Aprendizaje de algoritmia mediante desafíos de programación

Rafael C. Carrasco, Juan Ramón Rico-Juan, Miguel Ángel Varó
Departament de Llenguatges i Sistemes Informàtics
Universitat d'Alacant
03071 Alacant
{carrasco,juanra,mvaro}@dlsi.ua.es

Resumen

Esta presentación describe las características de un sistema de autoevaluación de código, similar a los utilizados en los concursos de programación, que permite a los estudiantes aplicar los conocimientos teóricos de la disciplina de algoritmia para resolver problemas prácticos y, a la vez, reforzar las competencias generales de programación adquiridas en cursos previos. La experiencia de varios cursos en su aplicación demuestra que la utilización de un sistema competitivo introduce un aliciente adicional para la realización de los ejercicios.

1. Motivación

Varios cursos de docencia en una asignatura de algoritmia de cuarto curso de ingeniería informática generaron las siguientes observaciones:

- Los mayoría de los estudiantes encontraban grandes dificultades para aplicar los conocimientos teóricos de algoritmia a la resolución de problemas simples inspirados por casos reales.
- Los fundamentos de programación adquiridos en los cursos previos a esta asignatura no parecían suficientes para generar programas robustos que resolviesen correctamente todos los casos de prueba planteados.
- La capacidad de los estudiantes para generar programas bien estructurados y documentados y para aplicar estrategias de control de la calidad de los programas no se manifestaba en la evaluación de los resultados (basada en la supervisión de prácticas y en un examen escrito).
- La compartición entre los estudiantes de fragmentos de código hacía difícil evaluar el desempeño real de cada estudiante en las horas de laboratorio.

Para intentar corregir estas deficiencias se decidió reorientar la asignatura hacia un carácter más práctico y, en consonancia, cambiar el sistema de evaluación por otro basado en la resolución de un problema en el laboratorio. Para facilitar la evaluación y el aprendizaje se implementó el sistema `javaluador`[1] que permite a los estudiantes comprobar la eficiencia de sus implementaciones. Este sistema se implementó en Java ya que la máquina virtual de Java ofrece un entorno robusto que minimiza los riesgos de ejecutar código externo dentro de nuestro servidor.

Problema	Código	Resultado	Tiempo	Fecha
P604. Días de permiso	P604.java	Tiempo máximo excedido		2010-02-02 09:13:18
P605. Días de permiso (solución completa)	P605.java	Correcto	6.996s	2010-01-17 09:24:44
P604. Días de permiso	P604.java	Correcto	6.144s	2010-01-17 09:21:21
P604. Días de permiso	P604.java	Error de compilación		2010-01-17 09:20:32
P604. Días de permiso	P604.java	Error de compilación		2010-01-17 09:19:34
P604. Días de permiso	P604.java	Error de compilación		2010-01-17 09:17:20
P605. Días de permiso (solución completa)	P605.java	Correcto	0.656s	2010-01-13 09:59:11
P604. Días de permiso	P604.java	Correcto	0.644s	2010-01-13 09:57:49
P604. Días de permiso	P604.java	Función restringida		2010-01-13 09:55:45
P604. Días de permiso	P604.java	Función restringida		2010-01-13 09:54:55

Figura 1: Información al estudiante sobre los problemas realizados y el nivel de progreso en cada uno de ellos.

La evaluación final de la asignatura se realiza mediante la resolución en el laboratorio de un problema similar a los utilizados durante la fase de aprendizaje. Esto hace posible (incluso deseable) que los estudiantes compartan ideas durante el curso sin que esta forma de trabajar perjudique la objetividad de la evaluación global.

2. Descripción general

El sistema contiene varias decenas de problemas que pueden ser utilizados durante la fase de entrenamiento. Cada enunciado (en formato PDF) contiene una descripción genérica del problema, la especificación de la función que debe implementarse y algunos ejemplos con valores posibles de los parámetros de entrada y los valores de salida correctos para cada caso.

La mayor parte de los problemas (aunque no todos) requieren optimizar un valor —mediante una función llamada `best` o `bestValue`— u obtener una solución óptima —mediante una función llamada

`bestSolution` o similar. En este último caso, si existen varias soluciones óptimas y el enunciado no establece un criterio de preferencia, cualquiera de ellas será considerada correcta. Como los datos del problema sólo indican el valor óptimo, el proceso de evaluación requiere comprobar que la solución devuelta es efectivamente una solución (esto es, que cumple los requisitos del problema) y que además conduce al valor óptimo cuando se evalúa con la función de mérito del problema.

Cuando se envía un código para su corrección, el sistema `javaluador` proporciona información detallada (véase la figura 1) en función de los siguientes resultados posibles.

