Recovery of 1990s emails from Dr Michael Lennon

Sunny Daniels
e-mail: sdaniels@lycos.com

Abstract:

I have an old computer, probably manufactured in the 1990s, with e-mails from the late Dr Michael Lennon of Auckland University (passed away in 1999) on its hard drive. I think that these e-mails might be of significant historical interest to mathematicians (and probably others) because of the significant contribution that I believe Dr Michael Lennon made (in the 1970s I believe) to the knot theory breakthrough in the 1980s that resulted in the New Zealand mathematician Sir Vaughan Jones (who passed away in 2020 I believe) being awarded a Fields Medal in 1990.

Because of the age of this computer and its hard drive, and the possible historical value of these old e-mails from Dr Michael Lennon, I wish to try to extract these e-mails from this old computer in such a way as to minimise the probability of triggering any hardware or software failure that could endanger this old data. I propose a method involving DTMF tones and also an alternative method involving a data logger attached to the serial port of this old computer.

I would appreciate feedback on these proposed methods from people knowledable in this area (extraction of valuable data from possibly unreliable old computers) before attempting to go any further with the extraction process.

Introduction:

What happened here
As the New York sunset disappeared
I found an empty garden among the flagstones there
Who lived here
He must have been a gardener that cared a lot
Who weeded out the tears and grew a good crop
And now it all looks strange
It's funny how one insect can damage so much grain

And what's it for?
This little empty garden by the brown stone door
And in the cracks along the sidewalk nothing grows no more

Who lived here
He must have been a gardener that cared a lot
Who weeded out the tears and grew a good crop
And we are so amazed, we're crippled and we're dazed
A gardener like that one no one can replace

And I've been knocking but no one answers
And I've been knocking most of the day
Oh and I've been calling, oh hey hey Johnny
Can't you come out to play?

And through their tears
Some say he farmed his best in younger years
But he'd have said that roots grow stronger, if only he could hear
Who lived there
He must have been a gardener that cared a lot
Who weeded out the tears and grew a good crop
Now we pray for rain, and with every drop that falls
We hear, we hear your name

And I've been knocking but no one answers
And I've been knocking most of the day
Oh and I've been calling, oh hey hey Johnny
Can't you come out to play?

And I've been knocking but no one answers
And I've been knocking most of the day
Oh and I've been calling, oh hey hey Johnny
Can't you come out, can you come out to play, Johnny?
Can't you come out to play in your empty garden, Johnny?
Can't you come out to play in your empty garden, Johnny?
Can't you come out to play in your empty garden, Johnny?
Can't you come out to play in your empty garden, Johnny?
Can't you come out to play in your empty garden, Johnny?
Can't you come out to play in your empty garden, Johnny?

(***Empty Garden* by Sir Elton John, I believe.** Above lyrics obtained from result of a Google search for “empty garden lyrics”; a few apparent minor mistakes in the lyrics corrected by myself).

For the record:

1) I don’t know whether or not Dr Michael Lennon was related to Mr John Lennon of the Beatles.

2) I am not generally a Beatles fan (not old enough to be likely to be a Beatles fan, I think), but I do like the above “empty garden” song.

3) Thanks to Mr Jonathan Thompson-Bean (for whom I have been working sporadically as an e-commerce developer since early 2018) for suggesting the possible relevance of the above Elton John song to the memory of Dr Michael Lennon.

Now, for the techincal issues:

I recalled, about six months ago, that, if I am not mistaken, I have an old Debian Linux computer with a hard drive containing a large collection of e-mail messages from the late Dr Michael Lennon of Auckland University (a New Zealander with a PhD in Mathematical Physics from Massechusetts Institute of Technology, if am correctly informed) who unfortunately passed away, very shortly after his retirement, in 1999.

I also recalled hearing somewhere that he had two children, so I Google searched and found that his daughter is Professor Tava Olsen of the Auckland University Business School. I am not entirely sure as to what has happened to his son.

I emailed Prof Olsen and she was delighted to hear abour the old emails, and certainly happy to have copies of all of them! They seem to be in e-mail storage files from the old Macintosh email program Eudora.

