

Re: how to speed up some lisp code?

Source: <http://coding.derkeiler.com/Archive/Lisp/comp.lang.lisp/2004-02/1889.html>

From: John H Palmieri (palmieri_at_math.washington.edu)

Date: 02/19/04

Date: Thu, 19 Feb 2004 14:44:13 -0800

On Feb 18 2004, Barry Margolin <barmar@alum.mit.edu> wrote:

> *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.*

To: Barry Margolin <barmar@alum.mit.edu>

Subject: Re: how to speed up some lisp code?

Bcc: palmieri@math.washington.edu

X-Face: 'mv.B*Lk2HZf}(gQV6`nM|p,xpL&Ic-@(Y*f3:)M" g\$~24P{a{\Zs<Y>Sy%(Bl([?i"VA sW@odDT:Q0Cw.f,:':PKOZw{bs0v2+)}`ulAVF{w,jDp(@OOZe2Mc5i'^S/^?NKn)`f#NnzHezZ_c7]{xSxM[NORal_[\^YN)%rM]&c4Kc|FJNxkIQcMue:h7L.h@s@j<ZDgIije

X-Draft-From: ("comp.lang.lisp" 140551)

References: <s5twu6k12gw.fsf@zeno1.math.washington.edu>

<barmar-D4AD4C.18012418022004@comcast.ash.giganews.com>

From: John H Palmieri <palmieri@math.washington.edu>

--text follows this line--

On Feb 18 2004, Barry Margolin <barmar@alum.mit.edu> wrote:

comp.lang.lisp: Re: how to speed up some lisp code?

> 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.

I tried that with fixnums, but it didn't seem to help much. That was a while ago, and I've sped up some other parts of the code, so I'll try again -- maybe now it will make more of a difference. (I think I'm doing a lot more list operations than arithmetic, which is why it wasn't much help.) Does it help to declare something as a list, or a list of fixnums?

> 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).

I think that the Allegro compiler has a setting which lets me avoid this. I'm away from the documentation now, but I think if I set various flags the right way (which I try to do), it assumes that sums of fixnums are fixnums, etc.

> --
> Barry Margolin, barmar@alum.mit.edu
> Arlington, MA
> *** PLEASE post questions in newsgroups, not directly to me ***

--
J. H. Palmieri
Dept of Mathematics, Box 354350 <mailto:palmieri@math.washington.edu>
University of Washington <http://www.math.washington.edu/~palmieri/>
Seattle, WA 98195-4350