Al Zimmermann's Programming Contest Factorials Example: 25! [last update 2013-04-21]

Here I try to explain my methods using 25! as an example. A table on the last page shows the shortest sequences for all n!.

Use normal backtracking where in each step k the number f(k), the "binary log code" of f(k) and the "cumulative prime flags integer" is calculated. You don't need big number arithmetic. Find more details in the chapter Remarks below the first tables.

step k	operation	number f(k)	prime factorization	binary log code (23-19-17-13-11-7-5-3-2)	prime flags
0		1	1	0	000000000
1	f(0) + f(0)	2	2	00001	000000001
2	f(1) × f(1)	4	2^2	00010	000000001
3	f(2) + f(1)	6	$3 \cdot 2$	0001-00001	000000011
4	f(3) × f(3)	36	$3^2 \cdot 2^2$	0010-00010	000000011
5	f(4) + f(3)	42	$7 \cdot 3 \cdot 2$	01-000-0001-00001	000001011
6	f(5) + f(4)	78	$13 \cdot 3 \cdot 2$	1-00-00-000-0001-00001	000101011
7	f(6) + f(4)	114	$19 \cdot 3 \cdot 2$	1-0-0-00-00-000-0001-00001	010101011
8	f(7) + f(3)	120	$5 \cdot 3 \cdot 2^3$	001-0001-00010	010101111
9	f(8) × f(5)	5040	$7 \cdot 5 \cdot 3^2 \cdot 2^4$	01-001-0010-00100	010101111
10	f(9) × f(8)	604800	$7 \cdot 5^2 \cdot 3^3 \cdot 2^7$	01-010-0011-00111	010101111
11	f(10) + f(9)	609840	$11^2 \cdot 7 \cdot 5 \cdot 3^2 \cdot 2^4$	10-01-001-0010-00100	010111111
12	f(11) + f(8)	609960	$23 \cdot 17 \cdot 13 \cdot 5 \cdot 3 \cdot 2^3$	1-0-1-1-00-00-001-0001-00011	111111111

In this case after 12 steps all prime factors are available because Prime Flags = $(111111111)_2$.
Now we can try to find a solution by multiplication of certain numbers f(1) to f(12).

In fact we can subtract recursively log codes of numbers from the log code of $25! = 23 \cdot 19 \cdot 17 \cdot 13 \cdot 11^2 \cdot 7^3 \cdot 5^6 \cdot 3^{10} \cdot 2^{22}$ until the result is equal to 0.

logCode(25!)	1-1-1-1-10-11-110-1010-10110
– logCode(f(12))	1-0-1-1-00-00-001-0001-00011
=	1-0-0-10-11-101-1001-10011
– logCode(f(7))	1-0-0-00-00-000-0001-00001
=	10-11-101-1000-10010
– logCode(f(11))	10-01-001-0010-00100
=	10-100-0110-01110
– logCode(f(10))	01-010-0011-00111
=	01-010-0011-00111
– logCode(f(10))	01-010-0011-00111
=	0

Therefore the sequence can be continued with the following multiplications:

13	$f(12) \times f(7)$	A
14	$f(13) \times f(11)$	B
15	$f(14) \times f(10)$	C
16	$f(15) \times f(10)$	25!

The numbers A, B, C and 25! can be calculated with another program or even “by hand” with a big number calculator.

Remarks

Backtracking

To avoid doublets check for each new number that is smaller than the actual maximum if this new number would also be available in a previous step where actual a larger number is calculated. If this is the case the new number will be ignored. The backtracking ends with the last subtraction or addition. In the worst cases like 33! this happens in step 16.

Binary Log Code

Without this idea I would not have reached a score over 24.00 in the contest. With this code you can describe all fractions of n!. As there are only a few millions of fractions you can use tables to find the code for a certain number immediately. For factorials up to 28! you need 31 bits for the code (41 bits for 37! and 86 bits for 100!). For each prime a certain number of bits are reserved. Prime factors that can occur only one time like 23 or 19 need only one bit. Between neighbor primes you need an overflow bit that is always cleared and represented by "-" in the table. If a subtraction of log codes results in a value that has common bits with the overflow mask $(1010101010010010001000010000 0)_2$ the subtraction is not legal. The log code works like a logarithm, an addition of log codes is equal to a multiplication of the associated numbers.

Prime Flags

We describe all flags in a single integer. Even 100! has got only 25 prime factors. The prime flags do not only describe the primes in the actual number but also in the numbers of all previous steps. The prime flags were not changed by a multiplication.

Numbers that are not fractions of n!

If such a number occurs set the log code equal to the overflow mask. Such numbers can't be used for the final multiplications. In fact you should avoid such numbers. Only in the case of 33! I had to work with a non-fraction namely the prime 107.
(In the contest I found this sequence later than all others.)
[Some contestants like Thomas Rokicki found SLPs for 33! that consist only fractions of 33!.]
[Herbert Kociemba found out that my SLP for 22! contains the number 23279477760 which is not a fraction of 22!.]

Monotone increasing sequences

In the above shown example for 25! no subtraction is used and f(k) > f(k–1) for all k > 0. Therefore the sequence is monotone increasing. A score of about 23.00 would have been possible by using only such sequences. For 13! , 15! , 25! and 29! the minimum number of steps can be reached by monotone increasing sequences.