I have sent her six of these e-mails simply by displaying them on the screen of the old computer (with "less" if i remember rightly), photographing this with my cellphone camera and emailing her the photos. Clearly, however, this is not an ideal method for retrieving the rest of the emails from this old computer.

However, this is an aging (manufactured before 1998 it appears) computer with a (if i am not mistaken) flat motherboard battery and aging hard drive, so I think that I need to treat it with great care given that:

1) Sir Vaughan Jones, who passed away in 2020, was, I believe, probably the only New Zealand citizen to have ever won a Fields medal.

2) Sir Vaughan Jones (I talked to him a few years ago, if I remember rightly) acknowledged Dr Michael Lennon for having made a major contribution, in the 1970s, to the knot theory breakthrough that got him the Fields Medal in 1990. I still remember the name of one witness to that conversation.

3) If I remember rightly, at least one photograph of Dr Michael Lennon appears in the University of Auckland obituary video for Sir Vaughan Jones:

https://www.youtube.com/watch?v=-i4uOWZvo_U

Also:

1) I don’t know exactly when and where Sir Vaughan Jones published his original knot theory breakthrough, but I am happy to try to find this out after I publish this, and then revise this article to include information on this if others are sufficiently interested.

2) I thank Associate Professor Warren Moors of the Auckland University Mathematics Department for giving me a copy of Kannan’s paper[1] some years ago, at a time when I believe I had no access to electronic journals in any University library. (This is unrelated to Dr Michael Lennon and Sir Vaughan Jones, as far as I know: this was for my research in Complexity Theory).

3) This was a bit before my time, but my understanding is that the British athlete Harold Abrahams, CBE, who I think won some medal at the olympics in the 1920s, passed away in 1978. I suspect that his life history was not particularly well-known to the British public in 1978, and someone at his funeral, or otherwise aware of his death, realised this, and *this* was the inspiration for the famous movie *Chariots of Fire*. I believe that *Chariots of Fire* premiered in 1981. I think that some of the music from it, by Vangelis, remained very popular for years after then, and a re-worked version of it was played at the London Olympics in 2012.

So maybe the death of Sir Vaughan Jones in late 2020 might inspire the creation of a movie about his life.

The Retrieval Process

The computer is an IBM compatible, running Debian Linux. I think that it has some sort of Pentium processor, 400 megabytes of RAM, a sound card, modem card and video card, and some USB sockets at the back. It says “year 2000 compliant” on the back, so I suspect that it was manufactured in the mid to late 1990s. It has a CD drive and a floppy drive. It also has serial and parallel ports, probably connected directly to the motherboard.

Given the possible unreliability of the computer and its hard drive, I am reluctant to simply open it up, pull out the hard drive, and put it into a newer computer. I thank a staff member of Rodhe and Schwartz here in New Zealand for suggesting jumpering the hard drive to make it read-only before attempting to recover the data, but again this would involve opening up the computer, which I presume would increase the risk of a hardware failure. Similarly, given that I suspect that this computer was manufactured in the very early days of USB, I am reluctant to plug any modern USB device into it (with the intention of copying the old e-mails to the USB device) for fear of triggering a hardware failure.

Somehow splitting up the e-mail archive files and copying them to floppies might also be a possibility, but I also see repeatedly taking floppy disks in and out of this old computer to be likely to trigger a hardware failure. Also, I don’t think I have any other floppy drives at home, but I think I have some in a storage unit here in Auckland. (Note added in January 2022: I was given a USB external floppy drive by a friendly security guard about a month ago who was vaguely aware of the existence of these old e-mails on this old computer, but I have yet to test it. My current thinking is that a floppy disk would be a good way of getting software onto the old computer to aid with the extraction, as I discuss below, but repeatedly taking floppies in and out of the old computer should be avioded, and hence splitting up the e-mails themselves to put onto floppies would not be desirable even if my new USB external floppy drive works).

The two approaches that seem, to me, to be the most promising and safe are:

1) Connect some sort of data logging device between signal ground and transmit data on the serial port. Somehow disable flow control and set the baud rate to something sensible with “setserial” if that is what it is called. Then type a command to send the e-mail files (probably one by one, one manually typed command per file) to the serial port. Then shut everything down, disconnect the data logger from the serial port, make a safe copy of the data on it, and then write some code to extract the contents of the e-mail files from the data on the data logging device (a sort-of software version of the receive side of a UART). The extraction software would, of course, be run on a newer, more reliable computer!

