Οικονομικό Πανεπιστήμιο Αθηνών, Τμήμα Πληροφορικής

Μάθημα: Τεχνητή Νοημοσύνη
Ακαδημαϊκό έτος: 2020–21
1η Προγραμματιστική εργασία
Παράδοση ως 13/12/2020

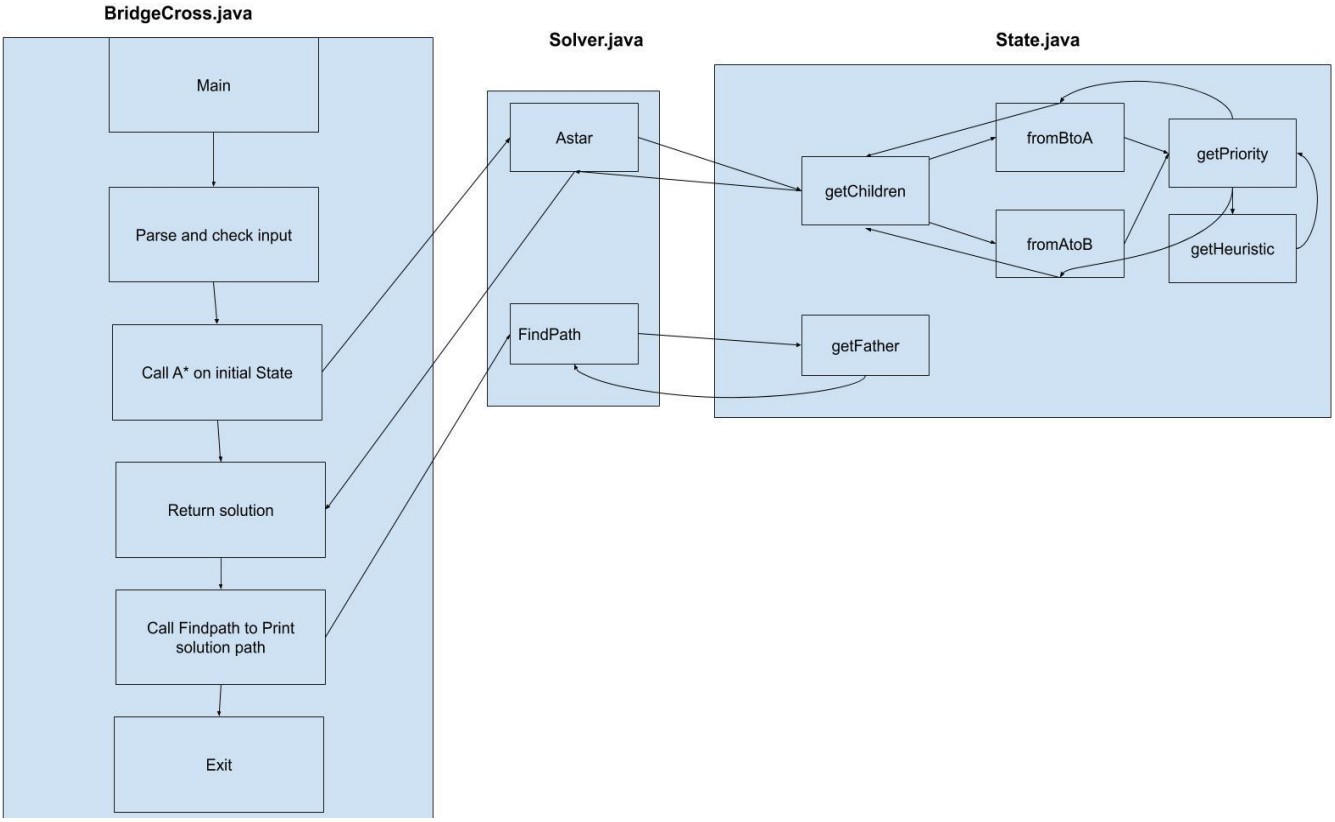
Μέλη:
Κυρμπάτσος Παναγιώτης -Χρήστος (ΑΜ:3180226)
Λιαρόπουλος Δημήτρης (ΑΜ:)
Γεώργιος- Στέφανος Μεϊδάνης (ΑΜ:3170107)

Περιγραφή Προβλήματος

Μια οικογένεια βρίσκεται στην δεξιά όχθη ενός ποταμού και επιθυμεί να περάσει απέναντι μέσω μιας γέφυρας. Κάθε μέλος της οικογένειας χρειάζεται διαφορετικό (και σταθερό προς οποιαδήποτε κατεύθυνση) χρόνο για να περάσει, και 2 άτομα μπορούν να περάσουν την φορά. Ο χρόνος που χρειάζονται είναι ίσος με τον μεγαλύτερο των 2 ατόμων. Η οικογένεια έχει και μια λάμπα, την οποία πρέπει να κρατάει ένα από τα μέλη που διασχίζει την γέφυρα.

