Soluciones entrenos OIE

Combo

Estás jugando a un video juego. El control del juego tiene botones A, B, X e Y. En este juego, puedes hacer combos para conseguir monedas. Para hacer un combo debes presionar una secuencia de botones.

Este juego contiene una secuencia secreta de botones, que puede ser representada mediante una cadena S de esos 4 caracteres. No conoces la cadena S, pero sí conoces su longitud N.

También sabes que el primer caracter de S nunca vuelve a aparecer en la cadena. Por ejemplo, S puede ser "ABXXY" o "XYYA", pero no puede ser "AAAAAA" o "BXYBX".

Puedes presionar una secuencia de como mucho $4N$ botones para hacer un combo. Sea p la cadena que representa la secuencia de botones que presionaste. La cantidad de monedas que obtienes por este combo se calcula como la longitud del prefijo más largo de S que también es subcadena de p. Una subcadena de una cadena t es una secuencia contigua (posiblemente vacía) de caracteres en t. Un prefijo de una cadena t es una subcadena de que contiene a su primer caracter, o es vacía.

Por ejemplo, si S es "ABXYY" y p es "XXYYABYABXAY", se obtienen 3 monedas porque "ABX" es el prefijo más largo de S que también es una subcadena de p.

Debes determinar la cadena secreta S realizando pocos combos.

Debes implementar la función *guess_sequence(int N)* de modo que dado N devuelva la cadena secreta S. Para ello puedes hacer llamadas a la función *press(string p)* que dada una cadena p te devuelve el número de monedas que obtienes al realizar ese combo.

Constraints

$1 \leq N \leq 2000$

Cada caracter de la cadena S es A, B, X o Y.

El primer caracter de S nunca reaparece en S.

- 5 puntos: $N = 3$
- Ninguna restricción adicional. Para esta subtarea, si q es la cantidad de llamadas a *press*, la puntuación de cada caso de prueba será:
  * 95 puntos si $q \leq N + 2$
  * $95 - 3(q - N - 2)$ puntos si $N + 2 \leq q \leq 2N + 1$
  * 25 puntos si $\max\{N + 10, 2N + 1\} \leq q \leq 4N$
  * 0 puntos en cualquier otro caso.

Solución

Vamos a ver 2 soluciones: una solución parcial (30 puntos) y la completa. Ambos códigos están la siguiente sección.

En primer lugar necesitamos saber cual es el primer caracter de la secuencia. Esto lo podemos hacer fácilmente con 3 llamadas a *press*, la llamamos con $p =$ "A", con $p =$ "B" y con $p =$ "X". Si alguna de estas llamadas ha devuelto una moneda, significa que la palabra secreta empieza por esa letra, si en ninguna de las llamadas hemos recibido una moneda entonces la palabra empieza por Y.

A partir de aquí cada caracter solo puede ser uno de los 3 distintos al primero, por lo que si t es lo que sabemos de momento de S y t tiene m caracteres, podemos llamar a t añadiendo uno de esos 3 caracteres y hacer otra llamada con t añadiendo otro de los 3 caracteres. De este modo si alguna de las dos llamadas devuelve $m + 1$ monedas querrá decir que S sigue con ese caracter. Si todas las llamadas devuelven m monedas querrá decir que S sigue con el caracter por el que no hemos preguntado. Por ejemplo:

Supongamos $S =$ "YBX". Para saber el primer caracter llamamos a *press* con $p =$ "A", nos devuelve 0 con lo que A no es el primer caracter de S, luego llamamos a *press* con $p =$ "B", nos devuelve 0 con lo que B no es el primer caracter de S, finalmente llamamos a *press* con $p =$ "X" y nos devuelve 0 con lo que X no es el primer caracter de S. Por lo tanto el primer caracter de S será Y.

Para saber el segundo carater preguntamos por $p =$ "YA" el resultado es 1 con lo que A no es segundo caracter de S. Luego preguntamos por $p =$ "YB", lo que nos devuelve 2, con lo que el segundo caracter de S será B. Para saber el último caracter de S, preguntaríamos primero por $p =$ "YBA" nos devuelve 2 y luego por YBB también nos devuelve 2 con lo que el tercer caracter no es ni A ni B, como tampoco puede ser Y (el primer caracter no vuelve a aparecer) entonces el tercer caracter es X.

En el peor caso acabaremos haciendo $2N + 1$ llamadas a *press*, con lo que obtendremos 30 puntos.

¿Como podemos hacer menos llamadas a *press*? En primer lugar, podemos saber cual es el primer caracter con solo dos llamadas a *press*. Primero con $p =$ "AB" si el resultado es > 0, entonces o A o B son el primer caracter, si no lo serán o X o Y. Con lo que tras una llamada solo nos quedarán dos candidatos a ser el primer caracter. Hacemos una llamada con uno de los candidatos; si el resultado es 1, ese es el primer caracter, si no, el primer caracter será el otro candidato.

