Ονόματα: Ελένη Τράμπαρη Λάρδα & Χαράλαμπος Σπανός
ΑΜ: 3180186 & 3180172 αντίστοιχα

Εργασία 1

(α) Διάσχιση της γέφυρας

Περιληπτική περιγραφή:

Στο συγκεκριμένο πρόβλημα μας δίνεται ως είσοδος (στο terminal) ένα συγκεκριμένο πλήθος ατόμων (N) μαζί με το χρόνο του καθενός (ntime[]). Σκοπός του προβλήματος αυτού είναι να καταφέρουμε να μεταφέρουμε όσο το δυνατόν γρηγορότερα αυτά τα άτομα από τη μια όχθη στην άλλη μέσω της γέφυρας (έχοντας ως περιορισμό ότι η γέφυρα αντέχει μέχρι 2 άτομα και χρειάζονται κάθε φορά ένα φακό).

StateBridge class:

Η συγκεκριμένη κλάση αφορά αντικείμενα που δείχνουν μια κατάσταση που μπορείς να βρεθείς από τις μετακινήσεις στη γέφυρα. Πιο συγκεκριμένα, έχουμε ορίσει 2 Array List Left & Right όπου στη Left είναι τα άτομα (οι χρόνοι τους) που βρίσκονται εκείνη τη στιγμή στην αριστερή όχθη και στη Right αντίστοιχα είναι τα άτομα (οι χρόνοι τους) που βρίσκονται εκείνη τη στιγμή στην δεξιά όχθη.

Όπως είναι αναμενόμενο έχουμε ορίσει επίσης διάφορους getters & setters ώστε να έχουμε πρόσβαση στις private μεταβλητές του StateBridge σε άλλες κλάσεις.

Η Boolean συνάρτηση isTerminal() ελέγχει αν μια κατάσταση είναι τερματική, δηλαδή αν δεν υπάρχουν άλλα άτομα στην λίστα Right.

Η void moveRight είναι μια συνάρτηση που κάνει τη μετακίνηση του πιο γρήγορου ατόμου από την αριστερή όχθη στην δεξιά, το οποίο είναι απαραίτητο να γίνει ώστε να επιστρέψει ο φακός για να περάσουν και οι υπόλοιποι.

Τέλος, έχει οριστεί η ευρετική συνάρτηση heuristic1 η οποία υπολογίζει το σκορ το οποίο είναι το άθροισμα αυτών που βρίσκονται στην αριστερή όχθη (η ευρετική καλείται στη κλάση Algorithm που θα συζητηθεί παρακάτω).

Όσον αφορά τον constructor περνάμε τις μεταβλητές N & ntime και βάζουμε στη λίστα Right τους χρόνους από το ntime.

Ο copy constructor μας χρησιμεύει στο πρόγραμμα αρκετές φορές και έχουμε περάσει στο αντικείμενο ό,τι πληροφορίες χρειάζονται (N, ntime, Right, Left, time, father).

Algorithm class

Σε αυτή τη κλάση υλοποιείται ο αλγόριθμος A*. Στον constructor έχουμε την αρχική κατάσταση (root), δηλαδή όλα τα στοιχεία στην λίστα Right.

Η Astar() είναι μια μέθοδος η οποία επιστρέφει τη τελική κατάσταση (StateBridge), δηλαδή όταν όλα τα στοιχεία είναι στη λίστα Left.

Αρχικά, δημιουργεί τα παιδιά της root δηλαδή όλες τις πιθανές περιπτώσεις μετακίνησης των ατόμων από δεξιά προς τα αριστερά σε δυάδες. Είναι σημαντικό να τονίσουμε ότι για κάθε αντικείμενο ορίζουμε τον πατέρα του (father) δηλαδή από ποιο αντικείμενο δημιουργήθηκε, αλλά και τον χρόνο (time) που είναι ο χρόνος του πατέρα του + το χρόνο μετακίνησης (από right σε left) για το συγκεκριμένο παιδί που δημιουργήθηκε (μέγιστος χρόνος εκ των δυο που μετακινούνται προς τα αριστερά). Το πλήθος των παιδιών που δημιουργούμε κάθε φορά το υπολογίζουμε με βάση το τύπο N*(N-1)/2 όπου N είναι ο αριθμός των ατόμων που έχουν απομείνει δεξιά.

Επίσης, έχουμε ορίσει μια Array List metopo η οποία περιέχει τα αντικείμενα StateBridge που ελέγχουμε κάθε φορά (αρχικά το metopo περιέχει μόνο το root) κάνοντας χρήση της ευρετικής. Για κάθε στοιχείο από το metopo καλούμε την ευρετική όπου υπολογίζεται το σκορ που αναφέραμε παραπάνω και ύστερα υπολογίζουμε το άθροισμα σκορ + total_time, όπου total_time είναι ο χρόνος που μας πήρε μέχρι να φτάσουμε σε αυτή την κατάσταση (=time). Έπειτα, κρατάμε το ελάχιστο άθροισμα (max_score) όπως και το αντικείμενο που αντιστοιχεί αυτό το άθροισμα (maxE). Στην συνέχεια , ελέγχουμε αν το πλήθος των ατόμων N έχει γίνει μηδέν, το οποίο θα σημαίνει ότι θα έχουμε βρει την τελική κατάστασή μας και θα την κάνουμε return . Αν όμως δεν έχει γίνει κάτι τέτοιο συνεχίζουμε εκτελώντας την moveRight και κρατώντας current πλέον την κατάσταση που θα μας γυρίσει η moveRight . Αρχικοποιούμε τις μεταβλητές και αφαιρούμε το πλήθος των ανθρώπων N που έχουμε κατά 1(αφού έχει περάσει ήδη 1 άτομο στην όχθη left).Η διαδικασία συνεχίζετε μέχρι να βρεθεί τελική κατάσταση (δηλαδή το N=0) και όταν γίνει αυτό τότε την επιστρέφουμε.

