Soluciones entrenos OIE

Deciphering the Mayan Writing

Descifrar la escritura maya es más difícil de lo que pronosticaron las primeras investigaciones. Después de casi 200 años hubo muy pocos progresos, y no ha sido hasta las últimas decadas que se han hecho avances meritorios.

La escritura maya está basada en unos pequeños dibujos llamados glifos que representan sonidos. Las palabras mayas normalmente se escriben como combinación de varios glifos.

Uno de los problemas a la hora de descifrar la escritura maya viene a la hora de decidir el orden de lectura. Para decidir la posición de los glifos, los mayas muchas veces se basaban en su propia visión de la estética más que en cualquier regla particular. Este hecho hace que, aunque se sepa pronunciar los glifos, muchas veces los arqueólogos no están seguros de cómo pronunciar las palabras.

Los arqueólogos están buscando una palabra especial W. Saben los glifos que contiene, pero no saben de cuántas maneras los pueden ordenar. Te piden ayuda. Te darán g glifos de la palabra W y una secuencia S de todos los glifos (en el orden en que aparecen) del fragmento que están estudiando ahora mismo. Ayúdales a contar el número de posibles apariciones de la palabra W en el fragmento.

Problema

Escribid un programa que, dados los glifos de W y la secuencia S de glifos en el fragmento de estudio, cuente el número de posibles ocurrencias de W en S, es decir, todas las secuencia de g glifos consecutivos en S tales que son una permutación de los glifos de W.

Entrada y salida

La entrada consiste en una 3 líneas. En la primera hay dos enteros g, n. En la segunda, los g glifos de W, y en la última la secuencia S de longitud n. Todos los glifos son letras minúsculas ('a'-'z') o mayúsculas ('A'-'Z').

Imprimid una única linea con el número de posibles ocurrencias.

Constraints

- $1 \leq g \leq 3000$
- $g \leq |S| \leq 3000000$

Subtareas y puntuación

- (50 puntos) $g \leq 10$
- (50 puntos) Sin restricciones adicionales.

Solución

La primera solución que nos puede venir a la cabeza es probar todas las permutaciones de W y para cada una, miramos cuántas veces coincide esa permutación con la secuencia S. Para el número de coincidencias, iteramos por todos los posibles inicios de W en la secuencia S. Obviamente esta solución es muy lenta. Para optimizarla, podemos darnos cuenta de que no nos importa el orden de los glifos en W, solo la cantidad que hay de cada. Así que podemos contar cuántas veces aparece cada glifo en W. Ahora, iteramos por todas las posibles posiciones de inicio i de W en S y contamos el número de veces que aparece cada glifo en la subsecuencia $S[i, i+1, \ldots, i+g-1]$. Por último, si las frecuencias de cada glifo coinciden con las de W, sumaremos uno a nuestra respuesta. Esta solución tiene complejidad $O(gn)$, así que sigue siendo lenta para obtener todos los puntos, pero es suficiente para obtener los primeros 50.

La observación clave para optimizar esta solución es darse cuenta de que cuando contamos la frecuencia de cada glifo en $S[i, i+1, \ldots, i+g-1]$ y en $S[i+1, i+2, \ldots, i+g]$, estamos repitiendo muchos cálculos que podemos reciclar. De hecho, para contar la frecuencia de cada glifo en $S[i+1, i+2, \ldots, i+g]$, podemos tomar las frecuencias de $S[i, i+1, \ldots, i+g-1]$, restar uno a la frecuencia de $S[i]$, y sumar uno a la de $S[i+g]$. Teniendo en cuenta esto, la solución sería la siguiente: Inicializamos un vector 52 posiciones (número de glifos distintos) a 0, iteramos por los glifos de W y contamos cuántos hay de cada. Ahora iteramos por los primeros g glifos de S y vamos restando uno al vector de frecuencias por cada glifo que nos encontramos. Si este vector está todo lleno de 0, es que las frecuencias coinciden, y hay una posible aparición de W en $S[0, 1, \ldots, g-1]$. Si no, significa que hay algún número distinto de cero en el vector. Si es positivo, significa que aparece más veces en W que en $S[0, 1, \ldots, g-1]$ y si es negativo al revés.

Ahora vamos iterando desde $i = g$ hasta $i = n-1$ y vamos actualizando el vector, añadiendo el glifo en la posición i y quitando el de la $i-g$. Después de cada iteración comprobamos si el vector tiene todo ceros, y en caso afirmativo, sumamos uno a la respuesta. La manera naif de comprobarlo es iterando las 52 posiciones, pero se puede hacer mejor si guardamos en una variable cuantas posiciones son cero a cada iteración. Esta variable la actualizaremos después de cada iteración si una posición ha pasado a ser un cero o ha dejado de serlo.

Código (C++)

El siguiente código de ejemplo obtiene 100 puntos.

```cpp
#include <bits/stdc++.h>

using namespace std;

int f(char c) { // codificamos los glifos con un entero entre 0 y 51
        if (c >= 'a' && c <= 'z') return c-'a';
        return c-'A'+'z'-'a'+1;
}

int main() {
        ios::sync_with_stdio(0);
        cin.tie(0);
        int g, n;
        cin >> g >> n;
        string w, s;
        cin >> w >> s;
        int m = f('Z')+1;
        vector <int>v(m, 0);
        // frecuencias en W
        for (int i = 0; i < g; ++i) ++v[f(w[i])];
        // restamos frecuencias en S[0...g-1]
        for (int i = 0; i < g; ++i) --v[f(s[i])];
        int ceros = 0; // cantidad de posiciones que son cero
        for (int i = 0; i < m; ++i) {
                if (v[i] == 0) ++ceros;
        }
        int ans = 0;
        if (ceros == m) ++ans;
        for (int i = g; i < n; ++i) { // iteramos los posibles inicios de W en S
                ++v[f(s[i-g])];
                if (v[f(s[i-g])] == 0) ++ceros;
                else if (v[f(s[i-g])] == 1) --ceros;
                --v[f(s[i])];
                if (v[f(s[i])] == 0) ++ceros;
                else if (v[f(s[i])] == -1) --ceros;
                if (ceros == m) ++ans;
        }
        cout << ans << endl;
}
```