Big O Notation

Khalid Hourani

February 14, 2017

We say a function $f(n)$ is $O(g(n))$, written

$$f(n) \text{ is } O(g(n)) \text{ or}$$
$$f(n) = O(g(n)) \text{ or}$$
$$f(n) \in O(g(n))$$

if and only if there exists a real constant $c > 0$ and an integer constant $m \geq 1$ such that

$$f(n) \leq cg(n) \text{ for all } n \geq m$$

In other words, $f(n)$ is $O(g(n))$ when it is bounded from above by $g(n)$. In particular, we use this for analyzing an algorithm to discuss its runtime and memory space. For example, the following algorithm for checking if a number is prime runs in $O(\sqrt{n})$ time:

```
int isPrime(int n)
{
        for (int i = 2; i * i < n; i++)
        {
                if (n % i == 0)
                {
                        return false;
                }
        return true;
        }
}
```

because this algorithm will perform an operation for each integer less than $sqrt(n)$.