DataStructures

ΆΣΚΗΣΗ 2 ΜΕΡΟΣ Ε

ΤΡΟΠΟΣ ΕΚΤΕΛΕΣΗΣ ΜΕ CMD:

- Άνοιγμα του cmd μέσα στον φάκελο src
- Εντολή : javac Main.java
- Εντολή : java Main fullpath/folders.txt
- Το πρόγραμμα θα ζητήσει να δώσετε έναν αριθμό για Folders βάζετε 100 μετα θα ζητήσει αριθμό αρχείων βάζετε 10 και μετα θα ρωτήσει αν θέλετε να σταματήσετε και βάζετε οποιοδήποτε αριθμό εκτος του μηδέν.Και 0 να πατήσετε δεν θα σας αφησει να σταματήσετε αν δεν φτιάξετε τουλάχιστον το παράδειγμα που ζητάει η άσκηση.Οπότε συνεχίζεται για 500 και για 1000 με 10 αρχεια και έτσι φτιάχνεται το παράδειγμα της άσκησης.Προφανώς μπορείτε να δοκιμάσετε όποιο άλλο θέλετε.

ΣΗΜΕΙΩΣΗ:Όταν έχετε τρέξει μια φορά το πρόγραμμα προσέξτε να έχετε κλείσει αν έχετε ανοίξει κάποια απο τα txt του πειράματος η το αρχείο csv με τα αποτελέσματα γιατί το πρόγραμμα διαγράφει τα δεδομένα απο την προηγούμενη φορα που έτρεξε οπότε αν τα έχετε ανοιχτά θα σας ενημερώσει ότι δεν έχουν διαγραφεί

1. **Node.java:** Χρησιμοποιούμε αυτή την κλάση για να δημιουργήσουμε αντικείμενα τύπου Node τα οποία τα χρησιμοποιούμε για να υλοποιήσουμε τις δομές δεδομένων ουρά και στοίβα
   - Τα attributes της κλάσης είναι τα data τύπου T ώστε να αποθηκεύει όλους τους τύπους δεδομένων,next τύπου Node<T> που δείχνει στον επόμενο κόμβο
   - Μέθοδοι set και get για να έχουμε πρόσβαση στα attributes της κλάσης

2. **TxTRead.java:**
   - Η κλάση αυτή έχει ως attributes την μταβλητή path ώστε να βρει το αρχείο που πρέπει να διαβάσει.
   - Μέσα στην μέθοδο read διαβάζουμε το αρχείο, φτιάχνουμε μια καινούργια λίστα μέσα στην οποία αποθηκεύουμε τα δεδομένα του αρχείου και λαμβάνουμε υπόψην όλες τις περιπτώσεις για τις οποίες το αρχείο αυτο μπορεί να μην είναι σωστό ώστε να τρέξει το προγραμμά μας.Με το πρώτο try ελένχουμε αν το αρχείο όντως υπάρχει,με το δεύτερο try ελένχουμε αν κάθε γραμμή έχει ένα στοιχείο και αν είναι αριθμός και με το τρίτο try ελένχουμε αν ο αριθμός είναι από 0 έως 1000000.Για όλα τα try αν δεν ολοκληρωθεί ο κώδικας που περιέχουν εμφανίζουμε στην οθόνη το λάθος και τερματίζουμε το πρόγραμμα.Τέλος επιστρέφουμε την λίστα με τα δεδομένα

3. **Disk.java:** Χρησιμοποιούμε αυτή την κλάση για να αναπαραστήσουμε ένα object τύπου Disk
    - Attributes τις κλάσεις είναι το diskId:μονοναδικό key για να ξέρουμε ποιος δίσκος είναι,folders:μια λίστα με τους φακέλους που περιέχει,freeSpace: ελεύθερος χώρος του δίσκου
    - Περιέχει default και μη κατασκευαστή,set και get μεθόδους
    - Υλοποιούμε την override μέθοδο για να συγκρίνουμε τους δίσκους ανάλογα το freespace τους.Γυρνάει 1 αν το αντικείμενο που παίρνει ως όρισμα είναι μικρότερο,0 αν είναι ίσα και -1 αν είναι μεγαλύτερο.
4. **Folder.java:** Χρησιμοποιούμε αυτή την κλάση για να αναπαραστήσουμε ένα object τύπου Folder
    - Attribute της κλάσης είναι το μέγεθος του φακέλου
    - Περιέχει default και μη κατασκευαστή,set και get μεθόδους
    - Υλοποιούμε την override μέθοδο για να συγκρίνουμε τους φακέλους ανάλογα το size τους.Γυρνάει 1 αν το αντικείμενο που παίρνει ως όρισμα είναι μικρότερο,0 αν είναι ίσα και -1 αν είναι μεγαλύτερο.

