

## Find the smallest number $N$ with exactly $K$ factors

For any number  $N$  can be expressed as the the product of prime factors:

$$N = p_1 \times p_2 \times \cdots \times p_i$$

For example, when  $N$  is 12 we get:

$$12 = 2 \times 2 \times 3$$

By grouping the same primes together we can express this as:

$$N = p_1^{e_1} \times p_2^{e_2} \times \cdots \times p_i^{e_i}$$

Again, for example when  $N$  is 12:

$$N = 2^2 \times 3^1$$

We can now split each term into two components as follows:

$$N = p_1^{e_1-n_1} p_1^{n_1} \times p_1^{e_2-n_2} p_2^{n_2} \times \cdots \times p_i^{e_i-n_i} p_i^{n_i}$$

where  $n_i$  can be any value between 0 and  $e_i$ ; ie  $(e_i + 1)$  different values.

By rearranging the above into two terms we get:

$$N = (p_1^{n_1} \cdot p_2^{n_2} \cdots p_i^{n_i}) \times (p_1^{e_1-n_1} \cdot p_2^{e_2-n_2} \cdots p_i^{e_i-n_i})$$

where each of the terms represent ALL of the factors of  $N$ .

In our example where:

$$\begin{aligned} N &= 12 \\ &= 2^2 \times 3^1 \end{aligned}$$

and therefore:

$$\begin{aligned} e_1 &= 2 \\ e_2 &= 1 \end{aligned}$$

we can choose the  $n$ 's in the following ways:

$$\begin{aligned} 12 &= (2^0 \cdot 3^0) \times (2^2 \cdot 3^1) : n_1 = 0, n_2 = 0 \\ &= (1 \cdot 1) \times (4 \cdot 3) \\ &= (1) \times (12) \\ 12 &= (2^1 \cdot 3^0) \times (2^1 \cdot 3^1) : n_1 = 1, n_2 = 0 \\ &= (2 \cdot 1) \times (2 \cdot 3) \\ &= (2) \times (6) \\ 12 &= (2^2 \cdot 3^0) \times (2^0 \cdot 3^1) : n_1 = 2, n_2 = 0 \\ 12 &= (2^0 \cdot 3^1) \times (2^2 \cdot 3^0) : n_1 = 0, n_2 = 1 \\ 12 &= (2^1 \cdot 3^1) \times (2^1 \cdot 3^0) : n_1 = 1, n_2 = 1 \\ 12 &= (2^2 \cdot 3^1) \times (2^0 \cdot 3^0) : n_1 = 2, n_2 = 1 \end{aligned}$$

To find how many ways each term can be constructed for a given  $N$  (and therefore given  $e_i$ 's), we need to determine how many different ways the  $n$ 's can be chosen. Since each  $n_i$  can be any value from 0 to  $e_i$  or  $(e_i + 1)$  distinct values, the total number of combinations to choose all  $n$ 's is:

$$K = (e_1 + 1) \times (e_2 + 1) \times \cdots \times (e_i + 1)$$

This gives an expression for the total number of factors for  $N$ .

Using our example where  $N = 12$ ,  $e_1 = 2$  and  $e_2 = 1$ , we can calculate  $K$ :

$$\begin{aligned} K &= (2+1)(1+1) \\ &= 3 \times 2 \\ &= 6 \end{aligned}$$

We can confirm this result by counting the distinct factors of 12 we know from brute force to be:

$$1, 2, 3, 4, 6, 12$$

Now that we have an expression that relates exponents of the prime factors of  $N$  to the total number of factors, we can just it to find  $N$ s for a given  $K$ .

First note that  $K$  is expressed as a product of terms which means each term is a factor of  $K$ .  $K$  may have many factorisations, however if we are looking for the smallest  $N$ , we want our exponents to be as small as possible. Therefore the factorisation we choose will be the prime factorisation.

