Χρονοδιάγραμμα Υλοποίησης

Ισουφάι Ντέιβιντ 3210056

Χρήστος Πέκος 3180153

Person.java

Αρχικά, ξεκινήσαμε κάνοντας develop την κλάση Person (βλ Person.java). Η κλάση αυτή δημιουργεί αντικείμενα τύπου Person, όπου κάθε αντικείμενο Person αναπαριστά ένα μέλος της οικογένειας.

Η κλάση αυτή έχει ένα στατικό γνώρισμα, το numberOfMembers, που δείχνει τον αριθμό των μελών της οικογένειας που έχουν δημιουργηθεί στην εκάστοτε εκτέλεση του προγράμματος. (το numberOfMembers έχει αρχικοποιηθεί με την τιμή 0)

Επίσης κάθε αντικείμενο Person περιέχει 2 γνωρίσματα, το pid (id με το οποίο ξεχωρίζουμε τα μέλη) και time (που συμβολίζει τον χρόνο που απαιτεί ένα άτομο για να διασχίσει την γέφυρα.

Η κλάση περιέχει τον κατασκευαστή Person(int time), όπου δημιουργεί ένα καινούργιο άτομο και αρχικοποιώντας τον χρόνο του. Επιπλέον, αυξάνει την στατική μεταβλητή numberOfMembers κατά 1.

Τέλος, έχουμε δημιουργήσει setters και getters για τα γνωρίσματα pid και time αντίστοιχα.

State.java

Ύστερα, ξεκινήσαμε την υλοποίηση της κλάσης State (βλ State.java) Η κλάση αυτή δημιουργεί αντικείμενα τύπου State, Όπου κάθε State αντικείμενο αναπαριστά μια κατάσταση του παιχνιδιού “Bridge Crossing”

Κάθε αντικείμενο τύπου State περιέχει 6 γνωρίσματα. Τα γνωρίσματα left και right είναι δύο arraylists τύπου Person, και συμβολίζουν τις 2 μεριές της γέφυρας, καθώς και ποια άτομα βρίσκονται στην κάθε μεριά. Το γνώρισμα Father είναι τύπου State και αποτελεί τον γονέα του αντικειμένου State, δηλαδή την προηγούμενη κατάσταση από την οποία δημιουργήθηκε το συγκεκριμένο State. Το totalTime είναι γνώρισμα τύπου int και κρατάει το συνολικό χρόνο που έχει περάσει κατά τις κινήσεις των ατόμων. Το γνώρισμα lamp, είναι τύπου Boolean και δείχνει σε ποια πλευρά της γέφυρας βρίσκεται η λάμπα (True αν βρίσκεται δεξιά, False αν βρίσκεται αριστερά). Επιπλέον, το f είναι γνώρισμα τύπου int, και συμβολίζει το συνολικό κόστος. Τέλος, για κάθε γνώρισμα έχουμε υλοποιήσει τους κατάλληλους Setters και Getters.

Για την κλάση State έχουμε υλοποιήσει 3 constructors. Ο πρώτος, δέχεται ως παράμετρο ένα Arraylist τύπου Person που αντιπροσωπεύει την αρχική κατάσταση παιχνιδιού. Ο δεύτερος,

δέχεται όλες τις παραμέτρους ενός αντικειμένου State, τα αντιγράφει, και δημιουργεί ένα νέο αντικείμενο State. Τέλος, ο τρίτος constructors δέχεται ως γνώρισμα ένα αντικείμενο State και αντιγράφοντας τις παραμέτρους του δημιουργεί ένα καινούργιο αντικείμενο State.

Η μέθοδος print() εκτυπώνει μια αναπαράσταση ενός state, έτσι ώστε να έχουμε μια καλύτερη εικόνα για το τι συμβαίνει σε κάθε κίνηση.

Οι μέθοδοι makeMoveRight(int id) makeMoveLeft(int id) δέχονται ως παράμετρο το id ενός ατόμου και εκτελούν την κίνηση που αναφέρει το όνομα τους (αριστερά στα δεξιά ή δεξιά στα αριστερά αντίστοιχα), επιστρέφοντας το τροποποιημένο state μετά την κίνηση. Η μέθοδος makeMoveLeft(int id1, int id2) απλά αντί για ένα άτομο την κίνηση την πραγματοποιούν δύο, ανάλογα με τα id που δίνονται ως παράμετροι.

