

Strings of Consecutive Happy Numbers

23 Feb 2008

1 Mar 2008

15 Mar 2008

21 Mar 2008

1 Sept 2008

6 Sept 2008

10 Sept 2008

12 Sept 2008

18 Sept 2008

19 Sept 2008

26 Sept 2008

9 Oct 2008

The goal here is to show that the first string of fifteen consecutive happy numbers.

2222222222319612 digit number

15 in a row $N = 77.(2222222222222220 \text{ nines}).3.(97388 \text{ nines}).3$

Note that Dr. Grundman uses S_2 but we will simply use S in the explanation, which in our program is the procedure 'onestep'.

Also note we will use a dot as the digit concatenation operator. Thus, 111112999994 could be written 11111.2.9999.94 if we wish.

First we define our procedures.

```
> 2 + 2222222222222220 + 1 + 97388 + 1; M1 := onestep(77) + 2222222222222220·81 + 32
    + 97388·81;
```

2222222222319612
 $M1 := 180000000007888355$ (1)

```
> restart;
```

```
> f := n → n2; #in case we ever want to investigate the cube of the digits, etc.
```

$f := n \rightarrow n^2$ (2)

```
> bs := 10;
```

#this is the base, in case we ever want to investigate binary or ternary or any other base.

$bs := 10$ (3)

```
> onestep := proc(n1)
```

#this is what Dr. Grundman calls $S_2(n1)$ and what we will simply call S below.

```
  local ans, n, d;
```

```
  n := n1;
```

```
  ans := 0;
```

```
  while n > 0 do
```

```
    d := n mod bs;
```

```
    ans := ans + f(d);
```

```
    n := (n-d) / bs;
```

```
  end do;
```

```
ans;  
end;
```

```
onestep := proc(n1)
```

```
  local ans, n, d;
```

```
  n := n1;
```

```
  ans := 0;
```

```
  while 0 < n do d := mod(n, bs); ans := ans + f(d); n := (n - d) / bs end do;
```

```
  ans
```

```
end proc
```

(4)

```
> happy := proc(n)
```

```
  # returns -1 if not happy, and returns the number of steps to reach 1 if it is happy
```

```
  local m, j, height;
```

```
  m := n;
```

```
  height := -1;
```

```
  for j from 1 to 100 while (m > 1 and m ≠ 4) do
```

```
    m := onestep(m);
```

```
  end do;
```

```
  if m = 1 then height := j; end if;
```

```
  height;
```

```
end;
```

```
happy := proc(n)
```

```
  local m, j, height;
```

```
  m := n;
```

```
  height := -1;
```

```
  for j to 100 while 1 < m and m <> 4 do m := onestep(m) end do;
```

```
  if m = 1 then height := j end if;
```

```
  height
```

```
end proc
```

(5)

```
>
```

The next procedure is only needed when we want to find the smallest N with $S(N) = n$ for a given n. A separate worksheet has the details on how this is constructed. The array contains the smallest N for $1 \leq n \leq 486 = 6 \cdot 81$.

```
> lowS := [1, 11, 111, 2, 12, 112, 1112, 22, 3, 13, 113, 222, 23, 123, 1123, 4, 14, 33, 133, 24,  
124, 233, 1233, 224, 5, 15, 115, 1115, 25, 125, 1125, 44, 144, 35, 135, 6, 16, 116, 1116,  
26, 45, 145, 335, 226, 36, 136, 1136, 444, 7, 17, 117, 46, 27, 127, 1127, 246, 227, 37, 137,  
1137, 56, 156, 1156, 8, 18, 118, 337, 28, 128, 356, 1356, 66, 38, 57, 157, 266, 238, 257,  
1257, 48, 9, 19, 119, 248, 29, 129, 1129, 466, 58, 39, 139, 1139, 258, 239, 1239, 448, 49,  
77, 177, 68, 168, 277, 1277, 268, 458, 59, 159, 666, 368, 259, 1259, 2666, 78, 178, 359,  
468, 69, 169, 1169, 2468, 269, 378, 577, 1577, 568, 369, 1369, 88, 188, 79, 179, 288, 469,  
279, 1279, 668, 388, 578, 379, 1379, 2388, 569, 1569, 488, 89, 189, 777, 1777, 289, 1289,  
2777, 4668, 588, 389, 579, 1579, 2588, 2389, 2579, 4488, 489, 99, 199, 688, 1688, 299,  
1299, 2688, 4588, 589, 399, 1399, 3688, 2589, 2399, 12399, 788, 499, 779, 1779, 689,  
1689, 2779, 12779, 2689, 3788, 599, 1599, 5688, 3689, 2599, 888, 1888, 789, 1789, 2888,  
4689, 699, 1699, 6688, 3888, 2699, 3789, 5779, 15779, 5689, 3699, 4888, 889, 1889, 799,  
1799, 2889, 4699, 2799, 12799, 5888, 3889, 5789, 3799, 13799, 23889, 5699, 15699,  
4889, 899, 1899, 6888, 16888, 2899, 12899, 26888, 45888, 5889, 3899, 5799, 15799,
```

