Henry Ibstedt
Glimminge 2036
280 60 Broby
Sweden

Smarandache Function Journal, Vol. 2-3, 1993, pp. 38-41.

The Florentin Smarandache Function S(n)

PCW February 1993

Problem (0)
A program *SMARAND* has been designed to generate S(n) up to a preset limit N (N up to 1000000 has been used in some applications). The program requires an input of prime number up to $\sqrt{N}$. Initially the program caculates $S(p_i^k)=D(i,k)$ for all primes p_i and all exponents k needed to reach the preset limit. It then proceeds to factorize consecutive values of n. If n is prime then S(n)=n otherwise p_i^k is replaced by D(i,k) whenever k>1. The largest component in the resulting array is determined and is equal to S(n). Slighly different versions of the program has been used depending on the application. Up to n=32000 both S(n) and D(i,k) were registed on files. The values of S(n) for $n \leq 4800$ were listed with the help of a program *SN_TAB* and the values of $S(p_i^k)$ were listed for $p_i \leq 73$, $k \leq 75$ with the help of a program *SNP_TAB*.

Problem (i)
(a) No closed expression for S(n) has been found
(b) No asymptotic expression for S(n) has been found. The behaviour of S(n) for $n \leq 32000$ has been graphically displayed using a program *SN_DISTR*.

Problem (ii)
S(n)/(n-m), $m \neq 0$ is equivalent to

$$m=n+kS(n),\ k \text{ integer} \quad \ldots. (1)$$

Let us assume that m is a given prime p. From the definition of S(n) it is evident that for every n there exists a prime q such that q/n (or n=lq, l integer) and S(n)=jq (i integer). We can therefore write (1) in the form

$$p=q(1+kj) \quad \ldots (2)$$

To find solutions to (1) when m is a prime p it is therefore sufficient to chose n as a multiple of p which fills the condition 1+kj=1.

In practice (as far as I have found) this means excluding from n those multiples of m(=p) which are divisible by primes larger than p and also cases where n-m has a different parity from S(n) as for example (n,m,S(n))=(5054,19,38) is not a solution while (2527,29,38) is a

solution.

When m is not a prime let p be the largest prime such that p/m, i.e. m=rp. Solutions to (1) will then be found when n is a multipe of p for which the GCD of n and S(n) is rp.

The above conditions are sufficient but may not be necessary. Lists of solutions are however easily obtained (not included) by looking for solutions to (n-m) mod S(n) = 0.

Problem (iii)
A number of solutions to $S(x^n)+S(y^n)=S(z^n)$ has been obtained and listed for n=3,5,7 and 11. The program *SMAR_iii* uses only Smarandache function values of the type $S(p_i^k)$ which had first been sorted in ascending order using a program *SNP_SORT*.

Problem (iv)
A program *SMAR_iv* has been designed to find solutions to the equation $S(k^n)^i=S(n^k)$ but no non-trivial solutions were found in the selected search area $n \leq 8000$.

Problem (v)
In a first attempt values saved on file up to 32000 were in a program *SMAR_v* to calculate the sums $Z(n)=1+1/S(2)+1/S(3)+ \ldots +1/S(n)$ for n= 800, 1600, 2400, ... 32000. These sums were used to study the behaviour of Z(n)-T(n) for various functions T(n):
T(n)=log(S(n)) gave a curve "parallel" to Z(n).
T(n)=log(largest prime<n) gave a similar result.
$T(n)=1+1/2^a+1/3^a+ \ldots +1/n^a$ gave interesting results. Supplemented with a linear term a "nearly straight horizontal" line was obtained.
To see if this holds for larger values the exercise was repeated for $n \leq 1000000$. Computer files to store S(n) is now out of question and the generating program *SMARAND* was revised so that the partial sums Z(n) were calculated in the same program. T(n) was calculated in a separate program for vaious values of a. For a=0.5 and it was found that

$$\mathbf{1+1/S(2)+1/S(3)+ \ldots +1/S(n) - (1+1/\sqrt{2}+1/\sqrt{3}+ \ldots +1/\sqrt{n}) - (20k-58)}$$

where k= n/25000 deviates from 0 with at most 10 in the interval 1 to 1000000 (at the points of representations in the graph, 1000000 was divided in 40 interval of 25000).

