J. Ángel Velázquez Iturbide
Gladys Lorena Aguirre
Jorge Huilca

Evaluación de un Nuevo Método Instruccional para el Aprendizaje de la Recursividad

Número 2014-08

Serie de Informes Técnicos DLSI1-URJC
ISSN 1988-8074
Departamento de Lenguajes y Sistemas Informáticos I
Universidad Rey Juan Carlos

Índice

Evaluación Preliminar de un Nuevo Método Instruccional para el Aprendizaje de la Recursividad

J. Ángel Velázquez Iturbide

Departamento de Lenguajes y Sistemas Informáticos I, Universidad Rey Juan Carlos,
C/ Tulipán s/n, 28933, Móstoles, Madrid
angel.velazquez@urjc.es

Gladys Lorena Aguirre, Jorge Huilca

Facultad de Informática y Electrónica, Escuela Politécnica Superior de Chimborazo
Riobamba, Ecuador
gaguirre@espoch.edu.ec, jhuilca@gmail.com

Resumen. Se presenta una evaluación de un nuevo método instruccional para mejorar la comprensión de la recursividad, basado en enseñar la eliminación de la recursividad lineal. Se describe el diseño de la evaluación, incluyendo la población y la asignatura, el protocolo seguido y las tareas realizadas, así como los modelos mentales identificados en los alumnos. Los resultados han sido positivos tanto para la comprensión de la ejecución de algoritmos lineales como múltiples. El número de alumnos con modelos viables completos ha aumentado sustancialmente, mientras se ha reducido el número de los que muestran un modelo de avance o simplemente no contestaban nada. Sin embargo, no parece haber variado su capacidad para diseñar algoritmos recursivos.

Palabras clave: Aprendizaje de la programación, recursividad, método instruccional, malentendidos, modelos mentales.

1 Introducción

La recursividad es uno de los conceptos de programación más difíciles de aprender, habiéndose realizado numerosas propuestas para facilitar su aprendizaje [1]. En diversos estudios se ha conjeturado que es distinta la dificultad de comprender la ejecución de algoritmos recursivos y de diseñarlos. En este informe se presenta una su evaluación realizada en la Escuela Superior Politécnica del Chimborazo (ESPOCH) del uso de la eliminación de la recursividad lineal [2] como método instruccional para mejorar la comprensión de la ejecución de los algoritmos recursivos.

La estructura del informe es la siguiente. El apartado 2 describe la evaluación, incluyendo población, asignatura, protocolo aplicado y tareas realizadas. El apartado 3 presenta el método de análisis y los modelos mentales identificados en los alumnos. En el apartado 4 se presentan los resultados obtenidos, tanto en el estudio principal como en una actividad secundaria. En el quinto apartado se comentan los resultados y

en el sexto se incluyen nuestras conclusiones. Finalmente, los Apéndices A y B contienen los enunciados de las dos pruebas utilizadas en la evaluación.

2 Descripción de la Evaluación

Se presenta la evaluación estructurada en tres subapartados: población y asignatura, protocolo aplicado y tareas realizadas por los participantes.

2.1 Población y Asignatura

La evaluación se realizó en la asignatura "Programación Avanzada", optativa de 2 créditos de 6º nivel de Ingeniería de Sistemas, con 25 alumnos matriculados.

La asignatura iba a tratar el diseño y análisis de algoritmos. Sin embargo, consultado el coordinador de la titulación (y después algunos profesores) sobre los conocimientos de los alumnos, resultó evidente que la instrucción de los alumnos en recursividad había sido muy baja. Se rediseñó el contenido de la asignatura para aumentar sus conocimientos de recursividad y así poder tratar posteriormente la complejidad de algoritmos y la técnica de divide y vencerás.

Debido a las restricciones de presencia del profesor en la ESPOCH y a diversas incidencias con el calendario, la asignatura se impartió durante cinco semanas con tres sesiones semanales de dos horas cada una. La Tabla 1 presenta las sesiones de clase impartidas. El tratamiento dado a los alumnos duró 3 semanas aproximadamente, resultando relevantes para este estudio sólo las sesiones 1-6 y 9. Se ha marcado en negrita las actividades relevantes de evaluación de los alumnos.

Tabla 1. Sesiones de impartición de la recursividad

Sesión	Contenidos
1.	Presentación de la asignatura Presentación informal de la recursividad
2.	Repaso de diseño y ejecución de algoritmos recursivos
3.	**Pretest** Motivación de la eliminación de la recursividad lineal
4.	Eliminación de la recursividad lineal
5.	Eliminación de la recursividad lineal
6.	**Práctica sobre eliminación de la recursividad**
7.	Conceptos de eficiencia y análisis complejidad de algoritmos iterativos
8.	Análisis de complejidad de algoritmos iterativos y notaciones asintóticas
9.	**Posttest** Análisis de complejidad de algoritmos recursivos
10.	Análisis de complejidad de algoritmos recursivos
11.	Test sobre complejidad
12.	Técnica de divide y vencerás
13.	Técnica de divide y vencerás
14.	Práctica sobre divide y vencerás

En el subapartado siguiente se revisa en detalle el contenido de las sesiones relevantes.