25889, 23899, 25799, 7888, 4899, 999, 1999, 6889, 16889, 2999, 12999, 26889, 37888, 5899, 3999, 13999, 36889, 25899, 8888, 18888, 7889, 4999, 7799, 17799, 6899, 16899, 27799, 38888, 26899, 37889, 5999, 15999, 56889, 36899, 25999, 8889, 18889, 7899, 17899, 28889, 46899, 6999, 16999, 58888, 38889, 26999, 37899, 57799, 157799, 56899, 36999, 48889, 8899, 18899, 7999, 17999, 28899, 46999, 27999, 127999, 58889, 38899, 57899, 37999, 137999, 238899, 56999, 78888, 48899, 8999, 18999, 68889, 168889, 28999, 128999, 268889, 378888, 58899, 38999, 57999, 157999, 258899, 88888, 188888, 78889, 48999, 9999, 19999, 68899, 168899, 29999, 129999, 268899, 378889, 58999, 39999, 139999, 368899, 258999, 88889, 188889, 78899, 49999, 77999, 177999, 68999, 168999, 277999, 388889, 268999, 378899, 59999, 159999, 568899, 368999, 259999, 88899, 188899, 78999, 178999, 288899, 468999, 69999, 169999, 588889, 388899, 269999, 378999, 577999, 1577999, 568999, 369999, 488899, 88999, 188999, 79999, 179999, 288999, 469999, 279999, 1279999, 588899, 388999, 578999, 379999, 1379999, 888888, 569999, 788889, 488999, 89999, 189999, 688899, 1688899, 289999, 1289999, 2688899, 3788889, 588999, 389999, 579999, 1579999, 2588999, 888889, 1888889, 788899, 489999, 99999, 199999, 688999, 1688999, 299999, 1299999, 2688999, 3788899, 589999, 399999, 1399999, 3688999, 2589999, 888899, 1888899, 788999, 499999, 779999, 1779999, 689999, 1689999, 2779999, 3888899, 2689999, 3788999, 599999, 1599999, 5688999, 3689999, 2599999, 888999, 1888999, 789999, 1789999, 2888999, 4689999, 699999, 1699999, 5888899, 3888999, 2699999, 3789999, 5779999, 8888888, 5689999, 3699999, 4888999, 889999, 1889999, 799999, 1799999, 2889999, 4699999, 2799999, 12799999, 5888999, 3889999, 5789999, 3799999, 13799999, 8888889, 5699999, 7888899, 4889999, 899999, 1899999, 6888999, 16888999, 2899999, 12899999, 26888999, 37888899, 5889999, 3899999, 5799999, 15799999, 25889999, 8888899, 18888899, 7888899, 4899999, 9999999] :

```
> minimalN := proc(n)
  local q, r, k, ans;
  global lowS;;
  if n < 487 then ans := lowS[n];
  else
    q := iquo(n, 81, 'r');
    ans := lowS[n - (q - 5) * 81] * 10q - 5 + (10q - 5 - 1);
  end if;
  ans;
end;
```

```
minimalN := proc(n)
  local q, r, k, ans;
  global lowS;
  if n < 487 then
    ans := lowS[n]
  else
    q := iquo(n, 81, 'r'); ans := lowS[n - 81 * q + 405] * 10(q - 5) + 10(q - 5) - 1
  end if;
  ans
end proc
```

