CB-252

MATEMÁTICA Y PROGRAMACIÓN: UNA EXPERIENCIA INTERDISCIPLINARIA E INTERINSTITUCIONAL

Teresa Pérez - Sylvia da Rosa
tperezan@gmail.com - darosa@fing.edu.uy
Consejo de Educación Secundaria (ANEP) Uruguay – Facultad de Ingeniería (UDELAR) Uruguay

Núcleo temático: Matemáticas y su integración con otras áreas.
Modalidad: CB
Nivel educativo: Formación y actualización docente
Palabras clave: algoritmo, programa, Python

Resumen

La experiencia interdisciplinaria e interinstitucional que se describe consiste en el desarrollo de un curso para duplas de los profesores formadas por un profesor de informática y un profesor de matemática de un mismo centro educativo de enseñanza media.

El curso tiene por objetivo trabajar en base a la integración de matemática y programación como forma de facilitar la comprensión y el aprendizaje de conceptos matemáticos, en el entendido de que programar una solución a un problema pone al descubierto aspectos del proceso de resolución que de otro modo quedan ocultos, como ser la necesidad tanto de plantear los problemas algorítmicos como tales, como de escribir las soluciones usando un lenguaje riguroso.

Este año ha participado en el desarrollo del curso un equipo formado por profesores de la inspección de matemática y de la coordinación de informática, del consejo de educación secundaria (CES) de ANEP, y por docentes del Instituto de Computación (InCo) de la Facultad de Ingeniería de la UDELAR. Han culminado exitosamente la edición 2016 del curso, 40 profesores de todo el país.

En la descripción de la experiencia se incluye una síntesis de los trabajos finales de los participantes, realizados en el aula con sus alumnos.

Introducción

La relación entre matemática y programación ha sido muy estrecha en la evolución de ambas disciplinas. Problemas matemáticos dieron impulso al desarrollo de la computación, como por ejemplo el *"Entscheidungsproblem"* o problema de decisión, que consiste en encontrar un algoritmo general que decida si una fórmula del cálculo de primer orden es un teorema. Este problema fue presentado por David Hilbert en 1928, y Kurt Gödel demostró que es insoluble en 1931, poniendo en discusión en el mundo matemático de la época, el problema de *lo computable* y la necesidad de formalizar la noción de *algoritmo.* Esta noción es una de las más fundamentales para la programación. Por otro lado, en los últimos años, la programación ha impactado

fuertemente en la noción de *prueba matemática.* Por ejemplo la prueba del *teorema de los cuatro colores* fue realizada en la década de 1970 con ayuda de un programa, y dio origen a un gran debate sobre su validez.

Desde el punto de vista educativo, la integración de matemática y programación facilita la comprensión y el aprendizaje de conceptos matemáticos, en el sentido de que programar una solución a un problema pone al descubierto aspectos del proceso de resolución que de otro modo quedan ocultos, Como lo ha expresado *Donald Knuth en "American Mathematical Monthly en 1974: "Se ha dicho que una persona no comprende algo realmente hasta que lo enseña a otro. En realidad, una persona no comprende algo profundamente hasta que puede enseñarlo a un computador, es decir, expresarlo como un algoritmo."*

Por otra parte, esta propuesta busca promover la introducción de la informática como ciencia básica en el sistema educativo medio, siguiendo recomendaciones de referentes internacionales (CSTA 2011, Dowek 2005, Peyton Jones 2013). Hacerlo a través de cursos de matemática resulta un camino natural, ya que proveen problemas algorítmicos interesantes cuyas soluciones pueden implementarse en algún lenguaje de programación. Esta perspectiva interdisciplinaria resalta aspectos que favorecen el aprendizaje de la algoritmia como disciplina de resolución de problemas y el análisis de soluciones como objetos de estudio. Por otro lado, implementar una solución obliga a tener en cuenta factores propios de la computación como el rigor del lenguaje y los recursos acotados.

Este curso se ha dictado durante varios años siendo la primera edición en 1999, y es un ejemplo de trabajo colaborativo entre el Instituto de Computación de la Facultad de Ingeniería de la UDELAR y la ANEP. Se ha presentado la propuesta en congresos internacionales (da Rosa, 2002). Entre 2009 y 2013 el apoyo del Instituto se concentró en la carrera del profesorado de informática y el curso dejó de dictarse. El curso se ha vuelto a dictar desde 2014, con apoyo de la inspección de matemática y la coordinación de informática de enseñanza secundaria. Desde entonces se utiliza el lenguaje Python.

