CS 401 Algorithms 04/12/2019

Assignment1

Due Date: 04/22/2019

Part I: Algorithm Analysis

1. Give the order of growth (as a function of n) of the running times of each of the following code fragments. Please show your work.

a.
```
while (n > 0) {
   for (int k = 0; k < n; k++)
         printReport(); // runs in O(N/2) time
   n = n / 2;
}
```

b.
```
for (int i = 0; i < n; i++)
   for (int j = i+2; j > i; j--)
         for (int k = n; k > j; k--)
               System.out.println(i+j+k);
```

c.
```
void aMethod(int n){
   if (n <= 1)
         return;
   anotherMethod(); //runs in O(1) time
   aMethod(n/2);
   aMethod(n/2);
}
```

d.
```
public static int mystery(int n) {
    int i = 1;
    int j = 0;
    while (i < n) {
        j++;
        if (j == i) {
            i++;
            j = 0;
        }
    }
    return j;
}
```

e.
```
public static int compute(int n) {
    int total = 0;
    for (int i = 1; i < n; i++) {
        int k = 1;
        while (k < i) { k = k + k;          }
        while (k > 1) { k /= 2;    total++; }
    }
    return total;
}
```

f.
```
//n, the length of arr
public static int removeDuplicates(char[] arr) {
    int len = arr.length;
    int i = 0;          // index of current item to find
    while (i < len) {
        int j;          // will be index of duplicate of arr[i]
        for (j = i + 1; j < len; j++) {
            if (arr[i] == arr[j]) break;
        }
        if (j == len) {   // no duplicate of arr[i] found; go to next i
            i++;
        } else {        // duplicate found; shift array over arr[j]
            for (int k = j + 1; k < len; k++) {
                arr[k - 1] = arr[k];
            }
            len--;
            arr[len] = 0;
        }
    }
    return len;
}
```

Part II: Programming

1. Write a method in Java that returns the total number of trailing zeroes for all integers from 1 to its parameter n; given 5, it returns 0 + 1 + 0 + 2 + 0 = 3. The *trailing zeroes* of an integer *a* are the zeroes following the last 1 in *a*'s binary representation (ex: 0100 ->2; 0101->0). Using big-O notation in terms of its parameter *n* and provide the time complexity of your algorithm.

2. Write a program that, given two sorted arrays of n in values, prints all elements that appear in both arrays, in sorted order. The **running time** of your program **should be proportional to n** in the worst case (Exercise 1.4.12)

3. Write a program that, given an array `a[]` of n distinct integers, finds a strict local minimum: an entry `a[i]` that is strictly less than its neighbors. Each internal entry (other than `a[0]` and `a[n-1]`) has 2 neighbors. Your program should use **`~2(lgn)`** compares in the worst case (Exercise 1.4.18). Provide tests for all possible cases.

Sample Cases:
```
Array-> {5, -4, 10, 16, 11, 20, 24, -10};
Local minimum: -4              //it can also output 11 as local minimum

Array-> {-8, -6, 18, 8, 20, 4, 40};
Local minimum: 8               //it can also output -8 or 4 as local minimum
```

4. a. Design and implement an algorithm in Java that performs substring pattern matching. Your algorithm needs to determine if the text contains the given pattern and returns position of the pattern.

   Input: A text string ***t*** and a pattern string ***p***.
   Output: Does ***t*** contain the pattern **p** as a substring, and if so where?

   b. What is the complexity of your algorithm. Please show your work.

5. Provide a sample program that draws points, lines, squares and circles using `StdDraw` class provided by the textbook. You can learn more about this class at https://algs4.cs.princeton.edu/code/javadoc/edu/princeton/cs/algs4/StdDraw.html.

HOW TO SUBMIT
You are supposed to submit your work as a single zip file via CANVAS. Zip file will include: • PartI.pdf • TrailingZeros.java • Exercise1412.java • Exercise1418.java • StringPatternMatcher.java • MyStdDraw.java Please use the following file format while naming the zip file: LastNameFirstnameX_Y.zip where LastNameFirstname is your last name with the first letter in capital, followed by your first name with the first letter in capital; the X is the course code; the Y is the assignment #. (ex: SerceFatmaCS401_1.zip)