CMP-5014Y Coursework Assignment 2

100108964 (kzy14tcu)

Wed, 20 Mar 2019 14:25

PDF prepared using PASS version 1.15 running on `Windows 10 10.0` (`amd64`).

☑ I agree that by submitting a PDF generated by PASS I am confirming that I have checked the PDF and that it correctly represents my submission.

Contents

5014YCW02-100108964-file.docx

Filename scrubbed (one or more forbidden characters found). Original name:

`DSA_ver2.docx`

SVD.java

```java
package question1;

import java.util.Arrays;
import java.util.HashMap;
import java.util.Random;

/**
 *
 * @author Nahim
 */
public class SVD {

    //Algorithm to find SVD whose run time complexity id O(n*n)
    public static void SVD_Onn(int arr[], int n){
        int maxCount = 0;
        int index = -1;
        long t1 = System.nanoTime();
        for(int i = 0; i < n; i++){
            int count = 0;
            for(int j = 0; j < n; j++)
            {
                if(arr[i] == arr[j])
                    count++;
            }


            if(count > maxCount)
            {
                maxCount = count;
                index = i;
        }
    }
        if(maxCount > n/2){
            System.out.println(arr[index]);

        }
        else
            System.out.print("no value found");



    }

    // Algorithm to find SVD whose worst runtime complexity is O(nlog(n))
    public static void SVD_Onlogn(int[] arr){

    Arrays.sort(arr);
    int count = 1;
    int x = arr[0];


    for (int i = 1; i <arr.length ; i++) {
        if(x==arr[i]){
            count++;
            if(count>arr.length/2) {
                System.out.println(x);
            }
        }else{
            x = arr[i];
            count = 1;
        }
    }


    }

    // Algorithm for finding SVD whose space complexity and worst case runtime is
        O(n) using a hashmap
    public static void SVD_On(int[] arr){

         long t1 = System.nanoTime();
         long t2 = 0;

        HashMap<Integer,Integer> map = new HashMap<Integer, Integer>();

        for(int i = 0; i < arr.length; i++){
            if(map.containsKey(arr[i])){
                int count = map.get(arr[i]) + 1;
                if(count > arr.length / 2){
                    System.out.println(arr[i]);


                }else
                    map.put(arr[i], count);
                t2 = System.nanoTime();

            }
            else
                map.put(arr[i], 1);
        }
            long end = t2 - t1;
        System.out.println(end);


    }

    public static void main(String[] args){

        Random rand = new Random();
        int randomNum = rand.nextInt();
        int n = 50000;
        int[] a = new int [n];
        int[] arr = {7,7,9,3,2,7,7};

        for(int i = 0; i < a.length; i++){
            a[i] = randomNum;
        }

        //testing

        //O(n*n)
        //SVD_Onn(a,n);

        //O(nlog(n))
        //SVD_Onlogn(a);

        //O(n)
        SVD_On(a);
    }

}
```