En el año 2016 el curso se llevó a cabo desde el 1 de agosto al 22 de octubre, y se introdujo la modalidad de trabajo en duplas formadas por un docente de matemática y uno de informática. Participaron 40 docentes de todo el país (20 duplas) y se dictó en forma semipresencial, usando la

plataforma moodle del CES y la sala de video conferencia de FING. Se realizaron tres encuentros presenciales, uno de ellos por video conferencia.

Desarrollo del curso

La propuesta consistió en que los profesores participantes trabajaran en duplas formadas por un profesor de informática y un profesor de matemática de un mismo liceo. Como no en todos los casos los profesores compartían algún grupo de alumnos, se los exhortó a que buscaran un espacio común de trabajo, por ejemplo, participando el profesor de informática en las clases de matemática donde se trabajarían los temas del curso.

La tabla abajo informa sobre la cantidad de egresados/estudiantes de los institutos de formación docente, IPA (matemática) o INET (informática), así como participantes con otras formaciones. Usualmente el curso está dirigido a egresados de formación docente en informática o matemática, pero para el trabajo en duplas se consideró importante eliminar dicha restricción, ya que en muchos liceos, los encargados de la asignatura informática no son profesores titulados.

	Matemática (inicio)	Matemática (final)	Informática (inicio)	Informática (final)
Egresado	27	19	9	7
Estudiante	3	1	14	9
Otro	0	0	7	4
Total	30	20	30	20

Durante el curso los profesores aprenden a programar soluciones a problemas computables sencillos, utilizando el lenguaje Python, que está instalado en las máquinas de que disponen muchos de los profesores y de los estudiantes (otorgadas por el Plan Ceibal). Asimismo el lenguaje es fácilmente accesible en las salas de informática de los centros educativos para los casos en que es necesario utilizar dichas salas.

Para aprobar el curso los profesores deben diseñar, planificar y llevar a cabo una propuesta para la clase con alumnos liceales utilizando lo visto en el curso, hacer una presentación oral y escribir un informe sobre ello, que incluye registros de las actividades de los alumnos.

Problemas algorítmicos y soluciones

A pesar de que se introduce y se usa un lenguaje de programación, el enfoque del curso consiste en enfatizar los pasos previos de la construcción de un programa, esto es, la especificación del problema algorítmico que se plantea y el diseño de una solución.

Si bien hay muchas definiciones de algoritmo (dado que no es una definición formal), todas requieren una anterior que es la de "problema algorítmico". Un problema algorítmico es:

1) una colección de datos de entrada (que puede ser infinita, por ejemplo el conjunto de los números enteros)

2) un resultado requerido *en función* de los datos de entrada

La función por la cual los datos de entrada se convierten en el resultado requerido es el algoritmo que soluciona el problema y que hay que elaborar. Esto implica pensar cuáles son los datos del problema, qué resultado se espera y como se obtendrá el resultado *en función* de dichos datos. Las características del lenguaje de programación empleado influyen en el diseño de la solución, por lo tanto la elaboración del programa es una actividad dialéctica, que exige sintetizar aspectos tanto del pensamiento algorítmico como del computacional, entendido este último en el sentido de: *"Computational Thinking is the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent."* (Computational Thinking, 2011). Por otra parte, en el curso se recomienda modularizar la solución, siguiendo el modelo que ilustra la siguiente tabla, (donde mcd refiere al ejemplo desarrollado abajo).

ingreso datos	a = input (" ingrese natural distinto de 0 o 0 para finalizar ") b = input ("ingrese numero natural distinto de 0 ")
proceso con funciones	mcd = mcd (a,b)
imprimo resultados	print "el max comun divisor de ", a, b, "es ", mcd

Las funciones son sub-programas cuya composición implementa el algoritmo. En el ejemplo se trata del siguiente problema: dados dos números naturales, hallar su máximo común divisor (mcd). Presentamos las dos soluciones trabajadas en el curso: por un lado, a partir de la definición matemática

