CIS 2168 Data Structures **Fall 2014**

Lecture 2: Sep. 2,4

Lecturer: Anwar Mamat

2.1 Array Based Collections

Listing 1: Bag Class

```java
import java.util.Iterator;
/**
 *  The Bag class represents a collection of generic items.
 *  It supports insertion and iterating over the items in arbitrary order.
 *  @author Anwar Mamat
 */
public class Bag<E extends Comparable<E>> implements Iterable<E>
{
        protected E[] items;  //array of items
        protected int arraySize = 0;  //number of items in the bag
        protected int capacity = 10;  //capacity of the bag

        /**
        *   Initializes an empty bag.
        */
        Bag()
        {
                items = (E[]) new Comparable[capacity];
        }
        /**
        *   Returns an iterator that iterates through the items in the bag
        * @return an iterator that iterates through the items in the bag
        */
        public Iterator<E> iterator() {
            return new BagIterator();
        }
        /**
        *   The iterator implementation
        */
        private class BagIterator implements Iterator<E> {
            private int i = 0;
            public boolean hasNext()  { return i < arraySize; }
            public void remove()      { System.out.println("to be implemented."); }
            public E next() {
                if(!hasNext()) {return null;}
                return items[i++];
            }

        }

        /**
        *  Insert new items into the bag
        * @param item the new item to be inserted.
        */
        public void insert(E item)
        {
                if(arraySize == capacity){
                        resize();
                }
                items[arraySize] = item;
                arraySize++;
        }

        /**
        * Returns an item by index
        * @param index is the item index
        */
        public E get(int index)
        {
                return items[index];
        }

        /**
        *   size of the bag
        * @return size the number of items in the bag.
        */
        public int size(){
            return arraySize;
        }

        /**
        *   if the bag contains a given item?
        * @return true if bag contains the item. false otherwise
        */
        public boolean contains(E item)
        {
            for(int i = 0; i < arraySize; ++i){
                if(items[i].equals(item)) return true;
            }
            return false;
        }
        /**
        *   is the bag empty?
        * @return true if bag is empty. false otherwise
        */
        public boolean isEmpty()
        {
                return arraySize == 0;
        }
        /**
        * Resize the bag when capacity is not enough
        */
        protected void resize(){
                capacity *= 2;
                int index =0;
                E[] temp = (E[]) new Comparable[capacity];
                for(E e: items){
                        temp[index++] = e;
                }
                arraySize = index;
                items = temp;
        }

        /**
        * unit test for bag
        */
        public static void main(String[ ] args)
        {
                Bag<Integer> bag = new Bag();
                for(int i = 1; i <= 20; i++){
                        bag.insert(i);
                }

                /*for(int i = 0; i < bag.size(); i++){
                        System.out.println(bag.get(i));
                }*/

                for(Integer i: bag){
                        System.out.print(i+",");
                }
        }
}
```

Listing 2: Bag Test Class

```
/*
 * Tests the Bag class
 */
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
/**
 *
 * @author anwar mamat
 */
public class BagTest {

    public static void main(String[] args){
        int sampleSize = 10000000;
        String fileName = "sorted"+Integer.toString(sampleSize)+".txt";
        Bag<Integer> bag = new SortedBag();
        Numbers num = new Numbers();
        try {
            num.read(bag, fileName);
        } catch (IOException ex) {
            System.err.println(ex.getMessage());
        }
        System.out.println("bag size=" + bag.size());
        //for(Integer b:bag){
        //     System.out.print(b +",");
        //}
        //bag.contains(7);

        long tStart = System.currentTimeMillis();
        //for(int j = 0; j < 10; ++j){
            for(int i = 0; i < 1000; ++i){
                int n = (int)(Math.random()*sampleSize);
                //System.out.println(n);
                bag.contains(n);
            }
        //}
        long tEnd = System.currentTimeMillis();
        long tDelta = tEnd - tStart;
        double elapsedSeconds = tDelta / 1000.0;
        System.out.println("Elapsed time: " + elapsedSeconds + " seconds");

    }
}
```