Η μέθοδος makeMove() με βάση το State, εξετάζει την μεριά που βρίσκεται η λάμπα στην συγκεκριμένη κατάσταση και δημιουργεί πιθανά παιδιά που μπορεί να υπάρξουν από το Δέντρο(δηλαδή κινήσεις ενός η δύο ατόμων), επιστρέφοντας στο τέλος ένα ArrayList τύπου State όπου περιέχει τα υποψήφια παιδιά. Αρχικά είχαμε υλοποιήσει το πρόγραμμα μας να δημιουργεί όλα τα πιθανά παιδιά που μπορούν να υπάρξουν ανάλογα την μεριά που βρίσκεται η λάμπα (κάνοντας όλες τις πιθανές κινήσεις, είτε ενός είτε δύο ατόμων). Επειδή όμως αυτό μεγάλωνε αρκετά τον χρόνο εκτέλεσης του προγράμματος, (το πρόγραμμα έτρεχε σωστά), αποφασίσαμε να απορρίψουμε την πιθανότητα να μετακινούνται δύο άτομα από αριστερά στα δεξιά ή ένα άτομο από δεξιά στα αριστερά (εκτός της περίπτωσης του να υπάρχει μόνο 1 άτομο στο παιχνίδι). Να σημειωθεί όμως, και στις δύο περιπτώσεις, το πρόγραμμα έλυνε το πρόβλημα με τον ακριβώς ίδιο τρόπο. Πριν εκτελέσουμε οποιαδήποτε κίνηση, παίρνουμε ένα αντίγραφο του State. Τέλος, για κάθε υποψήφιο παιδί, υπολογίζει και την f του και το πως διαμορφώνει την totalTime.

Η μέθοδος EstimateCost() είναι η ευρετική συνάρτηση του προγράμματος μας, με την οποία εκτιμούμε το κόστος που θα έχει το να πάμε από την τρέχουσα κατάσταση σε μια τελική κατάσταση. Αρχικά, ελέγχει την πλευρά που βρίσκεται η λάμπα. Στην EstimateCost(), κάνουμε άρση του περιορισμού το πολύ 2 άτομα μπορούν να περάσουν την γέφυρα σε μια κίνηση. Αν η λάμπα βρίσκεται στα δεξιά, ουσιαστικά περνάνε όλα τα μέλη της οικογένειας στην αριστερή πλευρά, και επιστρέφουμε τον χρόνο του πιο αργού. Αν η λάμπα είναι στα αριστερά, αρχικά ελέγχουμε αν βρισκόμαστε σε τελική κατάσταση. Αν όχι, επιστρέφει τον χρόνο που θα χρειαστεί το γρηγορότερο άτομο να διασχίσει την γέφυρα, προσθέτοντας τον χρόνο που θα χρειαστεί το αργότερο άτομο της δεξιάς πλευράς να περάσει στα αριστερά (και σε αυτή τη περίπτωση θεωρούμε ότι όλα τα άτομα της δεξιάς μεριάς περνάνε μαζι). Αλλιώς, η ευρετική επιστρέφει 0.

Η ευρετική συνάρτηση είναι αποδεκτή, γιατί υποτίμα το πραγματικό κόστος του βέλτιστου μονοπατιού από την τωρινή κατάσταση σε κάποια κατάσταση τερματισμού. Αυτό συμβαίνει γιατί στην ευρετική θα κάνουμε το πολύ δύο κινήσεις για να φτάσουμε σε τελική κατάσταση, ενώ στην πραγματικότητα θα χρειαστούν τουλάχιστον 2 (αφού στην πραγματικότητα έχουμε τον περιορισμό ότι το πολύ δύο άτομα μπορούν να διασχίσουν την γέφυρα).

Έστω ότι βρισκόμαστε σε μια κατάσταση n. Για να φτάσουμε σε μια κατάσταση n', το κόστος θα είναι c.

Αν η λάμπα βρίσκεται στα δεξιά το h(n) =χρόνος του αργότερου από τα δεξιά. Έστω ότι μεταβαίνουμε σε μια κατάσταση n' με χρόνο c>0.

