$r^*(p,q)$ in $(0,1)$ such that if f is p-valent in an annulus $A(\delta)$, and $0 < \delta < r^*(p,q)$, then f is at most q-valent in U.

The author's proof of this result — which is based on a *normal family* [1, p. 213] argument, as a complex analyst reader might anticipate — is somewhat technical in detail and sheds no light on how the questions raised by the theorem might be answered, so we do not include it here. One question which arises is whether the theorem is true if we replace the condition '$q \geq 2p$' with '$q > p$', but a more fundamental question is :

What is the value of $r^*(p,q)$ for each permissible pair (p,q)?

We conclude by leaving these open questions, unclouded by any conjectures, for the reader.

Reference

[1] R. P. Boas, Invitation to Complex Analysis. Random House, 1987.

J. B. Twomey,
University College,
Cork.

ON A QUESTION POSED BY GRAHAM HIGMAN

Gerard M. Enright

Consider a function f of the non-negative integers given by the following rules:

$$\begin{aligned} f(3n) &= 4n \\ f(3n+1) &= 4n+1 \\ f(3n+2) &\text{ is undefined} \qquad (n = 0,1,2,3,\ldots). \end{aligned}$$

Since $f(0) = 0$ and $f(1) = 1$, the function may be repeatedly and indefinitely applied to 0 and 1; that is, for $z = 0$ and 1, $f^k(z)$ is defined for all $k > 0$.

Question: Is there any integer $z > 2$ such that $f^k(z)$ is defined for all $k > 0$?

This function was introduced by Professor Graham Higman [1] during a lecture on explicit embeddings of finitely presented groups. He posed the question and he conjectured that the answer was "No". To be precise, he declared "No" to be his "first best guess".

In this paper, we will not prove Higman's conjecture but we will produce a good deal of evidence in its favour. Neither will we discuss the group theoretic context in which the question was raised. Instead we present an exploration of the problem as an example of computer-aided mathematics suitable for secondary school and college level students.

We use elementary programs in BASIC to obtain data on the function and we use this data in further development of the problem, leading to more efficient programming. Our suggestion is that students' knowledge and understanding of mathematics is reinforced by doing mathematics and that a computer is a very useful tool in this process. We demonstrate the power and scope of electronic computation. We also show its limitations when faced with a great volume of calculations and with very large numbers.

Let z be a positive integer. The sequence obtained by repeated application of the function f, $\{z, f(z), f^2(z), f^3(z), \ldots\}$, will be called the f-string of z. If z is such that $f^{(m-1)}(z)$ is congruent to 2 mod 3 for some m then $f^m(z)$ is undefined and the f-string of z, $\{z, f(z), f^2(z), f^3(z), \ldots, f^{(m-1)}(z)\}$, has length m. Otherwise z has infinitely long f-string and the question is whether any such $z > 2$ exists.

The following program may be used to compute f-strings.

```
10 INPUT S
30 PRINT S,;
40 IF(S MOD 3)=0 THEN S=4*S/3:GOTO 30
50 IF(S MOD 3)=1 THEN S=4*((S-1)/3)+1:GOTO 30
60 IF(S MOD 3)=2 THEN PRINT"STOP"
80 END
```

Here are two results:

```
7    9    12   16   21   28   37   49   65   STOP
19   25   33   44   STOP
```

Here are two more in f-string notation:

$\{264, 352, 469, 625, 833\}$

$\{961, 1281, 1708, 2277, 3036, 4048, 5397, 7196\}$.

Our first objective is to prove that there is no integer z between 2 and 1,000 for which $f^k(z)$ is defined for all $k > 0$. We do this by computing the f-strings of all such z. Our use of this method betrays the fact that we expect all f-strings to be finite, as we are inclined to support Higman's conjecture. We then consider the feasibility of extending our methods to a higher number range.

The program above may be suitably amended by the following additions:

```
10 FOR T=2 TO 1000
20 S=T
70 NEXT T
```

Running on a BBC Master 128 microcomputer all f-strings are screened in 51 seconds and transferred to paper on a Star LC-10 dot-matrix printer in 10.6 minutes. Program line 60 is not necessary but it is useful, on a large printout of numbers, to highlight where each f-string stops.

Students may be patient enough to wait 10.6 minutes for a listing of the first 1,000 f-strings but it is unlikely that either they or their teachers will want to extend this method much further. It would take two class periods to list 10,000 and more than a week of non-stop running to list all f-strings up to 1 million. Surely we can do better than that.