For example if  $K$  is 12 then:

$$\begin{aligned} 12 &= (e_1 + 1) \times (e_2 + 1)(e_3 + 1) \\ &= 3 \times 2 \times 2 \end{aligned}$$

given the prime factors of 12 are 2, 2, 3. This yields:

$$e_1 = 2, e_2 = 1, e_3 = 1$$

Plugging these values into our expression for  $N$  while selecting the smallest primes for the largest exponents (because we want  $N$  to be as small as possible), we get:

$$\begin{aligned} N &= q_1^{e_1} \times q_2^{e_2} \times q_3^{e_3} \\ &= 2^2 \times 3^1 \times 5^1 \\ &= 4 \times 3 \times 5 \\ &= 60 \end{aligned}$$

which can be verified by brute force methods.

## Numerical Results

### Smallest $N$ with $K$ factors - calculated by brute force

K	N Factors
A( 1)	1 [1]
B( 2)	2 [1, 2]
C( 3)	4 [1, 2, 4]
D( 4)	6 [1, 2, 3, 6]
E( 5)	16 [1, 2, 4, 8, 16]
F( 6)	12 [1, 2, 3, 4, 6, 12]
G( 7)	64 [1, 2, 4, 8, 16, 32, 64]
H( 8)	24 [1, 2, 3, 4, 6, 8, 12, 24]
I( 9)	36 [1, 2, 3, 4, 6, 9, 12, 18, 36]
J(10)	48 [1, 2, 3, 4, 6, 8, 12, 16, 24, 48]
K(11)	1024 [1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024]
L(12)	60 [1, 2, 3, 4, 5, 6, 10, 12, 15, 20, 30, 60]
M(13)	4096 [1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096]
N(14)	192 [1, 2, 3, 4, 6, 8, 12, 16, 24, 32, 48, 64, 96, 192]
O(15)	144 [1, 2, 3, 4, 6, 8, 9, 12, 16, 18, 24, 36, 48, 72, 144]
P(16)	120 [1, 2, 3, 4, 5, 6, 8, 10, 12, 15, 20, 24, 30, 40, 60, 120]

### Smallest $N$ with $K$ factors - calculated by factorising $K$

A( 1)	1
B( 2)	2
C( 3)	4
D( 4)	6
E( 5)	16
F( 6)	12
G( 7)	64
H( 8)	30
I( 9)	36
J(10)	48
K(11)	1024
L(12)	60
M(13)	4096
N(14)	192
O(15)	144
P(16)	210
Q(17)	65536
R(18)	180
S(19)	262144
T(20)	240
U(21)	576
V(22)	3072
W(23)	4194304
X(24)	420
Y(25)	1296
Z(26)	12288

## Python Code

### Find factors of lots of numbers then search for smallest $N$ for $K$

```
def factors(n):

    factors = [1]
    if n == 1:
        return factors

    for f in range(2,(n)//2 + 1):
        if not n%f:
            factors.append(f)
    factors.append(n)
    return factors

# factorise lots of number and count the factors
numbers = []
for n in range(1,13000):
    f = factors(n)
    numbers.append({
        "N":n,
        "K":len(f),
        "factors":f,
    })

print(f"    K          N Factors")
for k in range(1,27):          # A - Z
    letter = chr(64 + k)
    for n in numbers:
        if n["K"] == k:
            print(f"{letter}({k:2}){n[\"N\"]:8}", n["factors"])
            break
```

### Find smallest $N$ via prime factoring of $K$

```
small_primes = [2, 3, 5, 7, 11, 13, 17, 19, 23]
```

```
def prime_factors(n):
    factors = []
    for p in small_primes:
        while not n%p:
            n = n // p
            factors.append(p)
        if n == 1:
            return factors

print(F"{chr(64+1)}({1:2}){1:8}")
for k in range(2,27):
    N = 1
    j = 0
    for factor in reversed(prime_factors(k)):
        N = N * small_primes[j] ** (factor-1)
        j += 1
    print(F"{chr(64+k)}({k:2}){N:8}")
```