ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Τεχνητή Νοημοσύνη

Χειμερινό Εξάμηνο 2021-2022

Εργασία: 1

Απόστολος Γρηγόρης 3190335

Δέσποινα Παπαδοπούλου 3180146

Ευάγγελος Γεωργακόπουλος 3150019

Το πρόβλημα των κανιβάλων και ιεραποστόλων

Δομή του project:

State.java: είναι η κλάση στην οποία αποθηκεύει τις πληροφορίες που χρειάζεται να έχει κάθε κατάσταση του προβλήματος και δημιουργεί τα παιδιά που προκύπτουν από ένα δεδομένο State, ελέγχοντας αν αυτό είναι αποδεχτό.

Μεταβλητές:

- leftChemicalA, rightChemicalA: μετρητές οι οποίοι αποθηκεύουν το πλήθος των χημικών ενώσεων Α αριστερά και δεξιά πλευρά, αντίστοιχα, στην δεδομένη κατάσταση.
- leftChemicalB, rightChemicalB: μετρητές οι οποίοι αποθηκεύουν το πλήθος των χημικών ενώσεων Β αριστερά και δεξιά πλευρά, αντίστοιχα, στην δεδομένη κατάσταση.
- dropper: μια Boolean μεταβλητή η οποία αποθηκεύει την θέση του σταγονόμετρου το οποίο μας βοηθάει για την μεταφορά των χημικών στοιχείων, όπου false σημαίνει ότι βρίσκεται στην αριστερή πλευρά ενώ true στην δεξιά πλευρά, για την δεδομένη κατάσταση.
- father: μεταβλητή State η οποία αποθηκεύει τον πατέρα, από τον οποίο προέκυψε ως παιδί. Το πρώτο state του προβλήματος έχω ως πατέρα null.
- hypothesis: η μεταβλητή στην οποία έχει την πρόβλεψη μας ότι θα είναι το συνολικό πλήθος βημάτων που θα προκύψει από το δεδομένο State στην τελική κατάσταση. Αυτό υπολογίζεται από τις μετακινήσεις που έχουμε κάνει και τις επιπλέων κινήσεις που πρόκειται να συμβούν με βάση την ευρετική.
- moves: οι κινήσεις που προέκυψαν μέχρι την δεδομένη κατάσταση

- game: μια μεταβλητή τύπου Game από την οποία παίρνουμε τις πληροφορίες που έχουμε ως είσοδο από το χρήστη.

Συναρτήσεις:

- isTerminal: πρόγραμμα το οποίο ελέγχει αν η κατάσταση είναι τελική, δηλαδή αν στα αριστερά δεν υπάρχει καμία χημική ένωση, είτε A είτε B.
- isAcceptable: ελέγχει αν η κατάσταση είναι αποδεχτεί με βάση τους κανόνες, δηλαδή η B να ξεπερνούν ή να είναι ίσο πλήθος με των A, τόσο στην αριστερά όσο και στην δεξιά πλευρά.
- createChild: δημιουργεί τα παιδιά της δεδομένης κατάστασης. Αρχικά δημιουργεί μια κενή λίστα State με την ονομασία toReturn, στην οποία θα βάλουμε τα πιθανά παιδιά της δεδομένης κατάστασης και θα επιστρέψουμε κατά το τέλος του προγράμματος. Επίσης δημιουργούμε αντίγραφο της δεδομένης κατάστασης σε μια μεταβλητή father, που θα χρησιμοποιήσουμε για την δημιουργία των παιδιών του κατά την κλήση του constructor δεύτερου. Ύστερα ελέγχουμε σε ποια πλευρά βρίσκεται το σταγονόμετρο , με την χρήση της μεταβλητής dropper, ώστε να ξέρουμε πως να διαχειριστούμε τους μετρητές. Στην περίπτωση που το σταγονόμετρο βρίσκεται στα δεξιά, τότε ξεκινάμε ένα βρόχο for όπου με c από 0 έως το ελάχιστο μεταξύ το μέγιστο πλήθος που μπορεί να έχει το σταγονόμετρο και το πλήθος των B που υπάρχουν στην δεξιά πλευρά δημιουργεί καταστάσεις όπου αναχωρούν c πλήθος από την δεξιά πλευρά και μεταφέρονται στην απέναντι πλευρά. Βέβαια προτού υπολογίσει την υπόθεση του και το εισάγει στην λίστα, ελέγχουμε αν είναι μια αποδεχτεί κατάσταση. Να σημειωθεί πως σε αυτή την φάση του προγράμματος δεν μετακινούμε κανέναν A, γιατί θέλουμε να πάρουμε τις περιπτώσεις όπου μετακινούμε μόνο B. Το πλήθος των A το μετακινούμε μέσα σε έναν εμφωλευμένο βρόχο for, στον οποίο ξεκινάμε από m=c, ώστε να τηρούμε το γεγονός ότι θα ισχύει ο περιορισμός και στην σταγονόμετρο , μέχρι το ελάχιστο αριθμό από το πλήθος των υποψήφιων θέσεων που υπάρχουν αν αφαιρέσουμε τους c A και το πλήθος των B που υπάρχουν στη δεξιά πλευρά. Διατηρώντας την ίδια λογική, δημιουργούμε ένα νέο State με τις αλλαγές στους τέσσερις μετρητές και στον σταγονόμετρο και ελέγχουμε αν είναι επιθυμητή κατάσταση. Αν είναι επιθυμητή, τότε υπολογίζουμε την υπόθεση της και την τοποθετούμε στην λίστα. Την ίδια ακριβώς διαδικασία αλλά με τους αντίστροφους μετρητές ακολουθούμε και για την περίπτωση που η σταγονόμετρο βρίσκεται στα αριστερά.
- heuristic: ακολουθώντας την λογική της άσκησης μελέτης 4.3, υλοποιήσαμε σε κώδικα την ευρετική. Αρχικά ελέγχουμε αν είναι τερματική ώστε να επιστρέψουμε κατευθείαν μηδέν. Αλλιώς δημιουργούμε τον μετρητή κόστος, όπου θα εκχωρήσουμε την τιμή που θα επιστρέψουμε και έναν αθροιστή όπου έχουμε αποθήκευση το άθροισμα των δυο μετρητών που αφορούν το πλήθος των χημικών ενώσεων στην αριστερή πλευρά. Ύστερα ανάλογα σε ποια πλευρά είναι η σταγονόμετρο και το πλήθος των αθροιστή, το οποίο αφορά κυρίως αν είμαστε στα αριστερά, εισάγουμε την τιμή στο κόστος, όπως ορίζεται στην μελέτης 4.3 και το οποίο εν τέλη επιστρέφουμε.
- currentState: ασχολείται με την προβολή της δεδομένης κατάστασης.

SpaceSearcher.java: σε αυτή την κλάση γίνεται υλοποίηση της A*.

Μεταβλητές:

- q: είναι μια ούρα PQueue όπου εκεί θα αποθηκεύουμε τα παιδιά μιας κατάστασης ταξινομημένα έτσι ώστε να είναι πρώτο στην ουρά η κατάσταση με την μικρότερη υπόθεση.
- closedSet: ένα HashSet από State τα οποία αποτελούν το path το οποίο ακολουθήσαμε προκειμένου να φτάσαμε στην τελική κατάσταση.

