1 Notation

\(\mathbb{N}\) is the set of natural numbers (positive integers) \(\{0, 1, 2, 3, \ldots\}\). \(\mathbb{R}\) is the set of real numbers. \(\mathbb{R}_{\geq 0}\) is the set of real numbers greater than or equal to zero.

Suppose that \(f(n) := \sqrt{n}\) (using := means we are making a definition). We also write \(f : n \mapsto \sqrt{n}\). Then this defines a function from \(\mathbb{N}\) to \(\mathbb{R}\) (which we write as \(f : \mathbb{N} \to \mathbb{R}\)). \(f\) is also denoted by

\[n \mapsto \sqrt{n}\]

or by

\[\lambda n. \sqrt{n}\]

and is often just written \(\sqrt{n}\). All of these notations denote the same function.

So, for example, \(n\) denotes the identity function \(n \mapsto n\), \(\log n\) denotes the function \(n \mapsto \log n\), which is the same as \(\lambda n. \log n\) and so on.

Suppose that \(F\) is a function taking elements in some set \(A\) to boolean expressions. For example, if \(A = \mathbb{N}\),

\[F : n \mapsto n = 3\]

so \(F(2)\) is the boolean valued expression \(2 = 3\), which evaluates to false. Then

\[\exists x \in A \ F(x)\]

is a boolean valued expression whose meaning is “there exists \(x \in A\) such that \(F(x)\) is true” (in our example, this is true, since there exists an integer which is equal to 3).

\[\forall x \in A \ F(x)\]

is a boolean valued expression whose meaning is “for all \(x \in A\), \(F(x)\) is true”, which is false in our example, since not all integers are equal to 3.

If \(f, g : A \to \mathbb{R}\) for some set \(A\) then we define a new function from these as follows:

\[f + g : x \mapsto f(x) + g(x)\]

and if \(a \in \mathbb{R}\) we define

\[af : x \mapsto af(x)\]

so, for example, if \(b \in \mathbb{R}\) and \(x \in A\) then \((af + bg)(x) = af(x) + bg(x)\).

2 Big \(O\) Notation

2.1 Formal Definition

This is the formal definition: given \(f : \mathbb{N} \to \mathbb{R}\),

\[O(f) := \{g : \mathbb{N} \to \mathbb{R} | \exists c \in \mathbb{R}_{>0} \exists n_0 \in \mathbb{N} \forall n \geq n_0 | g(n) | \leq c | f(n) | \}\]
In words, this is “the set of functions $g$ mapping $\mathbb{N}$ to $\mathbb{R}$ for which there exists a real positive number $c$ and a natural number $n_0$ such that for all $n \geq n_0$, $|g(n)| \leq c|f(n)|$”. Intuitively, $O(f)$ is the set of functions $\mathbb{N} \to \mathbb{R}$ eventually bounded by $f$ to within a constant factor.

For example, define

$$f(n) := \begin{cases} 0 & \text{if } n < 100 \\ \frac{1}{2}(-1)^n n & \text{else} \end{cases}$$

Taking $c = 2$ and $n_0 = 100$, it is true that $\forall n \geq n_0 \quad n \leq c|f(n)|$, so that $n \in O(f)$ by definition.

### 2.2 Informal Notation

If $F$ is a function taking functions $\mathbb{N} \to \mathbb{R}$ to boolean valued expressions and $f : \mathbb{N} \to \mathbb{R}$ then $F(O(f))$ is informal notation (shorthand) for

$$\exists h \in O(f) \quad F(h)$$

For example, $\sum_{i=0}^{n} i^2 = \frac{1}{6}n^3 + O(n^2)$ is shorthand for

$$\exists h \in O(n^2) \quad \sum_{i=0}^{n} i = \frac{1}{6}n^3 + h(n)$$

As another example, $f = O(g)$ is shorthand for $\exists h \in O(g) \quad f = h$, which is logically equivalent to $f \in O(g)$. The shorthand is perhaps misleading because $O(g)$ is a set of functions, whereas $f$ is a single function. When combined with intuition, shorthand can be quite powerful, but don’t let it lure you into accepting flawed reasoning. If you don’t have that intuition yet, or you’re unsure, it’s always safer to go back to the formal definitions.

