Editorial Tarea 5

Profesor: Andrés Abeliuk
Auxiliares: Daniel Báez
Julieta Coloma
Máximo Flores Valenzuela
Blaz Korecic
Diego Salas

C. Apocalipsis

Podemos armar un grafo donde tendremos n nodos con valores a_i para $i = 1, \ldots, n$, y **nodos especiales** con los factores primos de cada a_i. Se puede demostrar que cada a_i tiene a lo más $\log_2 n$ factores primos.

Dem. Sea $n \in \mathbb{N}$ con descomposición en k números primos $n = \Pi_{i=1}^{k} p_i$. Podemos ver que el menor valor de n es cuando $p_i = 2$, $\forall i \in \{1, \ldots, k\}$. En dicho caso tenemos la igualdad $n = 2^k$, lo que implica directamente $\log_2 n = k$.

Ahora bien, tomemos el caso $n \neq 2^k$. La única forma que esto suceda es que exista algún factor primo mayor a 2. Tendremos lo siguiente:

$$n = \prod_{i=1}^{k} p_i \implies \log_2 n = \sum_{i=1}^{k} \log_2 p_i > k$$

La última desigualdad es porque para cualquier factor primo mayor que 2, $\log_2 p_i > 1$.

Ahora bien, ¿para qué nos sirve este análisis? Si modelamos el problema como un grafo, la respuesta se reduce a hacer un BFS desde s a t para encontrar la distancia mínima, como lo solemos hacer. Sabemos que la complejidad de aplicarlo es $\mathcal{O}(|V|+|E|)$ donde V es el conjunto de vértices y E el de aristas. Si hacemos un grafo por fuerza bruta, es decir, conectando todos los nodos que cumplan $\gcd(a_i, a_j) > 1$, tendremos aprox. $n(n-1)/2 \sim n^2$ combinaciones en el peor de los casos. Esto por tiempo no pasa.

La implementación descrita al principio tiene una complejidad $\mathcal{O}(n \log n)$. En código se puede realizar como sigue:

- Notemos que $N_{\text{máx}} = a_{\text{máx}} = 3 \cdot 10^5$, esto es, el nodo que represente el a_i más grande tendrá el valor $3 \cdot 10^5$.
- Los números primos dentro de un rango los podemos calcular usando la criba.
- Podemos implementar los factores primos de cada a_i con un desplazamiento para no repetir nodos. Por ejemplo, si $\exists j, a_j = 2$, el 2 es factor primo de sí mismo, pero no

podemos tener dos nodos con la misma etiqueta. Así que al nodo especial le pondremos $3 \cdot 10^5 + 2$ (sabemos que no estará ocupado).

- Hacemos el grafo conectando a cada a_i con sus nodos especiales representantes de sus factores primos de manera bidireccional.
- Al hacer el BFS, sumamos 1 a la respuesta sólo si el valor del nodo es menor o igual a $3 \cdot 10^5$ (porque sabemos que si es mayor, es un nodo especial, y esos no cuentan en el problema, sólo son para realizar las conexiones que cumplan la condición del gcd).

Otra implementación cambia sólo el último paso, podemos hacer el BFS sin revisar condiciones e imprimir la respuesta dada, pero dividida en 2, para quitar todos los nodos especiales.

D. Juego de divisores

En este problema nos pasa que n es 10^{12}, esto nos dice que:

- No podemos iterar por cada número.
- Nos sugiere ir por algo de $\mathcal{O}(\sqrt{n})$.