I did a Google search for such data logging devices, and discovered Rodhe and Schwartz; I e-mailed them and got a phone call back from them. I am happy to acknowledge the woman from Rodhe and Schwartz who called me if necessary; she was quite helpful. However, apparently Rodhe and Schwartz data loggers are designed specifically for logging RF signals from cellphone networks, and so require the input data to be in “I and Q” (in-phase and quadrature) form. I don’t know all of the details, but certainly the impression that I got is that some sort of non-trivial interface circuit would be required to convert the data from a PC serial port into a form that the Rodhe and Schwartz data loggers could accept.

However, there might be other brands of data logging devices on the market that could be directly connected to the serial port in the way that I described earlier. I think that one of them might be:

https://www.dataq.com/products/di-2108/

I would be happy to have feedback from scientists and engineers familiar with data logging devices on this.

2) Write a very simple C program to take an input file of e-mail messages, convert it into a sequence of DTMF tones in a format that the “pacat” program can accept (pacat is installed on the old computer: I have checked it) from stdin, and pipe the DTMF tone sequence into pacat to “blurt” out the contents of the input file through the sound card as DTMF tones! Each DTMF tone can be used to encode a nibble (the A, B, C and D tone pairs, which I believe were part of the original DTMF standard in the 1960s but are rarely used on telephones, can be used for the hex digits A to D. * and # can be used for E and F).

A sound recording device can be connected to the audio output of the sound card (if it works: I have not yet checked this) and then used to record the data from each file until either its capacity is used up (hopefully this can be avoided: if not, my software might have to divide the data up into chunks, with user intervention required to restart after each chunk: but this is fairly easy to implement, I think) or the e-mail file is finished.

The program dtmf_2.c (at the end of this article) is a mockup of this which worked successfully for me on my (modern) linux laptop (I don’t want to start up the old computer any more than absolutely necessary): this demostrates that this technique is potentially viable, I believe.

As input, it uses a hard-coded hex encoding of the string “I like old e-mails!”. It blurts it out as DTMF tones, one tone per hexidecimal nibble. The tones are 300 milliseconds long with 300 millisecond pauses between them. It outputs the audio in 8 bits per sample, mono (not stereo) format.

Sixteen-bit would give better sound quality I believe, but this is something that I have not yet had the time to try: I think that the DAC in most PC sound cards takes in digital audio in sixteen bit per sample per channel stereo format.

I ran this on my (modern Lenovo) laptop, with output piped to pacat. The DTMF tones came out of the speaker! I then ran it again with my cellphone voice recorder running, with the microphone of the cellphone held in front of the built-in speakers of the laptop.

I put the cellphone recording through an online DTMF decoder (I think that the URL was http://dialabc.com/sound/detect/index.html, but I am not sure: I might check it) and got the result (sorry about the slightly screwed-up formatting of the HTML table resulting from my cutting and pasting it as text):

```
:
Tone    Start Offset [ms]       End Offset [ms] Length [ms]
4
2,203 ± 15
2,505 ± 15
301 ± 30
9
2,776 ± 15
3,108 ± 15
331 ± 30
2
3,380 ± 15
3,682 ± 15
301 ± 30
0
3,983 ± 15
4,315 ± 15
331 ± 30
6
4,587 ± 15
4,889 ± 15
301 ± 30
C
5,191 ± 15
5,493 ± 15
301 ± 30
6
5,794 ± 15
6,096 ± 15
301 ± 30
9
6,398 ± 15
6,700 ± 15
301 ± 30
6
7,002 ± 15
```

7,303 ± 15
301 ± 30
B
7,605 ± 15
7,907 ± 15
301 ± 30
6
8,179 ± 15
8,511 ± 15
331 ± 30
5
8,782 ± 15
9,084 ± 15
301 ± 30
2
9,386 ± 15
9,688 ± 15
301 ± 30
0
9,990 ± 15
10,291 ± 15
301 ± 30
6
10,593 ± 15
10,895 ± 15
301 ± 30
#
11,197 ± 15
11,499 ± 15
301 ± 30
6
11,800 ± 15
12,102 ± 15
301 ± 30
C
12,404 ± 15
12,706 ± 15
301 ± 30
6
12,978 ± 15
13,310 ± 15
331 ± 30
4
13,581 ± 15
13,883 ± 15
301 ± 30
2
14,185 ± 15
14,487 ± 15
301 ± 30
0
14,788 ± 15

