1η Εργασία Τεχνητής Νοημοσύνης

Θεόδωρος Αλεξέγιεβ, p3200004

Κωνσταντίνος Ζαμπετάκης, p3200049

Ανδριάνα Καραγιάννη, p3200062

Διάσχιση ποταμού:

Η ευρετική που χρησιμοποιήσαμε μετράει πόσα άτομα έχουν μείνει στην αρχή, είναι ουσιαστικά ένας ποσοδείκτης που δείχνει ένα κομμάτι του κόστους της επόμενης κατάστασης. Το υπόλοιπο κομμάτι του κόστους είναι το κόστος της απόστασης της κατάστασης αυτής από τη ρίζα.

Η ροή εκτέλεσης:

- Όταν ξεκινάει το παιχνίδι ο χρήστης έχει την επιλογή μεταξύ ενός custom και ενός default παιχνιδιού. Αφού γίνει η επιλογή αυτή τα δεδομένα του παιχνιδιού(τα άτομα, ο χρόνος που κάνει κάθε άτομο, ο διαθέσιμος χρόνος), εισάγονται στο initialState τύπου State και με αυτό σαν παράμετρο καλείται ο A*(a_Star(State initialState)).
- Ο A* υλοποιεί αναζήτηση της κατάστασης με το μικρότερο κόστος. Αν δεν βρεθεί λύση επιστρέφει “Solution not found”. Εάν η αρχική κατάσταση (initialState) είναι η τελική κατάσταση/λύση, την επιστρέφει. Αν δεν ισχύει τίποτα από τα δύο, συνεχίζει την αναζήτηση λύσης καλώντας την getChildren(State state), η οποία όταν ολοκληρωθεί επιστέφει στον A* μέσω μίας ArrayList όλες τις δυνατές καταστάσεις από τις οποίες ο A* θα διαλέξει την καταλληλότερη.
- Η getChildren, δέχεται για παράμετρο ένα αντικείμενο τύπο State για το οποίο προσπαθεί να βρει τις δυνατές του, επόμενες καταστάσεις, να τις προσθέσει σε μια ArrayList children και να την επιστρέψει στην a_star() για περαιτέρω αναζήτηση. Αρχικά, ελέγχει το torch_position. Σύμφωνα με αυτή (αν torch_position = 0 βρίσκεται στην αρχή, αν torch_position = 1, βρίσκεται στο τέλος), υλοποιούμε την μετακίνηση προς την αντίστοιχη κατεύθυνση. Για να είναι μια μετακίνηση εφικτή, θα πρέπει το άτομο (ή άτομα) το οποίο θα μετακινηθεί, να μη κάνει περισσότερη ώρα από τη διαθέσιμη, (people_in_hashmap.getValue(person) <= state.available_time). Επειδή προς κάθε κατεύθυνση επιτρέπεται να μετακινηθούν το πολύ δύο άτομα, πρέπει να εξετάσουμε κάθε περίπτωση. Δηλαδή, δημιουργούμε πρώτα τη κατάσταση να μετακινηθεί ένα μόνο άτομο (για κάθε άτομο στη hashmap που μελετάμε εκείνη τη στιγμή), και ύστερα, μελετάμε κάθε εφικτό συνδυασμό των δύο ατόμων. Με ποια από όλες αυτές τις καταστάσεις θα προχωρήσει το παιχνίδι, είναι πλέον δουλειά του a_star αλγόριθμου στη Searcher.java

Output: αφού επιστρέψει η a_star την τελική κατάσταση, στην main εκτελείται το print_path(result), το οποίο εκτυπώνει το μονοπάτι που καταλήγει στην τελική κατάσταση.

1.Custom game(option 1): Στο custom παιχνίδι ο χρήστης επιλέγει πόσα άτομα διασχίζουν το ποτάμι, τους χρόνους που κάνουν και τον διαθέσιμο χρόνο που έχουν για την διάσχιση του. Εάν επιλέξει ο χρήστης μετακίνηση ατόμων πλήθους > 5, ο χρόνος εκτέλεσης του προγράμματος είναι μεγάλος οπότε δεν μπορούμε να διαπιστώσουμε εάν βρίσκει κάποια τελική κατάσταση ή αν η κατάσταση είναι τελική.

2.Default game(option 2): Στο default παιχνίδι είναι πέντε τα άτομα που διασχίζουν την γέφυρα με χρόνους 1, 3, 6, 8, 12 και διαθέσιμο χρόνο 30. Το παράδειγμα δίνεται παρακάτω.

```
Do you want to make a custom game (press 1), or play the default settings (press 2)?
2
This is the transfer order:
People at start: {Bob=1, Aunty=12, Alice=3, Marios=8, Hercules=6}
People at finish: {}
Torch position: 0
Available time: 30
Heuristic score (h): 0
Cost to reach this node from the initial node (g): 0
Heuristic score + Cost to reach this node (f): 0
People at start: {Aunty=12, Marios=8, Hercules=6}
People at finish: {Bob=1, Alice=3}
Torch position: 1
Available time: 27
Heuristic score (h): 3
Cost to reach this node from the initial node (g): 1
Heuristic score + Cost to reach this node (f): 4
People at start: {Bob=1, Aunty=12, Marios=8, Hercules=6}
People at finish: {Alice=3}
Torch position: 0
Available time: 26
Heuristic score (h): 4
Cost to reach this node from the initial node (g): 2
Heuristic score + Cost to reach this node (f): 6
People at start: {Bob=1, Hercules=6}
People at finish: {Aunty=12, Alice=3, Marios=8}
Torch position: 1
Available time: 14
Heuristic score (h): 2
Cost to reach this node from the initial node (g): 3
Heuristic score + Cost to reach this node (f): 5
People at start: {Bob=1, Alice=3, Hercules=6}
People at finish: {Aunty=12, Marios=8}
Torch position: 0
Available time: 11
Heuristic score (h): 3
Cost to reach this node from the initial node (g): 4
Heuristic score + Cost to reach this node (f): 7
People at start: {Hercules=6}
People at finish: {Bob=1, Aunty=12, Marios=8, Alice=3}
Torch position: 1
Available time: 8
Heuristic score (h): 1
Cost to reach this node from the initial node (g): 5
Heuristic score + Cost to reach this node (f): 6
People at start: {Bob=1, Hercules=6}
People at finish: {Aunty=12, Marios=8, Alice=3}
Torch position: 0
Available time: 7
Heuristic score (h): 2
Cost to reach this node from the initial node (g): 6
Heuristic score + Cost to reach this node (f): 8
People at start: {}
People at finish: {Bob=1, Aunty=12, Marios=8, Alice=3, Hercules=6}
Torch position: 1
Available time: 1
Heuristic score (h): 0
Cost to reach this node from the initial node (g): 7
Heuristic score + Cost to reach this node (f): 7
Search time:0.579 sec.
```