α) φτάνουμε στην n' και ο πιο αργός πέρασε τη γέφυρα

Σε αυτή τη περίπτωση έχουμε, h(n) = c. Και προφανώς h(n') >= 0 (0 αν φτάσαμε σε τελική κατάσταση). Συνεπώς h(n) <= c1 + h(n')

β) φτάνουμε στην n' χωρις τον πιο αργό να περνάει την γέφυρα

Αν το αργότερο άτομο δεν πέρασε την γέφυρα είχαμε κόστος c2. h(n') = χρόνος του γρηγορότερου από αριστερά + χρόνος αργότερου από δεξιά > h(n). Συνεπώς h(n) <= c2 + h(n')

Αν η λάμπα βρίσκεται στα αριστερά, το h(n) = χρόνος γρηγορότερου από τα αριστερά + χρόνος αργότερου από τα δεξιά. Το κόστος για να φτάσουμε σε μια n' θα είναι πάντα >= χρόνος γρηγορότερου από τα αριστερά. Η h(n') θα είναι χρόνος του αργότερου από τα δεξιά. Συνεπώς h(n) <= c + h(n') οπότε η ευρετική μας είναι και σε αυτή τη περίπτωση συνεπής.

Τελικά, συμπεραίνουμε ότι η ευρετική μας είναι αποδεκτή και συνεπής.

Η μέθοδος isFinal() ελέγχει αν η κατάσταση είναι τελική η όχι (Επιστρέφει true αν είναι τελική, false αν δεν είναι)

Η evaluate() υπολογίζει το f, προσθέτοντας το ευρετικό κόστος + το κόστος από την ρίζα στην τρέχουσα κατάσταση (totalTime).

Η compareTo(State s) συγκρίνει το f της τρέχουσας κατάστασης με την κατάσταση που δίνεται ως παράμετρος.

SpaceSearcher.java

Στην συνέχεια δημιουργήσαμε την κλάση SpaceSearcher που υλοποιεί τον αλγόριθμο A* με κλειστό σύνολο. Γνωρίζουμε ότι ο αλγόριθμος αυτός είναι βέλτιστος με συνεπή και αποδεκτή ευρετική συνάρτηση. Χρησιμοποιήσαμε τον αλγόριθμο A* με κλειστό σύνολο και όχι τον απλό A*, διότι σε περίπτωση που βρισκόμαστε σε μια κατάσταση την οποία έχουμε ξανασυναντήσει, το κλειστό σύνολο μας αναγκάζει να διακόψουμε την εξερεύνηση ενός μονοπατιού κάτω από την κατάσταση αυτή.
Για την υλοποίηση του κώδικα του αλγορίθμου, χρησιμοποιήσαμε τον κώδικα του φροντιστηρίου για τον BestFS με κλειστό σύνολο. Αυτό γιατί γνωρίζουμε ότι η διαφορά των δύο αλγορίθμων βρίσκεται στο ότι ο BestFS δεν χρησιμοποιεί τα προηγούμενα κόστη αλλά μόνο το κόστος της ευρετικής. Έχοντας έτσι υλοποιήσει στην State την μέθοδο evaluate(), ο κώδικας του φροντιστηρίου μπορεί να χρησιμοποιηθεί και για την υλοποίηση του A* με κλειστό σύνολο.

Πιο συγκεκριμένα χρησιμοποιούμε μια λίστα frontier που αντιπροσωπεύει το μέτωπο αναζήτησης και ένα HashSet closedSet που αντιπροσωπεύει το κλειστό σύνολο.

Η μέθοδος AStarClosedSet(State initialState) δέχεται ως παράμετρο μια αρχική κατάσταση. Αρχικά ελέγχουμε αν η κατάσταση αυτή είναι τελική, την περίπτωση δηλαδή που υπάρχει μόνο ένα άτομο που θέλει να διασχίσει την γέφυρα, όπου και επιστρέφουμε την ίδια την κατάσταση. Αν δεν είναι τελική κατάσταση, την προσθέτουμε στο μέτωπο αναζήτησης και ξεκινάμε μια επανάληψη για όσο το μέτωπο περιέχει καταστάσεις. Μέσα στον βρόγχο επανάληψης αφαιρούμε κάθε φορά το πρώτο στοιχείο και ελέγχουμε αν είναι τελική κατάσταση, όπου αν είναι το επιστρέφουμε. Αν δεν είναι, ελέγχουμε αν το κλειστό σύνολο περιέχει την κατάσταση που εξετάζουμε. Αν όχι, την προσθέτουμε στο κλειστό σύνολο, προσθέτουμε στο μέτωπο αναζήτησης όλα τα παιδιά της και τέλος ταξινομούμε το μέτωπο με βάση το συνολικό κόστος f.