15,090 ± 15
301 ± 30
6
15,392 ± 15
15,694 ± 15
301 ± 30
5
15,996 ± 15
16,297 ± 15
301 ± 30
2
16,599 ± 15
16,901 ± 15
301 ± 30
D
17,203 ± 15
17,505 ± 15
301 ± 30
6
17,807 ± 15
18,108 ± 15
301 ± 30
D
18,380 ± 15
18,712 ± 15
331 ± 30
6
18,984 ± 15
19,285 ± 15
301 ± 30
1
19,587 ± 15
19,889 ± 15
301 ± 30
6
20,191 ± 15
20,493 ± 15
301 ± 30
9
20,794 ± 15
21,096 ± 15
301 ± 30
6
21,398 ± 15
21,700 ± 15
301 ± 30
C
22,002 ± 15
22,304 ± 15
301 ± 30
7
22,575 ± 15

22,907 ± 15
331 ± 30
3
23,179 ± 15
23,481 ± 15
301 ± 30
2
23,782 ± 15
24,084 ± 15
301 ± 30
1
24,386 ± 15
24,688 ± 15
301 ± 30

Now, the hex input string in my C code is:

// hex string: 49 20 6c 69 6b 65 20 6f 6c 64 20 65 2d 6d 61 69 6c 73 21

while the output from the above (the digits manually extracted) is:

49 20 6C 69 6B 65 20 6# 6C 64 20 65 2D 6D 61 69 6C 73 21

Putting one above the other for comparison (using a fixed-pitch font):

```
49 20 6c 69 6b 65 20 6f 6c 64 20 65 2d 6d 61 69 6c 73 21
49 20 6C 69 6B 65 20 6# 6C 64 20 65 2D 6D 61 69 6C 73 21
```

I use # for f in my C encoding code, so this looks right. Success! I think I still have a copy of the cellphone sound recording if anyone is interested in it.

So this suggests that the DTMF tones approach is likely to work. Of course, an electrical connection between the sound card output and the sound recording device would obviously be preferable (less distortion, less background noise) to simply holding a microphone in front of a loudspeaker! I would be happy to have the assistance of an electrical engineer with this. I presume that an air-cored transformer would be ideal for electrically isolating the PC from the sound recorder while still giving good audio quality: as far as I know, air-cored inductors have been used in crossovers in loudspeakers for decades. (Thanks to Mr Brian Robar, formerly of SiGe systems of Canada, I believe, for telling me a bit about this a few years ago; I don't know where he currently lives).

Clearly this is a relatively slow way of extracting data from the old computer, so the question arises: is it fast enough to be practical at all!

To help answer this question, let's do some calculations. Suppose that we pipe the mail folders through gzip before blurting out as dtmf, and a "typical" e-mail message compresses down to the same size as dtmf_2.c itself; this is 1100 bytes. I think that this is plausible (I think that most of these old e-mails are plain text with no attachments). Suppose that we shorten the tones and pauses between tones from 300 ms to 250 ms, both to speed things up a bit and to simplify our calculations!

Suppose we enhance our C code a bit to take advantage of the (presumed) stereo capability of the sound card by blurting out the high nibble of each byte through one channel and the low nibble through the other channel at the same time! This gives two bytes per second (250 + 250 + 250 + 250 = 1000). 1100 / 60 is 18.3, so this gives 18.3 minutes per e-mail.

So suppose an old-fashioned DAT recorder were used as the recording device: I hope that there are better solutions than this around, but at least DAT does not use data compression (unlike a cellphone sound recorder, which I think uses lossy compression) at all! If Wikipedia is to be believed:

https://en.wikipedia.org/wiki/Digital_Audio_Tape

then 120-minute (60 metres long) DATs tapes are the longest ones that work reliably, If the lower-quality (32kHz at 12 bits per sample per channel) recording mode (which I think would be more than adequate for DTMF tones, given that DTMF tones are used over digital telephone trunk lines, which I think use an even lower sampling rate and dynamic range than 32kHz at 12 bits) were used, this would give 240 minutes per tape. This gives 13.1 e-mails per tape, which I think is doable if the computer is left running for weeks with a human operator to change the DAT tapes when necessary! It would be fairly easy to modify my C code, I believe, to stop when it has filled up a DAT tape with its audio output and wait for a key to be pressed before resuming. However I hope that a better audio recording solution than this is available for a reasonable price (just for one-off use). I don't know what equipment professional music recording studios use nowadays.