Para los demás caracteres, sea t lo que conocemos de S hasta ahora y sea m el número de caracteres de t, solo tenemos 3 candidatos a ser el siguiente caracter c_1, c_2, c_3, la idea es llamar a *press* con $p = tc_1tc_2c_1tc_2c_2tc_2c_3$. Vemos que como mucho el número de monedas que recibiremos será $m + 2$, ya que sinó, el primer caracter de S (que es el primer caracter de t) se repetiría en S, y eso no puede suceder, tal y como se indica en el enunciado. Además el mínimo número de monedas que nos pueden devolver es m, ya que t es una subcadena de p formada por los primeros m caracteres de S.

Si la función nos devuelve m, el siguiente caracter será c_3, ya que p cotiene la cadena t seguida de c_1 y también la cadena t seguida de c_2, de modo que si c_1 o c_2 fuesen el siguiente caracter de S, tendríamos que haber recibido como mínimo $m + 1$ monedas. Si la función nos devuelve $m + 1$, el siguiente caracter es c_1, ya sabemos que no puede ser c_3, y tenemos una subcadena de S formada por t seguida de c_2c_1, otra seguida de c_2c_2 y otra seguida de c_2c_3, si el siguiente caracter de S fuese c_2, como después vendría o c_1 o c_2 o c_3 nos tendrían que haber devuelto $m + 2$ monedas, no $m + 1$. Finalmente si la función nos devuleve $m + 2$ monedas el siguiente caracter es c_2.

Este procedimiento lo podemos hacer con todas las posiciones salvo la última, pues nos parasíamos de $4N$ caracteres, en ese caso, resolveremos el problema como hacíamos en el caso de 30 puntos.

El número total de llamadas a *press* será de $N + 2$ ya que para la primera y última moneda tenemos que hacer dos llamadas en vez de una.

Código

30 puntos

```cpp
#include <iostream>
#include <vector>

using namespace std;

string guess_sequence(int N){
    string t; // Aquí iremos construyendo S.

    // Buscamos el primer caracter de S.
    if (press("A") == 1) t.push_back('A');
    else if (press("B") == 1) t.push_back('B');
    else if (press("X") == 1) t.push_back('X');
    else t.push_back('Y');

    if (N == 1) return t; // Si S tiene un caracter ya hemos acabado

    string sequence = "ABXY";
    vector <string> left; // Aquí guardamos los tres caracteres que pueden salir a partir de ahora, en formato de palabra (nos será más cómodo)
    for (int i = 0; i < sequence.size(); ++i){
        if (sequence[i] != t[0]) { // El caracter sequence[i] es distinto al primer caracter de S
            string word;
            word.push_back(sequence[i]);
            left.push_back(word);
        }
    }

    while (t.size() < N){
        if (press(t + left[0]) == t.size() + 1) t += left[0];
        else if (press(t + left[1]) == t.size() + 1) t += left[1];
        else t += left[2];
    }

    return t;
}
```

100 puntos

```cpp
#include <iostream>
#include <vector>

using namespace std;

string guess_sequence(int N){
    string t;// Aquí iremos construyendo S.

    // Buscamos el primer caracter de S.
    if (press("AB") > 0){
        if (press("A") > 0) t.push_back('A');
        else t.push_back('B');
    }
    else {
        if (press("X") > 0) t.push_back('X');
        else t.push_back('Y');
    }

    if (N == 1) return t;

    string sequence = "ABXY"; // Si S tiene un caracter ya hemos acabado
    vector <string> left; // Aquí guardamos los tres caracteres que pueden salir a partir de ahora, en formato de palabra (nos será más cómodo)
    for (int i = 0; i < sequence.size(); ++i){
        if (sequence[i] != t[0]) { // El caracter sequence[i] es distinto al primer caracter de S
            string word;
            word.push_back(sequence[i]);
            left.push_back(word);
        }
    }

    vector <string> ends = {left[1], left[2] + left[0], left[2] + left[1], left[2] + left[2]}; // Ends contiene todos los sufijos que tendremos que añadir a t: c1, c2c1, c2c2 y c2c3

    while (t.size() < N - 1){
        int value = press(t + ends[0] + t + ends[1] + t + ends[2] + t + ends[3]) - t.size();
        if (value == 0) t += left[0];
        if (value == 1) t += left[1];
        if (value == 2) t += left[2];
    }

    // El último caracter lo tratamos a parte
    if (press(t + left[0]) == N) t += left[0];
    else if (press(t + left[1]) == N) t += left[1];
    else t += left[2];

    return t;
}
```