All integers in the sequence $2, 5, 8, 11, 14, \ldots$ have f-strings of length 1 simply because $f(3n+2)$ is undefined. So let's not waste time checking them. The following program is a little better.

```
10 FOR T=3 TO 999 STEP 3
20 FOR R=T TO T+1
25 S=R
30 PRINT S,;
40 IF(S MOD 3)=0 THEN S=4*S/3:GOTO 30
50 IF(S MOD 3)=1 THEN S=4*((S-1)/3)+1:GOTO 30
60 IF(S MOD 3)=2 THEN PRINT"STOP"
65 NEXT R
70 NEXT T
80 END
```

With this program it takes 45 seconds to screen the f-strings and 7.8 minutes to print them out.

Two sequences of numbers have f-strings of length 2. These are:

$4, 13, 22, 31, 40, \ldots$

$6, 15, 24, 33, 42, \ldots,$

the sequences $\{9n + 4 : n = 0, 1, 2, 3, \ldots\}$ and $\{9n + 6 : n = 0, 1, 2, 3, \ldots\}$ respectively.

We are prompted to look at the function definition with integers written in terms of their least residues mod 9 rather than mod 3. f is undefined on all integers of the form $9n + 2$, $9n + 5$ and $9n + 8$. f^2 is undefined in two cases since $f(9n+4) = 12n+5$ and $f(9n + 6) = 12n + 8$. Hence, 5 out of every 9 integers have

f-strings of lengths 1 or 2 and we need only examine the other 4. The revised program screens results in 37 seconds and prints them in 6.1 minutes.

From the pattern of STOPS indicating f-strings of length 3 we are led to examine the function definition with integers expressed in terms of least residues mod 27. The following results are easily established.

f is undefined on all integers of the following forms:
$27n+2,\ 27n+5,\ 27n+8,\ 27n+11,\ 27n+14,$
$27n+17,\ 27n+20,\ 27n+23,\ 27n+26.$

f^2 is undefined in the following situations:

$$\begin{aligned} f(27n+4) &= 36n+5,\\ f(27n+6) &= 36n+8,\\ f(27n+13) &= 36n+17,\\ f(27n+15) &= 36n+20,\\ f(27n+22) &= 36n+29,\\ f(27n+24) &= 36n+32. \end{aligned}$$

f^3 is undefined in the following situations:

$$\begin{aligned} f^2(27n+3) &= f(36n+4) = 48n+5,\\ f^2(27n+10) &= f(36n+13) = 48n+17,\\ f^2(27n+18) &= f(36n+24) = 48n+32,\\ f^2(27n+25) &= f(36n+33) = 48n+44. \end{aligned}$$

With this analysis we may confine attention to just 8 out of every 27 numbers and we know that all excluded integers have f-strings of length at most 3. The following is our best program so far.

```
 10 FOR T=7 TO 979 STEP 27
 20 FOR R=1 TO 8
 30 ON R GOTO 40,45,50,55,60,65,70,75
 40 S=T:GOTO 80
 45 S=T+2:GOTO 80
```

```
 50 S=T+5:GOTO 80
 55 S=T+9:GOTO 80
 60 S=T+12:GOTO 80
 65 S=T+14:GOTO 80
 70 S=T+20:GOTO 80
 75 S=T+21
 80 PRINT S,;
 90 IF(S MOD 3)=0 THEN S=4*S/3:GOTO 80
100 IF(S MOD 3)=1 THEN S=4*((S-1)/3)+1:GOTO 80
110 IF(S MOD 3)=2 THEN PRINT"STOP"
120 NEXT R
130 NEXT T
140 END
```

A screen run takes 30 seconds and printout time is 4.5 minutes. We have reduced running time by 40% and printing time by nearly 60%. It would still take 3 days of non-stop running to reach our 1 million target.

To continue with this approach, we should now analyse function behaviour with integers written in terms of least residues mod 81. We should exclude from testing all integers which have f-strings of length at most 4. The work involved might be considered rather cumbersome.

Alternatively, we may further exploit the fact that the only integers with f-strings of length 4 or more are those of form $27n +$ one of $\{0,1,7,9,12,16,19,21\}$. The general form of $f^3(z)$ may be calculated in each case. The values are $64n +$ the corresponding element of $\{0,1,16,21,28,37,44,49\}$. Let us restrict attention to integers of this latter form. Eliminate all integers congruent to 2 mod 3, all congruent to 4 or 6 mod 9 and all congruent to 3, 10, 18 or 25 mod 27. The f-strings produced, when the revised program is run through the range $16 = f^3(7)$ to $2369 = f^3(1000)$, are of length at least 7 but the first 3 elements of each one are omitted. A screen run takes 12.5 seconds and paper printout takes 1.5 minutes. Extension of this method might also be considered cumbersome.