So I would be happy to have feedback from experts in the relevant fields on how to proceed with the process of safely retrieving the emails from this old computer.

Getting dtmf_2.c onto the old computer

The other obvious question is: how do we get (an improved version of: it has to be able to take the input from a file or standard input rather than use a hard-coded input string) dtmf_2.c (or a compiled version of it) onto the old computer in the first place, and do it in such a way as to minimise the probability of causing a hardware or software failure? Putting a compiled version of it onto a CD and putting the CD into the computer would seem relatively safe and straightforward to me, although it might be necessary to copy it onto the hard drive in order to chmod it to make it executable; writing to the hard drive from which I am trying to recover data would not be ideal, I think, although of course harmess if everything is working correctly. I would be happy to have some feedback on this from others.

Using a floppy disk is another option, and maybe an ext2-formatted (or ext3, ext4: I don't know if ext4 is still the latest) floppy could be used to solve the executability problem: however I think that at the moment all of my floppy drives are in storage, except for the one in the old computer in question. (Note added in January 2022: I now also have a USB external floppy drive as mentioned earlier, but I have not yet tried it).

If all else fails (although this doesn't get around the problem of having to possible write to the hard drive in order to make the binary executable), if the standard linux hex dump utility xxd is on the old computer (I think that it probably is), then it can be used in reverse to create an arbitrary binary files from hex typed into the keyboard:

```
(linux command prompt)$ xxd -r -p
6865 6c6c 6f0a
hello
```

(linux command prompt)$

(output redirection would be necessary for this, of course). The (updated) object code of dtmf_2 could then be created on the old computer directly from a hex version typed into the command prompt. This process could be automated with a modified keyboard in which relays, controlled by a modern computer, are connected to the innards of the keyboard in parallel with the 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, a, b, c, d, e and f key switches. Having to modify a keyboard in this way would be a bit of a nuisance, however.

Possible other uses of this DTMF Solution

I believe that this DTMF tone method of retrieval of data from old computers could also be very useful for retrieving data from hard disks (or even floppy disks, particularly those from the 1980s that are incompatible with a standard IBM PC floppy drive, e.g. I believe the original Apple Macintosh floppies of the early 1980s) of computers from the pre-USB era: clearly putting a USB flash drive into such a computer (unmodified) is not an option! I know some university lecturers who still have old 1980s Macintoshes with interesting data on them. As far as I know, the Apple Macintosh has had good enough sound hardware to be able to "blurt out" a sequence of DTMF tones ever since its launch in the early 1980s. Of course, getting the software to do this onto a floppy disk readable by an early 1980s Macintosh might not be so easy: maybe the ROM debugger of these machines (accessible by installing the plastic "programmer's switch" which enables the "interrupt" switch behind a slot in the case to be pressed, I believe) could be used to put arbitrary code into RAM, if the documentation for it still exists. If not, maybe a spare old Macintosh could have its ROM changed to a customised ROM that allows arbitrary data to be typed into the keyboard (maybe automatically: relays controlled by a modern computer could be connected to the innards of the keyboard with their contacts in parallel with certain key switches) and written to an old Macintosh-format floppy disk.

