Logarithmic Runtime

In this lesson, we'll discuss when an algorithm can have a logarithmic runtime.

We'll cover the following

- Iterating powers of a number
- Harmonic series

Iterating powers of a number

Let's analyze the loop below where we iterate over all powers of 2

```
for (int i = 1; i <= N; i *= 2)
    x++;
```

- Iteration 1: $i = 1$
- Iteration 2: $i = 2$
- Iteration 3: $i = 4$
- ...
- Iteration x: $i = 2^{x-1}$

Let's say the loop terminates after the j^{th} iteration, i.e.,

$2^{j-1} <= N$

$j - 1 <= logN$

$j <= logN + 1$

Hence, the loop runs $(logN + 1)$ times.

In Big-O notation, the time complexity is $O(logN)$

A similar analysis gives $O(logN)$ runtime for the loop below.

```
for (int i = N; i >= 1; i /= 2)
```

```
for (int i = N; i >= 1; i /= 2)
        x++;
```

Harmonic series

Consider the piece of code below:

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

For all integers between 1 and N, we iterate over their multiples. The number of operations, therefore, will be:

$$N + \frac{N}{2} + \frac{N}{3} + \frac{N}{4} + ... + \frac{N}{N-1} + 1$$

$$= N[1 + \frac{1}{2} + \frac{1}{3} + ... + \frac{1}{N-1} + \frac{1}{N}]$$

Let’s define an upper bound on the second term. Grouping the terms, we get:

First term: $1 <= 1$

Second term: $\frac{1}{2} + \frac{1}{3} <= \frac{1}{2} + \frac{1}{2} <= 1$

Third term: $\frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \frac{1}{7} <= \frac{1}{4} + \frac{1}{4} + \frac{1}{4} + \frac{1}{4} <= 1$

and so on.

Let k be the number of groups we can make, then

$$1 + 2 + 4 + ... + 2^{k-1} <= N$$

$$2^k - 1 <= N$$

$$2^k <= N + 1$$

$$k <= log(N + 1)$$

Coming back to the number of operations, we have

$$N[1 + \frac{1}{2} + \frac{1}{3} + ... + \frac{1}{N-1} + \frac{1}{N}] <= N * log(N + 1)$$

Therefore, the time complexity is $O(NlogN)$.

In the next lesson, we’ll discuss an example where the runtime is not what it seems at a first glance.