

Τεχνητή Νοημοσύνη

Εργασία 1

Θέμα: Cross-bridge

ΑΜ: 3160074, 3170014, 3180230

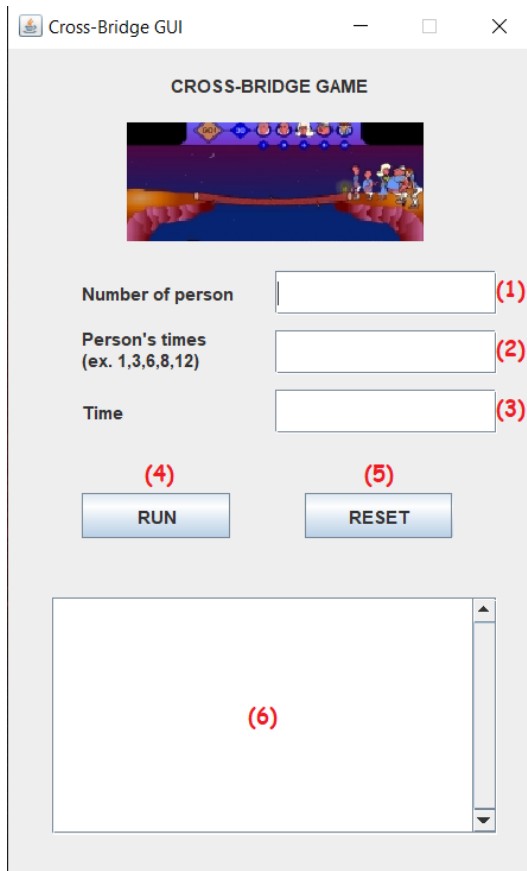
8/12/2020

1. Περιγραφή τρόπου χρήσης

- **Compile** : `javac CrossFrame.java State.java Pair.java Searcher.java`
- **Run** : `java CrossFrame`

Μετά το Run, θα εμφανιστεί στην οθόνη μας το CrossBridge GUI στο οποίο μπορούμε να δοκιμάσουμε οποιοδήποτε παράδειγμα.

Επεξήγηση GUI :



(1): Εισαγάγουμε το πλήθος των ατόμων που θέλουμε (π.χ.: 5).

(2): Εισαγάγουμε τους χρόνους των ατόμων χωρισμένους με κόμμα (",") (π.χ.: 1,3,6,8,12).

(3): Εισαγάγουμε τον μέγιστο χρόνο διάσχησης της γέφυρας (π.χ.: 30).

(4): Τρέχουμε το πρόγραμμα.

(5): Μηδενίζει τα στοιχεία για την δοκιμή ενός άλλου παραδείγματος.

(6): Εμφανίζει τα αποτελέσματα.

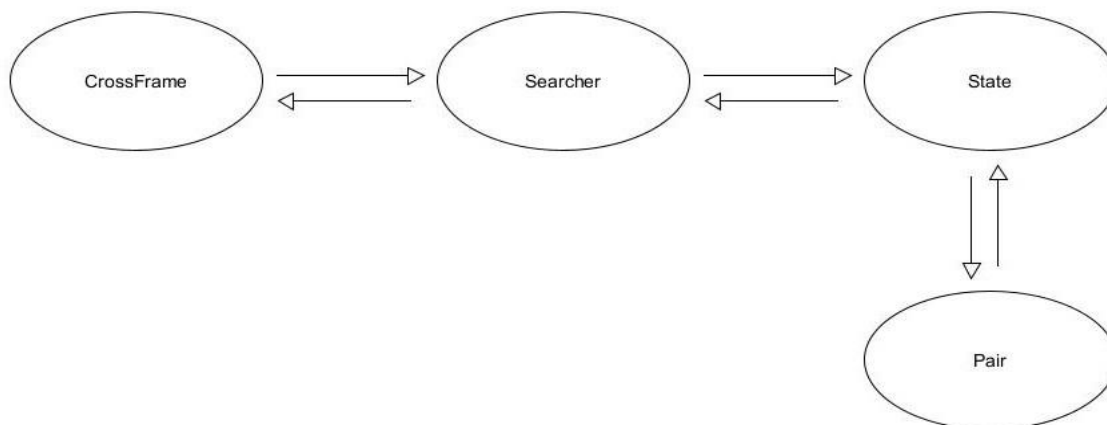
2. Δυνατότητες του προγράμματος

- Υπολογίζει το βέλτιστο μονοπάτι που πρέπει να ακολουθήσει, δηλαδή το ελάχιστο κόστος για τη διάσχιση της γέφυρας από τα άτομα, στα πλαίσια του δοσμένου χρόνου.
- Αναγνωρίζει αν το input είναι λανθασμένο και ενημερώνει κατάλληλα.
- Εάν, βάση του δοσμένου χρόνου δεν υπάρχει βέλτιστο μονοπάτι ενημερώνει αντίστοιχα.
- Ακόμη και για άτομα με ίδιους χρόνους το πρόγραμμα, μπορεί και βρίσκει το βέλτιστο μονοπάτι.

