Δομές Δεδομένων

Εργασία 2

2014-2015

Χρυσάνθη Τσαμπίκα Μαυράκη 3130128

Ελευθέριος Χατζηαράπης 3130225

- ## Interface PQEntry

Δημιουργούμε μια διεπαφή η οποία δέχεται δεδομένα τύπου T και Y, όπου τα Y πρέπει να είναι απόγονοι της διεπαφής PQKey.

(Τα T θα είναι Flights τα Y keyflight)

Μ'αυτό τον τρόπο εκτελούμε την εργασία με χρήση Generics.

Επιπλέον η διεπαφή ορίζει μεθόδους getKey() και setKey(Y k)

- ## Interface PQKey

Δημιουργούμε μια διεπαφή η οποία δέχεται δεδομένα τύπου Y Και κανει extend Comparable<Y>.

Επίσης ορίζουμε μεθόδους getPos(), setPos(int pos), και pushToTop() που επιστρέφει αντικέιμενο τύπου Y

- ## Class KeyFlight

Δημιουργούμε μια κλάση η οποία υλοποιεί την διεπαφή PQKey<KeyFlight> (δηλαδή θεωρούμε τον τύπο Y KeyFlight).

Η κλάση έχει τρία private int πεδία τα number(ο αριθμός πτήσεις), time(η ώρα που πετάει) και position(σε πια θέση βρίσκεται στην ουρά προτεραιότητας).

Αφού γνωρίζουμε απο την εκφώνηση πως κάθε μέρα αποδεκτά κλειδία είναι απο 0 έως 999, αρχικοποιούμε 1000 κλειδιά

και τους βάζουμε ως number τον αντίστοιχο αριθμό απο το 0-999 και επίσης βάζουμε position =-1 γιατί θεωρούμε πως δεν υπάρχουν ακόμα τα κλειδιά στην ουρά προτεραιότητας.

Δημιουργώ setters getters.

Υλοποιώ την compareTo(KeyFlight k)

Ελέγχουμε αν η ώρα του καλούντος αντικειμένου είναι μικρότερη απο του k τότε επιστρέφουμε

Αν είναι μικρότερη επιστρέφουμε -1

Αν είναι ίσες τότε ελέγχουμε τους αριθμούς πτήσεων και αν το

καλούν αντικέιμενο έχει μικρότερο αριθμό πτήσεις επιστρέφουμε

1 αλλιώς -1 (περίπτωσή να είναι ίσα δεν υπάρχει λόγω της αρχικοποίησης των 1000 κλειδιών)

Επίσης υλοποιώ την pushToTop() η οποία θέτει το time=-1 γιατί η πτήση με το μικρότερο time έχει την μεγαλύτερη προτεραιότητα οπότε αυτή η μέθοδος θα κάνει την πτήση μας πρώτη στην ουρά.

- ## Class Flight

Δημιουργούμε μια κλάση η οποία υλοποιεί την διεπαφή

PQEntry<Flight,KeyFlight>. Έχει ένα private boolean πεδίο departs το οποίο κρατά αν είναι αναχώρηση η άφιξη και ένα private KeyFlight πεδίο το οποίο είναι το "κλειδί " της πτήσης.

Υλοποιούμε setters getters και την compareTo με όρισμα flight

Έχουμε δύο κατασκευαστές. Ο ένας πέρνει boolean τιμή για το departs και ένα KeyFlight κλειδί. εκτελούμε τους αντίστοιχους setters

Ο δεύτερος κατασκευαστής πέρνει μόνο ένα κλειδί και θεωρεί ως default το departs false.

Επίσης υπερκαλύπτουμε την toString ώστε να εμφανιστεί το κατάλληλο μήνυμα όταν τυπώσουμε τα Flights στη main

- ## Class PQ

Δημιουργούμε μια κλάση PQ <T extends PQEntry<T,Y>,Y extends PQKey<Y>>

Δηλαδή για την κλάση μας το T θα είναι απόγονος της διεπαφής PQEntry και το Y θα είναι απόγονος της διεπαφής PQKey

Ορίζουμε έναν πίνακα τύπου T ο οποίος θα είναι η ουρά και ένα int το οποίο θα είναι το μέγεθος του πίνακα αυτού, ανα πάσα στιγμή.

