ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΕΦΑΡΜΟΣΜΕΝΩΝ ΜΑΘΗΜΑΤΙΚΩΝ ΚΑΙ ΦΥΣΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΟΜΕΑΣ ΜΑΘΗΜΑΤΙΚΩΝ

Γραπτή άσκηση #1

Μάθημα: Δομές Δεδομένων

Διδάσκων: Αντώνιος Συμβώνης

Φοιτήτρια: Ελένη Στυλιανού, ge21708

Email: elenistylianou03@live.com

Άσκηση 1

Έστω σύνολο S με πλήθος n στοιχεία. Για την εύρεση ελάχιστου και το μέγιστου στοιχείου του S με 3⌊n/2⌋ συγκρίσεις, αρκεί να συγκρίνουμε το 1ο στοιχείο με το 2ο, το 3ο με το 4ο, κ.ο.κ. Σε κάθε σύγκριση θα τοποθετούμε το μικρότερο στοιχείο σε ένα σύνολο X και το μεγαλύτερο στοιχείο σε ένα σύνολο Y. Αν το σύνολο έχει περιττό πλήθος στοιχείων, συγκρίνουμε και το τελευταίο με οποιοδήποτε στοιχείο του συνόλου, π.χ. με το 1ο. Επομένως, θα έχουμε το ελάχιστο στοιχείο του συνόλου στο σύνολο X και το μέγιστο στο Y (⌈n/2⌉ συγκρίσεις μέχρι αυτό το σημείο). Στη συνέχεια, ορίζουμε ως min το 1ο στοιχείο του X, και το συγκρίνουμε με τα υπόλοιπα στοιχεία του X. Αν βρούμε κάποιο μικρότερο στοιχείο, τότε αυτό ορίζεται ως το νέο min και τα υπόλοιπα στοιχεία συγκρίνονται με αυτό. Ομοίως για το σύνολο Y για την εύρεση μεγίστου. Η εύρεση του μεγίστου(max) και ελαχίστου(min) από τα X και Y χρειάστηκε συνολικά n-2 συγκρίσεις. Οπότε συνολικά απαιτήθηκαν ⌈n/2⌉ + n − 2 ≤ 3⌊n/2⌋ συγκρίσεις.

Άσκηση 2

Έστω n θετικός ακέραιος. Χρησιμοποιώντας δυαδική αναζήτηση στο διάστημα [0,n] θα βρούμε τον $[\sqrt{n}]$. Αναλυτικότερα, θα στέλνουμε σε μια συνάρτηση ως παράμετρο το n και θα ορίζουμε τις μεταβλητές y0=0 και yn=n. Στη συνέχεια, με τη χρήση του βρόγχου while, όσο ισχύει y0≤yn, η μεταβλητή x θα παίρνει την τιμή (y0+yn)/2. Αν x^2=n, το n θα είναι τέλειο τετράγωνο και η συνάρτηση θα επιστρέφει το x, που θα είναι η ρίζα του n σε αυτή την περίπτωση. Αν $x^2 <$ n, η μεταβλητή yn θα παίρνει την τιμή x-1, δηλαδή η αναζήτηση θα συνεχίζει στο 1ο μισό του τρέχοντος τμήματος. Αν $x^2 >$ n, η μεταβλητή y0 θα παίρνει την τιμή x+1, δηλαδή η αναζήτηση θα συνεχίζεται στο 2ο μισό του τρέχοντος διαστήματος. Όταν τερματίσει ο βρόγχος while, δηλαδή όταν y0>yn, τότε το n δεν είναι τέλειο τετράγωνο, αφού σε καμία επανάληψη δεν ίσχυσε x^2=n. Επίσης θα ισχύει y0= yn+1. Πράγματι, αν y0> yn+1, τότε yn<x< y0, αφού y0=x+1 ή yn=x-1. Υποθέτοντας χβτγ ότι y0=x+1, τότε $x^2 >$ n. Όμως x=(y0'+yn)/2 (2), όπου y0' η προηγούμενη τιμή του y0. Από την σχέση yn<x< y0 έχουμε ότι yn<x και από την σχέση x=(y0'+yn)/2 έχουμε ότι y0'>yn, γεγονός που είναι άτοπο αφού ο βρόγχος while δεν σταμάτησε και υπολόγισε τη νέα τιμή του y0. Επομένως, y0= yn+1. Επιπρόσθετα, θα ισχύει yn^2<n και $y0^2$> n. Από τις σχέσεις y0= yn+1, yn^2<n και $y0^2$> n εξάγεται το συμπέρασμα πως y0=$[\sqrt{n}]$. Συνεπώς, η συνάρτηση θα επιστρέψει τη μεταβλητή y0.

