CALCULATING THE SMARANDACHE FUNCTION WITHOUT FACTORISING

J. R. SUTTON
(16A Overland Road, Mumbles, SWANSEA SA3 4LP, UK)

Introduction

The usual way of calculating the Smarandache function S(n) is to factorise n, calculate S for each of the prime powers in the factorisation and use the equation

$$S(n) = \text{Max}\left(S\left(m_i^{\,p_i}\right)\right)$$

This paper presents an alternative algorithm for use when S is to be calculated for all integers up to n. The integers are synthesised by combining all the prime powers in the range up to n.

The Algorithm

The Pascal program at the end of this paper contains a procedure tabsmarand which fills a globally declared array, Smaran, with the values of S for the integers from 2 to the limit specified by a parameter. The calculation is carried out in four stages.

Powers of 2

The first stage calculates S for those powers of 2 that fall within the limit and stores them in the array Smaran at the subscript which corresponds to the value of that power of 2. At the end of this stage the array contains S for:-

2,4,8,16,32....

interspersed with zeros for all the other entries.

General case

The next stage uses succesive primes from 3 upwards. For each prime the S values of the relevant powers of the prime, and also the values of the prime powers are calculated, and stored in the arrays Smarpp and Prpwr, by the procedure tabsmarpp. This procedure is essentially the same as that in a previous paper except that:

a) the calculation stops when the last prime power exceeds the limit

and b) the prime powers are also calculated and stored.

Then for each non-zero entry in Smarand that entry is multiplied by successive powers of the prime and the S values calculated and stored in Smarand. Both of these loops terminate on reaching the limit value. Finally the S values for the prime powers are copied into Smarand. After the prime 3 the array contains:-

2,3,4,0,3,0,4,6,0,0,4....

This process is followed for each prime up to the square root of the limit. This general case could be continued up to the limit but it is more efficient to stop at the square root and treat the larger primes as seperate cases.

Largest primes

The largest primes, those greater than half the limit, contribute only themselves, S(prime)=prime, to the array of Smarandache values.

Multiples of prime only

The intermediate case between the last two is for primes larger than the square root but smaller than half the limit. In this case no powers of the prime are needed, only multiples of those entries already in Smarand by the prime itself. The prime is then copied into the array.

The Pascal program

The main program calls tabsmarand to calculate S values then enters a loop in which two integers are input from the keyboard which specify a range of values for which the contents of the array are displayed for checking.

The program was developed and tested with Acornsoft ISO-Pascal on a BBC Master computer. The function 'time' delivers the time lapse (in centiseconds) since last reset. On a computer with a 65C12 processor running at 2MHz the following timings were obtained:-

limit	seconds
1000	6.56
2000	12.87
3000	19.19
4000	25.64
5000	31.80

In this range the times appear almost linear. It would be useful to have this confirmed or disproved on a larger, faster computer.

```
program Testsmarand(input,output);
const limit=5000;
var count,st,fin:integer;
Smaran:array[1..5001] of integer;

procedure tabsmarand(limit:integer);
var count,t,i,s,is,pp,prime,pwcount,mcount,multiple: integer;
exit: boolean;
Prpwr:array[1..12] of integer;
Smarpp:array[1..12] of integer;

function max(x,y: integer):integer;
begin
if x>y then max:=x else max:=y;
end; {max}

function invSpp(prime,smar:integer):integer;
var n,x:integer;
begin
n:=0;
x:=smar;
repeat
x:=x div prime;
n:=n+x;
until x<prime;
invSpp:=n;
end; {invSpp}

procedure tabsmarpp(prime,limit:integer);
var i,s,is,pp:integer;
exit:boolean;
begin
exit:=false;
pp:=1;
i:=1;
is:=1;
s:=prime;
repeat
repeat
Smarpp[i]:=s;
pp:=pp*prime;
Prpwr[i]:=pp;
i:=i+1;
if pp>limit then exit:=true;
until (i>is) or exit;
s:=s+prime;
is:=invSpp(prime,s);
until exit;
end; {tabsmarpp}
```

```
begin writeln('Calculate Smarandache function for all integers up to
',limit);
for count:=1 to limit do Smaran[count]:=0;
Smaran[limit+1]:=limit+1;
t:=time;
     {powers of 2}
s:=2;
i:=1;
is:=1;
pp:=1;
exit:=false;
repeat
repeat
pp:=pp*2;
Smaran[pp]:=s;
i:=i+1;
if 2*pp>limit then exit:=true;
until (i>is) or exit;
s:=s+2;
is:=invSpp(2,s);
until exit;
     {general case}
prime:=3;
repeat
tabsmarpp(prime,limit);
mcount:=1;
repeat
pwcount:=1;
multiple:=mcount*prime;
repeat
if multiple<=limit then
     if Smaran[multiple]=0 then
          Smaran[multiple]:=max(Smaran[mcount],Smarpp[pwcount]);
pwcount:=pwcount+1;
multiple:=mcount*Prpwr[pwcount];
until multiple>limit;
repeat
mcount:=mcount+1;
until Smaran[mcount]<>0;
until mcount*prime>limit;
pwcount:=1;
repeat
Smaran[Prpwr[pwcount]]:=Smarpp[pwcount];
pwcount:=pwcount+1;
until Prpwr[pwcount]>limit;
repeat
prime:=prime+1;
until Smaran[prime]=0;
until prime*prime>limit;
```

```
     {multiple case}
repeat mcount:=1;
multiple:=prime;
repeat
if multiple<=limit then
     if Smaran[multiple]=0 then
          Smaran[multiple]:=max(Smaran[mcount],prime);
repeat
mcount:=mcount+1;
until Smaran[mcount]<>0;
multiple:=mcount*prime;
until multiple>limit;
Smaran[prime]:=prime;
repeat
prime:=prime+1;
until Smaran[prime]=0;
until prime>limit/2;
     {largest primes}
count:=1;
repeat
if Smaran[count]=0 then Smaran[count]:=count;
count:=count+1;
until count>limit;
writeln((time/t)/100,'seconds');
end; {tabsmarand}

begin
tabsmarand(limit);
repeat
writeln('Enter start and finish integers for display of results');
read(st,fin);
if (st>1) and (st<=limit) and (fin<=limit) then
     for count :=st to fin do writeln(count,Smaran[count]);
until fin=1;
end. {Testsmarand}
```