Hieu Trung Nguyen

# Worst Case Run-Time Analysis of day() Method

Version 1.0 - January 31, 2015

January 31, 2015

## 1 Introduction

This is a worst case run-time analysis of day() method in the Population class. In this analysis document, we will review the run-time analysis for day() using a line-by-line analysis, summation, simplified expression and conclude our analysis with a Big-Oh bound.

```
public void day() {
    for (int i = numGenomes - 1; i >= numGenomes / 2; i--) {
        myPopulation.remove(i);
    }
    boolean crossover;
    for (int i = 0; i < numGenomes / 2; i++) {
        Genome newGenome = new Genome(myPopulation.get(rand.nextInt(numGenomes / 2
        crossover = rand.nextBoolean();
        if (crossover) {
            newGenome.crossover(myPopulation.get(rand.nextInt(numGenomes / 2)));
            newGenome.mutate();
        } else {
            newGenome.mutate();
        }
        myPopulation.add(newGenome);
    }
    Collections.sort(myPopulation);
    mostFit = myPopulation.get(0);
}
```

# 2 Analysis

## 2.1 Line-By-Line

We will now review the line-by-line analysis of day(). Before we begin, let's briefly review the other method operations called inside day() with their Big-Oh bounds.

1. get() - Let $c_1$ stands for get(). This is an ArrayList get method. According to the ArrayList API, this operation runs on O(1).

2. add() - Let $c_2$ stands for add(). This is an ArrayList add method. According to the ArrayList API, this operation runs on O(1).

3. nextInt() - Let $c_3$ stands for nextInt(). This is a Random object method which runs on O(1).

4. nextBoolean() - Let $c_4$ stands for nextBoolean(). This is another Random object method that runs on O(1).

5. new Genome() - Let $c_5$ stands for new Genome(). This is a copy constructor from the genome class, it involves only constant operations and a for loop which loop for a fixed constant of 26 times. Therefore, this copy constructor is O(1). method which runs on O(1).

6. remove() - Let $c_6$ stands for remove(). This is an ArrayList remove method. Since in day(), we will always be removing from the end of the list, this operation is then O(1).

7. crossover() - Let $k_1$ stands for crossover(). This is a method from the genome class. Assuming that the genome length is in a reasonable range of a name, this operation runs on O(1). But since we are considering the worst-case, where a name can be unreasonably long, we will use n as the length of the Genome, where crossover() will operate on O(n).

8. mutate() - Let $k_2$ stands for mutate(). This is a method from the genome class. Considering the worst case, this operation is also O(n).

9. sort() - Let $s_1$ stands for sort(). This is a method from Collections and the cost of this operation is O(n log n).

After the review above we will use the representation, $k_1$ - crossover(), and $k_2$ - mutate(), all these methods will be O(n). $c_1$ - get(), $c_2$ - add(), $c_3$ - nextInt(), $c_4$ - nextBoolean(), $c_5$ - new Genome() and $c_6$ - remove(), all these methods will be O(1). $s_1$ - sort(), which will be O(n log n) in our analysis.

Below will be the line-by-line analysis:

1. We will consider the cost of the whole loop after. There is a declaration, an assignment, two arithmetic operations, one comparison and one assignment in this line, the cost of this line is 6.
2. There is a remove operation in this line, the cost of this line is $c_6$.
3. There is a declaration in this line, the cost of this line is 1.
4. There is a declaration, an assignment, a comparison, an arithmetic and a decrement operation in this line, the cost of this line is 5.
5. There is a declaration, an assignment, a new Genome(), a get(), a nextInt() and an arithmetic operation in this line, the cost of this line is $c_5 + c_1 + c_3 + 3$.
6. There is a declaration, an assignment, and a nextBoolean() operation in this line, the cost of this line is $c_4 + 2$.
7. There is a comparison in this line, the cost of this line is 1.
8. There is a crossover(), a get(), nextInt() and an arithmetic operation in this line, the cost of this line is $k_1 + c_1 + c_3 + 1$.
9. There is a mutate() operation in this line, the cost of this line is $k_2$.
10. There is a mutate() operation in this line, the cost of this line is $k_2$.
11. There is an add() operation in this line, the cost of this line is $c_2$.
12. There is a sort() operation in this line, the cost of this line is $s_1$.
13. There is a declaration, an assignment and a get() operation in this line, the cost of this line is $c_1 + 2$.

### 2.2 First Loop Analysis

We will now review the analysis of the first for loop in day(). Let f(n) be the cost of the for loop, where n is the size of the population.

$$
\begin{aligned}
f(n) &= \sum_{i=1}^{n/2} (c_6 + 5) \\
&= (c_6 + 5) \sum_{i=1}^{n/2} 1 \\
&= (c_6 + 5)(n/2 - 1 + 1) \\
&= (c_6 + 5)(n/2)
\end{aligned}
$$

Let $l_1$ be the total cost of the for loop, then $l_1$ = ($c_6$ + 5)(n / 2). We know that $c_6$ is a constant, therefore the total cost of the first for loop is O(n).

### 2.3 Second Loop Analysis

We will now review the analysis of the second for loop in day(). Let g(n) be the cost of the for loop, where n is the size of the population.

$$
\begin{aligned}
g(n) &= \sum_{i=1}^{n/2} (5 + c_5 + c_1 + c_3 + 3 + c_4 + 2 + 1 + k_1 + c_1 + c_3 + 1 + k_2 + k_2 + c_2) \\
&= (5 + c_5 + c_1 + c_3 + 3 + c_4 + 2 + 1 + k_1 + c_1 + c_3 + 1 + k_2 + k_2 + c_2) \sum_{i=1}^{n/2} 1 \\
&= (12 + c_5 + c_1 + c_3 + c_4 + k_1 + c_1 + c_3 + k_2 + k_2 + c_2) \sum_{i=1}^{n/2} 1
\end{aligned}
$$

Let $c_7 = c_5 + c_1 + c_3 + c_4 + c_1 + c_3 + c_2$. We know $c_7$ is O(1) since $c_1$, $c_2$, $c_3$, $c_4$, and $c_5$ are constants. Let $k_3 = k_1 + k_2 + k_2$, we know $k_3$ is O(n) since both $k_1$ and $k_2$ are O(n). Then now we have:

$$
\begin{aligned}
g(n) &= (c_7 + k_3) \sum_{i=1}^{n/2} 1 \\
&= (c_7 + k_3)(n/2 + 1 - 1) \\
&= (c_7 + k_3)(n/2)
\end{aligned}
$$

Let $l_2$ be the total cost of the second for loop, then $l_2$ = ($c_7$ + $k_3$)(n / 2). Since $k_3$ is O(n), the total cost of the second loop is O($n^2$).

# 3 TOTAL COST

Let $T$ be the total cost of the day() method. $T = l_1 + 1 + l_2 + s_1 + c_1 + 2$. Let $c = 1 + c_1 + 2$. $T = l_1 + l_2 + s_1 + c$. We know that $l_1$ is O(n), $s_1$ is O(n log n), $c$ is O(1) and $l_2$ is O($n^2$). Since $T = c + l_1 + s_1 + l_2$, T is therefore O($n^2$).