C01: Evaluar un polinomio

Estructuras de Datos
Facultad de Informática - UCM

Este ejercicio consta de dos entregas:

- **Entrega 1**. Corresponde a los apartados 1, 2, 3 y 4 de este enunciado. Debe entregarse en el problema *DOMjudge* que tiene identificador `C01-1` antes del **4 de febrero a las 23:55**.
- **Entrega 2**. Corresponde a los apartados 5, 6 y 7 de este enunciado. Debe entregarse en el problema *DOMjudge* que tiene identificador `C01-2` antes del **6 de febrero a las 23:55**.

Cada entrega debe consistir en un único fichero `.cpp` que se subirá a *DOMjudge*. Pueden subirse tantos intentos como queráis. Se tendrá en cuenta el último intento con el veredicto `CORRECT` que se haya realizado antes de la hora de entrega.

No olvidéis poner el nombre de los componentes del grupo en cada fichero `.cpp` que entregéis. Solo es necesario que uno de los componentes del grupo realice la entrega.

Las preguntas relativas a costes deben responderse como comentarios en el fichero `.cpp`.

Evaluación: Cada entrega se puntua de 0 a 5. Para poder obtener una calificación superior a 0 en una entrega es necesario obtener un veredicto `CORRECT`.

Un polinomio $P(x) = a_{n-1}x^{n-1} + \ldots + a_1 x + a_0$ de grado n puede ser representado mediante un array de n elementos en el que la posición i-ésima contiene el valor del coeficiente a_i. Partiremos, por tanto, la siguiente definición de clase:

```
const int GRADO_MAX = 5000;

class Polinomio {
public:
  Polinomio();
  // ...
private:
  int coeficientes[GRADO_MAX];
};
```

El ejercicio consiste en escribir un programa que lea un polinomio P por la entrada, un valor v, e imprima el resultado de evaluar el polinomio para el valor dado, esto es, el resultado de sustituir la variable x de P por el valor v. Para ello:

1. Añade a la clase `Polinomio` el siguiente método:

   ```
   {  this = P(x)  }
   void anyadir_monomio(int coef, int exp)
   {  this = P(x) + coef · x^exp  }
   ```

   Este método añade al polinomio `this` un monomio con el coeficiente y exponente pasados como parámetro.

2. Añade a la clase `Polinomio` el siguiente método:

```
{  this = P(x)  }
long evaluar(int valor)
{  result = P(valor)  }
```

Este método devuelve el resultado de sustituir la variable x del polinomio `this` por el `valor` indicado. Pon atención al coste. Si n es el número de componentes del array `coeficientes` (es decir, `GRADO_MAX`), no se van a admitir soluciones $O(n^2)$ ni $O(n \log n)$.

3. Indica el coste en tiempo del constructor de la clase `Polinomio`, así como de los métodos implementados en los apartados anteriores.

4. Escribe un programa que lea varios polinomios de la entrada, varios valores v, y utilice el método `evaluar` con el fin de evaluar cada polinomio para su correspondiente valor v, e imprimir cada resultado por pantalla. El formato de la entrada y la salida se describen en las secciones siguientes. Entrega el resultado de este apartado junto con los tres anteriores en el problema de *DOMjudge* que tiene identificador `C01-1`.

5. En los apartados anteriores hemos representado un polinomio mediante un array de enteros, donde la posición de cada entero indicaba el exponente del monomio asociado a dicho entero. Ahora vamos a representar un polinomio mediante un array de registros del tipo `Monomio`, donde cada uno de ellos contiene un coeficiente y un exponente:

```
const int MAX_MONOMIOS = 1000;

class Polinomio {
public:
  Polinomio();
  // ...
private:
  struct Monomio {
    int coeficiente;
    int exponente;
  };
  Monomio monomios[MAX_MONOMIOS];
  int num_monomios;
};
```

Suponemos que se cumple el siguiente invariante de representación: para toda instancia p de `Polinomio`,

$$\begin{aligned} I(\mathsf{p}) = \quad & 0 \leq \mathsf{p.num\_monomios} \leq \mathsf{MAX\_MONOMIOS} \\ & \wedge \forall i : 0 \leq i < \mathsf{p.num\_monomios} \Rightarrow \mathsf{p.monomios}[i] \neq 0 \\ & \wedge \forall i, j : 0 \leq i < j < \mathsf{p.num\_monomios} \Rightarrow \mathsf{p.monomios}[i]\mathsf{.exponente} < \mathsf{p.monomios}[j]\mathsf{.exponente} \end{aligned}$$

Es decir, el array de `monomios` debe estar ordenado de manera que la secuencia de exponentes sea estrictamente ascendente desde la posición 0 hasta `num_monomios - 1`. El array no puede contener exponentes duplicados, y no puede contener coeficientes nulos.

Modifica el constructor de la clase y los métodos `anyadir_modulo` y `evaluar` para acomodarlos a esta nueva representación.

6. Indica el nuevo coste en tiempo del constructor de la clase `Polinomio`, así como de los métodos anteriores.

7. Comprueba que el programa que enviaste en el apartado 4 sigue funcionando con la nueva implementación de la clase `Polinomio`. Para ello encontrarás un problema en *DOMjudge* con identificador `C01-2`. Envíalo allí.

Entrada

La entrada consta de una serie de casos de prueba. Cada caso ocupa dos lineas. La primera línea contiene dos números n y v (de tipo `int`), donde n indica el número de pares (*coeficiente*, *exponente*) que vendrán en la línea siguiente, y v es el valor para el cual quiere evaluarse el polinomio. La segunda línea contiene una secuencia de $2*n$ números de tipo `int` que indican los monomios del polinomio que quiere evaluarse. Cada pareja de números indica el coeficiente y el exponente del monomio correspondiente. Ten en cuenta que en la entrada puede haber varios monomios con el mismo exponente, y estos no están necesariamente ordenados.

La entrada finaliza con una línea que contiene dos ceros, caso que no deberá procesarse.

Salida

Para cada caso se escribirá una línea con el número resultante de evaluar el polinomio para el valor dado. Se garantiza que este número siempre estará contenido en el rango permitido por una variable de tipo `long` de 64 bits.

Entrada de ejemplo

```
2 4
3 2 5 0
4 2
3 2 1 0 7 1 -2 2
0 0
```

Salida de ejemplo

```
53
19
```

Créditos

Adaptación del problema *Evaluar un polinomio* de Alberto Verdejo e Isabel Pita.