INTRODUCTION TO JAVA

Activity 8: Algorithms and Data Structures (Part 2)

Exercises

1. Develop an implementation of Questions.java that takes the maximum number `N` as command-line input. Prove that your implementation is correct.

Output:

Think of an integer between 0 and 1023

Is it less than 512? false

Is it less than 768? true

Is it less than 640? false

Is it less than 704? true

Is it less than 672? true

Is it less than 656? false

Is it less than 664? false

Is it less than 668? true

Is it less than 666? false

Is it less than 667? true

Your number is 666

7. Modify BinarySearch.java so that if the search key is in the array, it returns the smallest index `i` for which `a[i]` is equal to key, and otherwise, it returns `-i`, where `i` is the smallest index such that `a[i]` is greater than key.

   Output:
   Done reading words
   Done sorting words
   Omar

9. Write a program Dedup.java that reads strings from standard input and prints them on standard output with all duplicates removed (in sorted order).

Output:

activity8

algorithms

data

development

java

june

lopez

introduction

omar

programming

structures

tiempo

to

12. Add code to LRS.java to make it print indices in the original string where the longest repeated substring occurs.

Input:

it was the best of times it was the worst of times

it was the age of wisdom it was the age of foolishness

it was the epoch of belief it was the epoch of incredulity

it was the season of light it was the season of darkness

it was the spring of hope it was the winter of despair

Output:

'st of times it was the '

13. Find a pathological input for which LRS.java runs in quadratic time (or worse)

Input:

MobyDick.txt

Output:

Exception in thread "main" java.lang.OutOfMemoryError: Java heap space

atjava.util.Arrays.copyOfRange(Unknown Source)

atjava.lang.String.<init>(Unknown Source)

atjava.lang.String.substring(Unknown Source)

atLRS.lrs(LRS.java:48)

atLRS.main(LRS.java:71)

18.**Median.** Add to StdStats.java a method `median()` that computes in linearithmic time the median of a sequence of *N* integers.

Hint: reduce to sorting.

Output:

min 1.000

mean 3.000

max 5.000

sum 15.000

stddev 1.581

var 2.500

stddevp 1.414

varp 2.000

median 2.000

19. **Mode.** Add to StdStats.java a method `mode()` that computes in linearithmic time the mode (value that occurs most frequently) of a sequence of *N* integers.

    Hint: reduce to sorting.

Output:

min 1.000

mean 3.000

max 5.000

sum 15.000

stddev 1.581

var 2.500

stddevp 1.414

varp 2.000

median 2.000

mode 3.000

20. **Integer sort.** Write a *linear*-time filter IntegerSort.java that takes from standard input a sequence of integers that are between 0 and 99 and prints the same integers in sorted order on standard output. For example, presented with the input sequence

```
98 2 3 1 0 0 0 3 98 98 2 2 2 0 0 0 2
```

your program should print the output sequence

```
0 0 0 0 0 0 1 2 2 2 2 2 3 3 98 98 98
```

Output:
0 0 0 0 0 0 1 2 2 2 2 2 3 3 98 98 98

33. **Quicksort.** Write a recursive program QuickSort.java that sorts an array of randomly ordered distinct `Comparable` elements.

Hint: Use a method like the one described in the previous exercise. First, partition the array into a left part with all elements less than *v*, followed by *v*, followed by a right part with all elements greater than *v*. Then, recursively sort the two parts.

Extra credit: Modify your method (if necessary) to work properly when the elements are not necessarily distinct.

Output:

Generating input: 0.0 seconds
Quicksort: 0.0 seconds
Comparisons: 819
Exchanges: 267

34. **Reverse domain.** Write a program to read in a list of domain names from standard input, and print the reverse domain names in sorted order. For example, the reverse domain of `cs.princeton.edu` is `edu.princeton.cs`. This computation is useful for web log analysis. To do so, create a data typeDomain.java that implements the `Comparable` interface, using reverse domain name order.

Output:

apple.com
bolle.cs.princeton.edu
cnn.com
cs.princeton.edu
ee.princeton.edu
google.com
princeton.edu
www.cs.princeton.edu