ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ

ΕΡΓΑΣΙΑ 2

ΙΑΚΩΒΟΣ ΕΥΔΑΙΜΩΝ : 3130059
ΣΤΕΦΑΝΟΣ ΠΑΥΛΟΠΟΥΛΟΣ : 3130168

ΜΕΡΟΣ Α

Στο μέρος Α πρέπει να δημιουργήσουμε 2 κλάσεις την PrintJob και την MaxPQ. Η 1η κλάση αφορά για την δημιουργία αντικειμένων τύπου PrintJob οπότε έχουμε μια κλάση με τον constructor της και τους getters και setters της. Επίσης η κλάση PrintJob υλοποιεί την διεπαφή Comparable. Δημιουργούμε έναν κατασκευαστή καθώς όταν θα δημιουργούμε αντικείμενα αυτής της κλάσης θα πρέπει να δίνεται και ο χρόνος που έρχεται το υποτιθέμενο προς εκτύπωση αρχείο αλλά και το μέγεθος του, το οποίο πρέπει να είναι ακέραιος αριθμός και μεταξύ του διαστήματος ένα με 128. Μόνο αυτές τις δυο παραμέτρους θέλουμε καθώς τις υπόλοιπες τις ορίζουμε εμείς. Οι υπόλοιπες είναι οι παράμετροι id, waitingTime, priority τις οποίες ουσιαστικά ορίζουμε εμείς. Την priority την ορίζουμε ως το μέγεθος του αρχείου μειωμένη κατά ένα διότι η priority πρέπει να ορίζεται στο διάστημα από μηδέν μέχρι 127. Ουσιαστικά η πιο σημαντική μέθοδος που ορίζουμε είναι η compareTo ,η οποία ουσιαστικά είναι η μέθοδος που υλοποιεί την Comparable. Σκοπός της είναι να μας επιστρέψει έναν αριθμό είτε αρνητικό είτε μηδέν είτε θετικό μετά από την σύγκριση δύο αντικειμένων PrintJob βάση με το priority τους.

Στην κλάση MaxPQ φτιάχνουμε μια priority queue κατασκευάζοντας έναν πίνακα με όνομα σωρό(heap) που θα έχει αρχεία τύπου PrintJob. Ο κατασκευαστής αυτής της κλάσης δημιουργεί έναν πίνακα ανάλογα με την χωρητικότητα που θα του δώσουμε αυξημένη κατά ένα καθώς αποθηκεύουμε αντικείμενα στον πινάκα από την 1η θέση και μετά αφού θέλουμε να έχουμε μια σχέση ανάμεσα σε πατέρα και παιδιά ώστε άμα τα παιδιά είναι στην θέση x και x+1 ο πατέρας να είναι στην θέση 2x. Έπειτα έχουμε τις μεθόδους insert, getMax, swim, swap, sink, empty, print, peek, size, getHeap, resize. Η insert χρησιμεύει για να εισάγουμε ένα αρχείο για το οποίο μετά καλούμε την swim για να το κατατάξει στη σωστή θέση ανάλογα με το priority του. Η getMax χρησιμεύει για να πάρουμε το αρχείο με την μεγαλύτερη priority(για τον σκοπό της εργασία μας αυτό με την

μικρότερη τιμή). Η μέθοδος resize καλείται άμα παρατηρηθεί ότι τα στοιχεία που έχουν εισαχθεί στον πίνακα φτάνουν το 75% της χωρητικότητας του. Τότε τον διπλασιάζουμε για να μην έχουμε προβλήματα. Την μέθοδο size την χρησιμοποιούμε για να ελέγξουμε πόσα αντικείμενα έχουμε εισάγει. Την μέθοδο getHeap την χρησιμοποιούμε για να πάρουμε ένα συγκεκριμένο αντικείμενο και την peek για να πάρουμε το πρώτο αντικείμενο του πίνακα.

