1η προγραμματιστική εργασία 2021-2022

Μέλη:

Αποστόλου Ιωάννης, 3190013

Καλαντζής Ηλίας, 3190068

Κατσάμης Κωνσταντίνος, 3190237

1ο Πρόβλημα: Missioners and Cannibals

Τρόπος χρήσης:

Το πρόγραμμα δέχεται τις τιμές N (πλήθος κανιβάλων και ιεραποστολών), M μέγιστος αριθμός θέσεων στη βάρκα και K τον μέγιστο αριθμό μετακινήσεων της βάρκας ως είσοδο στον κατασκευαστή State (int N, int M, int K).

Αρχιτεκτονική:

Το πρόγραμμα αποτελείτε από 3 κλάσεις από τη Main στην οποία γίνετε η ανάθεση τιμών η κλήση του αλγορίθμου επίλυσης και η χρονομέτρηση του προγράμματος. Έπειτα έχουμε τη State στην οποία γίνεται η δημιουργία των παιδιών του κόμβου που ορίσαμε στη Main αλλά και των παιδιών των παιδιών του κλπ. Ο έλεγχος αν είναι έγκυρο το παιδί (δεν παραβιάζεται κάποιος από τους περιορισμούς). Όπως και η ανάθεση identifier σε κάθε παιδί. Τέλος, η SpaceSearcher κάνει την αναζήτηση στο δέντρο για τη βέλτιστη λύση με χρήση αλγορίθμου A*

Μέθοδοι τεχνίτης νοημοσύνης που χρησιμοποιήθηκαν

Η αναζήτηση της βέλτιστης λύσης έγινε με τον αλγόριθμος αναζήτησης A* (= f(n)=h(n)+g(n)). Η εκτέλεση έγινε με h(n)= BestFS όπου διάλεγε την καλύτερη επομένη κατάσταση στον κόμβο n με τη βοήθεια της evaluate η οποία υπολογίζει πια είναι η «αξία» ενός κόμβου σύμφωνα με τις κινήσεις που θα έπρεπε να κάνει η βάρκα αν δεν υπήρχαν καθόλου περιορισμοί. Ως g(n) έχουμε το κόστος από τη ρίζα ως το n

Πειραματικά αποτελέσματα

Οι τιμές που δοκιμάστηκαν θα εκφραστούν παρακάτω στη μορφή:

(αριθμός κανιβάλων *, Θέσεις βάρκας, βήματα για εύρεση λύσης,χρόνος)

Τα αποτελέσματα:

(2, 2, 5, 0.001s)

(3, 2, 11, 0.001s)

(8, 4, 13, 0.001s)

(16, 4, 29,0.003s)

(32, 4, 61, 0.016s)

(64, 4, 125, 0.464s)

(100, 7, 80, 120s) Max steps reached

(100, 10, 49, 51.593s)

* έχουμε υποθέσει ότι έχουμε ίσο αριθμό ιεραποστόλων και κανίβαλων

Σημείωση: Οι παραπάνω τιμές υπολογίστηκαν εντός το IDE IntelliJ σε laptop με χαρακτηριστικά (R5-3500u, 16 GB RAM)

2ο Πρόβλημα: Το πρόβλημα των Ν βασιλισσών

Τρόπος χρήσης:

Το πρόγραμμα δέχεται τις τιμές populationSize(πλήθος αρχικών χρωμοσωμάτων), mutationProbability(πιθανότητα μετάλλαξης ενός χρωμοσώματος), maximumSteps(μέγιστος αριθμός βημάτων) και minimumFitness(αριθμός βασιλισσών που δεν απειλούνται).

Αρχιτεκτονική:

Το πρόγραμμα αποτελείτε από 3 κλάσεις. Η Main, στην οποία γίνεται η ανάθεση τιμών και κλήση του αλγορίθμου. Στη συνέχεια έχουμε την Chromosome, η οποία αποτελείται από ένα μονοδιάστατο πίνακα η οποία περιέχει γονίδια, όπου κάθε γονίδιο μας λέει σε ποια θέση βρίσκεται η κάθε βασίλισσα. Με αυτόν τον τρόπο η κλάση αυτή μοντελοποιεί την σκακιέρα. Τέλος έχουμε την GeneticAlgorithm, στην οποία υλοποιήσαμε τον γενετικό αλγόριθμο ο οποίος είναι υπεύθυνος για την επίλυση του προβλήματος.

Μέθοδοι τεχνίτης νοημοσύνης που χρησιμοποιήθηκαν:

Για την εύρεση της βέλτιστης λύσης χρησιμοποιήσαμε γενετικό αλγόριθμο ο οποίος είναι εμπνευσμένος από την βιολογία. Δυστυχώς οι γενετικοί αλγόριθμοι δεν εγγυούνται πως θα βρεθεί λύση κάθε φορά ενώ όσο μεγαλώνει το πλήθος (των N στην συγκεκριμένη περίπτωση), οπότε έπρεπε πέρα από τον αλγόριθμο να κάνουμε έξτρα λειτουργίες για να έχουμε ορθά αποτελέσματα.

