Τεχνητή Νοημοσύνη

Αναφορά 1ης εργασίας – Διάσχιση ποταμού

Χειμερινό εξάμηνο 2023-2024

Αγγελόπουλος Δημήτρης – 3200002
Μπουζιάνη Πηνελόπη – 3200115
Νεπιίβοντα Άνια – 3200120

Θέμα

Το πρόγραμμα που αναπτύχθηκε αφορά την επίλυση του προβλήματος διάσχισης ποταμού από μια οικογένεια. Στο πρόβλημα διάσχισης ποταμού μας δίνεται ένα πλήθος μελών μιας οικογένειας μαζί με τους χρόνους που κάθε μέλος χρειάζεται για να διασχίσει τον ποταμό και τον χρόνο που διαρκεί η λάμπα. Σκοπός του προβλήματος είναι να περάσουν όλα τα μέλη της οικογένειας στην απέναντι όχθη σε βέλτιστο χρόνο. Για το πρόβλημα ισχύει ότι όταν δύο μέλη διασχίζουν το ποτάμι μαζί ο χρόνος της διάσχισης είναι εκείνος του βραδύτερου. Οι περιορισμοί του προβλήματος είναι ότι ο κορμός αντέχει το πολύ δύο μέλη κάθε φορά και ότι η οικογένεια έχει μία μόνο λάμπα με περιορισμένη διάρκεια ζωής, την οποία πρέπει να έχει μαζί του ένα από τα μέλη που περπατούν κάθε φορά πάνω στον κορμό.

Η υλοποίηση της προγραμματιστικής εργασίας έγινε με χρήση της γλώσσας προγραμματισμού Java και για την βέλτιστη λύση του προβλήματος χρησιμοποιήθηκε ο αλγόριθμος A* με κλειστό σύνολο και χρήση ευρετικής συνάρτησης.

Τρόπος Χρήσης

Ο χρήστης μπορεί να εκτελέσει το πρόγραμμα, παρέχοντας χειροκίνητα τον αριθμό των μελών της οικογένειας, τους χρόνους διάσχισης τους και τον χρόνο που διαρκεί η λάμπα. (Αναλυτικές οδηγίες για την εκτέλεση του προγράμματος δίνονται στο αρχείο README.txt)

Δυνατότητες

Εύκολη επίλυση του προβλήματος διάσχισης ποταμού για οποιοδήποτε σενάριο με βέλτιστη λύση.

Αρχιτεκτονική

Το πρόγραμμα αποτελείται από 3 κλάσεις.

1. Η **State** χρησιμοποιείται για την αναπαράσταση των καταστάσεων κατά την εκτέλεση του αλγορίθμου.

   - Μεταβλητές :
     - **f, g, h :** Μέλη της συνάρτησης εκτίμησης για τον αλγόριθμο A*, όπου g είναι το κόστος από την αρχική κατάσταση μέχρι την τωρινή κατάσταση, h είναι η ευρετική εκτίμηση του κόστους από την τωρινή κατάσταση μέχρι και την τελική κατάσταση και f = g + h.
     - **father:** Προηγούμενη κατάσταση από την οποία φτάσαμε στον τωρινό κόμβο.
     - **total time:** Ο συνολικός χρόνος που έχει περάσει μέχρι την τρέχουσα κατάσταση.

- **members, membersCrossingTime, membersSide:** Σχετίζονται με τον αριθμό των μελών της οικογένειας, τους χρόνους διάσχισης τους και την πλευρά της όχθης που βρίσκονται. Για την membersSide ισχύει πως αν το μέλος είναι στα δεξιά τότε η τιμή του μέλους στον πίνακα είναι 1, ενώ αν είναι στα αριστερά είναι 0.
- **lantern:** Μεριά της όχθης που βρίσκεται η λάμπα.

- Constructors
  - **State(int members, int[] membersCrossingTime):** Αρχικοποιεί την αρχική κατάσταση με βάση τις τιμές που δίνει ο χρήστης.
  - **State(State s):** Copy constructor που δημιουργεί μία κατάσταση ίδια με αυτή που περνάμε ως όρισμα.