Κώδικας:

```
import java.util.*;
public class RootCeil
{
  public static void main(String args[])
  {
    System.out.println('\u000C');
    Scanner sc = new Scanner(System.in);

    System.out.println("Give a positive integer:");
    int n = sc.nextInt();
```

```
    while(n>0)
    {
      System.out.println("The ceiling of the square root of " + n + " is: " + ceilSqrt(n) );
      System.out.println("Give another positive integer (or 0 to stop):");
      n = sc.nextInt();
    }
  }

  /**
   * Return the ceiling of the square root of the given integer.
   * The given integer must be positive.
   */
  private static int ceilSqrt(int n)
  {
    int x,y0=0,yn=n;
    do
    {
      x = (y0+yn)/2;
      if(x*x == n)
        return x;
      else if(x*x > n)
        yn = x-1;
      else
        y0 = x+1;
    }while(y0<=yn);
    return y0;
  }
}
```

Άσκηση 3

Έστω S1 και S2 τα δύο σύνολα μήκους n. Για να βρούμε αν υπάρχει ζεύγος αριθμών, ένα από το S1 και ένα από το S1 με άθροισμα x, πρώτα ταξινομήσουμε το ένα από τα 2 σύνολα με merge sort, έστω το S1. Στη συνέχεια, για κάθε στοιχείο y του S2, θα βρίσκουμε τη διαφορά d(y)=x-y. Έπειτα, με χρήση της δυαδικής αναζήτησης στο ταξινομημένο σύνολο S1 θα εξετάζουμε αν ανήκει σε αυτό ο αριθμός d. Στην περίπτωση που για κάποιο y βρεθεί ο d(y) στο S1 τότε υπάρχει ζεύγος με άθροισμα x. Για τη merge sort απαιτείται O(nlogn) χρόνος. Για κάθε στοιχείο y του S2, η δυαδική αναζήτηση του d=x-y στο S1 απαιτεί O(logn) χρόνο και αφού το S2 έχει n το πλήθος στοιχεία, η συνολική αναζήτηση απαιτεί O(nlogn) χρόνο. Επομένως, η συνολική διαδικασία απαιτεί O(nlogn) χρόνο.

Άσκηση 4

Για την υλοποίηση ενός Αφηρημένου Τύπου Δεδομένων που υποστηρίζει το σύνολο των ακεραίων στο διάστημα {1…n} και τις λειτουργίες Insert(x), Delete(x) και Member(x) σε

χρόνο O(1) στη χειρότερη περίπτωση, μπορούμε να χρησιμοποιήσουμε ένα διάνυσμα από ακεραίους μήκους n, έστω X. Η i-οστή θέση του διανύσματος X θα αντιστοιχεί στον αριθμό i, όπου i ακέραιος στο [1,n]. Η τιμή στην i-οστή θέση του διανύσματος X θα αντιπροσωπεύει το πλήθος των εμφανίσεων του αριθμού i στο σύνολο. Όταν το σύνολο είναι κενό, όλες οι θέσεις του διανύσματος θα έχουν την τιμή 0. Οι τρεις πιο πάνω λειτουργίες υλοποιούνται ως εξής:

Insert(x): Αυξάνουμε την τιμή της x-οστής θέσης του διανύσματος κατά 1 (O(1) χρόνος) για να εισαχθεί ο θετικός ακέραιος x στο σύνολο.

Delete(x): Μειώνουμε την τιμή της x-οστής θέσης του X κατά 1, δεδομένου φυσικά ότι η τιμή στη θέση εκείνη είναι θετική (O(1) χρόνος), για να διαγράψουμε τον αριθμό x από το σύνολο.

Member(x): Αν η τιμή στην x-οστή θέση του διανύσματος X είναι θετική, τότε ο αριθμός x ανήκει στο σύνολο και έτσι γίνετε η εξέταση αν ο αριθμός ανήκει στο σύνολο. Αν είναι θετική επιστρέφεται η τιμή true, διαφορετικά false (O(1) χρόνος).

