Assignment 5: Big-O Sorting

Part 2: Miles

CS3305/W01 Data Structures

Casey Hampson

September 18, 2024

Program Output

```
Before conversion:
0.000 10.000 20.000 30.000 40.000 50.000 60.000 70.000 80.000 90.000
After conversion:
0.000 6.250 12.500 18.750 25.000 31.250 37.500 43.750 50.000 56.250
```

Worst Case Efficiency

Obviously, this code is going to be $\mathcal{O}(n)$, since the conversion from miles to kilometers doesn't have any relation to the number of input parameters we have, it's just a constant time calculation. The worst-case condition would perhaps be for the largest floats possible, but even then, as just mentioned, this has no relation to the size of the input, meaning it will still be considered to run at constant time, and the entire program will still run at $\mathcal{O}(n)$.

Source Code

```java
// Name: Casey Hampson
// Class: CS 3305/W01
// Term: Fall 2024
// Instructor: Sharon Perry
// Assignment: 05-Part-2-Miles

public class P2 {
    public static float MilesToKilometers(float input) {
        return (input / 1.6f);
    }

    // helper function to print our array
    // not general at all but suits the purpose of this program
    public static void PrintArray(float[] arr) {
        for (int i=0; i<10; i++) System.out.printf("%.3f ", arr[i]);
        System.out.printf("\n");
    }

    public static void main(String[] args) {
        float[] input = {0, 10, 20, 30, 40, 50, 60, 70, 80, 90};
        System.out.printf("Before conversion:\n");
        PrintArray(input);

        for (int i=0; i<10; i++) input[i] = MilesToKilometers(input[i]);
        System.out.printf("After conversion:\n");
        PrintArray(input);
    }
}
```