Sample: Algorithms Quantitative Methods - Fundamental Theorem of Arithmetic Algorithm

According to the fundamental theorem of arithmetic, we can write any integer k in the following form:

$$k = p_1^{\alpha_1} * p_2^{\alpha_2} * \dots * p_m^{\alpha_m} \quad (1)$$

Where $p_1, p_2, \dots, p_m$ are primes and $\alpha_1, \alpha_2, \dots, \alpha_m$ are positive integers.

Arrangements of the prime factors in k follow all the rules of permutations with repetitions. Thus, number of arrangements of the prime factors is:

$$n(k) = \frac{(\alpha_1 + \alpha_2 + \dots + \alpha_m)!}{\alpha_1!\,\alpha_2! \dots \alpha_m!} \quad (2)$$

The first observation we can make – number of arrangements does not depend on $p_1, p_2, \dots, p_m$. Thus, if values of $\alpha_1, \alpha_2, \dots, \alpha_m$ are known we should just sort it in descending order and use the first m primes for $p_1, p_2, \dots, p_m$ to get the smaller possible k (name this rule 1).

Example of using rule 1: if the powers are known to be {2, 5, 1} the smallest corresponding k equals $2^5 * 3^2 * 5^1$ (2, 3 and 5 are the first 3 primes).

So, to solve the problem it is enough to find $\alpha_1, \alpha_2, \dots, \alpha_m$ such that equation (2) holds.

When the rule 1 is used and sum of $\alpha_1, \alpha_2, \dots, \alpha_m$ is fixed (i.e., number of prime factors of k is fixed), the following is desirable (leads to smaller k). The value of α_1 is as large as possible, if α_1 reached its maximum, α_2 is as large as possible, etc. (name this rule 2).

Example of using rule 2: if sum of the powers is known to be 5 the smallest corresponding k equals 2^5, when the second or third powers are not zero, we get larger k: $2^4 * 3^1$ or $2^2 * 3^2 * 5^1$.

The rule 2 explains how to solve the problem if number of prime factors of k is given. Try to find limits of this number. It is clear that the lower limit is 1 (when k is prime). The upper limit is to be found from equation (2).

Fix the sum: $\alpha_1 + \alpha_2 + \dots + \alpha_m = \alpha$. Using properties of the factorial and the fact that $\alpha_1, \alpha_2, \dots, \alpha_m \geq 1$ we can write:

$$\min_{\alpha_1,\alpha_2,\dots,\alpha_m} \frac{\alpha!}{\alpha_1!\ \alpha_2! \dots \alpha_m!} = \min_{\alpha_1,\alpha_2,\dots,\alpha_m} \frac{\alpha!}{(\alpha - m + 1)!\ 1!\,1! \dots 1!} \geq \alpha$$

As a simple example, it is clear that:

$$\min_{1 \leq i \leq 5} \frac{5!}{i!\,(5-i)!} = \frac{5!}{4!} = 5$$

So, if we will choose $\alpha > n$, the equation (2) will newer hold.

Thus, range of the sums of $\alpha_1, \alpha_2, \dots, \alpha_m$ that is to be considered is $1..n$.

Now, the algorithm can be described.

1. Obtain a list of prime numbers. For the particular task this step should be done as pre-calculation step (before the test values of n are entered) to save the running time.
   The maximum prime to find is limited by the maximum k possible:

   $$p_{max} \leq \sqrt{k_{max}} = \sqrt{2^{63}} = 2^{31}\sqrt{2}$$

   A good algorithm for this step is Sieve of Eratosthenes.

   All the next steps are executed for each test separately (for each new n).

2. Find the set of integers $\alpha_1, \alpha_2, \dots, \alpha_m$ that satisfies (2) and (rule 2).

3. Use (rule 1) to find the minimum k.
Pseudo-codes for each step separately are shown below.

```
program Step1
Set A[i] values to true, i = 1.. p_max
for i = 2, 3, 4, ..., p_max
        if A[i] is true
                for j = i^2, i^2+i, i^2+2*i, ..., ≤ k_max
                        Set A[j] to false
                End for
        End if
End for
Declare PR for list of primes
Add 2 to PR
For i = 3, 5, 7, 9, …, ≤ k_max
        if A[i] is true
                add i to PR
        end if
end for
```

```
program Step2
declare a list of combinations α_1, α_2, …, α_m – Alphas (it is just an empty list now, something like list of arrays)
for m = 1 to n
        set currentCombination as array of m zeros.
        for  j = 1 to (n – m + 1)^m
                increment currentCombination(1)
                set i = 1
                while (currentCombination(i) > n-m+1)
                        set currentCombination(i) as 0;
                        increment currentCombination(i+1);
                        increment i
                end while
                (now currentCombination  = α_1, α_2, …, α_m)
                if (α_1 + α_2 + ⋯ + α_m ≤ n)
                        if equation (2) holds
                                sort the set (α_1, α_2, …, α_m) in non-increasing order
                                if α_1, α_2, …, α_m is not in Alphas
                                        add α_1, α_2, …, α_m to Alphas
                                end if
                        end if
                end if
        end for
        if Alphas is not empty
                break
```

end if
end for

program Step3

Set the minimum k equal to k_{max}

for each combination in Alphas

sort $\alpha_1, \alpha_2, \dots, \alpha_m$ in descending order

Set $p_1, p_2, \dots, p_m$ as the firs m elements of PR

Set $k = p_1^{\alpha_1} * p_2^{\alpha_2} * \dots * p_m^{\alpha_m}$

If (k < minimum k)

Set minimum k to be equal to k

End if

end for