$\forall a,b \in N, b > 0$, max (divisores(a) $\cap$ divisores(b)) = mcd (a,b)

puede plantearse una solución que involucra las siguientes funciones:

def **divisores** (a): return [x for x in range (1,a+1) if a%x== 0]	dado un natural devuelve el conjunto de sus divisores, usando la función predefinida "módulo" **(%)**

def **inter** (lis1, lis2):
lis = []
for element in lis1:
if element in lis2:
lis**.**append (element)
return lis
	lados dos conjuntos **lis1** y
	is2, devuelve su intersección **lis**
	Para representar conjuntos los l
	enguajes de programación suelen
	ısar listas (conjuntos ordenados sin elementos repetidos)

def **divisorescom** (a, b): d1 = divisores (a) d2 = divisores (b) dc = inter (d1, d2) return dc	lados dos naturales devuelve el conjunto de sus divisores comunes **dc)**
def **mcd** (a,b): dc = divisorescom (a,b) return max (dc)	lados dos naturales devuelve su náximo común divisor, usando las funciones anteriormente definidas y la función predefinida **max**

Importa observar que los lenguajes de programación cuentan con funciones predefinidas que es necesario conocer para que los programas resulten sencillos **(%, inter, append, max)**. De esta forma los estudiantes experimentan *la construcción* de una definición que deja de ser solamente "un texto que hay que aprender". Por otro lado, introduciendo la definición recursiva mcdR y la función que implementa el algoritmo de Euclides,

mcdR : N x (N - {0}) → N
mcdR (a,b) = ⎧ b, si (a mod b) = 0
⎩ mcdR (b, (a mod b)), si no
	def mcdR (a, b):
	r = a%b
	if r == 0:
	return b
	else:
	return mcdR (b, r)

se puede profundizar en conceptos esenciales y las relaciones entre ellos: la *especificación* del problema, la expresión matemática del algoritmo que lo soluciona como *función*, la *propiedad* que establece el vínculo entre especificación y algoritmo y la *prueba* de la misma: ¿cumple el algoritmo de Euclides la definición de mcd?

Organización del curso

Se pueden distinguir dos etapas en el desarrollo del curso: en una primera etapa se disponen en la plataforma virtual los materiales para el estudio de los conceptos básicos de algoritmia y el uso del intérprete Python. Los ejercicios están pensados para introducir a los participantes en la programación de soluciones teniendo en cuenta las recomendaciones mencionadas arriba, así como también para que reflexionen sobre la relación entre matemática y programación. Se intenta transmitir que lo importante que aporta la informática a la matemática es la necesidad del rigor, de pensar las situaciones "borde", de ordenar el pensamiento. Se enfatiza el hecho de que el computador hará lo que nosotros escribamos, sin pensar por sí mismo, o sea que nosotros tenemos que pensar las soluciones muy rigurosamente. Coincidimos con Dowek (2005) en que introducir esta manera de pensar desde temprano (por ejemplo desde 1ero de liceo, alumnos de 13 años) es muy beneficioso para la formación de todos los estudiantes. Ilustramos con el siguiente ejemplo:

Los estudiantes tienden a escribir una sentencia de lectura de datos de la siguiente manera:

```
numero = input ("ingrese un numero")
```

(en Python esto significa que se desplegará el mensaje entre comillas y luego lo que el usuario escriba se asignará a la variable n), con lo cual están diciendo que la entrada comprende varios conjuntos de números. ¿Es esperable que el número sea por ejemplo √2? ¿o -3345466778? Algunos participantes tomaron en cuenta las diferencias de representación de un número explícitamente. Por ejemplo, en uno de los trabajos, que presenta la secuencia didáctica para la introducción de Ecuaciones de primer grado en un 2° año de liceo (14 años), los autores dicen: *"En esa semana la profesora de Matemática acompañará dicho trabajo con análisis de situaciones que surjan y que necesiten fundamentación matemática. Por ejemplo la "diferencia" entre 2 y 2.0: ¿por qué al escribir 5/3 Python devuelve 1 mientras que al escribir 5.0/3.0 devuelve 1.66666666666667?"*

Contenidos del curso

Diseño, escritura y ejecución de programas en Python.

Resolución de pequeños problemas algorítmicos e implementación de soluciones en Python, enfatizando la comprensión de los mensajes de error. Selección y Sentencia for.