Συναρτήσεις:

- AstarClosedSet(State initialState): είναι η συνάρτηση η οποία με βάση το A* επιστρέφει την τελική κατάσταση. Ξεκινάμε με την αρχικοποίηση των μεταβλητών q και closedSet. Ύστερα κάνουμε add του initialState στην q. Μετά όσο ακόμα υπάρχουν στοιχεία στην ουρά εκτελούμε ένα πλήθος εντολών. Αρχικά δημιουργούμε μια μεταβλητή State currentState, όπου παίρνει ως είσοδο την κατάσταση της ουράς με την μικρότερη υπόθεση. Αν η κατάσταση αυτή είναι η τελική, τότε την επιστρέφει και τερματίζει η συνάρτηση. Αλλιώς, ελέγχει αν αυτή η κατάσταση περιέχεται στο closedSet, ώστε να μην αποθηκεύουμε διπλές καταστάσεις. Αν δεν υπάρχει, τότε την εκχωρούμε στο closedSet και δημιουργούμε τα παιδιά του με την κλήση της createChild(), τα οποία την αποθηκεύουμε σε μια λίστα. Στο τέλος κάνουμε add μια προς μια τις καταστάσεις της λίστας στην ουρά, ώστε να μπορέσει η ουρά να τα ταξινομήσει.

PQueue.java: υλοποιεί μια ουρά, σε μορφή πίνακα, η οποία αποθηκεύει αντικείμενα State σε αύξουσα σειρά με βάση την τιμή της hypothesis. Μέσα σε αυτήν έχουμε τις συναρτήσεις που μας εξυπηρετούν ώστε κάθε φόρα να βρίσκουμε το αντικείμενο με την ελάχιστη hypothesis.

IntegerComparator.java: μια κλάση όπου υλοποιούμε τον τρόπο σύγκρισης που χρειαζόμαστε στην PQueue.

Game.java: η κλάση στην οποία φτιάχνουμε το παιχνίδι του προβλήματος ιεραπόστολοι και κανίβαλοι. Οι μεταβλητές M,N και K είναι αντίστοιχα: το μέγιστο πλήθος που μπορεί να έχει η σταγονόμετρο , το πλήθος των χημικών ενώσεων για κάθε κατηγορία και το μέγιστο πλήθος κινήσεων, όπως και ορίζεται στην εργασία. Στην **main** για όσο είναι αληθής:

- συλλέγω τις πληροφορίες που χρειάζομαι από τον χρήστη και δημιουργώ ένα αντικείμενο Game.
- δημιουργώ ένα αντικείμενο SpaceSearcher και ένα State με αρχικοποίηση των μεταβλητών με τον εξής τρόπο:
    - οι μετρητές που αφορούν τις χημικές ενώσεις που βρίσκονται στα αριστερά να έχουν τιμή N, ενώ για τους μετρητές στην απέναντι πλευρά να είναι μηδέν
    - η σταγονόμετρο έχει την τιμή false, καθώς ξεκινάμε από την αριστερή πλευρά
    - ο μετρητής moves είναι μηδέν, γιατί δεν έχουμε κάνει καμία μεταφορά ακόμα
    - ο father είναι null, γιατί είναι το αρχικό State
    - και τέλος ο game έχει ως τιμή το game που μόλις δημιουργήσαμε

- αποθηκεύω το αποτέλεσμα που παράγει ο SpaceSearcher, εκτελώντας την συνάρτηση AstarClosedSet με είσοδο την αρχική κατάσταση, σε μια μεταβλητή State. Για να χρονομετρήσουμε την διαδικασία πρώτου και αφού καλέσουμε το AstarClosedSet εκτελούμε το currentTimeMillis().
- αν το State είναι κενό, τότε σημαίνει ότι δεν μπορέσαμε να βρούμε τελική κατάσταση
- σε περίπτωση που υπάρχει τελική κατάσταση, αποθηκεύουμε σε μια λίστα την διαδρομή που ακολουθήσαμε ώστε να φτάσουμε σε αυτό το σημείο, καλώντας αναδρομικά την εντολή getFather().
- ύστερα ελέγχουμε αν οι κινήσεις που κάναμε είναι μεγαλύτερες από το μέγιστο πλήθος βημάτων που όρισε ο χρήστης. Αν αληθεύει, τότε παρουσιάζει ένα ανάλογο μήνυμα.
- σε περίπτωση, όμως, που πληρούμε την προϋπόθεση των βημάτων, τότε με την κλήση currectState() προβάλουμε όλα τα States της διαδρομής στην τελική κατάσταση.
- προτού κλείσουμε το πρόγραμμα ζητάμε αν ο χρήστης θέλει να ξανά τρέξει το πρόγραμμα από την αρχή.