1. Error de presentación: el código enviado no satisface las especificaciones generales o utiliza funciones no permitidas.
2. Error de compilación: el programa no es Java válido o no contiene la función requerida.
3. Memoria máxima excedida: el programa requiere más memoria de la permitida (100MB por defecto).
4. Tiempo máximo excedido: el tiempo máximo de ejecución (10 segundos por defecto) ha sido superado.
5. Error de ejecución: se ha producido un error durante la ejecución con los datos de prueba.
6. Código incorrecto: el programa no resuelve adecuadamente algún caso de prueba.
7. Tiempo requerido: tiempo para completar todos los casos de prueba correctamente (cuando éste es inferior al máximo establecido).

3. Entrada y salida del sistema

En principio, parece deseable en un sistema de autoevaluación evitar los típicos problemas asociados a entradas no previstas por el programador que no están relacionados con cuestiones algorítmicas. Este tipo de errores se minimiza si el estudiante no necesita incluir funciones de lectura o escritura de ficheros en su programa sino únicamente funciones que obtienen los datos a partir de parámetros bien especificados en el enunciado del ejercicio. Así se evitan también los peligros asociados a permitir el acceso a ficheros del sistema por un programa ajeno.

Sin embargo, otros argumentos apoyan el uso de entradas basadas en texto:

Alumno	Código	Resultado	Temps	F.Envío	Desde
1. [illegible]n1	P06.java	OK	6.504s	2010-01-18 04:50:58	79.148.227.138
2. [illegible]am42	P06.java	OK	0.172s	2010-01-18 02:14:41	83.34.115.56
3. [illegible]ns4	P06.java	OK	0.172s	2010-01-18 01:54:23	87.221.201.74
4. [illegible]alb	P06.java	ERR_TEMPS		2010-01-18 01:49:04	85.56.154.44
5. [illegible]alb	P06.java	ERR_TEMPS		2010-01-18 01:48:13	85.56.154.44
6. [illegible]alb	P06.java	ERR_TEMPS		2010-01-18 01:46:39	85.56.154.44
7. [illegible]a11	P06.java	OK	0.144s	2010-01-18 01:32:25	84.232.42.215
8. [illegible]a11	P06.java	OK	0.524s	2010-01-18 01:31:21	84.232.42.215
9. [illegible]f9	P06.java	OK	0.992s	2010-01-18 01:19:40	95.19.38.12
10. [illegible]alb	P06.java	ERR_TEMPS		2010-01-18 01:19:31	85.56.154.44
11. [illegible]f9	P07.java	ERR_PRES		2010-01-18 01:19:30	95.19.38.12

Figura 2: Informe de progreso con información sobre la fecha de envío y URL origen.

- La mayor parte de los sistemas de evaluación on-line[3] utilizan como entrada y salida ficheros de texto.
- Durante el desarrollo y supervisión del programa es preciso leer y escribir datos (a menudo de gran tamaño para probar la eficiencia de la implementación) por lo que el estudiante tendrá que implementar en cualquier caso estas funciones.
- Con frecuencia, el programa debe ser supervisado por los profesores en el laboratorio utilizando algunos casos de prueba y esta tarea debe ser lo más sencilla posible.

Considerando todas las razones anteriores, se decidió utilizar una aproximación flexible, que permite determinar cualquier tipo de entrada. Es decir, el enunciado debe especificar qué función o funciones deben implementarse y los tipos de los parámetros y del valor de retorno de cada una. No obstante, en el caso de los problemas de prueba se optó sistemáticamente por la siguiente estrategia: los parámetros suelen ser de tipo cadena —`String`— o de tipo vector de cadenas —`String[]`— lo que hace la función muy similar a la que hay que implementar en otros evaluadores on-line y muy próxima a la entrada de la función principal —`main(String[]args)`—, lo que simplifica la supervisión en el laboratorio. Como la mayor parte de los problemas requiere procesar una cadena de texto de entrada, se recomienda usar la función `split` de la clase `String`[1] para procesar las entradas de texto.

4. Elementos de seguridad

La máquina virtual de Java, además de ofrecer un entorno robusto que minimiza los riesgos de ejecu-

[1] `String.split("\\p{Space}+")`

tar código externo en el servidor, permite fácilmente limitar la cantidad de memoria disponible para resolver un problema (mediante la opción `-Xmx`), que es uno de los parámetros de eficiencia de los programas.

La supervisión de los ejercicios la realiza un programa corrector que compara los resultados obtenidos por el programa enviado para un número arbitrario de casos de prueba con los resultados esperados. Los casos de prueba y la salida correcta se almacenan en un fichero de texto separado para simplificar su actualización. Como se discutirá más adelante, el programa corrector permite la inclusión de funciones de evaluación de la respuesta para los problemas que permiten múltiples respuestas correctas.

El límite de tiempo (10 s) se establece para que el servidor pueda atender cientos de peticiones por hora sin generar colas de espera significativas (sobre todo durante la evaluación). El control del tiempo que se dedica a la resolución de cada entrega se utiliza la instrucción `launchtool` que permite establecer un tiempo máximo de CPU para cada proceso.