Στοχοθεσία

Σκοπός της εργασίας είναι η ανάπτυξη προγράμματος σε Java για την επίλυση του προβλήματος διάσχισης γέφυρας. Ειδικότερα βρίσκει και εκτυπώνει τη βέλτιστη λύση, δηλαδή με ποια σειρά πρέπει να κινηθούν τα μέλη της οικογένειας, ώστε να περάσει όλη η οικογένεια απέναντι στον ελάχιστο χρόνο.



Διάγραμμα Σχέσεων

Το BridgeCross είναι το αρχείο που έχει την main. Αφού ολοκληρωθεί η αποθήκευση των παραμέτρων σε μεταβλητές και μια δυναμική λίστα ταξινομούμε την λίστα και αρχικοποιούμε τα δεδομένα(initial State και Solver). Τέλος η απάντηση στο πρόβλημα συγκρίνεται με τον δοθέντα χρόνο και αν δεν είναι εκτος ορίων παρουσιάζουμε την διαδρομή του αλγορίθμου που χρησιμοποιήσαμε από την ρίζα προς την λύση.

Κλάση State και αρχιτεκτονική

State.java: αναπαριστά τις καταστάσεις του προβλήματος

Ορίσματα

- private int elapsedTime: Για τον χρόνο που έχει περάσει.
- private ArrayList<Integer> arA,arB: Λίστες ατόμων για την κάθε πλευρά, arA=> δεξιά, arB =>αριστερά.
- private boolean torchAtA: Μεταβλητή που δηλώνει την τοποθεσία της λάμπας.
- private int priority: Γνώρισμα για την ουρά προτεραιότητας.
- private State father=null: Μεταβλητή που δείχνει στην προερχόμενη κατάσταση.

Μέθοδοι

- State() :default κατασκευαστής
- State(ArrayList<Integer> A, ArrayList<Integer> B, int etime, boolean torchAtA) κατασκευαστής με ορίσματα μια αλλης κατάστασης

- boolean fromAtoB(ArrayList<Integer> persons):Υπεύθυνη για το πέρασμα ατόμων από την λίστα Β στην Α. Διαγράφει άτομα από τη λίστα της όχθης Α και μεταφέρει τον πυρσό στην όχθη Β.
- boolean fromBtoA: Το ίδιο για την όχθη Β.
- ArrayList<State> getChildren:Παράγει τις καταστάσεις για όλες τις δυνατές μεταβάσεις με μια διάσχιση
- boolean IsTerminal(): Ελέγχει αν μια κατάσταση είναι τελική.
- void print(): Εκτυπώνει μια κατάσταση.
- int getMax(): Παίρνει σαν παράμετρο μια λίστα και επιστρέφει το μεγιστο στοιχείο της.
- getElapsedTime(): Επιστρέφει τον χρόνο που έχει παρέλθει
- getFather(): Επιστρέφει την προερχόμενη κατάσταση.
- setFather(): Θέτει την προερχόμενη κατάσταση ενός στιγμιότυπου
- int getHeuristic(): Χρησιμοποιεί την ευρετική συνάρτηση
- int getPriority(): Επιστρέφει το γνώρισμα για την ουρά προτεραιότητας
- int compareTo: Υλοποίηση της Comparable<State>. Χρησιμοποιείται στην σύγκριση καταστάσεων.
- public ArrayList<State> getChildren()
  Μέθοδος που επιστρέφει λίστα με όλες τις πιθανές επόμενες καταστάσεις παραγόμενες με μία μόνο διάσχιση της γέφυρας. Αρχικοποιεί την λίστα children η οποία αποθηκεύει τις παραγόμενες καταστάσεις. Διατάσσει τα πιθανά ζευγάρια σε μια λίστα που περιέχει όλα τα δυνατά ζευγάρια διάσχισης .Τα ζευγάρια βρίσκονται σε αύξουσα σειρα και για αυτό κάθε δυνατό ζευγάρι επιλέγεται μια μοναδική φορά. Στη συνέχεια για κάθε ζευγάρι υπολογίζεται η προτεραιότητά του, ο γονέας του και αλλάζει η θέση του πυρσού.Ολα τα πιθανά ζευγάρια επιστρέφουν στην κλάση Solver.Astar και εισάγονται στην σειρά προτεραιότητας queue.