Το πρόβλημα των Ν Βασιλισσών

Υλοποίηση με το γενετικό αλγόριθμο

Δομή του project:

BoardPanel.java: είναι το αρχείο που δέχεται τις τιμές των μεταβλητών από το ControlPanel, δημιουργεί αρχικό τυχαίο πληθυσμό, έπειτα επιλέγει από αυτό, χρωμοσώματα που είναι πιο κατάλληλα για γονείς με μια τυχαιότητα και από εκείνους παράγει παιδιά τα οποία μεταλλάσσει με μια τυχαιότητα. Τέλος, επιλέγει το πιο κατάλληλο χρωμόσωμα του πληθυσμού και το εκτυπώνει σε μορφή σκακιέρας. Αυτό το κάνει και για τις επόμενες γενιές μέχρι να βρεθεί χρωμόσωμα στο οποίο είτε δεν απειλείται καμία Βασίλισσα, είτε έχει παρέλθει ο μέγιστος επιτρεπτός χρόνος εκτέλεσης που έχει δοθεί σαν είσοδος.

Μεταβλητές:

- numOfQueens: αριθμός Βασιλισσών
- populationSize: αριθμός χρωμοσωμάτων του πληθυσμού κάθε γενιάς
- maxTime: μέγιστος επιτρεπτός χρόνος εκτέλεσης
- start: στιγμή εκκίνησης
- mutationChance: πιθανότητα μετάλλαξης χρωμοσώματος
- selectionFactor: παράγοντας που χρησιμοποιείται στην επιλογή γονέων
- bestChrom: το καταλληλότερο χρωμόσωμα της γενιάς
- isFinished: αν έχει τελειώσει η εκτέλεση, είτε λόγω χρόνου, είτε λόγω εύρεσης λύσης
- pauseTime: χρόνος παύσης, ώστε να προλαβαίνουν να φαίνονται στη σκακιέρα οι διαφορετικού συνδυασμοί Βασιλισσών που εκτυπώνονται
- width, height, tileWidth: αφορούν διαστάσεις της σκακιέρας
- population: πίνακας με όλα τα χρωμοσώματα της γενιάς

Constructor:

Κρατάει χρόνο εκκίνησης, δημιουργεί τυχαίο πληθυσμό, ορίζει διαστάσεις παραθύρου, βρίσκει το καλύτερο χρωμόσωμα της γενιάς και το ζωγραφίζει.

Συναρτήσεις:

