

Re: how to speed up some lisp code?

Source: <http://coding.derkeiler.com/Archive/Lisp/comp.lang.lisp/2004-02/1833.html>

From: Barry Margolin (barmar_at_alum.mit.edu)

Date: 02/19/04

Date: Wed, 18 Feb 2004 18:01:24 -0500

In article <s5twu6k12gw.fsf@zeno1.math.washington.edu>, John H Palmieri <palmieri@math.washington.edu> wrote:

> *I'm using a data type I'll call a "monomial": these are sorted lists
> of non-negative integers. So far, the lists have length at most 7 or
> 8, but if I can speed things up, I'll want to go up to 20, or more.
> The integers in the lists are small: less than 200. Right now, I'm
> representing these as lists. (I played around with representing them
> as integers: the list (a b c ...) gets translated into an integer
> where a is the left-most bunch of bits, then b is the next bunch, and
> so on. I was hoping that since I could use arithmetic operations
> instead of list operations, and since I could use = instead of equal
> for comparison, it would speed things up, but it didn't. Maybe I
> didn't do it well. I don't see any advantage in using bit-vectors,
> but I haven't tried, and you could try to persuade me if you feel
> differently...)*

The important thing is declaring the types of your variables. Since all the integers are small, declare the variables to be FIXNUM. Otherwise, even arithmetic operations have to be generic, because they might have to deal with floats, bignums, rationals, or complex numbers.

Make sure you declare all the temporary variables as well. Remember, even if the arguments to a function are FIXNUM, the result might overflow, and without declarations to the contrary the implementation has to handle this. E.g. if A and B are declared to be FIXNUM, it can't assume that (+ A B) will also be FIXNUM; consider the result of (+ MOST-POSITIVE-FIXNUM 1).

--

Barry Margolin, barmar@alum.mit.edu
Arlington, MA

*** PLEASE post questions in newsgroups, not directly to me ***