- void insertT(T flight)

Αν το flight που πάμε να εισάγουμε είναι null τότε βγάζουμε exception. Αν το size1 είναι ίσο με το μέγεθος του πίνακα που αρχικοποιήσαμε -1 τότε κάνουμε resize στον πίνακα και του προσθέτουμε άλλες 20 θέσεις.

Ύστερά ορίζουμε το position του κλειδιού και βάζουμε το flight στην ουρά. Τέλος καλούμε την smim με όρισμα την θέση στην οποία βάλαμε το τελευταίο στοιχείο.

- T Max()

Αν η ουρά προτεραιότητας είναι άδεια ρίχνουμε exception αλλιώς επιστρέφουμε τη ρίζα.

- T getMax()

Αν η ουρά προτεραιότητας είναι άδεια ρίχνουμε exception αλλιώς δημιουργούμε ένα αντικέιμενο τύπου T ίσο με τη ρίζα (heap[1])

Αν δέν είμαστε στη ρίζα τότε βάζουμε στη ρίζα το στοιχείο που είναι στη θέση size1(η οποία θα είναι το πιο δεξί φύλλο), κάνουμε το πιο δεξί φύλλο null (μειώνοντας παράλληλα και το size1 Κατα 1) και καλούμε την sink απο τη ρίζα. Βγάζουμε το flight απο την ουρά θέτωντας το Position του σε -1 και το επιστρέφουμε.

- void swim(int i)

Όσο το i είναι μεγαλύτερο του 1, βρίσκουμε τον πατέρα του (p=i/2), συγκρίνουμε το στοιχείο στη θέση ι με τον πατέρα του. Αν το αποτέλεσμα είναι αρνητικό τότε τελειώσαμε, αλλιώς αντιμεταθέτουμε τα στοιχεία και επανλαμβάνουμε.

- void sink(int i)

Θέτουμε αριστερό παιδί το 2*i, δεξί το αριστερό +1 και έστω max το αριστερό. Όσο το αριστερό είναι μικρότερο ή ίσο με το size1, αν το δεξί είναι μικρότερο του size1, τότε αν η σύγκρισή μεταξύ αριστερού και δεξίου παιδίου βγάλει αρνητικό αποτέλεσμα θέτουμε Max=right αλλιώς Max= left. Τώρα αν το στοιχείο στη θέση ι σε σύγκριση με το max βγάζει αρνητικό αποτέλεσμα τότε αντιμεταθέτουμε το ι με το max επαναπροσδιορίζουμε τα αριστερά και δεξία παιδιά και επαναλαμβάνουμε.

- void swap(int i, int j)

Πρόκειται για απλή αντιμετάθεση στοιχείων του πίνακα μας.

- int size()

Επιστρέφει το μέγεθος της ουράς εκείνη τη στιγμή.

- void updateKey(T x,Y k)

Αρχικοποιούμε ένα int = με τη θέση του κλειδιού του x μέσα στην ουρά. Δημιουργώ ένα προσωρινό αντικείμενο Y = με το κλειδί του x. Αλλάζουμε το κλειδί του x και το κάνουμε ίσο με k. Αν η συγκριση του νέου κλειδιού με το παλίο φέρει αρνητικό αποτέλεσμα τότε καλούμε την sink αλλιώς καλούμε την swim.

- T remove(Y k)

Ελέγχουμε αν το κλειδί k υπάρχει στην ουρά προτεραιότητας. Αν υπάρχει τότε αρχικοποιούμε ένα αντικείμενο T με όνομα f = το αντικείμενο της ουράς προτεραιότητας που υπάρχει στην θέση με κλειδί k. Αρχικοποιούμε ενα αντικείμενο Y knew και καλούμε την

pushToTop() ώστε να πάει στη ρίζα του δέντρου. Καλούμε την swim και ύστερα την getMax() τότε έχουμε αποκαταστήσει την ισορροπία στο δέντρο μας .