- paint: Ζωγραφίζει τη σκακιέρα και τις Βασίλισσες
- update: ελέγχει το χρόνο εκτέλεσης και το σκορ του bestChrom.
  Αν δεν έχει παρέλθει το maxTime και το bestChrom δεν είναι τελική κατάσταση, κάνει παύση την εκτέλεση του προγράμματος για pauseTime milliseconds. Έπειτα, παράγει καινούργιο πληθυσμό χρωμοσωμάτων ως παιδιά του τρέχοντος πληθυσμού, βρίσκει το χρωμόσωμα με το καλύτερο σκορ και το ζωγραφίζει.
  Αλλιώς, θέτει την isFinished = true
- restart: Παίρνει ως ορίσματα τις εισόδους που έχει περάσει ο χρήστης στην controlPanel, αρχικοποιεί από αυτά τις μεταβλητές της κλάσης, δημιουργεί τυχαίο πληθυσμό σαν πρώτη γενιά, βρίσκει το καλύτερο χρωμόσωμά της και το ζωγραφίζει.
- Reproduce: παίρνει σαν όρισμα 2 γονείς, τους κόβει σε τυχαίο σημείο και παράγει από αυτούς 2 παιδιά (ενώνοντας το πρώτο μέρος του ενός με το δεύτερο του άλλου και το πρώτο μέρος του τελευταίου με το δεύτερο του πρώτου). Έπειτα, τα μεταλλάσσει με την mutatePositions και τα επιστρέφει.
- mutatePositions: δέχεται σαν όρισμα ένα χρωμόσωμα και χρησιμοποιώντας τη random αποφασίζει αν θα το μεταλλάξει. Αν ναι, επιλέγει με τη random ποιο στοιχείο του θα μεταλλάξει και με τη random πάλι τι τιμή θα πάρει το στοιχείο αυτό. Τέλος, επιστρέφει το χρωμόσωμα.
- combineArrays: δέχεται σαν όρισμα 2 πίνακες, τους ενώνει και τους επιστρέφει σαν έναν.
- cutArray: δέχεται σαν όρισμα έναν πίνακα και 2 ακεραίους. Επιστρέφει τον υποπίνακά του, που ξεκινά από τη θέση που δηλώνει ο πρώτος ακέραιος και τελειώνει στη θέση που δηλώνει ο δεύτερος.
- selectParents: δέχεται σαν όρισμα ταξινομημένο τον πληθυσμό που πρόκειται να παράξει παιδιά. Από αυτόν επιλέγει 2 γονείς και τους επιστρέφει. Οι γονείς επιλέγονται με τον εξής τρόπο. Χρησιμοποιείται η random, της οποίας το αποτέλεσμα υψώνεται στη δύναμη selectionFactor και έπειτα η παραγμένη τιμή πολλαπλασιάζεται με το populationSize. Γονέας γίνεται το στοιχείο του πίνακα που έχει ως index το ακέραιο μέρος του αποτελέσματος του υπολογισμού.
  Συνειδητοποιούμε ότι από τη στιγμή που τα καλύτερα χρωμοσώματα είναι στην αρχή του πίνακα, όσο μεγαλύτερη η τιμή του selectionFactor, τόσο μεγαλύτερη η πιθανότητα να επιλεγούν καλύτερα χρωμοσώματα.
- isFinished: επιστρέφει την isFinished.
- getBestChrom: προσπελαύνει τον τρέχων πληθυσμό και επιστρέφει το στοιχείο με το καλύτερο σκορ.
- createRandomPopulation: δημιουργεί έναν πίνακα Chromosome με μέγεθος populationSize και τον γεμίζει κάνοντας χρήση της createRandomPositions.
- createRandomPositions: επιστρέφει πίνακα ακεραίων μεγέθους numOfQueens με τυχαίες τιμές 0 εώς numOfQueens -1
- loadImage: φορτώνει το εικονίδιο της Βασίλισσας
- drawQueens: ζωγραφίζει τις Βασίλισσες
- drawBoard: ζωγραφίζει τη σκακιέρα

ControlPanel.java: Είναι το αρχείο που αφορά το διαχειριστικό παράθυρο, μέσω του οποίου ο χρήστης περνάει εισόδους στο πρόγραμμα και το οποίο εμφανίζει το fitness του τρέχοντος χρωμοσώματος που είναι ζωγραφισμένο και του χρόνου από την εκκίνηση.
Μεταβλητές:

- start, now, end: χρησιμοποιούνται για τη μέτρηση χρόνου στο πρόγραμμα.
- Ended: χρήση για να μη συνεχίζει το πρόγραμμα να τυπώνει αλλαγές στο χρόνο, αφού έχει τελειώσει η διαδικασία.
- Board: τύπου BoardPanel
- Font: αφορά τη γραμματοσειρά
- BoardList, mutationList, selectionList, populationList, generationsList, pauseList: είναι int arrays που έχουν εκχωρημένες τις δυνατές τιμές που μπορεί να δώσει ο χρήστης μέσω των dropdown lists
- boardDropdown, mutationDropdown, selectionDropdown, populationDropdown, generationsDropdown, pauseDropdown: οι ανωτέρω αναφερθέντες dropdown lists
- run: το πλήκτρο που τρέχει το πρόγραμμα με τις εκάστοτε τιμές ορισμάτων.
- Img: το background image του control panel

Constructor:

> Δέχεται σαν όρισμα ένα BoardPanel, ορίζει τις διαστάσεις και τη διάταξη του παραθύρου, ορίζει τις δυνατές τιμές που μπορούν να επιλεγούν από τα dropdown lists, δημιουργεί τα τελευταία και ό,τι τα αφορά και δημιουργεί το πλήκτρο run και ορίζει τι θα γίνεται στο πρόγραμμα με το πάτημά του.