Listing 3: Generating Random Numbers

```
/**
 * This class writes integer numbers to a text file
 */
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 * @author anwar mamat
 */

public class Numbers{
    private int minValue = 0;
    private int maxValue = 0;

    Numbers(int low, int high){
        minValue = low;
        maxValue = high;
    }
    Numbers(){
        minValue = 0;
        maxValue = 0;
    }
    public void write(String fileName) throws IOException{
        try {
            File file = new File(fileName);
            BufferedWriter output = new BufferedWriter(new FileWriter(file));
            List<Integer> temp = new ArrayList();
            for(int i = minValue; i <=maxValue; ++i){
                temp.add(i);
            }
            Collections.shuffle(temp);
            for(Integer i: temp){
                output.write(Integer.toString(i) + '\n');
            }
            output.close();
        } catch ( IOException e ) {
            e.printStackTrace();
        }
    }
     public void read(Bag bag,String fileName) throws IOException{
        try {
            File file = new File(fileName);
            BufferedReader input = new BufferedReader(new FileReader(file));
            String line="";
            int count = 0;
            while((line = input.readLine()) != null){
                Integer t = Integer.parseInt(line);
                bag.insert(t);
                count++;
                if(count % 1000000==0){
                    System.out.println(count);
                }
            }
            input.close();
        } catch ( IOException e ) {
            e.printStackTrace();
        }
    }

    public static void main(String[] args){
        int size = 10000000;
        String fileName = Integer.toString(size)+”.txt”;
        Numbers w = new Numbers(1,size);
        try {
            w.write(fileName);
        } catch (IOException ex) {
            System.err.println(ex.getMessage());
        }
        System.out.println(”Done.␣Wrote␣”+ size + ”␣numbers␣to␣” + fileName);
    }
}
```

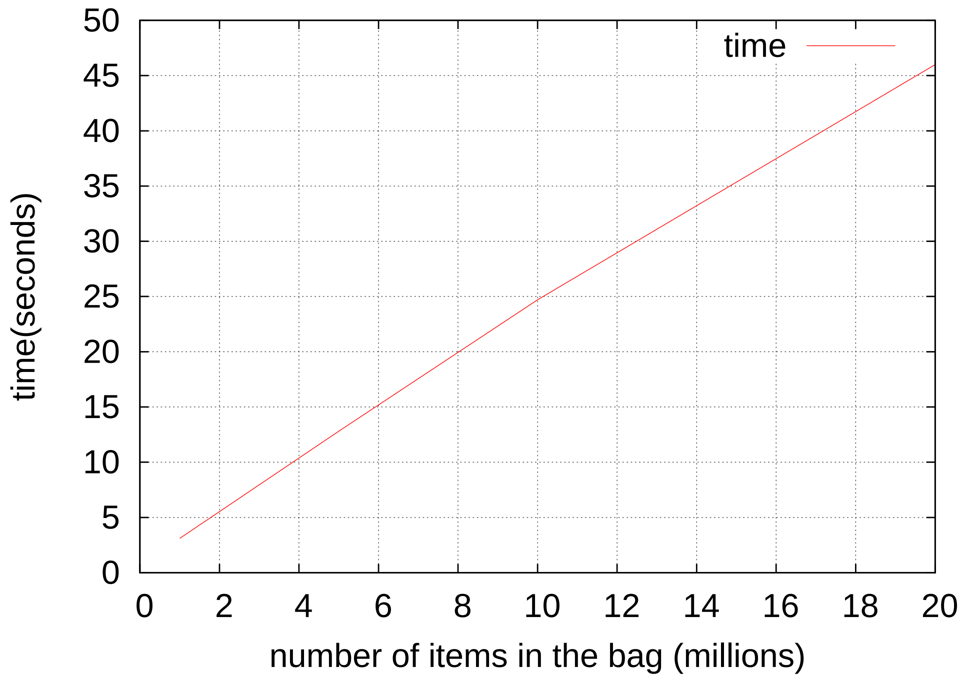

Figure 2.1: Processing time increases as the number of items in the bag increases