(6)

```
> minimalbigN := proc(n)
  local q, r, k, ans;
  global lowS;;
  if n < 487 then ans := lowS[n];
```

```

else
  if n < 1000 then
    q := iquo(n, 81, 'r');
    ans := lowS[n - (q - 5) · 81] · 10q-5 + (10q-5 - 1);
  else
    q := iquo(n, 81, 'r');
    ans := [ lowS[n - (q - 5) · 81], q - 5];
  end if; end if;
ans;
end;
minimalbigN := proc(n)
  local q, r, k, ans;
  global lowS;
  if n < 487 then
    ans := lowS[n]
  else
    if n < 1000 then
      q := iquo(n, 81, 'r'); ans := lowS[n - 81 * q + 405] * 10^(q - 5) + 10
      ^ (q - 5) - 1
    else
      q := iquo(n, 81, 'r'); ans := [lowS[n - 81 * q + 405], q - 5]
    end if
  end if;
ans
end proc

```

(7)

We first verify our candidate $N = 77.(2222222222222220 \text{ nines}).3.(97388 \text{ nines}).3$ works. First we check that the 7 before the carry work, then the 8 after the carry. .

```

> for j from 2 to 9 do
  print( j, happy(onestep(77) + 2222222222222220 · 81 + 32 + 97388 · 81 + j2) ) end do;

```

(8)

```

> for j from 0 to 8 do
  print( j, happy(onestep(77) + 2222222222222220 · 81 + 42 + j2) ) end do;

```



```

fini2 := false;
for i from 6 to 7 while fini2 = false do
if happy(b + 2·d2 + 1 - k·81 + onestep((c + i2) mod 100)) = -1 then fini2 := true; end
if;
end do;
if fini2 = false then print(b, c, k, d2, happy(b + 2·d2 + 1 - k·81 + onestep((c + 82)
mod 100))) end if;
end do;
end do;
end if;
end do;
end do;

```

```

> for b from 0 to 2000 do
for c from 975 to 983 do #so these have the last digit 5 carry but not any others.
fini := false;
for i from 0 to 4 while fini = false do
if happy(b + onestep(c + i2)) = -1 then fini := true; end if;
end do;
if fini = false then #now we check for last digit 5
for k from 0 to floor( $\frac{b}{81}$ ) do
for d2 from 0 to 8 do #d2 is the digit of the b part that precedes the nines that carry.
fini2 := false;
for i from 5 to 7 while fini2 = false do
if happy(b + 2·d2 + 1 - k·81 + onestep((c + i2) mod 100)) = -1 then fini2 := true; end
if;
end do;
if fini2 = false then print(b, c, k, d2, happy(b + 2·d2 + 1 - k·81 + onestep((c + 82)
mod 100))) end if;
end do;
end do;
end if;
end do;
end do;

```

1663, 978, 7, 4, 4

(11)

```

> for b from 0 to 2000 do
for c from 984 to 990 do #so these have the last digit 5 and 4 carry but not any others.
fini := false;
for i from 0 to 3 while fini = false do
if happy(b + onestep(c + i2)) = -1 then fini := true; end if;
end do;
if fini = false then #now we check for last digit 5,4
for k from 0 to floor( $\frac{b}{81}$ ) do
for d2 from 0 to 8 do
fini2 := false;
for i from 4 to 7 while fini2 = false do
if happy(b + 2·d2 + 1 - k·81 + onestep((c + i2) mod 100)) = -1 then fini2 := true; end
if;

```

```

end do;
if fini2 = false then print(b, c, k, d2, happy(b + 2·d2 + 1 - k·81 + onestep((c + 82)
  mod 100))) end if;
end do;
end do;
end if;
end do;
end do;