Συναρτήσεις:

- paintComponent: ζωγραφίζει το background
- Update: καλεί τη repaint()
- Paint: Ζωγραφίζει το υπόλοιπο κομμάτι του πάνελ. Την κεφαλίδα, το σκορ και το χρόνο.
- Getters για τις επιλεχθέντες τιμές των dropdown lists που γίνονται parse σε ακεραίους.

Chromosome.java: Είναι το αρχείο που αφορά τη δομή του χρωμοσώματος

Μεταβλητές:

- Positions: είναι int array με μέγεθος όσο και ο αριθμός των Βασιλισσών που έχουν επιλεγεί. Κάθε κελί αντιπροσωπεύει μια στήλη της σκακιέρας και η τιμή που παίρνει δηλώνει την κατακόρυφη θέση της Βασίλισσας που είναι στη στήλη αυτή.
- Fitness: Δηλώνει τον αριθμό «απειλών» μεταξύ των Βασιλισσών.

Constructor:

> δέχεται σαν όρισμα int array που αρχικοποιεί την Positions και εκχωρεί το αποτέλεσμα της calculateFitness στην fitness.

Συναρτήσεις:

- getPositions(): επιστρέφει την positions
- getFitness(): επιστρέφει την fitness
- calculateFitness(): Ελέγχει για κάθε Βασίλισσα πόσες φορές απειλείται από κάποια στα δεξιά της και επιστρέφει το άθροισμά τους. Αυτό γίνεται με τον εξής τρόπο. Για κάθε Βασίλισσα κοιτάει αν υπάρχει κάποια στα δεξιά της που να είναι στην ίδια πλειάδα ή αν η απόσταση της στήλης της πρώτης με τη δεύτερη είναι ίση με την απόσταση των πλειάδων τους. Με τον πρώτο έλεγχο ελέγχει αν υπάρχει Βασίλισσα στην ίδια γραμμή και με το δεύτερο αν υπάρχει στις διαγωνίους της. Από τη δομή του χρωμοσώματος δεν γίνεται να είναι 2 Βασίλισσες στην ίδια στήλη.

NQueens.java: Είναι το αρχείο που δημιουργεί το παράθυρο που περιλαμβάνει το BoardPanel και το ControlPanel και καλεί τις update των ανωτέρω.

Μεταβλητές:

- control: είναι το control panel
- board: είναι το board panel

Constructor:

Δημιουργεί παράθυρο, το παραμετροποιεί και βάζει σε αυτό τα control και board, των οποίων ορίζει τη διάταξη.

Συναρτήσεις:

- start: καθ’ όλη τη διάρκεια του προγράμματος καλεί τις update των control και board
- main: δημιουργεί αντικείμενο τύπου NQueens και καλεί τη start του.

Το πρόγραμμα με τις προκαθορισμένες τιμές και με 8 Βασίλισσες τρέχει σε χρόνο 0,069 seconds με αποτέλεσμα [4,2,0,5,7,1,3,6].

Σε χρόνο 0,071 seconds με 10 Βασίλισσες και αποτέλεσμα [3,6,0,5,8,1,7,4,2,9]

Σε χρόνο 0,099 seconds με 15 Βασίλισσες και αποτέλεσμα [9,5,10,4,1,11,13,7,2,14,12,0,3,8,6]

Το πρόγραμμα εκτελέστηκε με επεξεργαστή ryzen 3600