Para ajustar la dificultad del problema se generan casos de prueba que pueden resolverse en una fracción pequeña del tiempo máximo (típicamente unos segundos) siempre que se usen estrategias algorítmicas adecuadas. Afortunadamente, la utilización de estrategias inadecuadas suele conducir a tiempos extraordinariamente largos dado el carácter exponencial de la mayoría de los problemas.

Para evitar la extracción accidental o malintencionada de información interna del programa de corrección deben bloquearse las salidas estándar y de error durante la ejecución de las funciones externas. Sin embargo, para ofrecer al estudiante la máxima cantidad posible de información sobre los problemas de su código, sí que se permite la lectura de todos los errores y excepciones. Para evitar que estos errores revelen información sobre las pruebas el corrector informa de un error de presentación en los siguientes casos:

- Se crean o extienden clases cuyo nombre incluye `File`, `Error`, `Exception` o `Throwable`.
- Se utilizan funciones del sistema o de ciertas clases (por ejemplo, Runtime o Method) tales como `exec`, `exit`, `halt` o `invoke`.

5. Administración

El sistema `javaluador` permite establecer rangos de fechas para la realización de cada ejercicio (véase la figura 3. Esta característica es útil para realizar ejercicios con supervisión durante el curso (con fecha de entrega) y para establecer límites de tiempo en la evaluación.

Además el sistema genera informes sobre el grado de progreso global (véase la figura 2). también permite generar informes de resultados para cada problema y para cada estudiante. Para cada problema se informa además de quién lo ha resuelto en menor tiempo de ejecución (el estudiante recibe esta información inmediatamente). En todos los casos, se presenta un resumen estadístico con los siguientes datos:

- Problemas resueltos
- Problemas pendientes de comprobación
- Códigos aceptados
- Error de presentación
- Respuesta incorrecta
- Error de ejecución
- Excedido límite de tiempo
- Excedido límite de memoria
- Uso de funciones restringidas
- Error de compilación

Además se añade información sobre:

- Identificador del estudiante
- Código fuente enviado.
- Resultado (respuesta)
- Tiempo dedicado de CPU.
- Fecha y hora de envío.
- URL de acceso.

Esta información es útil para la supervisión de los trabajos y, especialmente, para el control de la evaluación. La gestión de los permisos de acceso se centraliza en los servicios administrativos de la Universidad, de forma que se comparte con otros servicios y se comparte mediante un protocolo LDAP (Lightweight Directory Access Protocol).

6. La experiencia

Actualmente, `javaluador` ofrece una treintena de problemas resolubles mediante procedimientos recursivos pero que, en su mayoría, para que sean efi-


Lista de problemas

Problema	Inicio	Fecha límite	Estado	Enunciado	Registro
P01. Corte de una barra			ok	pdf	...
P02. Supercrabas				pdf	...
P03. Reparto de tareas				pdf	...
P04. Poda ideal				pdf	...
P05. Memorias de traducción				pdf	...
P06. Dieta				pdf	...
P07. El juego de la cuerda				pdf	...
P08. America's cup				pdf	...
P09. Transporte con cubas				pdf	...
P10. Grabación óptima de CDs				pdf	...
P11. Sudoku				pdf	...
P12. Iluminación de calles	2010-01-01 00:00			pdf	...

Resultados

Envíos	31
Problemas resueltos	8
Pendientes de comprobación	
Códigos aceptados	12
Error de presentación	1
Respuesta incorrecta	
Error de ejecución	
Excedido límite de tiempo	4
Excedido límite de memoria	
Uso de funciones restringidas	9
Error de compilación	5

Figura 3: Página de administración de los casos de prueba e información general.

cientes, requieren aplicar técnicas de poda o de programación dinámica.

La publicación de los récords (los tiempos de ejecución menores para cada problema) demostró ser un aliciente para el trabajo de muchos estudiantes. En general, los estudiantes valoraron positivamente la objetividad de la evaluación y la orientación aplicada de la asignatura. Sin embargo, algunos casos problemáticos fueron detectados porque algunos estudiantes consideraban la evaluación en el laboratorio especialmente estresante lo que les impedía rendir en el nivel que aparentemente habían alcanzado previamente.

Otra dificultad considerable fue establecer unos criterios de gradación en la evaluación del examen (no así de la asignatura en la que se tenían en cuenta también los ejercicios realizados), ya que ni el tiempo de ejecución ni el número de entregas del ejercicio demostraron ser elementos suficientes para ponderar la calidad de la resolución presentada. Todos estos elementos serán objeto de reflexión en los próximos cursos en que se aplique esta metodología de aprendizaje y evaluación.

Globalmente, la experiencia ha resultado muy interesante y nos anima a experimentar nuevas vías similares.

Referencias

[1] `http://javaluador.dlsi.ua.es/`
[2] `http://java.sun.com/javase/6/docs/api/`
[3] `http://uva.onlinejudge.org/`
[4] Skiena, Steven S., Revilla, Miguel, *Programming Challenges*, Springer-Verlag, New York, 2003.