En vez de analizar cada número, veamos los divisores. Tomemos el caso de $n = 20$ y vamos (por cachativa) analizando los divisores presentes en la suma:

```
1   1
2   1 2
3   1 3
4   1 2 4
5   1 5
6   1 2 3 6
7   1 7
8   1 2 4 8
9   1 3 9
10  1 2 5 10
```

```
11  1 11
12  1 2 3 4 6 12
13  1 13
14  1 2 7 14
15  1 3 5 15
16  1 2 4 8 16
17  1 17
18  1 2 3 6 9 18
19  1 19
20  1 2 4 5 10 20
```

Nos fijamos que tenemos estos divisores:

```
divisor    1  2  3  4  5  6  7  8  9  10  11 12 13 14 15 16 17 18 19 20
ocurrencia 20 10 6  5  4  3  2  2  2  2   1  1  1  1  1  1  1  1  1  1
```

Esto es que **para cada divisor i** este ocurre floor($\frac{n}{i}$) veces. Ahora, con esta información, podríamos responder que la suma es $\sum_{i=1}^{n} \text{floor}(\frac{n}{i})$ pero esto todavía tomaría $\mathcal{O}(n)$.

Tenemos que reducir más el tiempo de cálculo. Notemos que hay valores que se repiten. Entre el 7 y el 10 el 2 se repite, y desde el 11 al 20 se repite el 1.

Podríamos identificar que tan largos son estos bloques y calcularlos de antemano. Esta solución es óptima y nos tomaría $\mathcal{O}$(cantidad de valores distintos para $\frac{n}{i}$).

Sea $q = \text{floor}(\frac{n}{i})$, para obtener el último valor i que cumple con $\text{floor}(\frac{n}{i}) = q$ tomamos el **i** que cumple con $\text{floor}(\frac{n}{i}) = q$. Este **i** es $\text{floor}(\frac{n}{q})$.

divisor	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
ocurrencia	20	10	6	5	4	3	2	2	2	2	1	1	1	1	1	1	1	1	1	1

```
i = 6  ->  floor(n/6) = 3 ^ floor(n/3) = 6
```

```
i = 7  ->  floor(n/7) = 2 ^ floor(n/2) = 10
```

Figure 1: Ejemplo para $i = 6$ e $i = 7$

Así obtenemos los limites de los valores repetidos. Luego es cosa de iterar de 1 a n, sumando $\text{floor}(\frac{n}{i}) \cdot i$ e identificando los bloques de números repetidos cuando $\text{floor}(\frac{n}{i}) \neq \text{floor}(\frac{n}{q})$

E. Alucinando con el GCD

Para este problema, calcularemos $G(n) = \sum_{i=1}^{n} \gcd(i, n)$ para cada n, luego, podremos usar la suma de los valores hasta ese n para que el valor de la consulta sea eficiente, se puede observar que: $\sum_{i=1}^{n} G(n)$ es equivalente a la respuesta.

Con esto, nos queda encontrar la forma de calcular $G(n)$ de manera eficiente. Se observa que el calculo de $G(n)$ puede realizar también como la suma de todos los posibles $\gcd(n)$ con otro número multiplicado por la cantidad de veces que se repite ese gcd con los valores menores a él. $\sum_{d \in D(n)} d \cdot \text{ocurr}(n)$ donde $D(n)$ son los divisores de n y $\text{ocurr}(n)$ es la ocurrencia de d como gcd en n.

Resulta que si $\gcd(i, j) = g$, entonces $\gcd(i/g, j/g) = 1$, llegando a que: $\sum_{d \in D(n)} d \cdot \text{ocurr}(n) = \sum_{d \in D(n)} d \cdot \phi(n/d)$, en donde ϕ son los primos relativos a n menores a n (son los i tal que $\gcd(i, n) = 1$, con lo que podemos desde los divisores calcular la suma de los valores, pasando iterativamente por todos los números por los que un i es divisor (de forma análoga a la criba).

Solo recorriendo los múltiplos de estos números como en la criba, podemos llegar a la solución en la misma complejidad, siendo esta $O(n \cdot \log(\log(n)))$, esto se puede hacer de la siguiente forma: recorremos cada número, identificando aquellos por los que aún no pasamos como números primos, y desde ellos hacer los cálculos de la sumatoria de la solución.