The idea of restricting output to the longer f-strings may be used in a more elegant process. Let us modify the program listed above so as to output only those f-strings of length at least 7. The new listing is as follows:

```
 10 DIM F(7)
 20 FOR T=7 TO 979 STEP 27
 30 FOR R=1 TO 8
 40 ON R GOTO 50,60,70,80,90,100,110,120
 50 S=T:GOTO 130
 60 S=T+2:GOTO 130
 70 S=T+5:GOTO 130
 80 S=T+9:GOTO 130
 90 S=T+12:GOTO 130
100 S=T+14:GOTO 130
110 S=T+20:GOTO 130
120 S=T+21
130 F(0)=S
140 FOR I=1 TO 6
150 IF (S MOD 3)=0 THEN S=4*S/3:GOTO 180
160 IF (S MOD 3)=1 THEN S=4*((S-1)/3)+1:GOTO 180
170 IF (S MOD 3)=2 THEN 250
180 F(I)=S
190 NEXT I
200 FOR J=0 TO 6:PRINT F(J),;:NEXT J
210 IF (S MOD 3)=0 THEN S=4*S/3:PRINT S,;:GOTO 210
220 IF (S MOD 3)=1 THEN S=4*((S-1)/3)+1:PRINT S,;:GOTO
                                                     210
230 IF (S MOD 3)=2 THEN PRINT "STOP"
240 PRINT:PRINT
250 NEXT R
260 NEXT T
270 END
```

On a run of this program, the 88 results are printed in 2.2 minutes. A re-run with the following changes:

```
 10 DIM F(13)
140 FOR I=1 TO 12
200 FOR J=0 TO 12:PRINT F(J),;:NEXT J
```

yields the 8 integers between 2 and 1,000 which have f-strings of length at least 13 in just 18 seconds. The longest f-strings in

this range are those of 163 and 331. Both of these have length 15 and they end with $f^{14}(163) = 9104$ and $f^{14}(331) = 18545$. We have a procedure now which we may reasonably hope to apply to higher numbers. Students might be encouraged to find out how many integers up to 10,000 have f-strings of length 19 or more and which of these has the longest f-string. How many integers up to 100,000 have f-strings of length 25 or more? Which of these is the longest?

The program is easily amended for these investigations. For example:

```
 10 DIM F(19)
 20 FOR T=7 TO 9997 STEP 27
140 FOR I=1 TO 18
200 FOR J=0 TO 18:PRINT F(J),;:NEXT J
```

A run of this modified program shows that there are just 5 integers up to 10,000 with f-strings of length at least 19. Output takes 2.8 minutes. The numbers are: 3475, 4633, 6177, 8236, 8607.

The first of these, 3475, has the longest f-string. It ends with

$$f^{23}(z) = 2596901.$$

The next three f-strings share this endpoint because

$$f(3475) = 4633, f(4633) = 6177, f(6177) = 8236.$$

Let us go a step further with the following changes:

```
 10 DIM F(25)
 20 FOR T=7 TO 49984 STEP 27
140 FOR I=1 TO 24
200 FOR J=0 TO 24:PRINT F(J),;:NEXT J
```

A run now takes 14.85 minutes. There are just 5 integers up to 50,000 with f-strings of length at least 25. The longest is that of 38,754, which ends with

$$f^{27}(38754) = 91549952.$$

Extending the range to 100,000 yields another 7 integers with f-strings of length 25 or more. There are 6 such integers between

100,000 and 200,000 and 6 more between 200,000 and 300,000. So far then, we have shown that there are just 24 integers z in the range 2 to 300,000 for which $f^{24}(z)$ is defined. The integer 38,754 still has one of the longest f-strings at 28 terms, a length equalled only by 65,610 and not exceeded.

Having reached 300,000 without finding an infinite f-string, we are inclined to rush onwards but we face two problems. We are approaching accuracy limits of the computer language (BBC BASIC) and program running times are rather slow (for classwork).

On the question of time, our current program, which is designed to list f-strings of length 25 or more, takes about 28 minutes for each 100,000 number range. Our 1 million time estimate stands at 4 hours and 40 minutes. We would rather not sacrifice program simplicity and legibility for minor efficiencies but one significant improvement would be to change all variables to integer type. Also integer division is executed faster than ordinary division. With these alterations, the program runs on a BBC Master 128 at about 22 minutes for each 100,000.