- isEmpty()

Ελέγχει αν η ουρά είναι άδεια

- PQ(int capacity)

Κατασκευαστής ο οποίος δέχεται το μέγεθος της ουράς και αρχικοποιεί τον πίνακα της ουράς (+1 επειδή το πρώτο πεδίο της ουράς είναι άδειο από σύμβαση)

- **Class FlightSchedule**

✓ String FlightListNumber

Είναι η πρώτη «λέξη» σε κάθε γραμμή του αρχείου. Πρόκειται για την ώρα στην οποία θα κάνει κάποια ενημέρωση ο ελεγκτής εναέρια κυκλοφορίας.

✓ int FlightListNum

Το παραπάνω string αφού το μετατρέψουμε σε ακέραιο.

✓ String FlightCondition

Ο τύπος τηw ενημέρωσης που θα γίνει, ακύρωση, εισαγωγή, αλλαγή ώρας.

✓ String FlightD_A

Το string αυτό μας δείχνει αν πρόκειται για αναχώρηση η

άφιξη.

- ✓ String FlightTime

  Η συμβολοσειρά με την ώρα στην οποία θα γίνει η άφιξη αναχώρηση.

  Ή παραπάνω ώρα αφού τη μετατρέψουμε σε ακέραιο.

- ✓ int FlightT

  Ή παραπάνω ώρα αφού τη μετατρέψουμε σε ακέραιο.

- ✓ int FlightKey

  Ο αριθμός πτήσης.

  Αρχικοποιούμε μια priority queue pq με αρχικό capacity 5.

  Ύστερα στην main

  Καλούμε την KeyFlight.initMethod(); Η οποία αρχικοποιεί τα 1000 κλειδιά. Ζητάμε απο τον χρήστη να εισάγει το όνομα του αρχείου και αφού κάνουμε τους απαραίτητους ελέγχους για τον αν υπάρχει η αν δεν μπορούμε να πάρουμε σωστά τα στοιχεία απο το αρχείο, ξεκινάμε τη διαδικασία, καλώντας την μέθοδο getData

  Όσο το αρχείο έχει επόμενη «λέξη»

  Αρχικοποιούμε ένα νεο flight f=null;

  Ύστερα ελέγχουμε το FlightCondition. Σε περίπτωση που

αυτό δεν είναι cancel τότε Και με την σειρά αρχικοποιούμε τις κατάλληλες μεταβλητές ελέγχωντας αν οι ώρες είναι αποδεκτές.

Αφού έχουμε τα δεδομένα στις κατάλληλες μεταβλητές καλούμε τον κατασκευαστή (αν πρόκειται για cancel καλούμε τον κατασκευαστή που παίρνει μόνο κλειδί).

Για να είναι συνεχώς ανανεωμένη η ουρά προτεραιότητας πριν κάνουμε την οποιαδήποτε ενημέρωση ελέγχουμε αν είναι άδεια και αν δεν είναι και το FlightListNum είναι μεγαλύτερο απο την ώρα πτήσης του στοιχείου με την μεγαλύτερη προτεραιότητα αφαιρούμε το στοιχείο και επαναλαμβάνουμε έως ότου η συνθήκη να είναι ψευδής.

Συνεχίζουμε με τις ενημερώσεις.

Αν το FlightCondition είναι insert τότε ελέγχουμε το FlightListNum αν είναι μεγαλύτερο απο την ώρα πτήσης τότε βγάζουμε μήνυμα λάθους. Αλλιώς εισάγουμε την ουρά προτεραιότητας το νέο στοιχείο.

Αν το FlightCondition είναι update τότε ελέγχουμε αν η ουρά είναι άδεια και αν δεν είναι τότε ορίζω το νέο time στο f και καλώ την updateKey

Αν είναι cancel τότε ελέγχω αν η ουρά είναι άδεια και αν δεν είναι αφαιρώ την πτήση απο την ουρά καλώντας την remove.

Τέλος αφού διαβαστεί ολόκληρο το αρχείο αρχίζω να τυπώνω και να αφαιρώ τα υπολοιπόμενα στοιχεία της ουράς (για πτήσεις που πετάνε μετά το τελευταίο FlightListNum).