The Big O Notation

Franck van Breugel

July 30, 2007

We use $\mathbb{N}$ to denote the set $\{0, 1, 2, \ldots\}$ of natural numbers. Let $f : \mathbb{N} \to \mathbb{N}$ be a function from natural numbers to natural numbers. We will associate such a function to a piece of Java code. This function will capture the answer to the question "How many elementary operations are at most performed when executing the Java code for input of a given size?"

Consider the following method.

```
1  /**
2     Returns the factorial of the given number.
3
4     @param n a number.
5     @pre. n > 0
6     @return the factorial of the given number.
7  */
8  public static int factorial(int n)
9  {
10    long factorial = 1;
11    int i = 2;
12    while (i <= n)
13    {
14       factorial *= i;
15       i++;
16    }
17    return factorial;
18 }
```

Let us first estimate how many elementary operations are performed at each line of the above method.

line	number
10	2
11	2
12	4
14	4
15	3
17	2

We will come back to the fact that these are just estimates. The total number of elementary operations depends on the value of `n`. Hence, the total number can be captured by a function

$f : \mathbb{N} \to \mathbb{N}$ defined by

$$f(n) = 11n + 10,$$

where n represents the value of `n`.

The actual number of elementary operations that are performed depends on many factors. For example, some computers can assign a value to a variable of type `long` in a single operation whereas other computers may need two. Hence, estimates like

$$f_1(n) = 14n + 12$$

and

$$f_2(n) = 17n + 14$$

may be reasonable as well. The big O notation provides us with an abstraction that can capture all these estimates. The functions f, f_1 and f_2 are all elements of $O(n)$, where we use n to denote the identity function, that is, the function that assigns to n the value n.

Why are f, f_1 and f_2 elements of $O(n)$? For that we need the formal definition of the big O notation. Let $f : \mathbb{N} \to \mathbb{N}$ and $g : \mathbb{N} \to \mathbb{N}$ be functions. Then $g \in O(f)$ if

$$\exists M \in \mathbb{N}\ \exists F \in \mathbb{N}\ \forall n \geq M\ g(n) \leq F \cdot f(n),$$

that is, we can find a "minimal size" M and a "factor" F such that for all inputs of size n that are greater than or equal to the "minimal size," $g(n) \leq F \cdot f(n)$.

To prove that $f \in O(n)$ we need to pick M and F. Let us pick $M = 10$ and $F = 12$. Then it remains to show that

$$\forall n \geq 10\ 11n + 10 \leq 12n,$$

that is, $11n + 10 \leq 12n$ for all $n \geq 10$. Let $n \geq 10$. Then $11n + 10 \leq 11n + n = 12n$. Hence, the above is true. Similarly, we can prove that $f_1 \in O(n)$. This time we pick $M = 12$ and $F = 15$. It remains to prove that

$$\forall n \geq 12\ 14n + 12 \leq 15n,$$

which is trivially true. To prove that $f_2 \in O(n)$, we pick $M = 14$ and $F = 18$. In this case, it remains to prove that

$$\forall n \geq 14\ 17n + 14 \leq 18n,$$

which is obviously true.

Let us consider another Java snippet.

```
1  /**
2     Sorts the given array.
3
4     @param a an array of integers.
5     @pre. a != null
6  */
7  public static void sort(int[] a)
8  {
9     for (int i = 0; i < a.length - 1; i++)
10    {
11       int min = i;
```

```
12          for (int j = i + 1; j < a.length; j++)
13          {
14             if (a[j] < a[min])
15             {
16                min = j;
17             }
18          }
19          int temp = a[i];
20          a[i] = a[min];
21          a[min] = temp;
22       }
23    }
```

Again, let us first estimate how many elementary operations are performed at each line of the above method.

line	number
9	10
11	2
12	10
14	8
16	2
19	4
20	6
21	4

Note that we do not know how often line 16 is executed. However, we do know that line 16 is executed at most

$$(n-1) + (n-2) + \cdots + 1 = \frac{n(n-1)}{2}$$

times, where n is the length of the array. An upperbound of the total number of elementary operations that is performed can be captured by a function $f : \mathbb{N} \to \mathbb{N}$ defined by

$$f(n) = 10n^2 + 16n - 26,$$

where n represents the length of the array.

To show that $f \in O(n^2)$, we pick $M = 16$ and $F = 11$. It remains to prove that

$$\forall n \geq 16 \; 10n^2 + 16n - 26 \leq 11n^2.$$

Let $n \geq 16$. Then $10n^2 + 16n - 26 \leq 10n^2 + 16n \leq 10n^2 + n^2 = 11n^2$. That concludes the proof.

For more details on the big O notation, we refer the reader to, for example, [1].

References

[1] Kenneth Rosen. *Discrete Mathematics and Its Applications.* McGraw-Hill, sixth edition, 2007.