COS 226 Data Structures and Algorithms
Computer Science Department
Princeton University
Fall 2015

Week 2 Activity

1. *Mathematical models for runtime analysis. Algorithms textbook 1.4*

   (a) Given the following functions, rank them from smallest(fastest algorithm) to largest(slowest algorithm) for large values of n. For example $logn < n$ and log algorithms are faster than linear algorithms for large n.
   $nlog_3n$, $2log_3n$, $\sqrt[3]{n}$, n^2, $3log_2n$, $n^{0.001}$

   (b) Suppose that an algorithm satisfies the following recurrence equation where T(n) is the time it takes to run the algorithm on a data set of size n.
   $T(n) = 2 * T(n/2) + n$
   Solve the recurrence relation to prove that this algorithm has a order of growth $nlog_2n$

   (c) Suppose that it takes 1 second to run the above algorithm when $N = 1000$. Estimate the runtime if $N = 1000000$.

2. *Code Tracing. Algorithms textbook 1.3*

   (a) What is the purpose of this code and what value does count return?

```
int[] intArray = new int[N];
for (int i=0; i<N; i++)
    intArray[i] = StdRandom.uniform(N);
int count = 0;
Arrays.sort(intArray);
int i = 0;

while (i < N) {
 int current = intArray[i];
 count += 1;
 int j = i + 1;
 while (j < N) {
  if (intArray[j] != current)
    break;
  j++;
 }
 i = j;
}
return count;
}
```

   (b) Determine (by counting operations) the order-of-growth of the program above. Assume that the order of growth of Arrays.sort is $nlog_2n$

3. *Iterators*

(a) Consider a collection called Bag (see Bag.java code at the Appendix section of this handout). The container for the Bag is an array a. Complete MyIterator code so that a Bag Iterator will print elements backwards from the order they were added. (that is, from $a[n-1]...a[0]$)

```
private class MyIterator implements Iterator<Item> {
  private _________;    // next item to return

  public MyIterator() {
      _____
  }
  public boolean hasNext()  { ______________________________}
  public void remove()      { throw new UnsupportedOperationException();}

  public _____  next() {
    if (!hasNext()) throw new NoSuchElementException();
      return _______;
  }
}
```

(b) What is the output of the following code?

```
Bag<Integer> myBag = new myBag<Integer>();
myBag.add(3);
myBag.add(1);
myBag.add(2);
for (int i : myBag){
  for (int j : myBag) {
     StdOut.println(i + " " + j);
  }
  StdOut.println();
}
```

4. *Memory*

Estimate the total memory used by a Bag object of N Integers. Note that Integers are objects and hence subject to object overhead in addition to space for an integer. Be sure to classify all components in your solution (eg: 16(object overhead) + 4(int) etc..) and identify the padding needed (if any). Express the answer in tilde notation.

1 Appendices

```
/*************************************************************************
 * Name: Andy Guna
 * Login: guna
 * Precept: P00
 *
 * Compilation: javac Bag.java
 * Execution: java Bag N
 * Dependencies: none
 *
 * Description: An implementation of a Bag class that demonstrate the use of Iterators and generics. The Bag class
 * implements Iterable. Also the Bag class is a generic class <Item>. The <Item> is
 * Initialized by the client Main (in this program as <Integer>. There are two
 * Iterator nested classes are provided, MyIterator and RandomIterator.
 * You can try to run the code with each one and see what happens.
 *
 *************************************************************************/

import java.util.Iterator;
import java.util.NoSuchElementException;

import edu.princeton.cs.algs4.StdOut;
import edu.princeton.cs.algs4.StdRandom;

public class Bag<Item> implements Iterable<Item> {

    private Item[] a;          // array of items
    private int N = 0;         // number of elements in the bag

    public Bag() {
        // need cast because Java does not support generic array creation
        a = (Item[]) new Object[1];
    }

    public boolean isEmpty() { return N == 0; }
    public int size()        { return N;      }

    // resize the underlying array holding the elements
    private void resize(int max) {
        Item[] temp = (Item[]) new Object[max];
        for (int i = 0; i < N; i++)
            temp[i] = a[i];
        a = temp;
    }

    // insert an item
    public void add(Item item) {
        if (item == null) throw new NullPointerException();
        if (N == a.length) resize(2*a.length);  // double size of array if necessary
        a[N++] = item;                          // add element
    }

    // remove the last item
    public Item removeLast() {
        if (isEmpty())
            throw new NoSuchElementException("bag is underflow");
        Item value = a[--N];
        return value;
    }
    public Iterator<Item> iterator()  { return new MyIterator();  }

    // The following iterator, doesn't implement remove() since it's optional
    private class RandomIterator implements Iterator<Item> {
        private int[] perm;   // return items in order a[perm[0]], a[perm[1]], ...
        private int n = 0;    // next item to return is a[perm[n]]

        public RandomIterator() {
            n = 0;
            perm = new int[N];
            for (int i = 0; i < N; i++)
                perm[i] = i;
            StdRandom.shuffle(perm);
        }
        public boolean hasNext()  { return n < N;                               }
        public void remove()      { throw new UnsupportedOperationException();  }

        public Item next() {
            if (!hasNext()) throw new NoSuchElementException();
            return a[perm[n++]];
        }
    }

     private class MyIterator implements Iterator<Item> {
        private _____ ;    // next item to return
        public MyIterator() {
            ----------
        }
        public boolean hasNext()  { _______________         }
        public void remove()      { throw new UnsupportedOperationException();  }

        public _____ next() {
            if (!hasNext()) throw new NoSuchElementException();
            return ______;
        }
    }
    // a test client
    public static void main(String[] args) {
        int N = Integer.parseInt(args[0]);
        Bag<Integer> MyBag = new Bag<Integer>();
```

```
        // add N integers
        for (int i = 0; i < N; i++) {
            MyBag.add(i);
        }
        // random iteration

        for (int i : MyBag){
         for (int j : MyBag) {
            StdOut.println(i + " " + j);
         }
         StdOut.println();
        }

        // delete all of the elements
        StdOut.println("remove all of the elements");
        while (!MyBag.isEmpty()) {
            int k = MyBag.removeLast();
            StdOut.print(k + " ");
        }
        StdOut.println();
    }
}
```