ACCESSING THE PRIMES AND EXPONENTS FROM THE TI-89 "FACTOR" COMMAND

Joseph Fadyn
Southern Polytechnic State University
1100 South Marietta Parkway
Marietta, Georgia 30060
jfadyn@spsu.edu

INTRODUCTION

We consider the problem of factoring positive integers using the TI-89. The TI-89 has a "factor" command in the "algebra" menu which will factor moderate sized positive integers. However, it does not seem to be easy to access the primes and exponents produced by that command individually. Thus, although one can "see" the primes and exponents produced by the "factor" command on the HOME screen, the problem of how to access those numbers during the execution of another program (say) is not trivial. This is unfortunate because many applications in number theory use the prime power decomposition of a positive integer, and they require that the individual primes and exponents be available. For example, knowing the individual primes and exponents in this decomposition is necessary when finding the Euler phi function and the sum and number of divisors of a positive integer. The individual primes and exponents are also of great value when working with Pythagorean triples (or Pythagorean n-tuples) and when solving polynomial congruences in which the modulus is not a prime. Of course writing your own program which by-passes the "factor" command altogether is always an option. However, the "factor" command is quite fast, and writing a naive program which relies on trial division by primes can result in a program which runs very slowly if the integer to be factored is a product of two or more large primes. I have, therefore, opted to write a program which uses the output of the "factor" command. This program is essentially a "utility" program. It can be utilized by other programs which find it necessary to have access to the individual primes and exponents in the prime power decomposition of a positive integer.

THE PROGRAM IDEA

Most of the rest of this paper contains the program listing and a simple application program which call our main result. Although I will not do a line by line analysis of the main program, let me explain some ideas which are used in the program. The program analyzes the "factor" command output of the TI-89 by converting it to a string. For example, the TI-89 command: string (factor (720)) produces the output string: "2 ^ 4 * 3 ^ 2 * 5" . This represents, of course, the factorization: $720 = 2^4 * 3^2 * 5$. A brief inspection of this string shows that, roughly speaking, exponents appear between ^ and * , and bases (primes) appear between * and ^ in the string. For example the exponent 4 on the base 2 appears between ^ and * as : ^ 4 * in the string. As another example, the base (prime) 3 appears between * and ^ as: * 3 ^ in the string. In addition, the first number in the string (2) is a base (prime), and the last number (5), since it is not followed by ^, is also a base (prime). Of course this is only a casual analysis, and there are a quite a number of special cases to consider. For example, consider the command: string (factor (5005)) which produces the output string: "5 * 7 * 11 * 13" . Now, here is the listing for the program.

THE PROGRAM

```
factors(n)
Prgm
ClrIO
DelVar p, e, s, r, u, v
If n =1 Then
{ } → e
0 → j
{ } → p
Goto stp1
EndIf
If isPrime(n) Then
n → p[1]
1 → e[1]
1 → j
Goto stp1
EndIf
string(factor(n)) → s
dim(s) → d
If inString(s ,"*" ,1) = 0 Then
inString(s, "^" ,1) → t
expr(mid(s, t+1, d)) → e[1]
expr(mid(s, 1, t-1)) → p[1]
1 → j
Goto stp1
End If
If inString(s, "^" ,1) = 0 Then
inString(s, "*" ,1)→ r[1]
expr(mid(s, 1, r[1] - 1)) → p[1]
1 → e[1]
1 → j
EndIf
r[1] + 1 → c
2 → i
If inString(s, "^" ,1) = 0 Then
While inString(s, "*" ,c) ≠ 0
inString(s,"*",c) → r[i]
```

```
expr(mid(s, r[i-1] + 1, r[i] – 1 - r[i-1])) → p[i]
1 → e[i]
r[i] + 1  → c
i + 1 → i
j + 1 → j
EndWhile
expr(mid(s, r[i-1] + 1, d - r[i-1])) → p[i]
1 → e[i]
j + 1 → j
Goto stp1
EndIf
DelVar r
1 → j : 1 → k : 1 → o
For i , 1, d
If mid (s, i, 1 ) = "*" Then
i → r[j]
j + 1 → j
EndIf
If mid (s, i, 1) = "^" Then
i → u[k]
k + 1 → k
EndIf
If mid( s, i, 1) = "*" Then
i → v[o]
o + 1 → o
ElseIf  mid( s, i, 1) = "^" Then
i → v[o]
o+ 1 →  o
EndIf
EndFor
DelVar i
expr (mid ( s, 1, v[1] – 1)) →  p[1]
2 →  c
For i , 1, o – 2
If mid( s, v[i] , 1 ) = "*" Then
expr( mid(s, v[i] + 1, v[i + 1] – v[i] –1)) → p[c]
c + 1 → c
EndIf
EndFor
If mid (s, v[o – 1], 1)="*" Then
expr( mid(s, v[o – 1] +1, d – v[o – 1])) → p[c]
EndIf
DelVar e
If mid(s , v[1], 1) = "*" Then
1 → e[1]
```

```
Else
expr( mid(x, v[1] + 1, v[2] – v[1] – 1) → e[1]
EndIf
2 → c
For I, 2, o – 2
If mid(s , v[i], 1 ) = "^" and mid(s, v[i + 1], 1) = "*" Then
Expr( mid( s, v[i] + 1, v[i + 1] – v[i] – 1)) → e[c]
c + 1 → c
ElseIf mid(s, v[ i – 1], 1) = "*" and mid(s, v[i] , 1) = "*" Then
1 → e[c]
c + 1 → c
EndIf
EndFor
If mid(s, v[o – 2], 1) = "*" and mid(s, v[o – 1], 1) = "*" Then
1→ e[c]
c + 1 → c
EndIf
If mid( s, v[o – 1] + 1, 1) = "^" Then
expr( mid(s, v[o – 1] + 1, d – v[o – 1])) → e[c]
Else
1 → e[c]
EndIf
Lbl stp1
EndPrgm
```

