COS 226 Data Structures and Algorithms
Computer Science Department
Princeton University
Spring 2016

Week 2 Activity

1. **Sorting**

   (a) Sorting Analysis

      i. Write a recurrence relation that describes the complexity of selection sort. (An example of a recurrence relation that describes binary search is $T(N) = T(N/2) + 1$)

      ii. Solve the recurrence relation of selection sort to show that selection sort is ~ $1/2n^2$

   (b) *Insertion Sort*

      i. Sort the word "ananda" using insertion sort.

      ii. Consider the assert statement (1) about the array segment $a[0..i]$ that is given at the end of the j loop in the sort below. Write another assert statement about the array before the beginning of the j loop.

```
public static void sort(Comparable[] a) {
        int N = a.length;
        for (int i = 1; i < N; i++) {


            for (int j = i; j > 0 && less(a[j], a[j-1]); j--) {
                exch(a, j, j-1);
            }
            assert isSorted(a, 0, i); _______ (1)
        }
        assert isSorted(a);
}
```

      iii. Suppose that you have an array of length 2N consisting of N Bs followed by N As. Below is the array when N = 10.
      B B B B B B B B B B A A A A A A A A A A
      How many compares does it take to insertion sort (ascending order) the array as a function of N? Use tilde notation to simplify your answer.

2. **Iterators**

(a) Consider an example collection called Bag (different from Bag.java implementation of algs4). Refer to the code given in the appendix below. Complete ReverseIterator 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 ReverseIterator implements Iterator<Item> {
  private _________;    // next item to return

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

(c) Rewrite the above code using explicit iterator methods hasNext and next.

3. **Amortized Analysis (Bonus Question)**

   Suppose we have a resizing array implementation of a stack, that increases in size by 10 when the array is full, and decreases in size by 10 if the array has 20 or more empty spaces.

   (a) What is the best case cost of push and pop operations?

   (b) What is the worst case cost of push and pop operations?

   (c) What is the amortized cost of push and pop operations?

   (d) Give a sequence of push and pop operations that can result in worst case performance.

   (e) Is there any advantage to using increments by 10 instead of doubling.

1 Appendix

```
/*************************************************************************
 *  Name: Andy Guna
 *  NetID:  guna
 *  Precept: P99
 *
 *  Partner Name:    N/A
 *  Partner NetID:   N/A
 *  Partner Precept: N/A
 *
 *  Operating system:
 *  Compiler:
 *  Text editor / IDE:
 *
 *  Hours to complete assignment (optional):
 *
 * A program that demonstrate the use of Iterators and generics. The Bag class (different from algs4)
 * implements Iterable. Also the Bag class is a generic class <Item>. The <Item> is
 * Initialized by the client Main (in this program as <int>. 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.
 *
 * To run <   java Bag  N
 * (where N is the Bag size - you provide that)
 *
 *
 *************************************************************************/

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

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

// A different implementation of a Bag different from algs4 Bags implementation

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

    private Item[] a;      // array of Items
    private int N;         // max number of elements in the bag
    private int size;      // actual number of elements in the bag

    public Bag(int N) {
     // need cast because Java does not support generic array creation
     a = (Item[]) new Object[N];
     this.N = N;
     this.size = 0;
    }

    public boolean isEmpty() { return size == 0; }
    public int size()        { return size; }
    public int capacity()    { 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;
        this.N = max;
    }
```

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

    // removes a random item
    public Item remove() {
        if (isEmpty())
            throw new NoSuchElementException("bag is underflow");
        Item value = a[--size];
        return value;
    }

    public Iterator<Item> iterator()  { return new ReverseIterator();  }

     private class ReverseIterator implements Iterator<Item> {

        private      ;    // next item to return

        public ReverseIterator() {

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

        public Item next() {


        }
    }

    // a test client
    public static void main(String[] args) {
        int N = Integer.parseInt(args[0]);
        Bag<Integer> MyBag = new Bag<Integer>(N);

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

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