```

```

> for b from 0 to 2000 do
  for c from 991 to 995 do #so these have the last digit 5,4,3 carry but not any others.
    fini := false;
    for i from 0 to 2 while fini = false do
      if happy(b + onestep(c + i2)) = -1 then fini := true; end if;
    end do;
    if fini = false then #now we check for last digit 5,4,3
      for k from 0 to floor( $\frac{b}{81}$ ) do
        for d2 from 0 to 8 do
          fini2 := false;
          for i from 3 to 7 while fini2 = false do
            if happy(b + 2·d2 + 1 - k·81 + onestep((c + i2) mod 100)) = -1 then fini2 := true; end
              if;
            end do;
          if fini2 = false then print(b, c, k, d2, happy(b + 2·d2 + 1 - k·81 + onestep((c + 82)
            mod 100))) end if;
          end do;
        end do;
      end if;
    end do;
  end do;

```

1582, 994, 16, 7, -1

(12)

```

> for b from 0 to 2000 do
  for c from 996 to 998 do #so these have the last digit 5,4,3,2 carry but not any others.
    fini := false;
    for i from 0 to 1 while fini = false do
      if happy(b + onestep(c + i2)) = -1 then fini := true; end if;
    end do;
    if fini = false then #now we check for last digit 5,4,3,2
      for k from 0 to floor( $\frac{b}{81}$ ) do
        for d2 from 0 to 8 do
          fini2 := false;
          for i from 2 to 7 while fini2 = false do
            if happy(b + 2·d2 + 1 - k·81 + onestep((c + i2) mod 100)) = -1 then fini2 := true; end
              if;
            end do;
          if fini2 = false then print(b, c, k, d2, happy(b + 2·d2 + 1 - k·81 + onestep((c + 82)
            mod 100))) end if;
          end do;
        end do;
      end if;
    end do;
  end do;

```



```

for  $k$  from 0 to floor( $\frac{b}{81}$ ) do
for  $d2$  from 0 to 8 do
 $fini2 := false$ ;
for  $i$  from 7 to 9 while  $fini2 = false$  do
if  $happy(b + 2 \cdot d2 + 1 - k \cdot 81 + onestep((c + i^2) \bmod 1000)) = -1$  then  $fini2 := true$ ; end if;
end do;
if  $fini2 = false$  then  $print(b, c, k, d2, happy(b + onestep(c + 1^2)), happy(b + onestep(c)),$ 
   $minimalN(b - d2^2 - k \cdot 81) \cdot 10^{(k+4)} + d2 \cdot 10^{k+3} + 10^{k+3} - 1000 + c,$ 
   $evalf(\log_{10}(minimalN(b - d2^2 - k \cdot 81) \cdot 10^{(k+4)} + d2 \cdot 10^{k+3} + 10^{k+3} - 1000 + c)))$ 
end if;
end do;
end do;
end if;
end do;
end do;
> for  $b$  from 0 to 2000 do
for  $c$  from 964 to 974 do #so these have the last digit 9,8,7,6 carry but not any others.
 $fini := false$ ;
for  $i$  from 2 to 5 while  $fini = false$  do
if  $happy(b + onestep(c + i^2)) = -1$  then  $fini := true$ ; end if;
end do;
if  $fini = false$  then #now we check for last digit 9,8,7,6
for  $k$  from 0 to floor( $\frac{b}{81}$ ) do
for  $d2$  from 0 to 8 do
 $fini2 := false$ ;
for  $i$  from 6 to 9 while  $fini2 = false$  do
if  $happy(b + 2 \cdot d2 + 1 - k \cdot 81 + onestep((c + i^2) \bmod 100)) = -1$  then  $fini2 := true$ ; end if;
end do;
if  $fini2 = false$  then  $print(b, c, k, d2, happy(b + onestep(c + 1^2)), happy(b + onestep(c)),$ 
   $minimalN(b - d2^2 - k \cdot 81) \cdot 10^{(k+4)} + d2 \cdot 10^{k+3} + 10^{k+3} - 1000 + c,$ 
   $evalf(\log_{10}(minimalN(b - d2^2 - k \cdot 81) \cdot 10^{(k+4)} + d2 \cdot 10^{k+3} + 10^{k+3} - 1000 + c)))$ 
end if;
end do;
end do;
end if;
end do;
end do;
> for  $b$  from 0 to 2000 do
for  $c$  from 975 to 983 do #so these have the last digit 9,8,7,6,5 carry but not any others.
 $fini := false$ ;
for  $i$  from 2 to 4 while  $fini = false$  do
if  $happy(b + onestep(c + i^2)) = -1$  then  $fini := true$ ; end if;
end do;
if  $fini = false$  then #now we check for last digit 9,8,7,6,5