5. **Main.java:** Διαβάζει από τα arguments της main το path και διαβάζει το txt αρχείο.Φτιάχνει ένα αντικείμενο τύπου Greedy και στέλνει true(αν θέλουμε να κάνει print τους δίσκους και τα δεδομένα τους).Περνάμε τα δεδομένα που διαβάσαμε σε μια λίστα και τρέχουμε τον αλγόριθμο.Αλλάζουμε το path και τρέχουμε το πείραμα.
6. **List.java:**Η λίστα είναι μια πιο απλοποιημένη έκδοση λίστας.Περιλαμβάνει μόνο της μεθόδους εισαγωγή και αφαίρεση στοιχείου απο μπροστά.Έχει size για να γνωρίζουμε τον αριθμό των στοιχείων που περιέχει και την sort που περιγράφεται στο μέρος Γ.

ΜΕΡΟΣ Α

7. **MaxPQ.java:** Η ουρά προτεραιότητας κάνει χρήση σωρού υλοποιήση με πίνακα(Η υλοποιήση έγινε με βοήθεια των διαφανειών πριν γίνει το εργαστήριο, με generics)
    - Attributes της ουράς είναι το μέγεθος της και ο σωρός
    - Μέδοδοι:
      Default κατασκευαστής ορίζει το μέγεθος ως 0 και φτιάχνει έναν κενό πίνακα
      isEmpy: Ελένχει αν ο σωρός είναι άδεια
      Insert: Στην εισαγωγή τσεκάρουμε το μέγεθος αν η ουρά είναι γεμάτη διπλασιάζουμε το μέγεθος της,η εισαγωγή γίνεται στο τέλος και κάνουμε swim για το στοιχείο που βάλαμε
      getMax:Παίρνουμε τα δεδομένα του στοιχείο με την μεγαλύτερη προτεραιοτητα.Το μεταφέρουμε στο τέλος , και αφού μειώσουμε το μέγεθος της ουράς κάνουμε sink το πρώτο στοιχειο.Οπότε το στοιχείο που είχε την μεγαλύτερη προτεραιότητα δεν το λαμβάνουμε υπόψην
      getMaxWithoutRemove:Απλά επιστρέφουμε τα δεδομένα του στοιχείου με την μεγαλύτερη προτεραιότητα αν είναι άδεια η ουρα πετάμε exception

swap:Αλλάζει 2 στοιχεία της ουράς
swim:Παίρνει ως όρισμα την θέση ενός στοιχείου και για όσο η θέση αυτή είναι μεγαλύτερη της μοναδας δηλαδή το στοιχείο δεν έχει φτάσει στην κορυφή ή ο πατέρας του είναι μικρότερος αλλάζει την θέση του με τον πατέρα και ενημερώνει την θέση.
sink: Παίρνει ως όρισμα την θέση ενός στοιχείου και και όσο το η θέση του παιδίου του είναι μικρότερου του μεγέθους της ουράς τσεκάρει ποιο παιδί είναι μεγαλύτερο και αν το μεγαλύτερο είναι και μεγαλύτερο από το στοιχείο που ελένχουμε κάνουμε swap και ενημερώνουμε την θέση αλλιώς κάνουμε break
size:γυρνάει το μέγεθος της ουράς
resize: Διπλασιάζει το μέγεθος του πίνακα
less: Ελένχει αν ο κόμβος είναι μεγαλύτερος η μικρότερος από ααυτόν που έρχεται ως όρισμα