ArrayHashTable.java

```java

package question2;

import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Random;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author Nahim
 */
public class ArrayHashTable extends HashTable {


    Object[][] table; //Create Object variable
    int chainSize = 5; //Initial size of chain
    int[] counts; //Array to store counts



    public ArrayHashTable(){
        table = new Object[capacity][]; //initialise 2D Object
        counts = new int[capacity];


        setTable(); //set table to null;


        setCount(); //set count to 0


    }

    public void setTable(){
        //initialise all values in table to null
        for(int i = 0; i<capacity; i++){
            table[i] = null;
        }
    }

    public void setCount(){
        //initialise all counts to 0
        for(int i = 0; i < capacity; i++){
            counts[i] = 0;
        }
    }




    @Override
    boolean add(Object obj) {

        // generate hash code
        int hash = obj.hashCode() % this.capacity;

        //check if chain arrays exists and make new one if it doesn't
        if(table[hash] == null){
            Object[] chain = new Object[chainSize];
            table[hash] = chain;
        }

        //check if object is already in hash table
        if(!contains(obj)){
            //Double chain capcity if max number
            if(table[hash][table[hash].length - 1] != null){

                // copy chain into temporary array
                Object[] tempArr = new Object[this.table[hash].length*2];
                System.arraycopy(this.table[hash], 0, tempArr, 0, this.table[hash
                    ].length);

                // Copy chains back doubled in size
                this.table[hash] = tempArr;
        }
            boolean addObj = false;
            int i = 0;
            while(!addObj){
                //checks if space is free.
                if(table[hash][i] == null){
                    table[hash][i] = obj;
                    counts[hash]++;
                    size++;
                    addObj = true;
                    System.out.println("Added " + table[hash][i] + " hash "
                    + hash + " to hash table. ");
                }
                i++;
            }

            return true;
    }
        return false;

    }

    @Override
    boolean contains(Object obj) {
        int hash = obj.hashCode() % this.capacity;

        // search through chain for Object
        for(int i = 0; i < counts[hash]; i++){
            if(table[hash][i] == obj){
                System.out.println(table[hash][i] + " found at position " +
                        hash + "," + i + ".");
                return true;
            }
        }

        return false;
    }

    @Override
    boolean remove(Object obj) {
        int hash = obj.hashCode() % this.capacity;

        for(int i = 0; i < counts[hash]; i++){
            if(table[hash][i] == obj){
                System.out.println("Removed " + table[hash][i] + " " + "from hash
                     table");
                table[hash][i] = null;
```

```java

                //Move all items back one
                for(int j = i + 1; j < counts[hash]; j++){
                    table[hash][j-1] = table[hash][j];
                }

                counts[hash]--;
                size--;

                return true;
            }
        }
        return false;
    }

    /**
     * Method to test run time experiment
     */
    public static void TimingExperiment(){


        Random r = new Random();
        ArrayHashTable hTable = new ArrayHashTable();
        int a = 100;
        int n = 1000; // matrix size

        while(n <= 50000) {

            int[] numbers = new int[n];

            for(int j = 0; j < n; j++){
                numbers[j] = Math.abs(r.nextInt());
            }

            // mean and std deviation

            double sum = 0;
            double sumSquared = 0;


                long t1 = System.nanoTime();

                for(int j = 0; j < n; j++){
                    hTable.add(numbers[j]);
                }

                for(int j = 0; j < n; j++){
                    hTable.remove(numbers[j]);
                }

                long t2 = System.nanoTime() - t1;

                // Recording it in milli seconds to make it more interpretable
                sum += (double)t2 / 1000000.0;
                sumSquared += (t2/1000000.0) * (t2/1000000.0);

                double mean = sum / a;
                double variance = sumSquared / a - (mean * mean);
                double stdDev = Math.sqrt(variance);

                if(n < 20000){
                    n += 1000;
                }
                else if(n < 50001){
                    n += 5000;
                }
        }

    }

  /**
    * Main method for testing
    * @param args
    */
    public static void main(String[] args) {

        //testing
        int t = 6;
        ArrayHashTable table = new ArrayHashTable();

        System.out.println("Testing");
        //add values to hash table
        for(int i = 0; i < t; i++ ){
            table.add(i);
        }

        //check values were added correctly
        for(int i = 0; i < t; i++){
            table.contains(i);
        }

        //remove all values
        for(int i = 0; i < t; i++){
            table.remove(i);
        }


       TimingExperiment();

    }


}
```

HashTable.java

```java
/*
Hash Table interface for the DS&A labs, week 1, semester 2. Note the emphasis
    here
is to get you to implement the algorithms, not write fancy code (although feel
    free to
do so! There are exercises for Programming 2 to engineer it more.
 */
package question2;

/**
 *
 * @author ajb
 */
public abstract class HashTable {
    int capacity=100;
    int size=0;

    public int size(){ return size;}

/**
 * Adds the specified element to this hash table if it is not already present
 *
 * @param obj
 * @return true if the element is successfully added
 */
   abstract boolean add(Object obj);

/**
 *
 * @param obj
 * @return  true if this hash table contains the specified element
 */
   abstract boolean contains(Object obj);
/**
 *
 * @param obj
 * @return Removes the specified element from this set if it is present
 */
  abstract  boolean remove(Object obj);

}
```