### 2.3 Example

Let $a, b \in \mathbb{R}_{\geq 0}$ and $f, g : \mathbb{N} \to \mathbb{R}_{\geq 0}$. We will show that

$$h = O(af + bg) \implies h = O(f + g)$$

This is shorthand for

$$h \in O(af + bg) \implies h \in O(f + g)$$

To prove this, suppose that $h \in O(af + bg)$. Then, by the definition, there exists $c$ and $n_0$ such that for all $n \geq n_0$,

$$|h(n)| \leq c|af(n)+bg(n)| = c(af(n)+bg(n)) \leq cm(f(n)+g(n)) = cm|(f(n)+g(n))|$$

where $m = \max(a,b)$. Thus, for all $n \geq n_0$, $|h(n)| \leq cm|(f + g)(n)|$. By the definition, $h \in O(f + g)$.
3 Induction

Let

\[ T(n) := \begin{cases} 
1 & \text{if } n < 4 \\
T(\lfloor n/4 \rfloor) + 1 & \text{else}
\end{cases} \]

We show by induction that \( a \log_2(n) + 1 \) bounds \( T(n) \) (we’ll choose \( a \geq 0 \) retrospectively). We check this is true for \( n = 1, 2, 3 \) by explicit computation. Then fix \( n \geq 4 \) and assume \( T(k) \leq a \log_2(k) + 1 \) is true for \( 0 < k < n \) (this is the induction hypothesis in strong induction). Then

\[
T(n) = T(\lfloor n/4 \rfloor) + 1 \quad \text{by definition}
\]
\[
\leq a \log_2(\lfloor n/4 \rfloor) + 2 \quad \text{by assumption}
\]
\[
\leq a \log_2(n/4) + 2 \quad \text{since } \log_2(x) \text{ is monotone and }
\]
\[
0 < \lfloor n/4 \rfloor \leq n/4
\]
\[
= a(\log_2(n) - 2) + 2
\]
\[
= a \log_2(n) + 1 \quad (†)
\]

(†) is true if we take \( a = \frac{1}{2} \). So we have shown \( T(n) \leq a \log_2(n) + 1 \). Showing that \( 2n \log(n) + 1 \) bounds the \( T \) in question 3.5 involves very similar reasoning, but it’s a little more work because of the extra terms.

4 Tasks

\textbf{task 0)} Show that question 3.4 is false if we allow the \( a \)s and \( g \)s to have negative values.

\textbf{task 1)} prove that \( n^2 \) is not in \( O(n) \)

\textbf{task 2)} prove that \( n \) is not in \( O(\log(n)) \)

\textbf{task 3)} Let \( a, b \in \mathbb{R} \) and \( f : \mathbb{N} \to \mathbb{R} \). Prove that if \( g_1 \in O(f) \) and \( g_2 \in O(f) \) then \( ag_1 + bg_2 \in O(f) \). Using this, prove that \( \frac{1}{2}n - 100 \log(n) + 7 \notin O(\log(n)) \)

\textbf{task 4)} write an ML \texttt{change} function that returns lists of pairs i.e. \texttt{change : int -> int list -> (int * int) list list}

where the first element in the pair is the multiplicity and the second is the coin value. For example, \texttt{change 10 [2,3];} evaluates to \texttt{[[(2,2),(2,3)],[(5,2)]]}.

Use \texttt{List.length} to check that there are 4366 ways of expressing 99 using 50,20,10,5,2,1 coins, as stated on page 55 of the notes.

\textbf{task 5)} Answer question 5.5

\textbf{task 6)} Implement tree sort in ML (cf p. 83 in the notes; \url{http://en.wikipedia.org/wiki/Tree_sort}). Test it out using a list of random numbers.