Η διεπαφή (interface) Listt που αντιστοιχεί στον παραπάνω ΑΤΔ:

```
public interface Listt
{
   /**
    * Insert the element e in the list.
    */
   public void insert(int i);

   /**
    * Delete the element e from the list.
    */
   public void delete(int i);

   /**
    * Return true if the element is in the list, false otherwise.
    */
   public boolean member(int i);
}
```

Η κλάση ListOfInts που υλοποιεί τη διεπαφή αυτή:

```
public class ListOfInts implements Listt
{
   private int[] array;

   public ListOfInts(int n)
   {
      array = new int[n+1];
```

```
    for(int i=0;i<=n;i++)
      array[i] = 0;
  }
  public void insert(int i)
  {
    array[i]++;
  }

  public void delete(int i)
  {
    if(array[i] > 0)
      array[i]--;
  }
  public boolean member(int i)
  {
    return array[i]>0;
  }
}
```

Άσκηση 5

Έστω P μία νόμιμη έκφραση σε μορφή postfix. Αρχικά, χωρίζουμε (split) την P στα διάφορα στοιχεία της (αριθμούς και τελεστές). Με χρήση του ΑΤΔ στοίβα θα υπολογίσουμε την τιμή της P. Η στοίβα θα περιέχει μόνο αριθμούς. Ο αλγόριθμος θα λειτουργεί ως εξής:

Αρχικά θα διαβάζει το 1ο στοιχείο της έκφρασης P, το οποίο θα είναι αριθμός, και θα το εισάγει (push) στη στοίβα. Ομοίως και το 2ο. Ακολούθως, για κάθε επόμενο στοιχείο x της P, αν το x είναι αριθμός, τότε το x θα εισάγεται (push) στη στοίβα. Θα αφαιρείται από τη στοίβα ο αριθμός που βρίσκεται στην κορυφή της (top), έστω n, εάν το x είναι τελεστής. Επίσης θα αφαιρείται και ο αμέσως επόμενος αριθμός από την κορυφή της στοίβας, έστω m. Τότε θα εισάγεται στην κορυφή της στοίβας (push) ο αριθμός mxn, όπου x θα είναι ένας από τους τελεστές +,-,*,/. Αφού η έκφραση P είναι νόμιμη, μετά την προσπέλαση όλης της S, στη στοίβα θα έχει απομείνει ένας μόνο αριθμός, ο οποίος θα είναι και η τιμή που αντιστοιχεί στην postfix έκφραση P.

Για κάθε στοιχείο της P, οι ενέργειες του πιο πάνω αλγορίθμου απαιτούν σταθερό χρόνο (O(1)), οπότε για μέγεθος n της postfix εκδοχής απαιτείται O(n) χρόνος.

Κλάση PostFix:

```
import java.util.*;
public class PostFix
{
  /**
   * The user inputs a valid expression in postfix form and the result of
   * the expression in printed in the terminal.
```

```
 */
public static void main(String args[])
{
   System.out.println('\u000C');
   Scanner sc = new Scanner(System.in);
   String p;

   System.out.println("Give a regular postfix form:");
   p = sc.nextLine();
   System.out.println(p);

   System.out.print("The result of the above postfix form is: ");
   System.out.println(postfixEvaluate(p));
}

/**
 * Return the result of the given expression in postfix form.
 * The expression must be valid and not be enclosed in double quotes (" ").
 *
 * @param pexp The expression in postfix form as a String.
 */
private static double postfixEvaluate(String pexp)
{
   double n,m;
   String p;
   LinkedStack<Double> stack = new LinkedStack<Double>();
   String[] elements = pexp.split(" ");

   for(int i=0;i<elements.length;i++)
   {
      if(isOperator(elements[i]))
         stack.push(operation(stack.pop(),stack.pop(),elements[i]));
      else
         stack.push(Double.parseDouble(elements[i]));
   }

   return stack.pop();
}

/**
 * Return true if the given String is one of the basic math operators (+,-,*,/)
 */
private static boolean isOperator(String p)
{
```

```
    if(p.equals("+") || p.equals("-") || p.equals("*") || p.equals("/"))
      return true;
    return false;
  }

  /**
   * Return the result of the operation between the numbers n and m
   * and the operator p (msn)
   *
   * @param n a double number
   * @param m a double number
   * @param p the operator
   * @return msn
   */
  private static double operation(double n, double m, String p)
  {
    if(p.equals("+"))
      return m+n;
    else if(p.equals("-"))
      return m-n;
    else if(p.equals("*"))
      return m*n;
    return m/n;
  }
}
```

Στην κύρια συνάρτηση (main) της κλάσης, ο χρήστης εισάγει μια νόμιμη έκφραση σε μορφή postfix (χωρίς εισαγωγικά “”) και εκτυπώνεται στο τερματικό η τιμή της έκφρασης.

