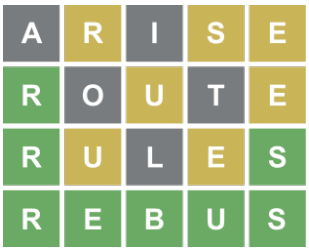
Wordle numérico



Wordle es un juego desarrollado en 2021 por Josh Wadler para la web. Consiste en adivinar una palabra secreta de cinco letras utilizando, como máximo, seis intentos. Tras cada intento, se ilumina cada una de sus letras de color verde, amarillo, o gris. El color verde significa que la letra correspondiente del intento se encuentra en la palabra secreta, y en esa misma posición. El color amarillo significa que la letra se encuentra en la palabra secreta, pero no en la misma posición. El color gris significa que la letra no se encuentra en la palabra secreta.

En esta práctica consideramos una variante de Wordle en la que, en lugar de adivinar una palabra secreta, hay que adivinar una secuencia de números enteros. Además, la secuencia puede tener cualquier longitud. Por ejemplo, si la secuencia secreta es `10 7 9 4 1 5` y el jugador introduce el intento `3 7 10 2 9 6`, los números deben iluminarse del siguiente modo:

Secuencia secreta:	**10**	**7**	**9**	**4**	**1**	**5**
Intento:	3	7	10	2	9	6

Hay una dificultad adicional: tanto la secuencia secreta como los intentos del jugador pueden contener números repetidos. En este caso, las reglas que determinan cómo iluminar los números del intento son las siguientes:

1. Si un número aparece *m* veces en la secuencia secreta, solo se iluminarán como mucho *m* repeticiones de ese número en el intento. En el siguiente ejemplo, aunque el intento tiene tres doses (2), solamente se iluminan los dos primeros, ya que la secuencia secreta tiene dos doses.

Secuencia secreta:	**1**	**5**	**4**	**2**	**2**	**3**
Intento:	2	2	2	7	9	8

2. A la hora de iluminar números repetidos, tienen prioridad las repeticiones que aparezcan en el lugar correcto de la secuencia y que, por tanto, se iluminarán de color verde:

Secuencia secreta:	**1**	**5**	**4**	**2**	**2**	**3**
Intento:	2	2	7	2	9	8

3. En cuanto a las repeticiones restantes, tienen prioridad para iluminarse de color amarillo aquellas que aparecen más a la izquierda en el intento. En el ejemplo anterior, se ha iluminado de amarillo la primera aparición del número 2, y no la segunda.

El objetivo de esta práctica es determinar cómo deben iluminarse los números de un intento, suponiendo que se conoce la secuencia secreta. Para ello nos apoyaremos en el TAD Multiconjunto.

Paso 1 - Implementación del TAD Multiconjunto

En matemáticas, un multiconjunto es una colección de elementos similar a un conjunto, con la diferencia de que en un multiconjunto pueden aparecer elementos repetidos. Para diferenciarlos de los conjuntos habituales, se utilizan dobles llaves para describir multiconjuntos. Por ejemplo, $\{\{2, 2, 3\}\}$ y $\{\{1, 5, 5, 3, 2, 2, 2\}\}$ son multiconjuntos.

Un multiconjunto de números enteros puede implementarse mediante un vector que almacena los elementos del multiconjunto, cada uno de ellos asociado con su multiplicidad (es decir, el número de veces que aparece).

```
const int MAX_ELEMS = 2000;

class Multiconjunto {
public:
  // ...
  // Interfaz
  // ...
private:
  struct Elem {
    int valor;
    int multiplicidad;
  };
  Elem elems[MAX_ELEMS];
  int num_elems;
}
```

Cada elemento del vector `elems` está compuesto por dos campos: el valor del elemento propiamente dicho, y su multiplicidad. Por otro lado, el atributo `num_elems` indica cuántas posiciones del vector `elems` están ocupadas. De este modo, se considera que las posiciones del intervalo $[0..\texttt{num\_elems})$ del vector están ocupadas.

