第二次作业

1.4.1

现在有 P(N) = N(N - 1)(N - 2) / 6
如果 P(N + 1) = (N + 1)N(N - 1) / 6 成立则公式成立

因为
P(N + 1)
= P(N) + N! / 2!(N-2)!
= P(N) + N(N - 1) / 2
= N(N - 1)(N - 2) / 6 + N(N - 1) / 2
= N(N - 1)(N - 2) / 6 + 3N(N - 1) / 6
= N(N - 1)(N - 2 + 3) / 6
= N(N - 1)(N + 1) / 6

所以原公式成立

1.4.2

代码:

```java
public class TreeSum {

    // 计算数组中三个数之和为0的元组数量
    public static int count(int[] array) {
        int length = array.length; // 获取数组的长度
        int count = 0; // 初始化计数器

        BigInteger bigInteger; // 用于存储三个数的和

        // 三重循环遍历数组中的所有可能的三元组
        for (int i = 0; i < length; i++) {
            for (int j = i + 1; j < length; j++) {
                for (int k = j + 1; k < length; k++) {
                    // 将三个数转换为BigInteger并相加
                    bigInteger = BigInteger.valueOf(array[i]);
                    bigInteger = bigInteger.add(BigInteger.valueOf(array[j]));
                    bigInteger = bigInteger.add(BigInteger.valueOf(array[k]));

                    // 如果三个数的和为0，则计数器加1
                    if (bigInteger.intValue() == 0) {
                        count++;
                    }
                }
            }
        }
        return count; // 返回计数器的值
    }

    public static void main(String[] args) {
        // 从命令行参数中读取文件名
        In in = new In(args[0]);
        // 读取文件中的所有整数并存储到数组中
        int[] array = in.readAllInts();
        // 输出数组中三个数之和为0的元组数量
        StdOut.println(count(array));
    }
}
```

1.4.4

语句块	运行时间	频率	总时间
D	t0	x	t0 * x
C	t1	N^2/2 - N/2	t1 * (N^2/2 - N/2)
B	t2	N	t2 * N
A	t3	1	t3
		总时间	(t1/2) N^2 + (-t1/2 + t2) N + t3 + t0x
		近似	~(t1/2) N^2 （假设x很小）
		增长的数量级	N^2

1.4.5

a. ~N
b. ~1
c. ~1
d. ~2N^3
e. ~1
f. lg(N^2 + 1) / lg N ~= lg(N^2) / lg N = 2 lg N / lg N = ~2
g. $(N^{100)/(2}N)$

1.4.6

a. O(N)
b. O(N)
c. O(N log(N))

1.4.7

if (a[i] + a[j] + a[k] == 0)

一个运算用于 if 检查；一个运算用于求和 a[i] 和 a[j]；一个运算用于将结果与 a[k] 相加
以及一个将结果与 0 进行比较的操作。

因此，主 “if ”的频率为 4 (N^3/6 - N^2/2 + N/3)

总计变为
t1(4N^3/6 - 4N^2/2 + 4N/3) + t2(N^2/2 - N/2) + t3*N + t4

1.4.8

代码

```java
public class CalculateNumber {

    public static void main(String[] args) {
        In in = new In(args[0]);
        int[] values = in.readAllInts();
        StdOut.println(countNumberOfPairs(values));

        // 测试数据
        int[] values1 = {1, 2, 4, 1, 2, 1, 2, 4, 5, 1, 2, 4, 5, 1, 2 ,5, 6, 7, 7, 8, 2, 1, 2, 4
        StdOut.println("Equal pairs 1: " + countNumberOfPairs(values1) + " Expected: 49");

        int[] values2 = {1, 1, 1};
        StdOut.println("Equal pairs 2: " + countNumberOfPairs(values2) + " Expected: 3");
    }

    // O(n lg n)
    private static int countNumberOfPairs(int[] values) {
        Arrays.sort(values);

        int count = 0;
        int currentFrequency = 1;

        for (int i = 1; i < values.length; i++) {
            if (values[i] == values[i - 1]) {
                currentFrequency++;
            } else {
                if (currentFrequency > 1) {
                    count += (currentFrequency - 1) * currentFrequency / 2;
                    currentFrequency = 1;
                }
            }
        }

        if (currentFrequency > 1) {
            count += (currentFrequency - 1) * currentFrequency / 2;
        }
        return count;
    }

    // O(n)
    private static int countNumberOfPairs2(int[] values) {
        Map<Integer, Integer> valuesMap = new HashMap<>();
        int equalNumbersCount = 0;
```

```
        for (int i = 0; i < values.length; i++) {
            int count = 0;
            if (valuesMap.containsKey(values[i])) {
                count = valuesMap.get(values[i]);
            }
            count++;
            valuesMap.put(values[i], count);
        }

        for (int numberKey : valuesMap.keySet()) {
            if (valuesMap.get(numberKey) > 1) {
                int n = valuesMap.get(numberKey);
                equalNumbersCount += (n - 1) * n / 2;
            }
        }
        return equalNumbersCount;
    }
}
```

1.4.15

代码

```
private static int threeSumFaster(int[] array) {
        Map<Integer, List<Integer>> elementIndexes = new HashMap<>();
        for (int i = 0; i < array.length; i++) {
            if (!elementIndexes.containsKey(array[i])) {
                elementIndexes.put(array[i], new ArrayList<>());
            }

            elementIndexes.get(array[i]).add(i);
        }

        int count = 0;

        for (int i = 0; i < array.length; i++) {
            for (int j = i + 1; j < array.length; j++) {
                int sum = array[i] + array[j];

                if (elementIndexes.containsKey(-sum)) {
                    for (int elementIndex : elementIndexes.get(-sum)) {
                        if (elementIndex > j) {
                            count++;
                        }
                    }

                }
            }
        }
        return count;
    }
```