US Solution:

Trước tiên chúng ta lập hai mảng:

+) S[1], S[2], ..., S[N] (ở đây $N=10^6$) với S[i] là số lượng các ước dương của i. Ta có công thức: S[1]=1; S[x]=(k+1).S[y] Ở đây $x = p^k.y$ và y không chia hết cho p (p là ước nguyên tố của x (SGK Chuyên tin tập 1). Chú ý rằng sử dụng sàng số nguyên tố ta có thể lấy p=E[p]

+) T[1], T[2], .., T[N] với T[i] là tổng các ước của i. Mảng T[i] cũng được lập bởi công thức qui hoạch động:

T[1]=1, T[x]=($1+p+p^2+...+p^k$).T[y] với $x = p^k y$

Sau đó với mỗi truy vấn bài toán trở thành tính tổng các phần tử liên tiếp trên mảng S và trên mảng T. Sử dụng kỹ thuật lập mảng tổng tiền tố để thực hiện điều này trong O(1)

ONES Solution:

Nhận xét rằng với mỗi số nguyên tố p thì số lần lớn nhất thực hiện được phép chia cho p đúng bằng lũy thừa cao nhất của p trong phân tích các số thành thừa số nguyên tố.

Vậy nên thuật toán có thể phát biểu đơn giản là với mỗi số nguyên tố trong phạm vi từ 1 đến 10^6 tìm lũy thừa lớn nhất của nó trong các phân tích $a_1, a_2, \dots, a_n$ thành thừa số nguyên tô. Đáp số là tổng các giá trị này.

Để thuật toán có hiệu quả về thời gian chú ý rằng trong thuật toán sàng số nguyên tố thì p là số nguyên tố khi và chỉ khi E[p]=p.

Do đó mỗi khi phân tích một số a thành thừa số nguyên tố p thì chỉ cập nhật lũy thừa max vào b[p] (khởi đầu các giá trị này bằng 0)

LCM Solution:

Trước tiên chúng ta phân tích $T = a \times (a+1) \times \dots \times b$ thành thừa số nguyên tố:

$$T = p_1^{x_1} \times \dots \times p_m^{x_m}$$

Ở đây $p_1, p_2, \dots, p_m$ là các số nguyên tố trong khoảng [1,10^6]

Mỗi cặp (x,y) có LCM(x,y)=T thì trong các phân tích x, y thành thừa số nguyên tố thì với số nguyên tố p_i phải có một lũy thừa bằng x_i (giá trị thứ hai có thể có $x_i + 1$) khả năng 0,..., x_i.

Do vậy số cặp (x,y) tìm được là:

$$(x_1 + 1) \cdot (x_2 + 1) \dots (x_m + 1)$$

PMOVE Solution:

Giả sử ta có các phân tích thành thừa số nguyên tố:

$$x_1 = p_1^{k_{11}} \times \dots \times p_m^{k_{1m}}$$
$$x_2 = p_1^{k_{21}} \times \dots \times p_m^{k_{2m}}$$
$$....$$
$$x_n = p_1^{k_{n1}} \times \dots \times p_m^{k_{nm}}$$

Và vị trí x_0 là vị trí tìm được có

$$x_0 = p_1^{k_{01}} \times \dots \times p_m^{k_{0m}}$$

Thì các giá trị $(k_{01}, \dots, k_{0m})$ phải làm cực tiểu hóa các biểu thức:

$$|k_{01} - k_{11}| + |k_{01} - k_{21}| + \dots + |k_{01} - k_{n1}|$$
$$....$$
$$|k_{0m} - k_{1m}| + |k_{0m} - k_{2m}| + \dots |k_{0m} - k_{nm}|$$

Dễ thấy các giá trị này là các giá trị trung vị trong các dãy:

$$(k_{11}, k_{21}, \dots, k_{n1})$$
$$\dots$$
$$(k_{1m}, k_{2m}, \dots, k_{nm})$$
Chú ý rằng nếu tìm trung vị bằng cách sử dụng lệnh sort của C++ hay quick sort thì sẽ quá thời gian các test cuối. Tuy nhên do số mũ không lớn (không vượt quá 100) nên đơn giản chỉ cần sử dụng kỹ thuật đếm phân phối.

NGAME Solution:

Dễ thấy rằng để số lượt chơi nhiều nhất có thể, mỗi lần chỉ thực hiện chia giá trị đang có x cho một ước nguyên tố của nó. Đặt $f(x)$ là hàm cho số lượng nhiều nhất đối với x nếu:
$$x = p_1^{k_1} \times \dots \times p_r\text{^}(k_r)$$
thì $f(x) = k_1 + \cdots + k_r$. Ngoài ra nếu $x = a.b$ thì $f(x) = f(a) + f(b)$.
Theo đề bài:
$$n = \frac{a!}{b!} = (b+1).(b+2)\dots a$$
Do đó $f(n) = f(b+1) + f(b+2) + \cdots + f(a)$
Bài toán được giải quyết nếu ta xây dựng được mảng f[1], f[2], ..., f[10^6]. Khi đó mỗi truy vấn sẽ qui về việc tính tổng một đoạn liên tục và bằng cách sử dụng mảng tổng tiền tố có thể thực hiện truy vấn trong thời gian O(1).
Để tính mảng f[...] ta có:

- f[1]=0
- f[x]=f[y]+1 với x>2 và x=E[x].y (Chú ý E[x] là ước nguyên tố của x)

Ở đây E[1], ..., E[10^6] là mảng sàng số nguyên tố