dtmf_2.c:

```
#include <stdio.h>
#include <math.h>

// https://www.rapidtables.com/convert/number/ascii-to-hex.html

// input string: "I like old e-mails!"

// hex string: 49 20 6c 69 6b 65 20 6f 6c 64 20 65 2d 6d 61 69 6c 73 21

char *input_hex_string = "49206c696b65206f6c6420652d6d61696c7321";

// (decoding success with:)
// http://dialabc.com/sound/detect/index.html

void dtmf_tone(int row_freq,int column_freq,int length_hunsec);

void output_pause(int length_hunsec);

void blurt_hex_string(char *input_string) {
  char *input_char;
  char input_char_value;

  int row_1 = 697;
  int row_2 = 770;
  int row_3 = 852;
  int row_4 = 941;

  int col_1 = 1209;
  int col_2 = 1336;
  int col_3 = 1477;
  int col_4 = 1633;

  int duration=30;

  input_char = input_string;
  while (*input_char != 0) {
    input_char_value = *input_char;

    if (input_char_value == '1') dtmf_tone(row_1,col_1,duration);
    if (input_char_value == '2') dtmf_tone(row_1,col_2,duration);
    if (input_char_value == '3') dtmf_tone(row_1,col_3,duration);
    if (input_char_value == 'a') dtmf_tone(row_1,col_4,duration);

    if (input_char_value == '4') dtmf_tone(row_2,col_1,duration);
    if (input_char_value == '5') dtmf_tone(row_2,col_2,duration);
    if (input_char_value == '6') dtmf_tone(row_2,col_3,duration);
    if (input_char_value == 'b') dtmf_tone(row_2,col_4,duration);

    if (input_char_value == '7') dtmf_tone(row_3,col_1,duration);
    if (input_char_value == '8') dtmf_tone(row_3,col_2,duration);
    if (input_char_value == '9') dtmf_tone(row_3,col_3,duration);
    if (input_char_value == 'c') dtmf_tone(row_3,col_4,duration);

    if (input_char_value == 'e') dtmf_tone(row_4,col_1,duration); // E is *
    if (input_char_value == '0') dtmf_tone(row_4,col_2,duration);
    if (input_char_value == 'f') dtmf_tone(row_4,col_3,duration); // F is #
    if (input_char_value == 'd') dtmf_tone(row_4,col_4,duration);

    output_pause(30);

    input_char++;
  }
}

int main(int argc, char **argv) {
  blurt_hex_string(input_hex_string);
}

// https://en.wikipedia.org/wiki/Dual-tone_multi-frequency_signaling

// at 44.1 kHz, one second is 44100 samples
// so one cycle of a 400 hertz tone is 110.25 samples

void output_pause(int length_hunsec) {
  int sample_loop;

  for (sample_loop = 1; sample_loop <= length_hunsec*441; sample_loop++) {
    putchar(0);
  }
}

// hunsec is hundredths of a second

void dtmf_tone(int row_freq,int column_freq,int length_hunsec) {
  int sample_loop;

  float row_sample_num_divisor;
  float column_sample_num_divisor;

  float float_sample_loop;
  float row_period_fraction;
  float column_period_fraction;

  float row_phase_angle;
  float column_phase_angle;

  float cosine_output;
  int zero_voltage = 127;
  float voltage_amplitude = 50; // 8 bit samples
  // sum of two cosine functions is in [-2,2]
  float output_byte_float;
  int output_byte_int;

  row_sample_num_divisor = 44100.0 / row_freq;
  column_sample_num_divisor = 44100.0 / column_freq;

  for (sample_loop = 1; sample_loop <= length_hunsec*441; sample_loop++) {
    float_sample_loop = sample_loop;

    row_period_fraction = float_sample_loop / row_sample_num_divisor;
    column_period_fraction = float_sample_loop / column_sample_num_divisor;

    row_phase_angle = 2 * M_PI * row_period_fraction;
    column_phase_angle = 2 * M_PI * column_period_fraction;

    cosine_output = cos(row_phase_angle) + cos(column_phase_angle);

    output_byte_float = zero_voltage + voltage_amplitude * cosine_output;
    output_byte_int = output_byte_float;
    // printf("%d\n",output_byte_int);
    putchar(output_byte_int);
  }
}
```

<u>Note added in January 2022:</u> I think that the above code doesn't use the full dynamic range available with 8-bit samples. It looks as if a sample value of 0 corresponds to maximum negative output voltage, which I think is about -2 volts for the line output of most digital audio equipment (https://forum.audiogon.com/discussions/standard-output-voltage-for-rca-line-outs-dac-cd-streamers-etc), 127 corresponds to zero volts, and 255 corresponds to the maximum positive output voltage (presumably around +2 volts). However, it looks to me as if the above code is adding a value in the interval [-100,100] to 127, producing a value in [27,227]. If so, then fixing this would improve the quality of the audio output slightly; the above code was basically just a quick hack! I am happy to investigate this properly if anyone is interested.