ΜΕΡΟΣ Β

Για το Β μέρος θέλουμε να διαβάσουμε από ένα text αρχείο τους χρόνους που φτάνουν τα αρχεία προς εκτύπωση και το μέγεθος τους. Καθώς τα διαβάζουμε δημιουργούμε αντικείμενα τύπου PrintJob και τα αποθηκεύουμε σε μια λίστα μονής σύνδεσης ώστε να μην ξανά ασχοληθούμε με το text αρχείο. Καθώς δημιουργούμε τα αρχεία ορίζουμε και το id του καθενός. Μετά έχουμε η οποία ελέγχει κάθε πότε τελειώνει ένα αρχείο από τον εκτυπωτή μετά ελέγχει ποια αρχεία έχουν έρθει από την περίοδο που άρχισε το αρχείο να εκτυπώνεται μέχρι που τελείωσε η εκτύπωση του και τα εισάγει στην priority queue μας και τα αφαιρει από την λίστα. Καθώς ένα αρχείο τελειώνει αφαιρείται από την priority queue. Ο χρόνος στον οποίο τελείωσε αλλά και το στιγμιότυπο του αρχείου περνούν ως ορίσματα σε μια μέθοδο Print η οποία εκτυπώνει σε ένα text αρχείο τον χρόνο στον οποίο τελείωσε η εκτύπωση του αρχείου καθώς και ο id του αρχείου. Αφού τελειώσουν όλα τα αρχεία την εκτύπωση τους εκτυπώνεται ο μέσος χρόνος που περίμεναν τα αρχεία καθώς και το αρχείο που περίμενε περισσότερο μαζί με τον αντίστοιχο χρόνο αναμονής του στο τέλος του αρχείου text με τα αποτελέσματα. Για να υπολογίσουμε το μέσο χρόνο αναμονής αλλά και τον μέγιστο χρόνο αναμονής χρησιμοποιήσαμε δυο μεταβλητές. Για τον μέσο χρόνο απλά κάθε φορά που ένα αρχείο θα άρχιζε να εκτυπώνεται τότε παίρναμε τον χρόνο αναμονής του και τον προσθέταμε σε μια μεταβλητή που είχαμε το άθροισμα των χρόνων αναμονής των προηγούμενων αρχείων που ήδη έχουν εκτυπωθεί και στο τέλος την διαιρούμε με το σύνολο των αρχείων που εκτυπώθηκαν. Για τον μέγιστο χρόνο αναμονής απλά κρατήσαμε τον χρόνο αναμονής του πρώτου αρχείου και όταν επιλέχτηκε το επόμενο προς εκτύπωση συγκρίναμε τους δυο χρόνους και κρατήσαμε σε μια μεταβλητή τον μεγαλύτερο. Αυτή η σύγκριση έγινε και με τους χρόνους αναμονής των επόμενων αρχείων σε σύγκριση με το μεγαλύτερο χρόνο αναμονής που είχαμε αποθηκεύσει μέχρι τότε.

ΜΕΡΟΣ Γ

Στο Γ μέρος εργαζόμαστε σαν το μέρος Β με μοναδική διαφορά την αλλαγή της priority των αρχείων που θα πάρουν σειρά προς εκτύπωση. Η λογική εδώ είναι να μην εκτυπώνονται πρώτα τα μικρότερα αρχεία αλλά να λαμβάνεται υπόψη και ο χρόνος αναμονής ενός αρχείου. Κάθε 15 δευτερόλεπτα από την στιγμή που άρχισε να εκτυπώνεται το πρώτο αρχείο ξανά υπολογίζουμε το priority κάθε αρχείου που περιμένει για εκτύπωση. Για το λόγο αυτό χρησιμοποιούμε μια μεταβλητή T2 η οποία ενημερώνει το priority των αρχείων που βρίσκονται στην priority queue ανάλογα κιόλας και με την μεταβλητή που ελέγχει πότε ένα αρχείο τελειώνει την εκτύπωση.
Το priority γίνεται ίσο με την διαφορά του προηγούμενου priority και του χρόνου αναμονής του αρχείου που περιμένει να εκτυπωθεί. Αφού ενημερώσουμε την priority του αρχείου μετά καλούμε την μέθοδο swim της κλάσης MaxPQ για να ελέγξουμε άμα ένα αρχείο πρέπει να αναδυθεί και να εκτυπωθεί πιο νωρίς από άλλα που ήταν πιο ψηλά από αυτό προηγουμένως. Ο ελάχιστος αριθμός που μπορεί να φτάσει το priority ενός αρχείου είναι μηδέν και αυτό σημαίνει ότι το αρχείο θα βρίσκεται σε μεγαλύτερη προτεραιότητα από τα άλλα.