Schools and colleges will also have other equipment. Comparisons may be made of running times of similar programs on different computers, in different versions of BASIC and in other languages. The author also used an RM Nimbus X20. This 80186 based 8MHz computer supports BBC BASIC and runs it faster than the BBC Master Series. It is not necessary to use integer variables to take advantage of quicker integer calculations. The Nimbus runs our current program about 40% faster. The printer used in this experiment was an Epson LQ800 but printer speeds are not very significant now that output volume is considerably reduced.

Whether or not one is satisfied with this time, roundoff errors will ruin any attempt to go further. The next integer for which $f^{24}(z)$ is defined is $z = 335167$. Either computer will accurately produce the f-string of z up to

$$f^{27}(z) = 791783233$$

but then give 1.05571098 E9 and the message "Too big at line 240". Now the student can have the satisfaction of doing a few simple divisions and multiplications with pen and paper to produce the rest of the f-string beyond the capability of the machines. Results are as follows:

$$f^{28}(z) = 1055710977$$
$$f^{29}(z) = 1407614636$$
$$f^{30}(z) \text{ is not defined.}$$

We might amend the program by inserting brackets to give division priority over multiplication in the calculation of successive function values in the variable S. This however merely postpones the inevitable breakdown to $z = 491731$. The maximum integer which can be handled by BBC BASIC is 2,147,483,647. Larger real numbers can be stored but tenth and subsequent digits will be rounded and accuracy lost soon after the maximum integer value. Again the student who is not afraid of a few long calculations can go beyond the computer for the rest of the f-string of 491,731. It ends at a length of 40 terms with

$$f^{39}(491731) = 36672278528.$$

The student who survives that calculation will not want to give up before reaching the 1 million target. Would you be prepared to omit those program lines which compute the 26th and subsequent terms of the long f-strings? Let the computer stop at the 25th term and thus avoid roundoff errors and numbers which are too big. The Nimbus works at about 13.5 minutes for each 100,000 number range or 2 hours 15 minutes for a million run and there are a total of 69 integers z between 2 and 1,000,000 for which $f^{24}(z)$ is defined.

Since the Nimbus can handle real numbers which are just a little bigger than the maximum integer. some more help may be squeezed from the computer. The program can be extended successfully to give 28 terms of sufficiently long f-strings. There are just 17 survivors and it seems reasonable to complete that number of calculations by hand. The results are as follows:

Seven integers less than 1,000,000 have f-strings of length 28:

```
.   38754      65610      563401     595852
    725097     972979     988666.
```
Four integers have f-strings of length 29:
```
    422551     446889     543823     827199.
```
Three integers have f-strings of length 30:
```
    335167     794422     940402.
```
The three remaining integers are:
```
    491731     655641     874188.
```
The f-strings of these three have the exceptional lengths of 40, 39 and 38 respectively. All end with the same number 36,672,278,528. In fact, the last two are substrings of the first because

$$f(491731) = 655641 \quad f(655641) = 874188.$$

More powerful personal computers in the 80286 and 80386 ranges may be available to some students. Microsoft GW BASIC is normally supplied in the MS-DOS package. Greater accuracy and faster running times can be achieved. The author transferred the program to a 25 MHz Morse 486 personal system. Double precision numbers have an accuracy level of 17 digits internally with up to 16 displayed. The MOD operator, however, so useful for modulus arithmetic, has an upper limit of 32,767 and must be replaced by direct computation. A million run can be achieved, in a time of 24.4 minutes, on the Morse 486, printing full f-strings for all integers z between 2 and 1,000,000 for which $f^{24}(z)$ is defined.

It is observed that the hand completed results are all verified by the machine and it is felt that this is a good point at which to end the article. The patient reader might, however, like to know that the f-string of length 40 arising from 491,731 is not only the longest of any integer less than 1 million but it is also the longest f-string of any integer less than 10 million. Soon after that, it is equalled and then surpassed. The longest f-string of any integer less than 100 million is that of 95,305,399 which has 47 terms.

Reference

[1] G. Higman, *Some explicit embeddings of finitely presented groups* (Lecture at Groups in Galway Conference, May 1990) (Unpublished).

Gerard M. Enright,
Department of Mathematics and Computer Studies,
Mary Immaculate College,
Limerick.