2.2 Diseño y Tratamiento

El diseño adoptado ha sido de medidas repetidas, ya que no había otra posibilidad. La medida se tomó mediante un cuestionario formado por cuatro problemas, que se usó como pretest y como posttest. Para evitar un efecto de repetición, a los alumnos se le atendió en tutoría sobre cualquier duda que tuvieran, salvo su rendimiento en el cuestionario. No obstante, se les argumentó que se les atendería en sus dudas sobre el cuestionario tras conocer sus resultados en una práctica distinta. A continuación se describe el tratamiento realizado, incluyendo ambas pruebas. Además, entre la primera realización del cuestionario y la segunda se dejó transcurrir el máximo tiempo admisible para que no se distorsionara el ritmo de la asignatura, que fueron dos semanas.

Se comenzó con dos sesiones de repaso de la recursividad. En la primera sesión, se presentó la asignatura y se realizó una introducción informal a la recursividad, con ejemplos visuales, de texto [3] y de dramatización [4]; se tuvieron en cuenta los problemas de algunas metáforas [5].

En la segunda sesión se recordaron los elementos de los algoritmos recursivos, ilustrado con ejemplos numéricos y sobre vectores, así como las distintas clases de recursividad. Se propuso a los alumnos que diseñaran algunas variantes de los ejemplos presentados. Por último, se mostraron algunas formas de analizar y representar los procesos recursivos: pila de control, rastros (o "trazas"), reescritura y árbol de recursión, con ayuda de los sistemas BlueJ, WinHIPE y SRec. Los sistemas BlueJ y SRec estuvieron disponibles en el campus virtual durante todo el periodo de impartición de la asignatura.

En la tercera sesión se entregó en papel un cuestionario de conocimientos a los alumnos. Este cuestionario sería parte de su evaluación en la asignatura. Aunque no tuvieron límite de tiempo, los alumnos lo contestaron en hora y media. La media hora restante se dedicó a motivar la eliminación de la recursividad.

En los días siguientes se informó a los alumnos de la calificación obtenida en el cuestionario, pero se retrasó explicarles los errores cometidos.

Las sesiones cuarta y quinta se dedicaron al método instruccional, en el que se presentó un método "general" de eliminación de la recursividad lineal y cuatro casos particulares, más sencillos [2].

La sesión sexta fue una sesión de laboratorio en la que los alumnos debía iniciar la realización de una práctica sobre eliminación de la recursividad lineal. Los alumnos disponían del resto de la semana (al ser martes, el resto del día y cinco días más) para realizar la práctica y entregar un informe. Mientras tanto se impartieron dos sesiones del tema siguiente.

Al realizar la sesión novena, ya había acabado el plazo de entrega de la práctica. Se volvió a entregar a los alumnos el mismo cuestionario.

En los días siguientes y una vez corregido, se les informó de sus nuevas calificaciones y a los alumnos interesados ya se les pudo explicar en tutorías los errores que habían cometido.

2.3 Tareas

El cuestionario utilizado como pre- y post-test constaba de cuatro preguntas:

1. Traducir una definición recursiva a Java (Q9 de [6]).
2. Detallar la ejecución de una función recursiva lineal.
3. Detallar la ejecución de una función recursiva múltiple (tarea 2 de [7]).
4. Diseñar un algoritmo iterativo y otro recursivo para un mismo problema (tarea 1 de [7]).

Denotaremos las cuatro preguntas como P1, P2, P3 y P4. P1 se usó para comprobar los conocimientos de los alumnos de Java imperativo. P2 y P3 eran los elementos principales para nuestra evaluación, en las que se pedía dar un rastro detallado (*trace*, "traza" en español de España, "prueba de escritorio" en español de Ecuador) de la ejecución del algoritmo correspondiente. P4 se utilizó de forma complementaria, para ver si el método instruccional también afectaba a la capacidad de los alumnos para diseñar algoritmos recursivos. El cuestionario se encuentra en el Apéndice A; puede verse que la operación de combinación en ambas definiciones recursivas es una suma.

La práctica consistía en convertir dos algoritmos recursivos (uno numérico y otro sobre vectores) en iterativos utilizando las técnicas vistas en clase. Los denotaremos Q1 y Q2, respectivamente. Para cada problema se les pedía ejecutar detalladamente el algoritmo sobre dos ejemplos, mostrando un rastro detallado de su ejecución. También se pedía identificar la regla a usar para eliminar la recursividad y presentar el algoritmo iterativo resultante. Los rastros de Q1 se utilizaron como forma secundaria de obtener información sobre la comprensión de la recursividad por los alumnos, intentando conseguir triangulación de resultados.

