

Re: efficiently accumulating values

Source: <http://coding.derkeiler.com/Archive/Lisp/comp.lang.lisp/2006-07/msg00273.html>

- *From:* liyer.vijay@xxxxxxxxxx
 - *Date:* 6 Jul 2006 20:14:05 -0700
-

Alexander Schmolck wrote:

liyer.vijay@xxxxxxxxxx writes:

liyer.vijay@xxxxxxxxxx wrote:
[snip my own stuff]

Hi,

I found the reference in PCL (Peter, if you're reading this, thanks) to VECTOR-PUSH-EXTEND which dramatically reduces time to 0.346 seconds. Here is the program:

```
(defun find-words (graph)
  "Given GRAPH find all possible words."
  (destructuring-bind (m n) (array-dimensions graph)
    (loop with result = (make-array 0 :adjustable t :fill-pointer 0)
      as i from 0 to (1- m)
      do (loop as j from 0 to (1- n)
          do (loop as word in (get-words graph i j)
              if (and (zerop i) (zerop j))
              do (vector-push-extend word result)))
          finally (return (coerce result '(or cons list))))))
```

However, I'd still like to know if there's a better way, and since I didn't mention it in my last post, any suggestions, comments, criticisms on coding style, programming are greatly appreciated.

I was about to suggest vector-push-extend. I have only glanced over your code but I'd think the next thing you could do is pass the vector in to get-words to be destructively modified there.

Hi,

Alexander, I've tested all the other suggestions except yours, I'll do

Re: efficiently accumulating values

that next :-)

Thanks for all the comments, I did some better timing tests with the various suggestions.

Since single calls to CL:TIME do not give consistent results, and I didn't like SB-PROFILE:PROFILE I wrote the following macros.

```
(in-package :boggle)
```

```
(shadow '#:time)
```

```
(defmacro time (form &optional (run 50))
```

```
"Execute FORM RUN times and give the average time taken in seconds."
```

```
(let ((runval (gensym))
```

```
(start (gensym))
```

```
(stop (gensym))
```

```
(sum (gensym)))
```

```
`(loop with ,runval = ,run and ,start and ,stop
```

```
repeat ,runval
```

```
do (setq ,start (get-internal-real-time))
```

```
do ,form
```

```
do (setq ,stop (get-internal-real-time))
```

```
summing (- ,stop ,start) into ,sum
```

```
finally (return (float (/ ,sum ,runval 1000))))))
```

```
(defmacro run (&rest defuns)
```

```
(let ((stack (gensym)))
```

```
`(let ((,stack '()))
```

```
,@(loop
```