Η print(StateBridge x) είναι μια μέθοδος η οποία δέχεται σαν όρισμα την τερματική κατάσταση που θα έχει προκύψει (StateBridge x), αφού εφαρμοστεί ο Astar(). Έχουμε δημιουργήσει μια Array List path με αντικείμενα StateBridge όπου αρχικά περιέχει τη τελική κατάσταση (current). Προσθέτουμε στο path κάθε φορά τον father του συγκεκριμένου αντικειμένου μέχρι να μην υπάρχει πια πατέρας δηλαδή μέχρι να φτάσουμε στο root. Όταν τελειώσει η διαδικασία αυτή εκτυπώνουμε το path από το τέλος στην αρχή του ώστε να εκτυπωθούν οι λίστες Left, Right του κάθε στοιχείου από το root στο current.

main class:

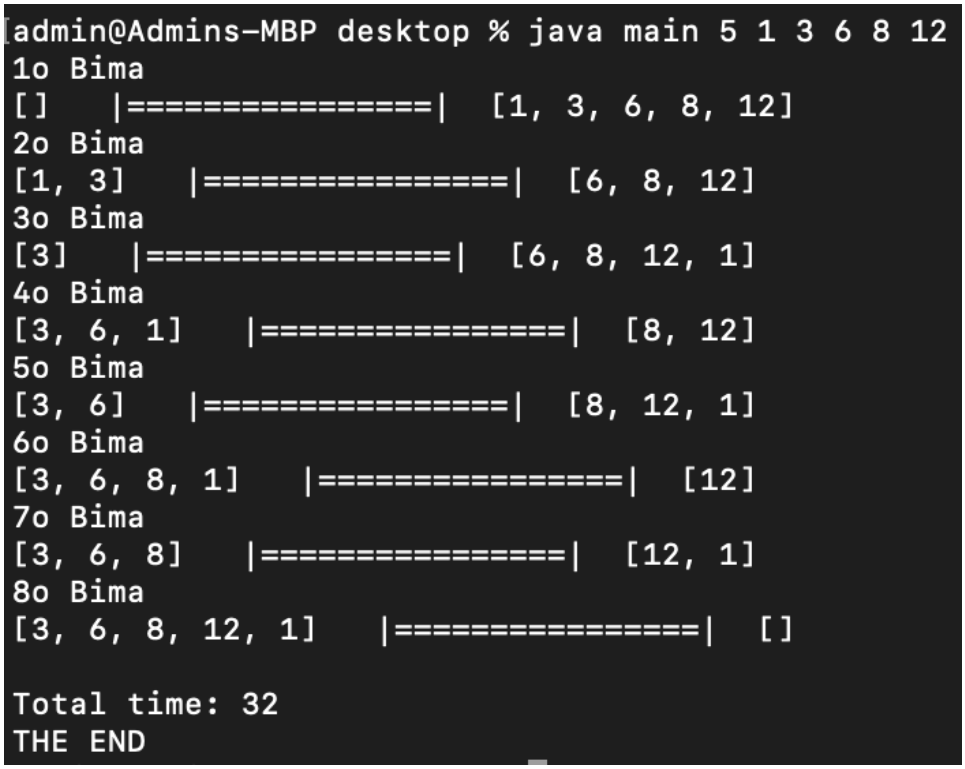
Σε αυτή τη κλάση περνάμε τα ορίσματα από τα arguments σε αντίστοιχες μεταβλητές, ελέγχουμε αν το N είναι θετικό εκτυπώνοντας κατάλληλο μήνυμα και δημιουργούμε ένα αντικείμενο StateBridge root με αυτά τα στοιχεία. Έπειτα, δημιουργούμε ένα αντικείμενο Algorithm a δίνοντας του σαν όρισμα το root και καλούμε τη μέθοδο Astar. Τέλος καλούμε τη print με όρισμα τη τελική κατάσταση που έχουμε μετά από το κάλεσμα της μεθόδου Astar.

Πειραματικά αποτελέσματα:

```
admin@Admins-MBP desktop % javac main.java
```

```
admin@Admins-MBP desktop % java main 3 10 20 30
1o Bima
[]   |================|  [10, 20, 30]
2o Bima
[10, 20]   |================|  [30]
3o Bima
[20]   |================|  [30, 10]
4o Bima
[20, 30, 10]   |================|  []

Total time: 60
THE END
```



Παρατηρήσεις:

Το πρόγραμμα έτρεξε σε περιβάλλον macOS Big Sur , 8GB RAM , Dual-Core
java 15.0.1
Java(TM) SE Runtime Environment (build 15.0.1+9-18)
Java HotSpot(TM) 64-Bit Server VM (build 15.0.1+9-18, mixed mode, sharing)