Los alumnos debían entregar sus prácticas siguiendo una plantilla establecida. El enunciado de la práctica y la plantilla de informe se encuentran en el Apéndice B.

3 Método de Análisis de los Modelos Mentales

En el apartado primero se resume el método de análisis utilizado y los modelos mentales identificados en este estudio como relevantes.

3.1 Método de Análisis

Primero, se corrigieron los cuestionarios del pretest para entregar las calificaciones a los alumnos. Esto sirvió de análisis preliminar ya que permitió tener una idea aproximada de las respuestas más frecuentes o más llamativas de los alumnos.

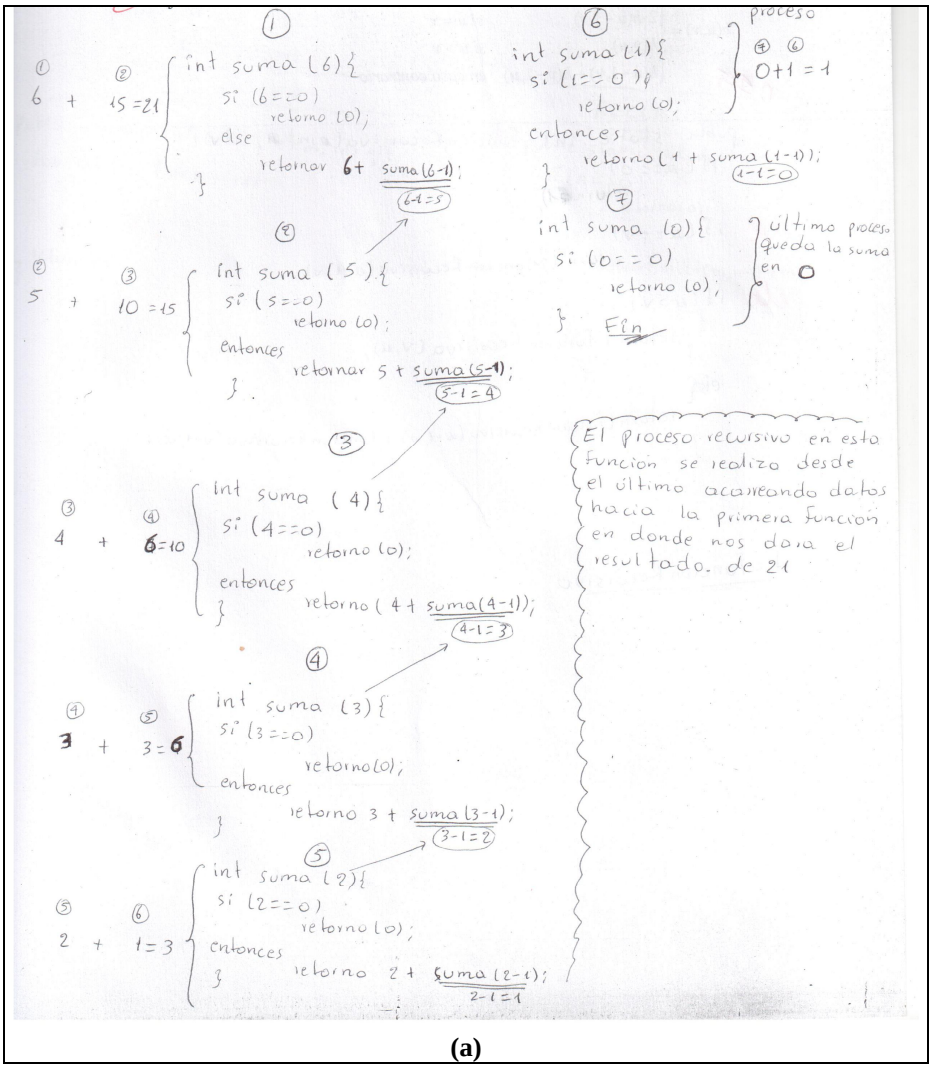
Una vez corregidos ambos tests, se analizaron en detalle. Se propusieron como categorías iniciales para nuestro análisis varias categorías extraídas de otros estudios anteriores: modelo de copias [6], modelo frágil [8], modelo activo [6] y "otros". También se utilizó un modelo no encontrado antes en la literatura, "definición recursiva". A partir de estos modelos, dos autores se repartieron las soluciones del pretest y del posttest y las clasificaron por separado. Celebraron tres reuniones hasta llegar a un consenso en su catalogación. A continuación, un tercer autor revisó la catalogación. Como resultado, se vio la conveniencia de introducir otro nuevo modelo ("modelo dudoso") y resolver algunas cuestiones, que presentamos a continuación. Posteriormente se realizó otra ronda más de análisis, con frecuentes revisiones de casos particulares.