Por ejemplo, el multiconjunto $\{\{5, 5, 8\}\}$ puede representarse mediante un objeto `Multiconjunto` cuyo atributo `num_elems` tiene valor 2, `elems[0]` es un objeto de tipo `Elem` con atributos `valor = 5`, `multiplicidad = 2`, y `elems[1]` es otro objeto con atributos `valor = 8`, `multiplicidad = 1`. El resto de posiciones del array `elems` contienen valores indefinidos.

Utilizaremos el siguiente invariante de representación I, donde `m` es un objeto de la clase `Multiconjunto`:

$$\begin{aligned} I(\texttt{m}) \quad \equiv \quad & 0 \leq \texttt{m.num\_elems} \leq \texttt{MAX\_ELEMS} \\ & \wedge \forall i, j : 0 \leq i < j < \texttt{m.num\_elems} \Rightarrow \texttt{m.elems}[i].\texttt{valor} < \texttt{m.elems}[j].\texttt{valor} \\ & \wedge \forall i : 0 \leq i < \texttt{m.num\_elems} \Rightarrow \texttt{m.elems}[i].\texttt{multiplicidad} > 0 \end{aligned}$$

Es decir, los objetos `Elem` están ordenados en el array `elem` de manera ascendente según el atributo `valor`, y las multiplicidades son siempre estrictamente mayores que cero.

1. Implementa las siguientes operaciones en el TAD Multiconjunto:

```
{  true  }
Multiconjunto()   // constructor: crea un multiconjunto vacío
{  this = ∅  }
```

```
{  this = {{x_1,...,x_n}}  }
void anyadir(int elem)   // añade un elemento al multiconjunto
{  this = {{x_1,...,x_n}} ∪ {{elem}}  }

{  this = {{x_1,...,x_n}}  }
void eliminar(int elem)   // elimina un elemento del multiconjunto
{  this = {{x_1,...,x_n}} − {{elem}}  }

{  this = {{x_1,...,x_n}}  }
bool pertenece(int elem)   // comprueba si un elemento está en el multiconjunto
{  result = elem ∈ {{x_1,...,x_n}}  }
```

En la operación `eliminar`, si el elemento pasado como parámetro se encuentra varias veces en el multiconjunto, solamente se elimina una de las apariciones. Si no se encuentra en el multiconjunto, la operación no realiza nada.

2. Indica el coste en tiempo de cada una de las operaciones.

Paso 2 - Uso del TAD Multiconjunto

Escribe un programa que, dadas una secuencia secreta y un intento, determine cómo deben iluminarse los números del intento. En las siguientes secciones se describe el formato de la entrada y la salida.

Entrada

La entrada consta de una serie de casos de prueba. Cada caso de prueba consta de tres líneas. La primera línea contiene el número M de elementos de la secuencia secreta y del intento ($1 \leq M \leq 2000$). La segunda línea contiene los M números que describen la secuencia secreta, y la tercera línea contiene los M números del intento. Cada uno de estos números está comprendido entre 0 y 10^9.

La entrada finaliza con el número `0`, que no se procesa.

Salida

Para cada caso de prueba debe escribirse una línea con M caracteres. El carácter i-ésimo indica el color con el que debe iluminarse el número i-ésimo del intento. El carácter (`.`) denota el color gris, el carácter (`0`) denota el color amarillo, y el carácter (`#`) denota el color verde.

Entrada de ejemplo

```
6
10 7 9 4 1 5
3 7 10 2 9 6
5
4 1 5 2 2
2 2 2 6 8
5
4 1 5 2 2
2 2 2 6 2
6
1 2 3 4 5 6
1 2 3 4 5 6
0
```

Salida de ejemplo

```
.#0.0.
00...
0...#
######
```

Créditos

Autor: Manuel Montenegro.