Consecutive [chaining] sequences

At least for small factorials you can find the shortest possible sequences by doing all new operations with the last element. That means for all integers k > 0: f(k+1) = f(k) +/–/× f(a) with 0 ≤ a ≤ k.
The above example for 25! fulfills this principle. It also works from 13! up to 23!. The first factorial of the contest where you can't find the shortest sequence with this principle seems to be 24!.
If you would have used this principle from step 7 to the last +/- operation you would have reached a score of 24.04.

Program output

The last +/– operation occurs at step 12. The sequence ends with 4 multiplications. Output of my program:

```
16 steps:  1,2,3,6,36,42,78,114,120,5040,604800,609840,609960,
           1*(12)=609960,1*(11)=609840,2*(10)=604800,1*(07)=114
```

That means: $25! = 604800^2 \cdot 609960 \cdot 609840 \cdot 114$

Improving sequences

If your backtracking works only until step 11 start a new backtracking by applying the first numbers of the found sequences. Now you can handle a few steps more. You can get our sequence with 16 steps from a sequence with 17 steps where the last +/– operation is already done in step 11. See that the first 9 numbers are equal:

17 steps: 1,2,3,6,36,42,78,114,120,5040, 5082, 5083 and $25! = 114 \cdot (5083 \cdot 120) \cdot (5082 \cdot 120) \cdot (5040 \cdot 120)^2$
16 steps: 1,2,3,6,36,42,78,114,120,5040,604800,609840,609960 and $25! = 114 \cdot 609960 \cdot 609840 \cdot 604800^2$

Walter Trump, 2013-04-20 [2013-04-21]

Shortest SLPs [just one sample for each n! – not all mentioned properties are represented] Walter Trump, 2013-04-20

n	steps	factorization of n!	sequence that generates the factors
13	11	$273\cdot275\cdot288^2$	1,2,4,16,15,18,288,273,275
14	11	$20160\cdot20020\cdot36\cdot6$	1,2,4,6,36,144,140,20160,20020
15	12	$6048\cdot6006\cdot6000\cdot6$	1,2,4,6,36,42,1512,6048,6006,6000
16	12	$144144\cdot144000\cdot1008$	1,2,4,5,7,12,144,1008,1001,144144,144000
17	12	$3740\cdot3744\cdot5040^2$	1,2,4,6,36,1296,1260,5040,3744,3740
18	13	$891072\cdot891000\cdot8064$	1,2,4,6,36,72,108,112,8064,7956,891072,891000
19	13	$2394\cdot7128576\cdot7128000$	1,2,4,6,24,576,600,2400,2394,2970,7128000,7128576
20	14	$6\cdot25194\cdot25344\cdot25200^2$	1,2,4,6,36,144,140,176,25344,25200,25194
21	14	$25840\cdot44478720\cdot44452800$	1,2,4,16,20,80,1600,1620,25920,25840,27440,44452800,44478720
22	14	$23279256000\cdot71850240\cdot672$	1,2,3,6,18,324,330,336,672,221760,71850240,23279477760,23279256000
23	15	$1428134400\cdot1432569600\cdot12636$	1,2,3,9,27,36,324,351,12636,4435236,4435200,1437004800,1432569600,1428134400
24	15	$11629094400\cdot4590\cdot11623772160$	1,2,4,6,24,96,2304,2280,2184,2310,4590,5322240,11623772160,11629094400
25	16	$114\cdot609960\cdot609840\cdot604800^2$	1,2,4,6,36,42,78,114,120,5040,604800,609840,609960
26	15	$3744216\cdot(3744000\cdot2772)^2$	1,2,3,6,36,38,216,1368,2736,2737,2772,3744216,3744000
27	16	$15237331200\cdot15190156800\cdot47044800$	1,2,3,6,18,324,322,342,360,363,129600,47044800,47174400,15190156800,15237331200
28	16	$342144\cdot326876\cdot(336960\cdot4900)^2$	1,2,4,8,64,72,70,4900,5184,331776,326876,336960,342144
29	17	$14616\cdot1404081\cdot(1404000\cdot14784)^2$	1,2,3,9,81,84,87,96,168,14616,14625,14784,1404000,1404081
30	17	$45356\cdot380\cdot(45760\cdot45360\cdot1890)^2$	1,2,4,5,20,24,400,380,378,1890,45360,45356,45760
31	18	$40398976800\cdot40239309600\cdot159667200\cdot31680$	1,2,4,6,36,1296,1260,5040,6336,25344,25346,31680,31935960,159667200,40239309600,40398976800
32	18	$20026\cdot20010\cdot150\cdot(20020\cdot20160\cdot144\cdot36)^2$	1,2,4,6,36,144,140,150,20160,20010,20020,20026
33	18	$2531847783628800\cdot025316575764480000$ $\cdot13547$	1,2,3,9,11,121,110,107,130,13310,13440,13547,120960,15724800,1902071808000, 25316575764480000,25318477836288000
34	17	$6661729122048000\cdot6661517403340800$ $\cdot6652800$	1,2,4,6,36,144,216,210,220,31680,31464,31824,6652800,211718707200, 6661517403340800,6661729122048000
35	19	$899\cdot874\cdot6683040^2\cdot6652800^3$	1,2,4,6,10,36,35,216,220,864,874,899,30240,6652800,6683040
36	18	$39970374732288000\cdot1398918654701568000$ $\cdot6652800$	1,2,4,6,36,35,216,220,864,899,30240,6652800,6683040,44460928512000, 39970374732288000,1398963115630080000,1398918654701568000
37	20	$1002478\cdot62640\cdot(238\cdot62400)^2\cdot(4158\cdot240)^3$	1,2,4,16,256,240,238,260,4160,4158,62400,62640,1002240,1002478