Example of `algorithms` in Action

Rogério Brito

May 29, 2005

1 Objective

The intent of this document is to serve as a simple example of how the algorithms package can be used for typesetting pseudo-code.

It shows one recommended way of achieving this, but, of course, you are free to use the package the way it suits you best. In particular, the algorithmic package is used with the option `noend` (so that things like **end if**, **end for** etc clauses aren't printed), saving vertical space, which is convenient when submitting papers to journals.

To compensate for the lack of the "end" clauses, I have increased the indentation of the statements from the default length of 1.0em (roughly the length the letter "m") to 2.0em. This is specially important for long algorithms, with many nested constructions, so that the reader of your algorithm doesn't loose track of its structure.

Algorithm 1 FACTORIAL(n)

Require: An integer $n \geq 0$.
Ensure: The value of $n!$.

```
1: if n = 0 then
2:     return 1
3: else
4:     return n · FACTORIAL(n − 1)
```

2 Hints for Typesetting Algorithms

Here are some short hints on typesetting algorithms:

- Don't overcomment your pseudo-code. If you feel that you need to comment too much, then you are probably doing something wrong: you should probably detail the inner workings of the algorithm in regular text rather than in the pseudo-code;

- Similarly, don't regard pseudo-code as a low-level programming language: *don't pollute your algorithms* with punctuation marks like semi-colons, which are necessary in C, C++ and Java, but not in pseudo-code. Remember: your readers *are not* compilers;
- Always document what the algorithm receives as an input and what it returns as a solution. Don't care to say in the `\REQUIRE` or in the `\ENSURE` commands *how* the algorithm does what it does. Put this in the regular text of your book/paper/lecture notes;
- If you feel that your pseudo-code is getting too big, just break it into sub-algorithms, perhaps abstracting some tasks. Your readers will probably thank you.

Of course, you should follow those hints with common sense. Well, anything should be done with common sense.

3 Internals

Just as a starting point for you to typeset your algorithms with the `algorithms` package, the output of the Algorithm 1 cited in the previous section was generated by the following sequence of commands:

```
\algsetup{indent=2em}
\newcommand{\factorial}{\ensuremath{\mbox{\sc Factorial}}}

\begin{algorithm}[h!]
  \caption{$\factorial(n)$}\label{alg:factorial}
  \begin{algorithmic}[1]

    \REQUIRE An integer $n \geq 0$.
    \ENSURE The value of $n!$.

    \medskip

    \IF {$n = 0$}
       \RETURN $1$
    \ELSE
       \RETURN $n \cdot \factorial(n-1)$
    \ENDIF
  \end{algorithmic}
\end{algorithm}
```