Άσκηση 6

Stack: Ο ΑΤΔ Stack υποστηρίζει τις μεθόδους push,pop,top,size και isEmpty. Παρακάτω περιγράφεται η υλοποίηση του ΑΤΔ Stack με χρήση των δομών δεδομένων DoublyLinkedList, και ArrayList:

Υλοποίηση Stack με χρήση DoublyLinkedList: Για την υλοποίηση ενός Stack με χρήση ενός DoublyLinkedList κατασκευάζουμε την κλάση LinkedStack, στην οποία χρησιμοποιούμε ένα πεδίο τύπου DoublyLinkedList, έστω list. Το πεδίο αυτό είναι στην ουσία το Stack μας, το οποίο αποθηκεύει αντικείμενα τύπου E. Οι μέθοδοι του Stack υλοποιούνται όπως παρακάτω:

void push(E e): Για την εισαγωγή του αντικειμένου e (τύπου E) στο Stack, αρκεί να τοποθετήσουμε το αντικείμενο αυτό στην αρχή της διπλά συνδεδεμένης λίστας list με τη μέθοδο addfirst().

E pop(): Για την αφαίρεση του στοιχείου που βρίσκεται στην κορυφή της στοίβας, θα χρησιμοποιήσουμε τη μέθοδο removeFirst() της διπλά συνδεδεμένης λίστας. Με αυτόν τον τρόπο αφαιρούμε το στοιχείο από τη λίστα και αυτό επιστρέφεται μέσω της μεθόδου.

E top(): Για να εξετάσουμε ποιο στοιχείο βρίσκεται στην κορυφή της στοίβας, αρκεί να βρούμε ποιο είναι το 1ο στοιχείο της λίστας και γι' αυτό χρησιμοποιούμε τη μέθοδο first() της κλάσης DoublyLinkedList.

int size(): Το μέγεθος του Stack θα είναι το μέγεθος της διπλά συνδεδεμένης λίστας, άρα επιστρέφουμε το μέγεθος της λίστας με τη μέθοδο size() της κλάσης DoublyLinkedList.

boolean isEmpty(): Η στοίβα είναι κενή αν η λίστα είναι κενή, άρα επιστρέφουμε την αληθοτιμή της μεθόδου isEmpty() για τη λίστα μας.

Υλοποίηση Stack με χρήση ArrayList: Για την υλοποίηση ενός Stack με χρήση ενός ArrayList κατασκευάζουμε την κλάση ArrayStack, στην οποία χρησιμοποιούμε ένα πεδίο τύπου ArrayList, έστω data. Η μεταβλητή αυτή είναι στην ουσία το Stack που αποθηκεύει αντικείμενα τύπου E. Οι μέθοδοι του Stack υλοποιούνται όπως παρακάτω:

void push(E e): Για την εισαγωγή του αντικειμένου e (τύπου E) στο Stack, αρκεί να τοποθετήσουμε το αντικείμενο αυτό στο τέλος του ArrayList, δηλαδή στη θέση data.size() και για να το πετύχουμε αυτό χρησιμοποιούμε την εντολή add(data.size(),e).

E pop(): Για την αφαίρεση του στοιχείου που βρίσκεται στην κορυφή της στοίβας, θα κάνουμε χρήση της μέθοδο remove() της κλάσης ArrayList. Αφαιρούμε το στοιχείο της τελευταίας θέσης, δηλαδή της θέσης data.size()-1, οπότε χρησιμοποιούμε την εντολή data.remove(data.size()- 1).

E top(): Για να ελέγξουμε ποιο στοιχείο βρίσκεται στην κορυφή της στοίβας, αρκεί να βρούμε ποιο στοιχείο βρίσκεται στην τελευταία μη κενή θέση του ArrayList και για να το επιτύχουμε αυτό κάνουμε χρήση της μεθόδου data.get(data.size()-1) της κλάσης ArrayList.

int size(): Το μέγεθος του Stack θα είναι το μέγεθος της δεικτοδοτούμενης λίστας μας, άρα επιστρέφουμε το μέγεθος της λίστας data με τη μέθοδο size() της κλάσης ArrayList.

boolean isEmpty(): Η στοίβα είναι κενή αν η δεικτοδοτούμενη λίστα είναι κενή, άρα επιστρέφουμε την αληθοτιμή της μεθόδου isEmpty() για τη λίστα μας.

