(algorithms)

Richard C. Waters, editor

Mitsubishi Electric Research Laboratories
201 Broadway
Cambridge MA 02139
Dick@MERL.COM

This issue's Algorithms article discusses a minor but entertaining question that has been a source of argument between Lisp programmers for decades. Suppose that you are performing an iterative computation and wish to create a list of values in the order that they are computed. An easy thing to do is to initialize a list to `nil`, **`push`** each value onto the list as it is computed, and then **`nreverse`** the list before returning it, to get the values into the correct order. Alternatively, you can write more complex code that enters the values in the list in the correct order in the first place and avoids having to call **`nreverse`**. The question is, is this **`nreverse`**-avoiding code better or not?

Over the years, many people have insisted that the **`nreverse`**-avoiding approach is much faster and therefore better. Others have insisted that using **`nreverse`** is faster and since it is simpler, is therefore doubly better.

The following article argues that as a practical matter, using **`nreverse`** is probably a bit faster in most Lisp implementations. However, more importantly there is no reason to believe that the speed difference either way in any Lisp implementation is large enough to get excited about. As a result, the decision between the two approaches should be made on the basis of simplicity and clarity, not speed. From this perspective, **`nreverse`** is pretty clearly better.

Richard C. Waters