```



```

if happy( $b + \text{onestep}(c + i^2)$ ) = -1 then fini := true; end if;
end do;
if fini = false then #now we check for last digit 9,8,7,6,5,4,3
  for k from 0 to floor( $\frac{b}{81}$ ) do
    for d2 from 0 to 8 do
      fini2 := false;
      for i from 3 to 9 while fini2 = false do
        if happy( $b + 2 \cdot d2 + 1 - k \cdot 81 + \text{onestep}((c + i^2) \bmod 100)$ ) = -1 then fini2 := true; end
          if;
        end do;
      if fini2 = false then print( $b, c, k, d2, \text{happy}(b + \text{onestep}(c + 1^2)), \text{happy}(b + \text{onestep}(c)),$ 
        minimalN( $b - d2^2 - k \cdot 81$ )  $\cdot 10^{(k+4)} + d2 \cdot 10^{k+3} + 10^{k+3} - 1000 + c,$ 
        evalf(log10(minimalN( $b - d2^2 - k \cdot 81$ )  $\cdot 10^{(k+4)} + d2 \cdot 10^{k+3} + 10^{k+3} - 1000 + c$ )))
        end if;
      end do;
    end do;
  end do;
end if;
end do;
end do;
end do;
> for b from 0 to 2000 do
  for c from 996 to 999 do #so these have the last digit 9,8,7,6,5,4,3,2 carry.
    #fini := false;
    #for i from 3 to 4 while fini = false do
      #if happy( $b + \text{onestep}(c + i^2)$ ) = -1 then fini := true; end if;
      #end do;
      #if fini = false then #now we check for last digit 9,8,7,6,5,4,3,2
        for k from 0 to floor( $\frac{b}{81}$ ) do
          for d2 from 0 to 8 do
            fini2 := false;
            for i from 2 to 9 while fini2 = false do
              if happy( $b + 2 \cdot d2 + 1 - k \cdot 81 + \text{onestep}((c + i^2) \bmod 100)$ ) = -1 then fini2 := true; end
                if;
              end do;
            if fini2 = false then print( $b, c, k, d2, \text{happy}(b + \text{onestep}(c + 1^2)), \text{happy}(b + \text{onestep}(c)),$ 
              minimalN( $b - d2^2 - k \cdot 81$ )  $\cdot 10^{(k+4)} + d2 \cdot 10^{k+3} + 10^{k+3} - 1000 + c,$ 
              evalf(log10(minimalN( $b - d2^2 - k \cdot 81$ )  $\cdot 10^{(k+4)} + d2 \cdot 10^{k+3} + 10^{k+3} - 1000 + c$ )))
              end if;
            end do;
          end do;
        end if;
      end do;
    end do;
  end do;
  end do;
  evalf(log10(minimalN(1710)  $\cdot 1000 + 534$ ));
  24.60205999

```

(27)

So the candidate values of b are 1710, 1654, 1602, 1647, 1654, and 1663. The smallest M1 value resulting from these is the 25 digit number

M1 = 188889999999999999799999949

corresponding to $(b,c,k,d2) = (1602, 949, 5, 7)$ which has precisely eight consecutive happy numbers before the carry.

This M1 much larger than the eighteen digit M1 value we got from the 7/8 split over the carry, hence the 8/7 split over the carry cannot give a smaller N.

Thus, the 18 digit number above must yield the best value for N.

[>
=>
=>