

INTRODUCTION TO JAVA

ACTIVITY 7: Algorithms and Data Structures (Part 1)

Exercises

1. Implement the method `printAll()` for [ThreeSum.java](#), which prints all of the triples that sum to zero.

```
public static void printAll(int[] a) {
    int N = a.length;
    for (int i = 0; i < N; i++) {
        for (int j = i + 1; j < N; j++) {
            for (int k = j + 1; k < N; k++) {
                if (a[i] + a[j] + a[k] == 0) {
                    StdOut.println(a[i] + " " + a[j] + " " + a[k]);
                }
            }
        }
    }
}
```

8ints.txt:

Elapsed time = 0.0

4

30 -30 0

30 -20 -10

-30 -10 40

-10 0 10

2. Modify [ThreeSum.java](#) to take a command-line argument `x` and find a triple of numbers on standard input whose sum is closest to `x`.

```
public static void TripleNumbersClosestToX(int x, int[] a) {
    int N = a.length;
    for (int i = 0; i < N; i++) {
        for (int j = i + 1; j < N; j++) {
            for (int k = j + 1; k < N; k++) {
```

```

        if (a[i] + a[j] + a[k] == x) {
            StdOut.println("(" + a[i] + ") + (" + a[j] + ") + (" + a[k] + ") = " +
x);
        }
    }
}

```

Entered number = 10

(30) + (-30) + (10) = 10

(30) + (-20) + (0) = 10

(-30) + (40) + (0) = 10

(-20) + (-10) + (40) = 10

Elapsed time = 0.001

- Write a program [FourSum.java](#) that takes an integer N from standard input, then reads N long values from standard input, and counts the number of 4-tuples that sum to zero. Use a quadruple loop. What is the order of growth of the running time of your program? Answer: exponential. Estimate the largest N that your program can handle in an hour. Then, run your program to validate your hypothesis.

DONE.

```

public static int count(long[] a) {
    int N = a.length;
    int cnt = 0;
    for (int i = 0; i < N; i++) {
        for (int j = i+1; j < N; j++) {
            for (int k = j+1; k < N; k++) {
                for (int l = k+1; l < N; l++) {
                    if (a[i] + a[j] + a[k] + a[l] == 0) {
cnt++;
                    }
                }
            }
        }
    }
    return cnt;
}

```

6. What is the value of x after running the following code fragment?

```
int x = 0;
for (int i = 0; i < N; i++)
    for (int j = i + 1; j < N; j++)
        for (int k = j + 1; k < N; k++)
            x++;
```

Answer: $N \text{ choose } 3 = N(N-1)(N-2) / 6$.

16. Each of the four Java functions below returns a string of length N whose characters are all x. Determine the order of growth of the running time of each function. Recall that concatenating two strings in Java takes time proportional to the sum of their lengths.

```
public static String method1(int N) {
    if (N == 0) return "";
    String temp = method1(N / 2);
    if (N % 2 == 0) return temp + temp;
    else return temp + temp + "x";
}

public static String method2(int N) {
    String s = "";
    for (int i = 0; i < N; i++)
        s = s + "x";
    return s;
}

public static String method3(int N) {
    if (N == 0) return "";
    if (N == 1) return "x";
    return method3(N/2) + method3(N - N/2);
}

public static String method4(int N) {
    char[] temp = new char[N];
    for (int i = 0; i < N; i++)
        temp[i] = 'x';
    return new String(temp);
}
```

Output:

```
xxxxxxxxxx
Elapsed time for method 4 = 0.0
xxxxxxxxxx
Elapsed time for method 3 = 0.0
xxxxxxxxxx
Elapsed time for method 1 = 0.0
xxxxxxxxxx
Elapsed time for method 2 = 0.0
```

17. The following code fragment (adapted from a Java programming book) creates a random permutation of the integers from 0 to N-1. Determine the order of growth of its running time as a function of N. Compare its order of growth with [Shuffle.java](#) from Section 1.4.

```
int[] a = new int[N];
boolean[] taken = new boolean[N];
int count = 0;
while (count < N)
{
    int r = StdRandom.uniform(N);
    if (!taken[r])
    {
        a[r] = count;
        taken[r] = true;
        count++;
    }
}
```

DONE

Creative Exercises

4. **Young tableaux.** Suppose you have in memory an N-by-N grid of integers a such that $a[i][j] < a[i+1][j]$ and $a[i][j] < a[i][j+1]$ for all i and j like the table below.

5	23	54	67	89
6	69	73	74	90
10	71	83	84	91
60	73	84	86	92
99	91	92	93	94

Devise an $O(N)$ time algorithm to determine whether or not a given integer x is in a Young tableaux.

Answer. Start at the upper right corner. If element = x , done. If element $> x$, go left. Otherwise go down. If you reach bottom left corner, then it's not in table. $O(N)$ since can go left at most N times and down at most N times.

6. Moore's Law. This problem investigates the ramifications of exponential growth. Moore's Law (named after Intel co-founder Gordon Moore) states that microprocessor power doubles every 18 months. See [this article](#) which argues against this conventional wisdom. Write a program `MooresLaw.java` that takes an integer parameter `N` and outputs the increase in processor speed over a decade if microprocessors double every `N` months.

- a. How much will processor speed increase over the next decade in speeds double every `N = 18` months?
- b. The true value may be closer to speeds doubling every two years. Repeat (a) with `N = 24`.

Subset sum. Write a program `Exponential.java` that takes a command line integer `N`, reads in `N` long integer from standard input, and finds the *subset* whose sum is closest to 0. Give the order of growth of your algorithm.

DONE.

Web Exercises

Write a program `OneSum.java` that takes a command-line argument `N`, reads in `N` integers from standard input, and finds the value that is closest to 0. How many instructions are executed in the data processing loop?

DONE.

Write a program `TwoSum.java` that takes a command-line argument `N`, reads in `N` integers from standard input, and finds the pair of values whose sum is closest to 0. How many instructions are executed in the data processing loop?

DONE.