3. Αρχιτεκτονική Προγράμματος

- **Class Pair** - δημιουργεί instances ζευγαριών που θα διασχίζουν την γέφυρα κάθε φορά. Τα επιστρέφει στη State.
- **Class Searcher** – Καλείται από τη CrossFrame. Η βασική της λειτουργία είναι να ελέγχει εάν είμαστε σε τερματική κατάσταση από τα παιδιά που της έχουν επιστραφεί από την κλάση State. Στο τέλος αποθηκεύει την τελική κατάσταση στο Path.
- **Class State** – Καλείται από την Searcher και χρησιμοποιεί την Pair. Βασική λειτουργία η εύρεση των παιδιών της κάθε κατάστασης. Επιστρέφει στην Searcher τα παιδιά.
- **Class CrossFrame** - Αποτελείται από την main() μας και δημιουργεί το GUI για την εύχρηστη χρήση του προγράμματος. Τέλος, εκτυπώνει τα αποτελέσματα με την print().

Η επικοινωνία των κλάσεων φαίνεται στην παρακάτω εικόνα:



4. Μέθοδοι ΑΙ που χρησιμοποιήθηκαν

Χρησιμοποιείται ο αλγόριθμος αναζήτησης A* με μια heuristic συνάρτηση (int heuristic()), η οποία βρίσκεται στην State.java. Ουσιαστικά η A* αναζητά στην ταξινομημένη λίστα την επόμενη κατάσταση και για όσο υπάρχουν paths to explore. Αν ο χρόνος τελειώσει τότε σταματά την αναζήτηση. Σε κάθε άλλη περίπτωση είναι ένα βήμα πιο κοντά στο να χτίσει το μονοπάτι.

5. Παραδείγματα (CPU: i7700HQ, RAM: 12GB, JAVA: 14.0.1)

The image displays two side-by-side screenshots of the 'Cross-Bridge GUI' application. Both windows show a top-down view of a game area with a bridge and several characters. Below the game view, there are input fields for 'Number of person', 'Person's times (ex. 1,3,6,8,12)', and 'Time'. At the bottom of each window are 'RUN' and 'RESET' buttons, and a scrollable text area showing game results.

Left Screenshot (5 people, 30s time):

- Number of person: 5
- Person's times (ex. 1,3,6,8,12): 1,3,6,8,12
- Time: 30
- Buttons: RUN, RESET
- Results:
 - Astar running time: 0.0053286 sec
 - Hypothetical optimal path time: 29 sec.
 - Optimal path took: 7 step(s).
 - Path list:
 - [1, 3, 6, 8, 12] || []
 - [6, 8, 12] || [1, 3]
 - [6, 8, 12, 1] || [3]
 - [8, 12] || [3, 6, 1]
 - [8, 12, 1] || [3, 6]

Right Screenshot (7 people, 500s time):

- Number of person: 7
- Person's times (ex. 1,3,6,8,12): 5,7,8,202,203,204,205
- Time: 500
- Buttons: RUN, RESET
- Results:
 - Astar running time: 0.1813027 sec
 - Hypothetical optimal path time: 466 sec.
 - Optimal path took: 11 step(s).
 - Path list:
 - [5, 7, 8, 202, 203, 204, 205] || []
 - [8, 202, 203, 204, 205] || [5, 7]
 - [8, 202, 203, 204, 205, 5] || [7]
 - [8, 204, 205, 5] || [7, 202, 203]
 - [8, 204, 205, 5, 7] || [202, 203]

Below the right screenshot, a third scrollable text area shows a continuation of the path list:

- [8, 202, 203, 204, 205, 5] || [7]
- [8, 204, 205, 5] || [7, 202, 203]
- [8, 204, 205, 5, 7] || [202, 203]
- [8, 204, 205] || [202, 203, 5, 7]
- [8, 204, 205, 5] || [202, 203, 7]
- [204, 205] || [202, 203, 7, 8, 5]
- [204, 205, 5] || [202, 203, 7, 8]
- [5] || [202, 203, 7, 8, 204, 205]
- [5, 7] || [202, 203, 8, 204, 205]
- [] || [202, 203, 8, 204, 205, 5, 7]