Solution to Car Talk problem

I can no longer find the transcript of this episode online, but the problem is reproduced below. I assigned it as a homework problem in an honors intro mechanics (physics) class in 2009.

1. You're driving your car on the highway at 75 mph, and you notice a sign that says you are 75 miles from your destination. So if you continue driving at that speed, you'd be there in an hour. But, you're not going to do that, because then it wouldn't be a puzzler. When you have driven one mile and you are now 74 miles from your destination, you drop your speed down to 74 mph.

So, you drive that first mile at 75 mph; when you are 74 miles from your destination, you drop your speed down to 74 mph; and then 73 mph, 72 mph ... and so on. Until, finally, you get down to 1 mile from your destination and you're driving at one mile per hour.

And the question is, if you do this, how long is it going to take you to travel the entire 75 miles?

Solution: This was a problem originally posted on the radio show *Car Talk*, you can find their response here [9 June 2012: dead link]:

`http://www.cartalk.com/content/puzzler/transcripts/200642/answer.html`

Incidentally, the hosts Tom and Ray Magliozzi both have degrees from MIT. Anyway: we can figure this out mile-by-mile. At constant speed, the time t taken to travel a distance d at velocity v is just $t = d/v$. If we traveled a total distance of 75 mi at 75 mi/h, the time would simply be

$$t_o = \frac{d}{v} = \frac{75\,\text{mi}}{75\,\text{mi/h}} = 1\,\text{h} \tag{1}$$

Of course, the problem presented is not this simple. For the first mile, we travel 1 mi at 75 mi/hr, for the second, we travel 1 mi at 74 mi/h, and so on. This leads to a series of times for each mile, the sum of which is the total trip time. Adding up the times for each mile,

$$t_{\text{tot}} = t_1 + t_2 + \ldots + t_{75} = \frac{1}{v} + \frac{1}{v-1} + \ldots \frac{1}{1} = \sum_{k=75}^{1} \frac{1}{v-(k-1)} = \sum_{k=1}^{75} \frac{1}{k} \tag{2}$$

Looking at the series we end up with, it is nothing more than the harmonic series $1/n$ summed from $n = 1$ to $n = 75$. There is no closed-form solution for the harmonic series through n terms, but there is a nice approximation:

$$\lim_{n\to\infty} \sum_{k=1}^{n} \frac{1}{k} = \ln n + \gamma \tag{3}$$

Here $\gamma \approx 0.57721$ is the Euler-Mascheroni constant. For sufficiently large n, we can approximate the harmonic sum:

$$\sum_{k=1}^{n=75} \frac{1}{k} \approx \ln k + \gamma \approx 4.3175 + 0.57721 \approx 4.895 \tag{4}$$

Thus, the trip should take approximately 5 hours. One can also sum this series by brute-force, ideally using a computer (see code examples below). The exact result is:

$$\sum_{n=1}^{75} \frac{1}{n} = \frac{670758981768141571449624262218133}{136851726813476721146087646859200} \approx 4.901355630553047 \tag{5}$$

The approximation is good in this case to about 1%, and it gets better for larger series. The trip takes just under 5 hours. (The exact solution was generated by the LISP program below. LISP is just cool that way.)

The car talk solution was to use the approximation we noted above:

> RAY: What you wind up with is a series of numbers. If you start counting from your destination of one hour, plus a half an hour, plus a third of an hour, plus a fourth, plus a fifth, plus a sixth, and when you're driving at 60 miles an hour, that mile takes you one sixtieth of an hour obviously which is a minute.
>
> I got the answer by adding up one plus a half, plus a third, and so on. I converted them all to decimals.
>
> There's no discrete answer, the best you can get is an approximation. Here's how you get it: you have 75 terms. You're going to look up the natural log of 75, which happens to be 4.317, and you're going to add it to something called Euler's constant.
>
> Euler was a Swiss mathematician who lived in the 1700s, and he came up with a bunch of interesting stuff, not the least of which is this. His constant is something like 0.57712. If you add the log of 75 and .577 you come up with 4.89.
>
> TOM: So choice two, approximately five hours is the correct answer.

On can also do this in code pretty easily. I used this problem as a teaching exericse - the exact solution would be extremely tedious on paper or with a calculator, but trivial to generate from a computer program. I whipped up solutions to this problem in various programming languages, mainly so the students in my class could see different ways of numerically solving the problem (most of them knew programming, but had a pretty diverse background of languages and depth). I made the C versions a bit fancier, taking command line arguments and such, but all versions below work: standard C (iterative and recursive), LISP, pascal, fortran, java, perl, postscript, python, and even an approximate version with a bash shell script (it is a hack, but plenty accurate). Keep in mind that the programs were supposed to be readable for students with minimal programming background, not elegant per se.

Postscript was probably the most interesting to figure out. I had never programmed in Postscript before this exercise, and it seemed wildly inefficient for the current task, but it does work. Since it is stack-based, the solution also looks much different than the other languages. The output is very nice looking, however. Really - copy and paste the postscript code below into a file and save it as (say) "harmonic.ps." Open that file in, e.g., Preview, Acrobat Distiller, Ghostview and you'll have a nicely printed table of the harmonic series. You aren't just computing the harmonic sum, you're computing how to display the output nicely on a page!

