
Subject: prime factors (was Re: All day FFT....)
Posted by [Brian Jackel](#) on Fri, 08 Feb 2002 20:47:36 GMT
[View Forum Message](#) <> [Reply to Message](#)

Hi Bob, Martin

Here's my contribution to the prime number wars.
Not recursive, I'm afraid.

A single benchmarks shows it as being 10% faster
than Martin's code. Your mileage may vary.

```
IDL> num= 124123L*7L*3L*5L & st= systime(1) & for indx=0,9999 do dummy=
ifactors(num) & print,systime(1)-st
6.6720001
IDL> num= 124123L*7L*3L*5L & st= systime(1) & for indx=0,9999 do dummy=
prime_factors(num) & print,systime(1)-st
5.5940000
```

;All bug reports cheerfully accepted

;Brian Jackel

;bjackel@phys.ucalgary.ca

;+

; NAME: Prime_Factors

;

; PURPOSE: This function accepts a single (scalar) value, and returns a

; vector containing all the prime factors of that value. This
is

; useful for seeing if FFT's will be fast, or reducing
fractions.

;

; CATEGORY: Math

;

; CALLING SEQUENCE: Result= PRIME_FACTORS(Value)

```

;
; INPUTS: Value  a scalar byte, integer, or long integer value.
;
; KEYWORDS: SORT  if set, then the result will be sorted in increasing
;               order. Otherwise, factors may be scattered in no
;               particular order.
;
; UNIQUE  if set, then the result will only contain one of each
;          factor ie. multiple occurances will be removed. This
;          is done using the library function UNIQ. Note that
;          this requires SORTing.
;
; OUTPUTS: The result of this function will be a vector containing all
;          prime factors of the input value. If the input value is
;          prime, then the result will have only one element, equal
;          to the input.
;
; RESTRICTIONS: Fastest if no prime factor is greater than 97, quite
slow
;               after that, approximately order(sqrt(N)), where N is the
;               largest prime factor.
;
;               Only works for positive numbers.
;
; PROCEDURE: Do a fast search for all primes up to 97, then slowly loop

```

; through the rest (if any).

;

; EXAMPLES:

;

;IDL> test=PRIME_FACTORS(1L) & PRINT,test

; 1

;

;IDL> test=PRIME_FACTORS(5414145L) & PRINT,test

; 3 5 11 19 11 157

;

;IDL> test=PRIME_FACTORS(5414147L) & PRINT,test

; 5414147

;

; MODIFICATION HISTORY:

; Written February 14 1995, Brian Jackel, University of Western Ontario

; September 3 1995 Bjj Increased the list of primes to 97, improved
the dumb

; loop considerably: $O(n)$ to $O(\sqrt{n}/2)$

; Screened input better, added /SORT and /UNIQUE

;-

FUNCTION PRIME_FACTORS,value,SORT=sort,UNIQUE=unique

```
IF (N_PARAMS() LT 1) THEN MESSAGE,"Error- this function requires a  
scalar input parameter"
```

```
IF (N_ELEMENTS(value) GT 1) THEN MESSAGE,"Error- this function only  
accepts scalar input"
```

```
IF (value EQ 0) THEN BEGIN
```

```
    MESSAGE,'Warning- input value was zero ',/INFORMATIONAL
```

```
    RETURN,[0L]
```

```
ENDIF
```

```
IF (value LT 0) THEN MESSAGE,'Warning- input value was  
negative',/INFORMATIONAL
```

```
IF ((value - LONG(value)) NE 0) THEN BEGIN
```

```
    MESSAGE,"Warning- Value should be an integer, but is  
"+STRING(value),/INFORMATIONAL
```

```
    RETURN,[1L]
```

```
ENDIF
```

```
work= ABS(value)    ;make a working copy
```

```
factors= value/work ;1 (or maybe -1) is always a factor, albeit a  
trivial one
```

```
;
```

```
;For this first bit we just have a list of prime numbers (up to 97),
```

```
;and check if "work" is divisible by any of them. If so, make a note
```

```
;of it, and divide "work" by the appropriate factors. Repeat until
```

```
;"work" is no longer divisible by anything in the list. This either
```

```
;means that we've got all the factors, or the remaining ones are
```

;larger than 97.

;

some_primes=

[2,3,5,7,11,13,17,19,23,29,31,37,41,43,47,53,59,61,67,71,73, 79,83,89,97]

REPEAT BEGIN

w= WHERE((work MOD some_primes) EQ 0 ,nw) ;see if any
thing in the list matches

IF (nw GT 0) THEN BEGIN

some_primes= some_primes(w) ;throw away
everything but the prime factors

factors= [factors,some_primes]

temp= some_primes(0)

FOR indx=1,nw-1 DO temp=temp*some_primes(indx)

work= work/temp ;divide the
working value by all prime factors

ENDIF

ENDREP UNTIL (nw EQ 0)

;

;At this point we've found all the prime factors up til 97.

;Not having any better idea, I'll just keep trying to divide "work"

;by larger and larger numbers, until I've removed all the factors,

;or the Universe ends.

;

;Really, we should only be trying to divide by prime numbers, but if

```

;I had a quick way to test the primeness of numbers I'd be rich and
;famous by now.
;
;Note, however, that even numbers aren't prime, so we can halve the
;search space by concentrating only on odd numbers. We really should
;also ignore anything that ends in 5, but that actually slows things
;down a bit. Ideally we would use a base 6 number system, which would
;allow us to ignore 2/3 of the numbers instead of 1/2 or 6/10.
;
;Also, we can only have to search up to SQRT(work), which changes the
;time from O(n) to O(sqrt(n)), a significant improvement.

```

```

upper_limit= FIX( SQRT(work) + 1 ) ;highest number to check, about
SQRT(2^31)=45000, so worst case should still be pretty fast

```

```

current_try= 101L

```

```

WHILE (current_try LT upper_limit) DO BEGIN

```

```

  IF ((work MOD current_try) EQ 0) THEN BEGIN

```

```

    nfactors= 0

```

```

    REPEAT BEGIN

```

```

      work= work / current_try

```

```

      nfactors= nfactors+1

```

```

    ENDREP UNTIL (work MOD current_try) NE 0

```

```

    factors= [factors, REPLICATE(current_try,nfactors)]

```

```
upper_limit= FIX( SQRT(work) + 1 )
```

```
ENDIF
```

```
current_try= current_try + 2L
```

```
ENDWHILE
```

```
;At this point, if "work" isn't 1, then it must be prime.
```

```
;Also, throw away the first element in "factors" (was a
```

```
; dummy 1) unless the input value was simply 1.
```

```
;
```

```
IF (work NE 1) THEN factors= [factors,work] ;anything left at this  
point must be a prime
```

```
IF (value NE 1) THEN factors= factors(1:*)
```

```
IF KEYWORD_SET(SORT) OR KEYWORD_SET(UNIQUE) THEN factors= factors(  
SORT(factors) )
```

```
IF KEYWORD_SET(UNIQUE) THEN factors= factors( UNIQ(factors) )
```

```
RETURN,factors
```

```
END
```
