw3.zip`** που δημιουργήσατε παραπάνω.