Κλάση Solver- Μέθοδοι Τεχνητής Νοημοσύνης

Solver.java: Παράγει την λύση του προβλήματος ,και περιέχει την Μέθοδο Τεχνητής Νοημοσύνης(A*).

Ορίσματα

- MinPQ<State> queue Η ουρά προτεραιότητας Χρησιμοποιείται για αποθήκευση των καταστάσεων και την υλοποίηση του A*. Έχει χρησιμοποιηθεί η υλοποίηση του Sedgewick.
- int time; Μεταβλητή του χρονικου ορίου που δόθηκε κατά την είσοδο. Χρησιμοποιείται για τον τερματισμό του αλγορίθμου στην περίπτωση που παραβιαστεί, μια και το μονοπάτι του A* είναι και το βέλτιστο.

Μέθοδοι

- public Solver(): Κατασκευαστής.
- List<State> findPath(): Μέθοδος που τυπώνει την διαδρομή από την λύση προς την αρχική κατάσταση.Χρησιμοποιεί συνδεδεμένη λίστα για την αποτύπωση της σχέσης γονέα παιδιού μεταξύ αρχικής (initial) και τελικής κατάστασης (solution).
- State Astar(State initialState): Μέθοδος Τεχνητής Νοημοσύνης

Για την επίλυση του προβλήματος χρησιμοποιήθηκε ο αλγόριθμος εύρεσης μονοπατιού A*.Το συνολικό κόστος κάθε κατάστασης αξιολογείται ως το άθροισμα ElapsedTime + Heuristic . Ως elapsed time ορίζεται το αθροισμα των χρονων για τις διαδοχικες διασχισεις της γεφυρας μέχρι εκείνη την κατάσταση. Η ευρετική προκύπτει ως εξής:
Αν ο πυρσός βρίσκεται στην όχθη Α, χαλαρώνω τον περιορισμό πως πρέπει να περάσουν μόνο δύο άτομα την γέφυρα και υποθέτω ότι μπορούν να περάσουν όλοι μαζί έτσι η ευρετική ισούται με το χρόνο του πιο αργού ατόμου στην όχθη Α.
Αν ο πυρσος βρίσκεται στην όχθη Β,υποθέτω πως θα ξαναπάει στην όχθη Α με το ταχύτερο άτομο και στη συνέχεια χαλαρώνω τον περιορισμό και υποθέτω ότι μπορούν να περάσουν όλοι μαζί (ο χρόνος του πιο αργού ατόμου στην όχθη Α). Ετσι η ευρετική είναι το άθροισμα των δυο αυτων χρόνων.

Η παραπάνω ευρετική είναι αποδεκτή αφού h(n)<=C(n)* για όλες τις δυνατές καταστάσεις.Δηλαδή η ευρετική ποτέ δεν υπερεκτιμά το κόστος για να φτάσουμε στην τελική κατάσταση. Αυτό προκύπτει από την χαλάρωση του περιορισμού πως το πολύ δύο άτομα περνάνε τη γέφυρα.Εναλλακτικά μπορούμε να πούμε ότι η ευρετική υποεκτιμά τις διασχίσεις της γέφυρας που θα χρειαστούν.

Η ευρετική είναι επίσης συνεπής επειδή η τιμή της για όλους τους κόμβους δεν είναι ποτέ μεγαλύτερη από το άθροισμα της εκτιμώμενης απόστασης μιας γειτονικής κατάστασης και του κόστους μετάβασης σε αυτή η γειτονική κατάσταση.Με άλλα λόγια ο αλγόριθμος A* πάντα ακολουθεί το βέλτιστο μονοπάτι για την εύρεση λύσης.

Επειδή η συνάρτηση που χρησιμοποιούμε για να αναθέσουμε προτεραιότητα σε όλες τις καταστάσεις είναι συνεπής και επειδή κάθε φορά επιλέγουμε μέσω της MinPQ να εξερευνήσουμε την κατάσταση με την μικρότερη προτεραιότητα ο αλγόριθμος βρίσκεται πάντα στο βέλτιστο μονοπάτι δίχως τη χρήση κλειστού συνόλου.