ΜΕΡΟΣ Β

8. **Greedy.java:** Η κλάση αυτή έχεις ως σκοπό να υπολογίσει πόσους δίσκους χειαζόμαστε ανάλογα το μέγεθος τον φακέλων που δίνεται.
    - Έχει ως attributes το flag που δείχνει αν θα εμφανίσουμε το αποτέλεσμα η όχι και το numberOfDisks που μετράει πόσοι δίσκοι χρησιμοποιήθηκαν.
    - Η μέθοδος greedyAlg παίρνει μια λίστα με τα μεγέθοι των φακέλων και φτιάχνει μια άδεια ουρα προτεραιότητας και 3 ακέραιες μεταβλητές που μετραμε τα συνολικά TB που έποιαναν οι φάκελοι, τον αριθμό τον φακέλων που θα επεξεργαστούμε και το id για κάθε νέο δίσκο που φτιάχνουμε.
      ‘Ολος ο αλγόριθμος τρέχει μέσα στο while(head != null) δηλαδή όσο έχουμε κάποιον φάκελο να επεξεργαστούμε και μέσα τσεκάρουμε 3 περιπτώσεις.**Αν η ουρά είναι αδεία** απλά δημιουργούμε έναν καινούργιο δίσκο , τον βάζουμε στην ουρά ,βάζουμε τον φάκελο μέσα σε αυτόν τον δίσκο και ανανεωώνουμε το freespace του.
      Η δεύτερη περίπτωση είναι ο δίσκος να μην έιναι άδειος , κάνουμε getMaxWithoutRemove() απο την ουρά και ελένχουμε αν ο στον δίσκο με την μεγαλύτερη χωρητικότητα χωράει ο φάκελος που τσεκάρουμε.**Αν δεν χωράει** τότε είναι σίγουρο ότι δεν χωράει σε κανέναν άλλον αφού πήραμε τον δίσκο με την περισσότερη χωρητικότητα .αρα δημιουργούμε καινούργιο δίσκο ,βαζουμε τον φάκελο στην λίστα με τους φακέλους που έχει αναναιώνουμε το id και το freespace και τον κάνουμε insert στην ουρά προτεραιότητας.**Αν χωράει** τότε βάζουμε βάζουμε στην λίστα του τον φάκελο που ελένχουμε,ανανεώνουμε το freespace και κάνουμε sink ώστε να πάρει την σωστή θέση μέσα στην ουρά προτεραιότητας.

ΜΕΡΟΣ Γ

9. Η μέθοδος ταξινόμησης που υλοποιήθηκε είναι της Insertion sort με με την βοήθεια του εργαστηρίου.Εφόσον τα δεδομένα τα περνάμε σε λίστα η sort υλοποιήθηκε μέσα στην κλάση List ως μέθοδος και χρησιμοποιείται πάνω στις λίστες.

Η βασική ιδέα της insertion sort είναι οτι παίρνουμε κάθε φορά τον header από την λίστα που θέλουμε να κάνουμε sort και τον αποθηκεύουμε σε ένα temp Node και τσεκάρουμε τι γίνεται στην καινούργια λίστα που θέλουμε να τον τοποθετήσουμε.
Αν η λίστα η καινούργια είναι άδεια δεν χρειάζεται να τσεκάρουμε τίποτα απλά ανανεώνουμε τον header και tail.
Αν η καινούργια λίστα δεν είναι άδεια χρειάζεται να την διατρέξουμε για να μπορέσουμε να βρούμε που θα τοποθετηθεί ο κόμβος που έχουμε πάρει απο την λίστα που ταξινομούμε.Οπότε ενώ για κάθε κόμβο έχουμε τον next αυτό δεν φτάνει για να καταλάβουμε που πρέπει να μπει ο Node που επεξεργαζόμαστε οπότε χρειάζεται να κρατάμε για κάθε κόμβο να κρατάμε και τον προηγούμενο του.Έτσι είμαστε σε θέση να προσδιορίσουμε την ακριβή θέση του κόμβου και να τον τοποθετήσουμε στην σωστή θέση της καινούργιας λίστας.