Αν ο αλγόριθμος δεν καταφέρει να βρει κάποια τελική κατάσταση, τότε επιστρέφει null.

Main.java

Τέλος, υλοποιήσαμε την κλάση και μέθοδο Main.

Αρχικά ζητάμε από τον χρήστη να εισάγει τον συνολικό αριθμό των μελών της οικογένειας και έπειτα μέσα σε έναν επαναληπτικό βρόχο ζητάμε τον χρόνο που χρειάζεται να διασχίσει την γέφυρα το κάθε μέλος, όπου κάθε φορά δημιουργούμε νέο αντικείμενο τύπου Person, το

οποίο το προσθέτουμε σε μια λίστα familyMembers. Επίσης ζητάμε και τον χρόνο που η λάμπα μένει αναμμένη.

Υλοποιούμε την αρχική κατάσταση του παιχνιδιού, φτιάχνοντας ένα νέο αντικείμενο τύπο State με παράμετρο την λίστα familyMembers. Επίσης δημιουργούμε νέο αντικείμενο με όνομα searcher τύπου SpaceSearcher.

Χρησιμοποιήσαμε την μεταβλητή start, ώστε να αποθηκεύσουμε την χρονική στιγμή πριν εκτελέσουμε τον αλγόριθμο αναζήτησης, έπειτα εκτελέσαμε τον αλγόριθμο και τέλος αποθηκεύσαμε την χρονική στιγμή που τέλειωσε ο αλγόριθμος σε μια μεταβλητή end.

Στην συνέχεια ελέγχουμε αν ο αλγόριθμος αναζήτησης επέστρεψε κάποια τελική κατάσταση, αν όχι εκτυπώνουμε πως δεν βρήκαμε κάποια λύση. Αν ναι, ελέγχουμε αρχικά αν ο συνολικός χρόνος που χρειάστηκαν τα μέλη να διασχίσουν την γέφυρα ξεπερνά τον χρόνο που η λάμπα είναι αναμμένη και εκτυπώνουμε το κατάλληλο μήνυμα. Αλλιώς σε μια λίστα path προσθέτουμε την τελική κατάσταση και επαναληπτικά προσθέτουμε τον πατέρα της κάθε νέας κατάστασης, μέχρι να φτάσουμε στην ρίζα.

Έπειτα χρησιμοποιούμε την μέθοδο reverse ώστε να αναστρέψουμε την λίστα, για να μπορέσουμε να εκτυπώσουμε το μονοπάτι από την ρίζα μέχρι τον τελικό κόμβο. Τέλος, εκτυπώνουμε και τον χρόνο που χρειάστηκε το πρόγραμμα για να βρει λύση.

Παραδείγματα εκτέλεσης προγράμματος

Παράδειγμα 1ο: Χρόνοι: 1,3,6,8,12 Χρόνος λάμπας: 30

Υπολογιστής 1:

```
Welcome to a simulation of family at the Bridge Game! Please type the number of family members:
5
Whats the time of family member 0 ? :
1
Whats the time of family member 1 ? :
3
Whats the time of family member 2 ? :
6
Whats the time of family member 3 ? :
8
Whats the time of family member 4 ? :
12
Last but not least, please type the duration time of the lamp:
30
|____________________| L 0 1 2 3 4
The total time is 0
0 1 L |____________________|2 3 4
The total time is 3
1 |____________________| L 2 3 4 0
The total time is 4
1 2 0 L |____________________|3 4
The total time is 10
1 2 |____________________| L 3 4 0
The total time is 11
1 2 3 4 L |____________________|0
The total time is 23
2 3 4 |____________________| L 0 1
The total time is 26
2 3 4 0 1 L |____________________|
The total time is 29

Search time:0.016 sec.
```