Η υλοποίηση του αλγορίθμου έγινε ως εξής. Στον κατασκευαστή, γίνεται η αρχικοποίηση του πλήθους των βασιλισσών(αρχικά με N), της λίστας με τα αρχικά χρωμοσώματα και της λίστας που περιέχει όλα τα χρωμοσώματα και το καθένα τόσες φορές όσο είναι το σκορ του(fitness). Και οι δυο λίστες αρχικοποιούντα με Null. Ακολούθως, με τη μέθοδο run η οποία υλοποιεί και τον αλγόριθμο, δημιουργείται η λίστα με όλα τα χρωμοσώματα. Στη συνέχεια έχουμε την αναπαραγωγή όπου επιλέγονται τυχαία δύο χρωμοσώματα με βάση το σκορ τους(όσο μεγαλύτερο σκορ έχουν, τόσο μεγαλύτερη πιθανότητα έχουν να επιλεγούν). Γίνεται αποκοπή και διασταύρωση των χρωμοσωμάτων κι έτσι δημιουργούμε τα παιδιά. Έπειτα, με τη μέθοδο mutate, υπάρχει η πιθανότητα να μεταλλαχτεί κάποιο χρωμόσωμα, δηλαδή να πάρει καινούργια και τυχαία τιμή. Αφού γίνουν τα πιο πάνω, ελέγχουμε εάν έχουμε κάποια σύγκλιση, δηλαδή αν το γονίδιο με το καλύτερο σκορ(fitness) έχει φτάσει στο επιθυμητό όριο(minimumFitness). Τότε ο αλγόριθμος το επιστρέφει και τερματίζει. Αλλιώς συνεχίζει, ενημερώνοντας τη λίστα με τα πολλαπλά χρωμοσώματα και συνεχίζεται η πιο πάνω διαδικασία μέχρι να βρει το καλύτερο αποτέλεσμα εκτός και αν τελειώσει ο αριθμός των βημάτων που δόθηκε στην αρχή και επιστρέφει το καλύτερο αποτέλεσμα που έχει βρει μέχρι εκεί.

Πειραματικά αποτελέσματα:

Οι χρόνοι του αλγορίθμου:

Για τον υπολογισμό των χρόνων χρησιμοποιήσαμε την μέθοδο nanoTime. Μέσα σε μια for η οποία ξεκινούσε από το 3, αλλάζαμε τις τιμές του N. Έπειτα η πρώτη εντολή της επανάληψης ήταν μια ανάθεση τιμής για τον χρόνο εκείνη την στιγμή, ενώ το ίδιο κάναμε στην τελευταία εντολή πριν κλείσει η for. Τέλος βρίσκαμε την διαφορά αυτών των 2 και μετατρέπαμε τα nanoseconds σε second για πιο εύκολη σύγκριση και κατανόηση. Λόγο του ότι η java δεν είναι ασύγχρονη γλώσσα και δεν χρησιμοποιήσαμε multithreading δεν υπάρχει λάθος στο πρόβλημα, όμως εξαρτάται αρκετά από την υπολογιστική δύναμη του υπολογιστή που εκτέλεσε το πρόγραμμα. Παραθέτουμε κάτω το αποτέλεμα.

Παρακάτω θα έχουμε τα αποτελέσματα του αλγορίθμου από N = 3 έως N = 15, όμως από την φύση του αλγορίθμου όταν θα ξανατρέξετε το πρόγραμμα οι θέσεις των βασιλισσών, ειδικά όσο αυξάνεται το N, θα είναι σε διαφορετική θέση. Οι τιμές που δοκιμάστηκαν θα εκφραστούν παρακάτω στην μορφή:

(μέγεθος σκακιέρας, χρωμόσωμα, σκορ(fitness) , χρόνος)

Τα αποτελέσματα:

N = 3, |2|0|2|, 2, 2

Για N = 3 διακρίνουμε πως οι βασίλισσες απειλούνται όμως αυτό είναι λογικό, καθώς δεν υπάρχει μαθηματική λύση για 3χ3:

Έστω Q η βασίλισσα και 1 οι θέσεις που απειλεί και 0 οι θέσεις που δεν απειλούνται

|Q|1|1|

|1|1|0|

|1||0||1|

ή

|1|Q|1|

|1|1|1|

|0|1||0|

Ή

|1|1|1|

|1|Q|1|

|1|1|1|

Ήταν ενδεικτικά κάποιες λύσεις αλλά επειδή υπάρχει μια συμμετρία ισχύουν τα ίδια και για τις άλλες λύσεις. Διακρίνουμε πως είναι αρκετά δύσκολο να βρούμε θέση για μια 2η βασίλισσα η οποία μόλις μπει καλύπτει όλες τις υπόλοιπες κενές θέσεις και δεν υπάρχει κενός χώρος για 3η βασίλισσα που να μην απειλείται. Επομένως το πρόβλημα έχει λύσεις για N >= 4.

N = 4, |1|3|0|2|, 6, 0

N = 5, |4|1|3|0|2|, 10, 0

N = 6, |4|2|0|5|3|1|, 15, 0

N = 7, |3|6|2|5|1|4|0|, 21, 7 : 0

N = 8, |4|6|0|2|7|5|3|1|, 28, 0

N = 9, |2|8|5|7|1|3|0|6|4|, 36, 1

N = 10, |4|0|7|5|1|8|6|3|9|2|, 45, 0

N = 11, |4|8|5|0|2|10|7|1|3|9|6|, 55, 3

N = 12, |7|3|11|9|2|10|3|11|8|5|1|6|, 64, 3

N = 13, |2|6|8|10|4|0|3|9|12|5|11|7|11|, 75, 3

N = 14, |5|9|12|6|2|10|1|4|0|11|3|8|13|4|, 88, 4

N = 15, |9|5|6|11|6|0|7|14|14|9|4|1|8|2|12|, 100, 4