ΜΕΡΟΣ Δ

10. CreateTxtFilesOfExperiment.java: Χρησιμοποιούμε αυτή την κλάση για να φτιάξει ο χρήστης τα αρχεία του πειράματος και επίσης χρησιμποποιούμε την βοήθεια των μεθόδων των κλάσεων **ReadCsvFile** kai **CreateCsvFile**

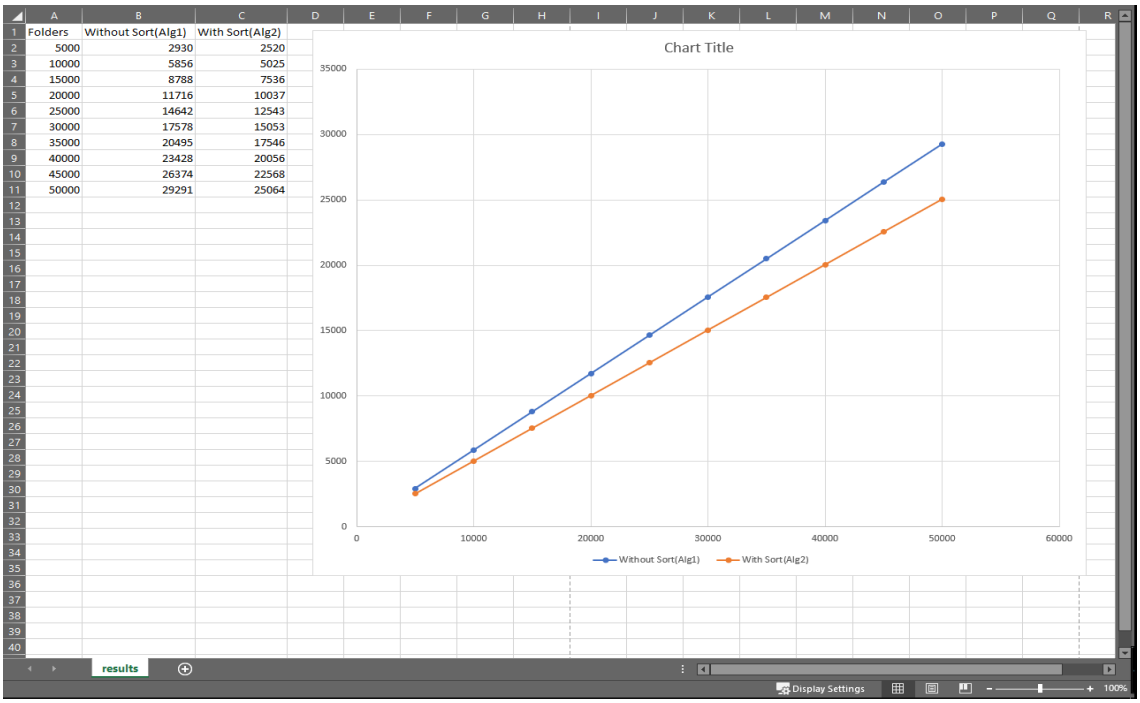
- Attribute της κλάσης είναι το numberOfFilesCreate και χρησιμοποιείται για να δώσει σε κάθε αρχείο έναν αριθμό.
- Η μέθοδος createTxtFiles έχει ως όρισμα το path που έχει δωσει ο χρήστης και το επεξεργαζόμαστε ώστε να τοποθετήσουμε τα δεδομένα σε έναν φάκελο Data
- Οι μεταβλητές τις μεθόδου:
  Το **n** είναι το πλήθος των φακέλων που δίνει ο χρήστης
  Το **numberOfFiles** είναι το πλήθος των αρχείων που θέλει ο χρήστης να φτιάξει με συγκεκριμένο n
  Το **i** ξεκινάει απο 0 , αυξάνεται μέχρι να φτάσει τον αριθμό τον αρχείων που έδωσε ο χρήστης και μηδενίζει για το επόμενο input του χρήστη
  Το **j** ξεκινάει από το 0 , φτάνει μέχρι το πλήθος των φακέλων που έδωσε ο χρήστης και μηδενίζει για το επόμενο input του χρήστη
  Το **exit** είναι μεταβλητή που κρατάει αν ο χρήστης θέλει να συνεχίσει να δημιουργεί αρχεία,με 0 σταματάει και με οποιαδήποτε άλλη τιμή συνεχίζει
  Το **ret** είναι τύπου StringBuilder και χρησιμποιηείται για να φτιάξουμε το path που θα δημιουργηθούν τα αρχεία
  Το **directory** περιέχει το path του φακέλου data
  Το **extension** έχει το .txt
  Το **extension1** έχει το .csv
  Το **randomNumber** είναι η τυχαία μεταβλητή που θα χρησιμποιήσουμε
  Το **numberOfDiskAlg1** είναι ναι ουσιαστικά το σύνολο τον αρχείων που χρησιμοποιήσε ο αλγόριθμος greedy χωρις sort
  Το **numberOfDiskAlg2** είναι ναι ουσιαστικά το σύνολο τον αρχείων που χρησιμοποιήσε ο αλγόριθμος greedy με sort