<u>gzipped and base64 encoded:</u>

In case you are interested in copying the code out of this document (should be possible for PDF, I think) and compiling it and running it, here is a base64-encoded gzipped version, to make the process easier for you:

H4sICNHbXmEAA2R0bWZfMi5jALVWXY/qNhB951fMZVVdvvNJWJZLn9rq3ocr9b2tImMbYm2IaezA
3a32v3fsBDYQglhVldBufDxzPHM8mcmDyGhaMA5flGZCTpKfOw8naEt0YpCO40Ci9U49Oc7hcJjk

```
ZCeYJquUqwmVW4fKbM9z7WTFdsVzhygqxFjLccJ/TBK9TS2ByHaFBqVzkW2eoPsNUvHMQaYM+HhL
RKo+dcuD+I+TVTgH34WIQjSHaAXR1C7XFgntMyIMIvx51obCLADf63RoQnIY2DNjZIxLRlhCN5z7
bkSjebSKpvi0xucQ/099FrHIQ5zOAt/rLmwwPcapZMZTFZRypeAgdPLUPyqCgjBBUrKiVggli4w5
jGtOtSMydsp/LwUDprfrWMuM90SmIZeHeJ3zv0dmQWVabLP3dcqzjU7ipMgUp/1FRSALbfLZkUKV
HNfNVmmR17Pu1cUooT780wGo4+Z5ccTeoXhP0oKbDfwdw/ZQx2g+W9QgH6HZzK1DAUKPU78OhQjN
Qw8DLTFM25J5vjtf1DDD5gVBVMcMnRfOZnXM8HlREJwIWZETLWS2DNxTzMdU0LYugdk/JCLl0KtJ
AJ+W4JbqQEMGZDhTqzRaQ69puITP3ud+7cqtbiOb8OgYZX9xm8FvYfDvZghaGIK7GUgLQ1hnuE0R
Nij8DwoxbWG4X4ioheF+IVYtDB8QYtagCD4oxGMLw/1CzFsY7heCtjB8QAjeoAgbQgC22F9BKBjc
5nJbuO6XZN3CEFxE85uJ5uE2F2vhaopz1swDtwryou8MhwZ+67x1Oqa94ZzMbOMn+YaOqv49wMW+
bFmNzn85APGUt7NxzrPJQTyLHccxNpH5xjEr55eCpGOTQbwtUi3GZi4VPKMvsRKbjKTIZFmIhjCc
ePD89XUEaA44hmTGjE5h6LkuKLLd4VeCsVXSWtAXii1XroFAiAYJfje8gjnKOHmeO/GnJ6+7pp5N
3MClV5xKuat6/1rm0KvBZlos6nbwZXnONsCwzyyGw+MwwBiM3r3ypsyV2PsyWlpXEz8+sZwznagy
w1KO/zr+r2do8kslKZmqHfwAi5nYCyXtLC/3K/LrJicr+ze+0PD9gB3PhWQYIaGmiBvsV/bP/ROi
eEyyTcqbvud7tW0lsATL+z+O/Veey3gvU0023H45zN75Kjg2SQhtPmGXMHXtu/sIK6FPhQUGUsXW
3JI+yOokWBeZjV6ZuxQZ/DH2R/5fJ/qqEFcvmscWOcZU38C1vZvrt4IB2Rdj4oJzKgH72dV2SWce
tTr5P0q8UQPIdlERVYe6UhJo2/R3blQntNROC9HNMq6H9V5NyOTDAL7Hv3/Dfy1lDFfK8NyxvcSr
Y88KFX1x3bsIpQ9DCzfPqjf+RoUh2VnBD6/U+KD5osBlSSLP1eq1L8IOJ4Ne97o/sT+z7ujCswrv
2P2u7drx1PkXLBfBYkoOAAA=
```

It needs to be base64 decoded and then guzipped.

References:

1: Ravi Kannan: Circuit-size lower bounds and non-reducibility to sparse sets. *Information and Control*, Volume 55, Issues 1-3, October-December 1982, pages 40-56.