C:

```
//sums the partial harmonic series H(n) = 1 + 1/2 + ... + 1/n
//usage: pass "n" as a command line argument
//e.g., harmonic 75 <enter>
//returns H(N), the sum of the first N terms
//BY THE WAY: the input is restricted to N<=1e5
//(c) P. LeClair 2009

#include <stdio.h>

int main (int argc, const char * argv[])
{
        int i=0, DIST;
        double t=0;

        if (argc!=2) {
                fprintf(stderr,"\nUsage: %s NUM\n\n",*(argv));
                return(-1);
        }

        else DIST= atoi(*(argv+1));

        if (DIST>1e5) {
                printf("N out of range; try a number less than 1e5.\n");
                return(-1);
        }

        printf("\nPartial sums of the harmonic series through n=%i terms:\n",DIST);

        do      {
                t+=1.0/((double) ++i);
                printf("n=%i, H(%i)=%g\n",i,i,t);
        }
        while(i<DIST);

        return(0);
}
```

C (recursive):

```
//sums the partial harmonic series H(n) = 1 + 1/2 + ... + 1/n
//usage: pass "n" as a command line argument
//e.g., harmonic 75 <enter>
//returns H(N), the sum of the first N terms
//BY THE WAY: the input is restricted to N<=1e5
//(c) P. LeClair 2009

#include <stdio.h>

double sum(double N)
{
        if (N == 0) return 0;
        return 1.0/N + sum(N - 1.0);
}

int main (int argc, const char * argv[])
{
        int i=0, DIST;
        double t=0;

        if (argc!=2) {
                fprintf(stderr,"\nUsage: %s NUM\n\n",*(argv));
                return(-1);
        }

        else DIST= atoi(*(argv+1));

        if (DIST>1e5) {
                fprintf(stderr,"N out of range; try a number less than 1e5.\n");
                return(-1);
        }

        printf("\nPartial sum of the harmonic series through n=%i terms:\n",DIST);
        printf("%g\n",sum(DIST));

        return(0);
}
```

Bash script:

```
#!/bin/bash
#simple bash script to calculate first 75 terms of the harmnoic series, approx
#since bash only does integer math, we make some allowances
#multiply everything by 1e6 and compute sum in millionths :-)
count=1
sum=0
while [ $count -lt 76 ] ; do
 sum=$(( $sum + 1000000/$count ))
 count=$(( $count + 1 ))
done
sum=$sum/1000000
echo "Sum = $sum"
```

Common LISP:

```
(defun sum (N)
  ( loop for i from 1 to N
         for sum = 1 then (+ sum (/ 1 i))
         finally (return sum)))

(format t "Sum of harmonic series through 75 terms is ~A~%" (sum 75))
```

Postscript:

```
%!PS
/LM 72 def                                   %define left margin
/Times-Roman findfont 8 scalefont setfont
/nstr 7 string def

/newline
{ currentpoint 8 sub                         % move y down by 8 points
exch pop                                     % drop old x coordinate
LM exch                                      % replace it with left margin
moveto} def                                  % go there

/harmonic
{     /num 0 def
           /current 1 def                    % loop counter
          {/num 1 current div num add def    % add 1/current to running sum
          /current current 1 add def         %incr counter
      } repeat
      num
} def

/prt-n                                       %convert given value to a string
{ nstr cvs show } def                        %then print it

/prtharmonic                                 %given N, calc first N harmonic sums
{ dup prt-n                                  %print N =
( = ) show
harmonic prt-n%                              %get current partial sum & print it
newline                                      %along with a newline
} def

% ---------- Program ---------------
LM 700 moveto                                %go to upper left corner of page
(Partial sums of the first 75 terms of the harmonic series:)show
newline
newline
1 1 75 {prtharmonic} for                     %show every term from 1 to 75
showpage
```

Python:

```
print '\nSum of the first 75 terms of the harmonic series 1/n.'

sum=0

for i in range(1,76):
    sum  = sum  + 1/float(i)

print 'sum: ' + str(sum)
```

Java:

```java
// sum first N terms of harmonic series
// compile: javac harmoic.java
// run: java harmonic N

import java.io.PrintWriter;

public class harmonic {

  public static void main(String[] args) {
        // command line argument gives how many terms to sum over
        int N = Integer.parseInt(args[0]);

        double sum = 0.0;
        for (int i=1; i<=N; i++)
            sum += 1.0/i;

        System.out.println("The␣sum␣of␣the␣first␣" + N
                        + "␣terms␣of␣the␣harmonic␣series␣is␣"  + sum);

    }

}
```

Fortran:

```fortran
PROGRAM HARMONIC
REAL :: SUM = 0
INTEGER :: I=1

DO I=1,75
    SUM = SUM + 1.0/I
END DO

WRITE (*,*) "Sum␣of␣harmonic␣series␣through␣75␣terms:␣",SUM

END PROGRAM harmonic
```

Perl:

```perl
#!/usr/bin/perl
$sum=0;
for ($counter = 1; $counter <= 75; ++$counter) {
    $sum = ($sum + 1/$counter);
}

print "Sum␣of␣harmonic␣series␣through␣n=75:␣", $sum,"\n";
```

Pascal:

```
program harmonic;

var
   i    : Integer;
   sum  :  Real;

begin
    sum := 0;
    i := 1;
    repeat
       sum := sum + 1.0/i;
       i := i + 1;
    until (i=76);

    Write('Sum␣of␣first␣75␣terms␣of␣the␣harmonic␣series:␣');
    Writeln(sum);

end.
```