Queue: Ο ΑΤΔ Stack υποστηρίζει τις μεθόδους enqueue,dequeue,first,size και isEmpty. Πιο κάτω δίνετε περιγραφή για την υλοποίηση του ΑΤΔ Queue με χρήση των δομών δεδομένων DoublyLinkedList, και ArrayList:

Υλοποίηση Queue με χρήση DoublyLinkedList: Για την υλοποίηση ενός Queue με χρήση ενός DoublyLinkedList κατασκευάζουμε την κλάση LinkedQueue. Στην κλάση αυτή, χρησιμοποιούμε ένα πεδίο τύπου DoublyLinkedList, έστω list. Το πεδίο αυτό είναι στην ουσία το Queue μας, το οποίο αποθηκεύει αντικείμενα τύπου E. Οι μέθοδοι του Queue υλοποιούνται ως εξής:

void enqueue(E e): Για την εισαγωγή του στοιχείου e (τύπου E) στο τέλος της Ουράς, αρκεί να τοποθετήσουμε το στοιχείο στο τέλος της διπλά συνδεδεμένης λίστας με τη μέθοδο addLast() της κλάσης DoublyLinkedList. Θα κάνουμε χρήση της εντολής list.addLast(e).

E dequeue(): Για να αφαιρέσουμε το 1ο στοιχείο της ουράς, αρκεί να αφαιρέσουμε το 1ο στοιχείο της διπλά συνδεδεμένης λίστας list χρησιμοποιώντας την μέθοδο removeFirst(), επιστρέφοντας έτσι το 1ο στοιχείο της Ουράς.

E first: Το 1ο στοιχείο της ουράς θα είναι το 1ο στοιχείο της λίστας μας, οπότε αρκεί να επιστρέψουμε το 1ο στοιχείο της λίστας με την εντολή return list.first().

int size() και boolean isEmpty(): Οι μέθοδοι size() και isEmpty() υλοποιούνται ακριβώς όπως υλοποιούνται και για τον ΑΤΔ Stack μέσω της DoublyLinkedList.

Υλοποίηση Queue με χρήση ArrayList: Για την υλοποίηση ενός Queue με χρήση ενός ArrayList κατασκευάζουμε την κλάση ArrayQueue, στην οποία γίνεται χρήση ενός πεδίου τύπου ArrayList, έστω data. Το πεδίο αυτό είναι στην ουσία το Queue μας. Οι μέθοδοι του Queue υλοποιούνται ως εξής:

void enqueue (E e): Για την εισαγωγή του στοιχείου e (τύπου E) στο τέλος της Ουράς, αρκεί να τοποθετήσουμε το στοιχείο στο τέλος της δεικτοδοτούμενης λίστας με την εντολή data.add(data.size(),e). Δεν χρειάζεται έλεγχος αν η ουρά έχει γεμίσει, καθώς η κλάση ArrayList διπλασιάζει το μέγεθος των αντικειμένων της όταν αυτά γεμίσουν.

E dequeue (): Για την αφαίρεση του 1ου στοιχείου της ουράς, αρκεί να αφαιρέσουμε το 1ο στοιχείο της δεικτοδοτούμενης λίστας data κάνοντας χρήση της μεθόδου remove() και επιστρέφοντας έτσι το 1ο στοιχείο της Ουράς. Για την επίτευξη αυτού χρησιμοποιούμε την εντολή data.remove(0).

E first: Το 1ο στοιχείο της ουράς θα είναι το 1ο στοιχείο της λίστας μας, οπότε αρκεί να επιστρέψουμε το 1ο στοιχείο της δεικτοδοτούμενης λίστας με την εντολή return data.get(0).

int size() και boolean isEmpty(): Οι μέθοδοι size() και isEmpty() υλοποιούνται ακριβώς όπως υλοποιούνται και για τον ΑΤΔ Stack μέσω της ArrayList.

Στο BlueJ project που συνοδεύει το κείμενο έχουν ενσωματωθεί υλοποιήσεις των παραπάνω διαπροσωπειών και κλάσεων. Για την υλοποίηση των παραπάνω έχει χρησιμοποιηθεί κώδικας από την ιστοσελίδα του μαθήματος, ο οποίος έχει τροποποιηθεί κατάλληλα.