Problem (vi)
Not attempted.

Equipment
Calculations were done on an dtk-computer with 486/33 Mhz processor. Programs were written in Borlands Turbo Basic. Printouts were done on an HP IIP Laser printer. Some graphs were done on an HP Paintjet. The run time to calculate S(n) up to 1000000 was 2 h 50 m. The initial calculation up 32767 took 198 s.

```
'SMARAND, H. Ibstedt, 930320

'The Smarandache function S(n) calculated by comparing largest prime      and S(P^A). The
values of S(n) are calculated and registered in a file SN.DAT up to n = 32000. The calculation
goes further in other applications.

DEFLNG A-S
CLS :T=TIMER
DIM P(168),D(168,75),K(168),L(168)
OPEN "PA" FOR INPUT AS #1
FOR I=1 TO 168 :INPUT #1,P(I) :NEXT :CLOSE #1

'This part of the program calculates S(P(I)^A) and saves the result in the array D(I,A), P(I) is the
Ith prime number. The routine uses the fact that D(I,A)<=P(I)*A for a downward search for the
value of D(I,A). This calculation goes beyond what is required to calculate S(n) up to n =
32000.

FOR I=1 TO 42
A=2 :P=P(I) :D(I,1)=P
WHILE A<76
C=0 :N=0
 L:
C=C+1
N=N+P
IF C>=A THEN D(I,A)=N :GOTO LWEND
PP=P*P
 L1:
IF N-PP*INT(N/PP)=0 THEN C=C+1 :PP=PP*P :GOTO L1
IF C>=A THEN D(I,A)=N :GOTO LWEND :ELSE L
 LWEND:
INCR A
WEND
NEXT

'The array D(I,A) is stored in a file SNP.DAT for future use.

OPEN "SNP.DAT" FOR OUTPUT AS #2
FOR I=1 TO 42 :FOR J=1 TO 75
PRINT #2,P(I),J,D(I,J)
NEXT :NEXT :CLOSE #2

'This part of the program calculates S(N). It calls on the subroutine NFACT to express N in
prime factor form. Factors P(I)^A with A>1 are replaced by D(I,A) and placed in array L()
together with the factors P(I) of multiplicity 1. S(N) is then the largest component of L(). S(N) is
stored in a file SN.DAT.

N=1
OPEN "SN.DAT" FOR APPEND AS #3
WRITE #3,1
WHILE N<32000
INCR N :print n
'Factorize N.
```

```
GOSUB NFACT
IF K(0)>0 THEN S=P(0) :GOTO LWR

'Construct L().

 FOR I=1 TO 168 :L(I)=0 :NEXT
 C=0
 FOR I=1 TO M
 INCR C
 IF K(I)=1 THEN L(C)=P(I)
 IF K(I)>1 THEN L(C)=D(I,K(I))
 NEXT

'Find the largest value of L() and hence S(N).

 S=0
 FOR I=1 TO C
 IF L(I)>S THEN S=L(I)
 NEXT
  LWR:
 WRITE #3,S
 WEND
 CLOSE #3
 T=TIMER-T :PRINT T
 END

'Subroutine for factorization of N.

  NFACT:
 FOR I=0 TO 168 :K(I)=0 :NEXT :P(0)=0
 N1=N :M=0
 FOR I=1 TO 168
  LA:
 IF N1-P(I)*INT(N1/P(I))=0 THEN K(I)=K(I)+1 :M=I :N1=N1/P(I) :GOTO LA
 IF N1=1 THEN I=168
 NEXT
 IF N1>1 THEN P(0)=N1 :K(0)=1
 RETURN
```