L01: Evaluar un polinomio

Estructuras de Datos
Facultad de Informática - UCM

Este ejercicio consiste en implementar un TAD utilizando dos representaciones distintas. La primera de ellas (apartados 1, 2, 3 y 4) es opcional. La segunda de ellas (apartados 5, 6, 7 y 8) es la que será evaluada.

Si tienes problemas para abordar el problema con la segunda representación, empieza intentándolo con la primera. Cuando tengas el veredicto `CORRECT`, adapta tu solución a la segunda representación.

La entrega consiste en un único fichero `.cpp` que se subirá a *DOMjudge*. Podéis subir 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 por parte de alguno de los miembros de la pareja.

No olvidéis poner el nombre de los componentes de la pareja en el fichero `.cpp` que entreguéis. Solo es necesario que uno de los componentes de la pareja realice la entrega.

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

Evaluación: La práctica se puntuará de 0 a 10, desglosada del siguiente modo:

- Si la pareja ha obtenido el veredicto `CORRECT`, se obtendrán hasta 9 puntos por la entrega, más 1 punto por la evaluación que haga de la entrega de otra pareja.
- Si la pareja no ha obtenido el veredicto `CORRECT`, pero ha realizado al menos una entrega en el juez, podrá obtener hasta 2 puntos por la evaluación que haga de las entregas de otras dos parejas.
- Si la pareja no ha realizado ninguna entrega, la calificación de la práctica será de 0.

Un polinomio $P(x) = a_{n-1}x^{n-1} + \ldots + a_1 x + a_0$ con coeficientes enteros puede ser representado de distintas maneras. Supongamos que tenemos una clase `Polinomio` con la siguiente interfaz:

```
class Polinomio {
public:
  Polinomio();
  void anyadir_monomio(int coef, int exp);
  long evaluar(int valor) const;
private:
  // ...
};
```

Una instancia recién construida de esta clase denota el polinomio $P(x) = 0$. El método `anyadir_monomio` añade al polinomio `this` un monomio con el coeficiente y exponente pasados como parámetro. El método `evaluar` devuelve el resultado de sustituir la variable x del polinomio `this` por el `valor` indicado como parámetro. La especificación del constructor y de ambos métodos es la siguiente:

```
{  true  }
Polinomio()   // constructor
{  this = 0  }
```

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

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

En este ejercicio trataremos dos posibles representaciones de los polinomios, que se describen en las siguientes secciones. En ambos casos, el programa a escribir debe leer un polinomio P por la entrada y un valor v, y a partir de ambos imprimir 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.

Primera representación (opcional)

En esta primera sección representaremos el polinomio $P(x) = a_{n-1}x^{n-1} + \ldots + a_1 x + a_0$ mediante un array de números enteros, en el que la posición i-ésima del array contiene el valor del coeficiente a_i.

```
const int GRADO_MAX = 10000;

class Polinomio {
public:
  // ...
private:
  int coeficientes[GRADO_MAX + 1];
};
```

Suponiendo esta representación, se pide:

1. Implementar el constructor de la clase `Polinomio`, y los métodos `anyadir_monomio` y `evaluar`. Sobre este último método, pon atención al coste. Si n es el grado del polinomio, no se van a admitir soluciones $O(n^2)$ ni $O(n \log n)$.
2. Indicar el coste en tiempo, en el caso peor, del constructor de la clase y de los métodos mencionados en el apartado anterior.
3. Completa la función `tratar_caso()` de la plantilla proporcionada en el Campus Virtual (o en este mismo PDF, haciendo clic en el icono del clip al lado del título). Esta función debe leer un único polinomio de la entrada estándar y construir el objeto `Polinomio` correspondiente. Después deberá leer un valor sobre el cual llamar al método `evaluar()`, e imprimir el resultado por pantalla. El formato de la entrada se describe al final de este enunciado.
4. Entrega el resultado en el problema de *DOMjudge* con identificador `L01-1`.

Segunda representación

Otra forma de representar un polinomio consiste en tener un array de registros de tipo `Monomio`, donde cada uno de ellos guarda un coeficiente y un exponente:

```
const int MAX_MONOMIOS = 10000;

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`:

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

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.

Utilizando esta representación, se pide:

5. Implementar el constructor de la clase `Polinomio`, y los métodos `anyadir_monomio` y `evaluar`. Sobre este último método, pon atención al coste. Si n es el grado del polinomio, no se van a admitir soluciones $O(n^2)$ ni $O(n \log n)$.
6. Indicar el coste en tiempo, en el caso peor, del constructor de la clase y de los métodos mencionados en el apartado anterior.
7. Completa la función `tratar_caso()` de la plantilla proporcionada en el Campus Virtual (o en este mismo PDF, haciendo clic en el icono del clip al lado del título). Esta función debe leer un único polinomio de la entrada estándar y construir el objeto `Polinomio` correspondiente. Después deberá leer un valor sobre el cual llamar al método `evaluar()`, e imprimir el resultado por pantalla. El formato de la entrada se describe al final de este enunciado.

   Si has implementado el apartado 3 de la representación anterior, no debería ser necesario realizar ningún cambio sobre la versión que ya tengas implementada.
8. Entrega el resultado en el problema de *DOMjudge* con identificador `L01-2`.

Entrada

La entrada consta de una serie de casos de prueba. Cada caso ocupa dos líneas. La primera línea contiene dos números n y v ($0 \leq n \leq 2000$, $-10 \leq v \leq 10$), 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 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.