- Μέθοδοι:
  - **getters() και setters(…):** Για να έχουμε πρόσβαση στις private μεταβλητές της State από άλλες κλάσεις.
  - **evaluate():** Ενημερώνει την συνάρτηση εκτίμησης (f) προσθέτοντας τις τιμές των g και h.
  - **moveLeft(int firstMember, int secondMember):** Υλοποιεί σε μία κατάσταση την μετακίνηση δύο μελών από τη δεξιά πλευρά στην αριστερή πλευρά ενημερώνοντας τις θέσεις των μελών, της λάμπας και τον συνολικό χρόνο.
  - **moveRight(int member):** Υλοποιεί σε μια κατάσταση την μετακίνηση ενός μέλους από την αριστερή πλευρά στη δεξιά πλευρά ενημερώνοντας τη θέση του, την θέση της λάμπας και τον συνολικό χρόνο.

- **heuristic():** Υπολογίζει την ευρετική εκτίμηση (h) για την τρέχουσα κατάσταση και ενημερώνει το h. (Παρουσιάζεται παρακάτω με μεγαλύτερη λεπτομέρεια (βλέπε Μέθοδοι τεχνητής νοημοσύνης)).
- **getChildren():** Δημιουργεί και επιστρέφει μια λίστα με τις αμέσως επόμενες καταστάσεις που μπορούμε να μεταβούμε με βάση τους περιορισμούς του προβλήματος. Έχει επίσης αποφευχθεί η δημιουργία των καταστάσεων μετακίνησης ενός ατόμου από τα δεξιά στα αριστερά και η μετακίνηση δύο ατόμων από τα αριστερά στα δεξιά καθώς βάση λογικής δεν θα οδηγήσουν σε μια πιο βέλτιστη λύση.
- **isFinal():** Ελέγχει αν η τωρινή κατάσταση είναι τελική κατάσταση τσεκάροντας αν όλα τα μέλη της οικογένειας βρίσκονται στην αριστερή πλευρά.
- **print():** Εκτυπώνει πληροφορίες σχετικά με την τωρινή κατάσταση, όπως τα μέλη που βρίσκονται στην αριστερή μεριά, τα μέλη που βρίσκονται στην δεξιά μεριά, την μεριά της λάμπας, τον συνολικό χρόνο που έχει περάσει και την συνολική εκτίμηση.
- **equals(Object obj):** Έλεγχος ισότητας δυο αντικειμένων State συγκρίνοντας τις μεταβλητές lantern, membersCrossingTime και membersSide.
- **hashCode:** Παράγει έναν μοναδικό hash code για κάθε κατάσταση.
- **compareTo(State s):** Συγκρίνει δύο καταστάσεις με βάση την τιμή της f.

2. Η **SpaceSearcher** υλοποιεί τον αλγόριθμο A* με κλειστό σύνολο.

- Μεταβλητές :
  - **frontier:** ArrayList με τις καταστάσεις που δεν έχουν εξερευνηθεί ακόμα.

- **closedSet:** Hashset με τις καταστάσεις που ήδη εξερευνήθηκαν για την αποφυγή εξερεύνησης μιας ίδιας κατάστασης σε χαμηλότερο επίπεδο.
- **laternTime:** Μια ακέραια μεταβλητή που αναπαριστά την διάρκεια ζωής της λάμπας σε λεπτά.

- Constructor:
  - **SpaceSearcher(int lanternTime):** αρχικοποιεί το frontier, το closedSet ως κενά σύνολα και θέτει το lanternTime με την τιμή που έδωσε ο χρήστης.

- Μέθοδοι :
  - **aStar(State initialState):** Υλοποιεί τον A* αλγόριθμο με κλειστό σύνολο, εξερευνώντας καταστάσεις προκειμένου να βρει την βέλτιστη τελική κατάσταση. Επιστρέφει αντικείμενο State εφόσον βρεθεί τελική κατάσταση, ειδάλλως την τιμή null εφόσον δεν βρεθεί τελική για τον δοσμένο χρόνο διάρκειας της λάμπας.

3. Η **Main** αφορά την εκτέλεση του προγράμματος. Αρχικά ζητάει από τον χρήστη να εισάγει τιμές για τον αριθμό των μελών, τον χρόνο διάσχισης του καθενός, την διάρκεια ζωής της λάμπας και εκτελεί τους απαραίτητους ελέγχους ορθότητας σχετικά με τις τιμές που δίνει ο χρήστης για τους χρόνους των μελών. Στη συνέχεια δημιουργεί την αρχική κατάσταση με ορίσματα τις τιμές που έδωσε ο χρήστης και εφαρμόζεται ο αλγόριθμος A*. Μετά την εκτέλεση του αλγορίθμου αν βρεθεί λύση, επιστρέφουμε τη διαδρομή που ακολούθησε ο A* μέχρι να φτάσει στην τελική κατάσταση μέσω της μεθόδου

getFather(). Επίσης εμφανίζεται ο χρόνος που έκανε ο A* να βρει λύση και πόσα βήματα χρειάστηκε. Αν δεν βρέθηκε λύση επιστρέφει το ανάλογο μήνυμα στον χρήστη.