Αποτελέσματα διαφορετικών εισόδων σε δύο διαφορετικά PC

Πλήθος μελών	Ξεχωριστός χρόνος κάθε μέλους	Χρονικό όριο	Αποτέλεσμα	Χρόνος εκτέλεσης (pc1)	Χρόνος εκτέλεσης(pc2)
5	1 3 6 8 12	30	29	88 ms	55ms
4	1, 2,5,10	18	17	73 ms	46 ms
5	1, 4, 5, 8, 11	30	30	96 ms	50 ms
5	10 12 5 4 1	35	31	94 ms	39 ms
8	3 7 1 23 16 8 1 6 2	50	wrong number of parameters given	-	-
8	3 7 1 23 16 8 6 2	50	no solution found in	683 ms	42ms

			given time		
8	3 7 1 23 16 8 6 2	60	54	989 ms	1245ms
3	4 5 26	40	35	63 ms	30ms
5	90 40 23 68 3	400	205	91 ms	37ms

Το παράδειγμα της εκφώνησης δίνει τα εξής αποτελέσματα:

File - BridgeCross

```
"C:\Program Files\Java\jdk-11.0.2\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2019.1.2\lib\idea_rt.jar=59996:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2019.1.2\bin" -Dfile.encoding=UTF-8 -classpath "C:\Users\pchrk\OneDrive\Desktop\cs\3rd year\texnhth nohmosynh\ask1\ai1\out\production\ai1" com.company.BridgeCross 5 1 3 6 8 12 70
Terminal State reached. Elapsed Time= 29
Route to initial:
----------------
A:
     1   3   6   8   12
B:

elapsed time: 0
----------------
----------------
A:
     3   8   12
B:
     1   6
elapsed time: 6
----------------
----------------
A:
     3   8   12  1
B:
     6
elapsed time: 7
----------------
----------------
A:
     8   12
B:
     6   1   3
elapsed time: 10
----------------
----------------
A:
     8   12  3
B:
     6   1
elapsed time: 13
----------------
----------------
A:
     3
B:
     6   1   8   12
elapsed time: 25
```

File - BridgeCross

```
45 ----------------
46 ----------------
47 A:
48      3   1
49 B:
50      6   8   12
51 elapsed time: 26
52 ----------------
53 ----------------
54 A:
55
56 B:
57      6   8   12  1   3
58 elapsed time: 29
59 ----------------
60 Execution Time: 45
61
62 Process finished with exit code 0
63
```

```
"C:\Program Files\Java\jdk-11.0.2\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2019.1.2\lib\idea_rt.jar=60652:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2019.1.2\bin" -Dfile.encoding=UTF-8 -classpath "C:\Users\pchrk\OneDrive\Desktop\cs\3rd year\texnhth nohmosynh\ask1\ai1\out\production\ai1" com.company.BridgeCross 8 3 7 1 23 16 8 6 2 69606
Terminal State reached. Elapsed Time= 54
Route to initial:
----------------
A:
     1   2   3   6   7   8   16  23
B:

elapsed time: 0
----------------
----------------
A:
     2   3   7   8   16  23
B:
     1   6
elapsed time: 6
----------------
----------------
A:
     2   3   7   8   16  23  1
B:
     6
elapsed time: 7
----------------
----------------
A:
     3   7   8   16  23
B:
     6   1   2
elapsed time: 9
----------------
----------------
A:
     3   7   8   16  23  1
B:
     6   2
elapsed time: 10
----------------
----------------
A:
     7   8   16  23
B:
     6   2   1   3
elapsed time: 13
```

```
----------------
----------------
A:
     7   8   16  23  2
B:
     6   1   3
elapsed time: 15
----------------
----------------
A:
     16  23  2
B:
     6   1   3   7   8
elapsed time: 23
----------------
----------------
A:
     16  23  2   1
B:
     6   3   7   8
elapsed time: 24
----------------
----------------
A:
     16  23
B:
     6   3   7   8   1   2
elapsed time: 26
----------------
----------------
A:
     16  23  1
B:
     6   3   7   8   2
elapsed time: 27
----------------
----------------
A:
     1
B:
     6   3   7   8   2   16  23
elapsed time: 50
----------------
----------------
A:
     1   2
B:
     6   3   7   8   16  23
elapsed time: 52
----------------
```



```
----------------
A:

B:
     6   3   7   8   16  23  1   2
elapsed time: 54
----------------
Execution Time: 1253

Process finished with exit code 0

```