Ejemplos avanzados

Trabajo con ejemplos utilizando listas. Recursión e iteración (while). Definición de funciones.

Lecturas relacionadas a la programación para estudiantes de Enseñanza Media.

En una segunda etapa, los cursillistas trabajan en el diseño de una propuesta de aula y en la experimentación de la misma en alguno de sus grupos. Para la evaluación final deben elaborar un informe y presentarlo oralmente. El contenido en esta etapa se organiza como sigue:

Diseño de propuesta didáctica para su aplicación en el aula

Cada dupla docente elegirá un tema que pueda experimentar en alguno de sus grupos. Se trata de diseñar una secuencia de actividades a proponer a los alumnos, con su correspondiente análisis y detalles de implementación.

Aplicación de la propuesta y registro

Eventualmente puede acompañar al docente, como observadores, en la presentación en alguna de las clases, compañeros de plataforma y/o un Inspector de Matemática del CES y/o la Coordinadora de Informática.

Elaboración de informe

Elaboración del informe de la experiencia con registro de las producciones de los alumnos y reflexión sobre el aporte de la propuesta al aprendizaje matemático de los alumnos. Presentación en el encuentro presencial final.

En ambas etapas los participantes disponen de foros para consultas atendidos con alta frecuencia. La modalidad es fomentar la discusión y las distintas soluciones entre cursillistas.

La siguiente tabla muestra los temas elegidos por los docentes para el trabajo final:

Temas elegidos	**Cantidad**
Raíces de polinomios	2
Divisibilidad	5
Función polinómica de primer grado	6
Estadística	1
Geometría Analítica	2
Temas varios (operaciones con números enteros, variables, etc)	4

La siguiente tabla muestra la cantidad de grupos y años de liceo en que se trabajó.

Año de liceo		**Cantidad**
Primero	(12-13 años)	7
Segundo	(13-14 años)	6
Tercero	(14-15 años)	3
Cuarto	(15-16 años)	2
Quinto	(16-17 años)	2

Conclusiones

Todos los profesores resaltan el estímulo que significa para la motivación de los estudiantes trabajar con un lenguaje de programación. Muchos señalan además que constataron un impacto significativo entre aquellos estudiantes que, en general, no muestran interés en la asignatura

matemática. Asimismo la gran mayoría destaca aportes importantes para el aprendizaje de los contenidos trabajados.

El trabajo en duplas fue considerado exitoso por la gran mayoría de los profesores, tanto para ellos como para sus alumnos. Si bien los participantes tienen a su disposición un manual de Python elaborado para el curso 2013 por los mismos cursillistas, trabajar en conjunto con un profesor de informática ahorra a los profesores de matemática dedicar tiempo a cuestiones técnicas que no aportan al contenido pero que es necesario superar para poder continuar con las tareas.

Como en ediciones anteriores hemos obtenido información valiosa para diseñar estrategias para mejorar el curso, por ejemplo, la elaboración de pruebas de autoevaluación, especialmente para el apoyo a aquellos profesores con mayores dificultades.

A pesar de que los estudiantes liceales cuyos profesores han seguido nuestro curso se benefician de esta experiencia integradora del aprendizaje de matemática y programación y que muchos de los participantes han dado continuidad al trabajo integrado de matemática y programación más allá del alcance del curso, no hemos logrado que la propuesta sea aceptada institucionalmente por razones que no tienen que ver con lo académico, sino con el hecho de que la educación en informática en nuestro país es un campo atravesado por opiniones de la más diversa índole. De esta forma, nuestro curso y sus efectos tienen un alcance limitado, lo que dificulta ofrecer una valoración más adecuada de los resultados de los aprendizajes.

Referencias bibliográficas

Computational Thinking (2011) https://www.cs.cmu.edu/~CompThink/ Consultado 15/04/2017

CSTA K12 Computer Science Standards. 2011.
http://www.csteachers.org/?page=CSTA_Standards

Dowek, G. (2005) Quelle informatique enseigner au lycée? Bulletin de l'APMEP, 480. Peyton Jones, S. (2013) Bringing Computer Science Back into Schools: Lessons from the UK, SIGCSE'13.

da Rosa, S. (2002) The Role of Discrete Mathematics and Programming in Education. Proceedings of Functional and Declarative Programming in Education Workshop.