EXAMPLE

First, note that 72456 = $2^3 * 3 * 3019$. If you execute factors(72456), no output is produced. This is in accordance with the philosophy that this program will be called by other programs, so that output would not necessarily be desired. The primes in the factorization are stored in (sequence) variable p: p = {2, 3, 3019} and exponents are stored in (sequence) variable e: e= {3, 1, 1}. Although the TI-89 displays these results in set notation, the order of the elements is actually preserved. The number of primes (and exponents) in the factorization of 72456 can be found using the command: dim(p). In this case, dim(p) = 3. To access the individual primes and exponents we use the notation p[i] or e[i] for i between 1 and 3. For example, p[3] = 3019, and e[1] = 3.

APPLICATION

The following is a simple program which efficiently produces for a positive integer n, the Euler phi function $\varphi(n)$, the number of divisors function $\tau(n)$, and the sum of the divisors function $\sigma(n)$ by calling our main program factors(n) to produce the prime power factorization of n. Our program uses the following well known formulas for these quantities, assuming that $n = p_1^{a_1} \cdot p_2^{a_2} \cdot \ldots \cdot p_k^{a_k}$:

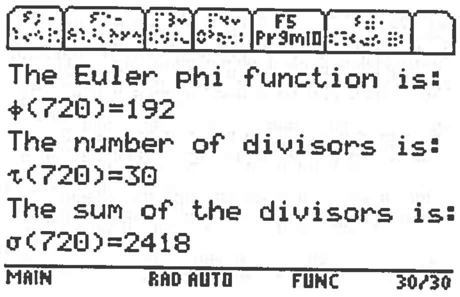
$$\phi(n) = n \cdot \left(1 - \frac{1}{p_1}\right) \cdot \left(1 - \frac{1}{p_2}\right) \cdot \ldots \cdot \left(1 - \frac{1}{p_k}\right)$$

$$\tau(n) = \prod_{j=1}^{k} (a_j + 1)$$

$$\sigma(n) = \prod_{j=1}^{k} \frac{\left(p_j^{a_j+1} - 1\right)}{(p_j - 1)}$$

```
phitausi()
Prgm
ClrIO
DelVar p, e, ph, ta, si, s, r, u, v, φ, τ, σ
Disp "Enter Positive Integer n"
Input n
factors(n)
dim(p) → j
n * ∏ (1 – 1 / (p[i] ) , i, 1, j)  → ph
Disp "The Euler phi function is:"
Disp "φ(" &string(n) &")=" &string(ph)
Pause
∏ (e[i] + 1, i, 1, j) → ta
Disp "The number of divisors is:"
Disp "τ(" &string(n) &")=" &string(ta)
Pause
∏ ( (p[i]^(e[i] + 1) – 1)/(p[i] – 1), i ,1 ,j) → si
Disp "The sum of the divisors is:"
Disp "σ(" &string(n) &")=" &string(si)
EndPrgm
```

Here is a screen capture of output corresponding to input n = 720:



```
The Euler phi function is:
φ(720)=192
The number of divisors is:
τ(720)=30
The sum of the divisors is:
σ(720)=2418
MAIN          RAD AUTO      FUNC      30/30
```