Υπολογιστής 2:

```
Welcome to a simulation of family at the Bridge Game! Please type the number of family members:
5
Whats the time of family member 0 ? :
1
Whats the time of family member 1 ? :
3
Whats the time of family member 2 ? :
6
Whats the time of family member 3 ? :
8
Whats the time of family member 4 ? :
12
Last but not least, please type the duration time of the lamp:
30
|____________________| L 0 1 2 3 4
The total time is 0
0 1 L |____________________|2 3 4
The total time is 3
1 |____________________| L 2 3 4 0
The total time is 4
1 2 0 L |____________________|3 4
The total time is 10
1 2 |____________________| L 3 4 0
The total time is 11
1 2 3 4 L |____________________|0
The total time is 23
2 3 4 |____________________| L 0 1
The total time is 26
2 3 4 0 1 L |____________________|
The total time is 29

Search time:0.015 sec.
```

Παράδειγμα 2ο: Χρόνοι: 1,3,6,8,12,15 Χρόνος λάμπας: 45

Υπολογιστής 1:

```
Welcome to a simulation of family at the Bridge Game! Please type the number of family members:
6
Whats the time of family member 0 ? :
1
Whats the time of family member 1 ? :
3
Whats the time of family member 2 ? :
6
Whats the time of family member 3 ? :
8
Whats the time of family member 4 ? :
12
Whats the time of family member 5 ? :
15
Last but not least, please type the duration time of the lamp:
45
|____________________| L 0 1 2 3 4 5
The total time is 0
0 1 L |____________________|2 3 4 5
The total time is 3
1 |____________________| L 2 3 4 5 0
The total time is 4
1 2 3 L |____________________|4 5 0
The total time is 12
2 3 |____________________| L 4 5 0 1
The total time is 15
2 3 0 1 L |____________________|4 5
The total time is 18
2 3 1 |____________________| L 4 5 0
The total time is 19
2 3 1 4 5 L |____________________|0
The total time is 34
2 3 4 5 |____________________| L 0 1
The total time is 37
2 3 4 5 0 1 L |____________________|
The total time is 40

Search time:0.091 sec.
```

Υπολογιστής 2:

```
Welcome to a simulation of family at the Bridge Game! Please type the number of family members:
6
Whats the time of family member 0 ? :
1
Whats the time of family member 1 ? :
3
Whats the time of family member 2 ? :
6
Whats the time of family member 3 ? :
8
Whats the time of family member 4 ? :
12
Whats the time of family member 5 ? :
15
Last but not least, please type the duration time of the lamp:
45
|____________________| L 0 1 2 3 4 5
The total time is 0
0 1 L |____________________|2 3 4 5
The total time is 3
1 |____________________| L 2 3 4 5 0
The total time is 4
1 2 3 L |____________________|4 5 0
The total time is 12
2 3 |____________________| L 4 5 0 1
The total time is 15
2 3 0 1 L |____________________|4 5
The total time is 18
2 3 1 |____________________| L 4 5 0
The total time is 19
2 3 1 4 5 L |____________________|0
The total time is 34
2 3 4 5 |____________________| L 0 1
The total time is 37
2 3 4 5 0 1 L |____________________|
The total time is 40

Search time:0.079 sec.
```

Παράδειγμα 3ο: Χρόνοι: 1,3,6,8,12,15 Χρόνος λάμπας: 30

Υπολογιστής 1:

```
Welcome to a simulation of family at the Bridge Game! Please type the number of family members:
6
Whats the time of family member 0 ? :
1
Whats the time of family member 1 ? :
3
Whats the time of family member 2 ? :
6
Whats the time of family member 3 ? :
8
Whats the time of family member 4 ? :
12
Whats the time of family member 5 ? :
15
Last but not least, please type the duration time of the lamp:
30
No path found within the lamp's duration time (30 sec.)
```

Υπολογιστής 2:

```
Welcome to a simulation of family at the Bridge Game! Please type the number of family members:
6
Whats the time of family member 0 ? :
1
Whats the time of family member 1 ? :
3
Whats the time of family member 2 ? :
6
Whats the time of family member 3 ? :
8
Whats the time of family member 4 ? :
12
Whats the time of family member 5 ? :
15
Last but not least, please type the duration time of the lamp:
30
No path found within the lamp's duration time (30 sec.)
```