Μέθοδοι Τεχνητής Νοημοσύνης

Στην κλάση SpaceSearcher έχει υλοποιηθεί ο αλγόριθμος A* με κλειστό σύνολο, που εξερευνά καταστάσεις προκειμένου να βρει την βέλτιστη τελική κατάσταση.

Η ευρετική που χρησιμοποιήθηκε προήλθε από αφαίρεση του περιορισμού των ατόμων που μπορούν να περπατήσουν κάθε φορά στον κορμό. Με άλλα λόγια ο περιορισμός ότι μπορούν να περπατήσουν το πολύ δύο άτομα στον κορμό κάθε φορά δεν ισχύει. Συνεπώς η ευρετική υπολογίζει το κόστος εκτίμησης σύμφωνα με το παρακάτω:

- Αν η λάμπα βρίσκεται στην δεξιά μεριά, τότε το κόστος ισούται με τον χρόνο του βραδύτερου ατόμου στα δεξιά. Αυτό συμβαίνει καθώς έχει αφαιρεθεί ο περιορισμός των δύο ατόμων και συνεπώς μπορούν να περάσουν όλοι μαζί στην απέναντι όχθη με χρόνο ίσο του βραδύτερου μέλους.
- Αν η λάμπα βρίσκεται στην αριστερή μεριά, τότε το κόστος ισούται με το άθροισμα του γρηγορότερου στα αριστερά και τον χρόνο του βραδύτερου στα δεξιά.

Παραδείγματα Χρήσης

- Η εκτέλεση του κώδικα για 5 μέλη με χρόνους : 1, 3, 6, 9, 12.

Στο πρώτο παράδειγμα δίνεται συνολικός χρόνος διάρκειας της λάμπας 30 λεπτά.

```
------ RIVER CROSSING SOLUTION ------

Step 1:
Members on the left side: 1 3
Members on the right side: 6 9 12
Lantern's side: left
Elapsed time: 3 minutes
f(n)=3+13=16

Step 2:
Members on the left side: 3
Members on the right side: 1 6 9 12
Lantern's side: right
Elapsed time: 4 minutes
f(n)=4+12=16

Step 3:
Members on the left side: 1 3 6
Members on the right side: 9 12
Lantern's side: left
Elapsed time: 10 minutes
f(n)=10+13=23

Step 4:
Members on the left side: 3 6
Members on the right side: 1 9 12
Lantern's side: right
Elapsed time: 11 minutes
f(n)=11+12=23

Step 5:
Members on the left side: 3 6 9 12
Members on the right side: 1
Lantern's side: left
Elapsed time: 23 minutes
f(n)=23+4=27

Step 6:
Members on the left side: 6 9 12
Members on the right side: 1 3
Lantern's side: right
Elapsed time: 26 minutes
f(n)=26+3=29

Step 7:
Members on the left side: 1 3 6 9 12
Members on the right side:
Lantern's side: left
Elapsed time: 29 minutes
f(n)=29+0=29
------------ SUMMARY ------------

Total time required: 29 minutes.
Time left: 1 minutes.
Finished in 7 steps.
A* algorithm search time: 0.002 seconds.
```

Στο δεύτερο παράδειγμα δίνεται συνολικός χρόνος διάρκειας της λάμπας 25 λεπτά.

```
------ RIVER CROSSING PROBLEM ------

Please enter number of family members: 5
Enter the crossing time for member 1: 1
Enter the crossing time for member 2: 3
Enter the crossing time for member 3: 6
Enter the crossing time for member 4: 9
Enter the crossing time for member 5: 12
Enter the lantern time: 25


---------- INITIAL STATE ----------
Members on the left side:
Members on the right side: 1 3 6 9 12
Lantern's side: right
Elapsed time: 0 minutes
f(n)=0+12=12

Needed at least or more than: 26 minutes.
Could not find a solution.
```