Το **currentNumbersOfFilesCreate** είναι ο αριθμός των αρχείων που χρησιμοποιήθηκαν για συγκεκριμένο n
Το **flag** χρησιμοποιείται για τον τίτλο ώστε να μην τρέξουν σημεία του κώδικα που χρησιμοποιούνται για τα πειραματικά δεδομένα
Το **secondFlag** χρησιμοποιείται απο την μέδοδο ReadCsvFile επειδή το πρώτο πλήθος φακέλων γράφεται μαζί με τον header την πρώτη φορά που θα τσεκάρουμε τα αρχεία του csv να υπολογίσουμε και αυτό
Το **greedy** είναι αντικείμενο της κλάσης greedy και το θέουμε για να τρέξουμε τον αλγόριθμο
Το **currentFileList** είναι η λίστα που κρατάει τα μεγέθη των φακέλων για συγκεκριμενο n
Το **csvBuilder** είναι αντικείμενο της κλάσης CreateCsvFile και το θελουμε για να χρησιμποιήσουμε της μεθόδους της κλάσης
Το **csvReader** είναι αντικείμενο της κλάσης ReadCsvFile CreateCsvFile και το θελουμε για να χρησιμποιήσουμε της μεθόδους της κλάσης
Το **countN** μετράει πόσα διαφορετικά n έχει δώσει ο χρήστης

- Στο πρώτο try της μεθόδου διαγράφουμε το csv αρχείο με τα αποτελέσματα αν υπάρχει,προσθέτουμε στο path το Data και φτιάχνουμε τον φάκελο Data αν δεν υπάρχει.Αν ο φάκελος Data υπήρχε και δεν είναι άδειος διαγράφουμε τα txt περιεχομένα του.
  Στο while(exit != 0) ξεκινάει η δημιουργία των δεδομένων και τρέχει μέχρι ο χρήστης να πατήσει το 0.
  Στο επόμενο while ζητάμε απο τον χρήστη να δώσει το πλήθος των φακέλων που θελει και ελένχουμε αν είναι αρχικά αριθμός ,μέσω της μεθόδου τσεκάρουμε αν το n που έδωσε ο χρήστης υπάρχει είδη και ενημερώνουμε τα attributes του αντικειμένου csvReader.Αν το countN δεν είναι μεγαλύτερο του 1 δεν χρησιμοποιούμε την μέθοδο.Αν το n δεν υπάρχει αυξάνουμε το CountN
  Στο επόμενο while(csvReader.getExist()) παίρνουμε από τον χρήστη των αριθμό των αρχείων που θα δημιουργηθούν και τσεκάρουμε αν είναι μεγαλύτερος του 10 και αν είναι αριθμός
  Το while(i < numberOfFiles && csvReader.getExist()) τρέχει για όλα τα αρχεία και τα δημιουργεί.Αρχικά φτιάχνει το path κάθε αρχείου και μετά μέσα στο try τρέχουμε άλλο ένα while που τρέχει όσες φορές είναι το πλήθος των φακέλων και παίρνει απο την τυχαία μεταβλητή έναν αριθμό από το 0-1000000 και τον γράφει μέσα στο txt αρχείο.Για κάθε αρχείο έχουμε μια λίστα που κρατάει τα δεδομένα του ώστε να τρέξουμε τους 2 αλγοριθμούς και ανανεώνουμε τις τιμές των numberOfDiskAlg1, numberOfDiskAlg2.Τελος αδειάζουμε την λίστα για το επόμενο αρχείο του ίδου πλήθους.
  Το επόμενο while απλά ρωτάει τον χρήστη αν θέλει να συνεχίσει η όχι ελένχει το input του χρήστη και απλά ελένχει αν έχουν δωθεί τουλάχιστον 3 διαφορετικά n και η διαφορά τους είναι μεγαλύτερη από 900 αν όχι ο χρήστης συνεχίζει αναγκαστικά.
  Τέλος για κάθε n χρησιμοποιούμε την μέθοδο createCsvFile η οποία βγάζει τον μέσο όρο και γράφει τα αποτελέσματα

