A COMPUTER SEARCH FOR INTEGER VALUES OF 2^C AND 3^C

Neal R. Wagner

1. Introduction. In [1], Wilansky posed the following question.

> PROBLEM. *Do the equations $2^C = N$ and $3^C = M$ have any solutions for N and M integers, C not an integer?*

This article concerns a computer search for solutions. Section 2 describes an initial "brute-force" search for $N < 10^6$. Section 3 gives an improvement of the search procedure which was used for $N < 10^7$. The search showed that there were no solutions in the range $N < 10^7$, $C < 23.25349\ \ 652$, $M < 12\ \ 43762\ \ 43617$. Finally, Section 4 gives a few values of C which came closest to yielding a solution.

2. The Initial Search. A very simple method was used at first. Starting with an integer limit L, we consider each positive integer $N \leq L$ which is not a power of 2. Within the accuracy of the computer calculations, we solve for C in $2^C = N$ and then calculate $M = 3^C$ as a real number. There is no possibility of a solution unless M is as close to an integer as the accuracy of our calculations would allow.

The calculations were carried out using double precision hardware on a CDC 6000 series computer and using CDC FORTRAN subroutines for double precision natural logarithms and exponentiation. We were dependent on this particular machine because its double precision uses 96 bits (the equivalent of about 28.8 decimal digits) for the significant part of a floating point number.

Other common machines cannot carry out calculations with this accuracy at the hardware level, and software simulation would be very time-consuming.

3. Improved Search. A simple observation allows us to greatly speed up the search. Having checked a value of $N = 2^C$ by our method and calculated $M = 3^C$ as the same time, we can check $N \cdot K$ for fixed K as follows: since

$$N \cdot K = 2^C \cdot K = 2^{(C+\log_2 K)}$$

and

$$3^{(C+\log_2 K)} = M \cdot 3^{\log_2 K},$$

we need only multiply M by the fixed constant $3^{\log_2 K}$ and check if the result is close to an integer. In this new check, the logarithm and exponentiation of the original check

are eliminated In theory, we would only have to do an original check for prime values of N. In practice, we only eliminate values of N with the first few primes as factors. If we assume that the logarithm and exponentiation take up most of the computation time, then by using the quick check for multiples of 2, we would reduce the time to 50% of the original. Similarly, using the quick method for multiples by 2 and 3, we get a time of 33.3%; multiples by 2, 3, and 5 give 26.3%; multiples by 2, 3, 5, and 7 give 22.857%; and so forth.

In an actual run, the algorithm below, which uses the quick method for multiples by 2 and 3, ran in about 42% of the original computation time. Because of the increased overhead of adding extra multiples, it did not seem worthwhile to incorporate multiples by any other primes.

In the algorithm below, N takes on values 1, 5, 7, 11, 13, 17, 19, 23, 25, . . ., i.e., those not divisible by 2 or 3. Inner loops check multiples of N by mixed powers of 2 and 3. All variable are floating point except for the integer variables N, N_1, N_2, and L, and the logical flags F_1 and F_2. (The algorithm uses F_1 to increment N and F_2 to prevent redundant checks.) This algorithm as given does not eliminate values of N which are exact powers of 2.

ALGORITHM. Solutions to $2^C = N$ and $3^C = M$, where $N \le L$.

1. Set L. Set $D \leftarrow 3^{\log_2 3}$, $N \leftarrow 1$, $F_1 \leftarrow$ *true*.
2. Repeat steps 3–13 until $N > L$.
    3. Set $C \leftarrow \ln N / \ln 2$, $R \leftarrow 3^C$.
    4. Check if R is close to an integer.
    5. Set $N_1 \leftarrow N$, $R_1 \leftarrow R$, $F_2 \leftarrow$ *false*.
    6. Repeat steps 7–11 until $N_1 > L$.
        7. Set $N_2 \leftarrow N_1$, $R_2 \leftarrow R_1$.
        8. Repeat steps 9 and 10 until $N_2 > L$.
            9. If F_2 is *true*, then check if R_2 is close to an integer.
            10. Set $N_2 \leftarrow N_2 \cdot 2$, $R_2 \leftarrow R_2 \cdot 3$, $F_2 \leftarrow$ *true*.
        11. Set $N_1 \leftarrow N_1 \cdot 3$, $R_1 \leftarrow R_1 \cdot D$.
    12. If F_1 is *true*, then set $N \leftarrow N + 2$.
    13. Set $N \leftarrow N + 2$, $F_1 \leftarrow$ *not* F_1.

Using this algorithm, no solutions were found in the range $N \le 10^7$.

It would also be possible to speed up the search by doing a preliminary check using less precision. However, on our machine single precision did not suffice even for a preliminary check. Our only other option would have been to modify or replace the CDC exponentiation

routine to give a faster routine with less precision. This method might have cast doubts on the validity of the results and was not attempted.

4. Near Solutions. During execution, we printed out values of N, C, and M for which M was within 10^{-4} of an integer. The Table gives successive values of N for which the corresponding value of M is closer to an integer than any preceding value. The last line of the Table gives the value of M closest to an integer for $N \leq 10^7$. All numbers are given with 25 significant figures and are easily that accurate. Changing C so that M is an exact integer gives, for example, in the last line,

$$\begin{aligned} C &\doteq 22.76468\ 10957\ 11037\ 01900\ 883 \\ M &= 7\ 26964\ 50718 \\ N &\doteq 71\ 26098.99999\ 99999\ 99699\ 557 \end{aligned}$$

Reference.

1. A. Wilansky, "What if 2^C and 3^C are integers?" *Amer. Math. Monthly* 83 (1976), p. 473.

C	$N \doteq 2^C$	$M \doteq 3^C$
12.42862 18860 91111 35346 962	5513	8 51058.99992 94174 27425 7055
13.04080 34098 47554 54001 529	8427	16 67418.00002 17414 82579 523
13.18750 67277 58278 55727 088	9329	19 59024.00001 65308 19179 244
16.93574 05760 08516 25131 170	1 25362	1203 37724.00000 82893 63002 6
17.53128 24546 09020 56268 167	1 89427	2314 98127.00000 04724 86197 0
21.74165 04914 69766 91641 454	35 06622	2 36267 26945.00000 03933 3511
22.09092 90814 55665 18216 181	44 67168	3 46778 21059.00000 00984 9639
22.14826 24410 67421 50979 174	46 48270	3 69323 34957.99999 99927 9639
22.76468 10957 11037 01906 966	71 26099	7 26964 50718.00000 00048 5783

TABLE