11. **ReadCsvFile.java:** Χρησιμοποιούμε αυτή την κλάση για να ελένχουμε αν το πλήθος των φακέλων για τους οποίους ο χρήστης θα δημιουργήσει πειραματικά αρχεία έχει είδη δημιουργηθεί και αν η διαφορά του πλήθους των φακέλων που έχει δημιουργήσει ο χρήστης είναι τουλάχιστον 900
    - Η κλάση αυτή έχει ως atributes τους ακεραίους max,min και τις Boolean μεταβλητές exist,maxDifference.Οι μεταβλητές max,min ανανεώνονται με κάθε καινούργιο **πλήθος φακέλων n** που δίνει ο χρήστης,η μεταβλητή **exist** γίνεται true αν το n δεν έχει ξαναδωθεί από τον χρήστη και false αν έχει ξαναδωθεί, η μεταβλητή maxDifference επιστρέφει true αν η διαφορά είναι μικρότερη από 900 και αν όχι επιστρέφει false
    - Η μέθοδος **readCsvFile** παίρνει ως ορίσματα το path του αρχείου ,το n που είναι το πλήθος των φακέλων που δίνει ο χρήστης και το flag που μέσα στο if που το περιέχει ελέχουμε το στοιχείο που γράφαμε μαζί με τους τίτλους μια φορά.Μετά γίνεται false.Τσεκάρει αν το αρχείο υπάρχει και ανανεώνει τα values των attributes.
12. **CreateCsvFile.java:** Η κλάση αυτη περιλαμβάνει την μέθοδο CreateCsvFile η οποία παίρνει ως ορίσματα το πλήθος,το path για το csv αρχείο,τους δίσκους που χρησιμοποιήσαν οι 2 αλγοριθμοι, τα αρχεία που δημιουργήθηκαν με το συγκεκριμένο n και το flag για τον τίτλο και γράφει στην πρώτη στήλη το πλήθος των φακέλων ,στην δεύτερη τον μέσο όρο του πρώτου αλγορίθμου και στην 3^{η} στήλη τον μέσο όρο των δίσκων του δεύτερου αλγορίθμου.

ΑΠΟΤΕΛΕΣΜΑΤΑ



ΣΥΜΠΕΡΑΣΜΑ

Μπορεί στο σχεδιάγραμμα να μην φαίνεται πόσο μεγάλη είναι η διαφορά του greedy αλγορίθμου με sort και χωρις sort αλλά αν κοιτάξουμε τα νούμερα φαίνεται πως ανά 5000 φακέλους ο greedy αλγόριθμος χωρίς sort χρησιμοποιεί 410 περίπου δίσκους παραπάνω από τον greedy με sort και στα

50000 έχουν διαφορά 4200 δίσκους περίπου.Αν ο φθηνότερος HDD δίσκος είναι περίπου 30 ευρώ τότε μιλάμε για μια διαφορά 126000 ευρώ.Και αν παμε σε δίσκους SSD τα λεφτά ανεβαίνουν ακόμα πιο πολύ.Άρα με sort έχουμε καλύτερη κατανομή του χώρου των δίσκων το οποίο μπορεί να γλιτώσει σε κάποιον αρκετά χρήματα.