3.2 Modelos Mentales

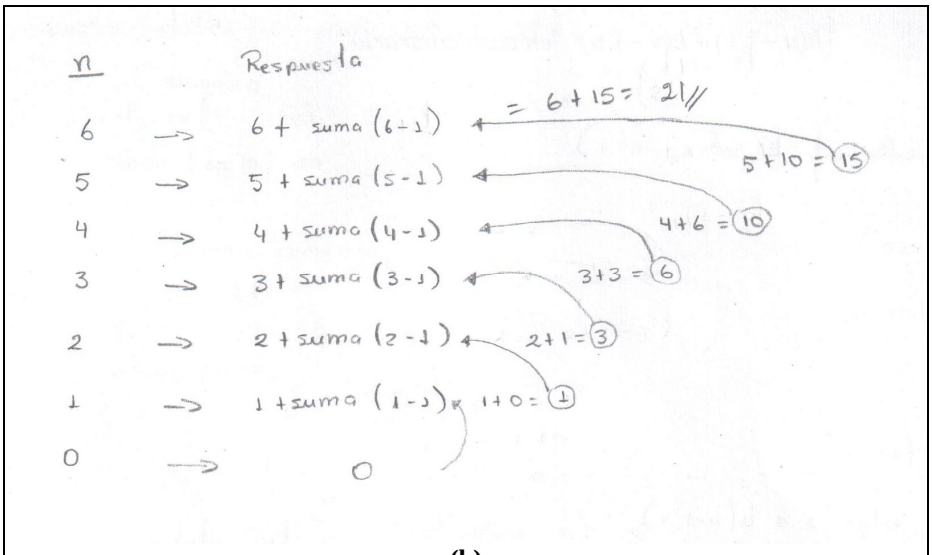
Veamos las categorías identificadas. Las ilustramos con las respuesta dadas por algunos alumnos en el pretest.

- Modelo viable. Corresponde al modelo de copias [6] o coherente [8], según los autores. A veces se hace explícito el proceso de vuelta (Fig. 1.a, A4 en P2); en otros casos queda implícito mediante el uso de flechas (Fig. 1.b, A2 en P2), de paréntesis (Fig 1.c, A8 en P2) o de resultados parciales (Fig 1.d, A13 en P2). Para la pregunta P3, a veces se utilizan árboles de recursión (Fig. 1.e, A10, y 1.f, A14) o notaciones similares (Fig. 1.g, A5).

6


int suma (6){
si (6==0)
retorno (0);
else
retornar 6+ suma(6-1);
}
int suma (5){
si (5==0)
retorno (0);
entonces
retornar 5 + suma(5-1);
}
int suma (4){
si (4==0)
retorno (0);
entonces
retorno (4 + suma(4-1));
}
int suma (3){
si (3==0)
retorno(0);
entonces
retorno 3 + suma(3-1);
}
int suma (2){
si (2==0)
retorno (0);
entonces
retorno 2 + suma(2-1);
int suma (1){
si (1==0)
retorno (0);
entonces
retorno (1 + suma (1-1));
int suma (0){
si (0==0)
retorno (0);
Fin
6+15=21
5+10=15
4+6=10
3+3=6
2+1=3
6-1=5
5-1=4
4-1=3
3-1=2
2-1=1
0+1=1
1-1=0
proceso
último proceso queda la suma en 0
El proceso recursivo en esta función se realizo desde el último acarreando datos hacia la primera función en donde nos daria el resultado. de 21

(a)



(b)

suma (6) = 6 + 5+4+3+2+1+0 = 21

suma (6) = 6 + suma(5) = 21

suma (6) = 6 + [5 + 4+3+2+1+0] = 21

Suma (6) = 6 + [5 + suma(4)] = 21

Suma (6) = 6 + [5 + (4 + 3+2+1+0)] = 21

Suma (6) = 6 + [5 + (4 + suma(3))] = 21

Suma (6) = 6 + [5 + (4 + [3+2+1+0])] = 21

Suma (6) = 6 + [5 + (4 + [3 + suma(2)])] = 21

Suma (6) = 6 + [5 + (4 + [3 + (2+1+0)])] = 21

Suma (6) = 6 + [5 + (4 + [3 + (2 + suma(1))])] = 21

Suma (6) = 6 + [5 + (4 + [3 + (2 + (1+0))])] = 21

Suma (6) = 6 + [5 + (4 + [3 + (2 + (1 + suma(0))])]) = 21

Suma (6) = 6 + [5 + (4 + [3 + (2 + (1 + (0)))])] = 21

caso general : suma(n) = n + suma (n-1);

caso base : n == 0 ; 0

(c)

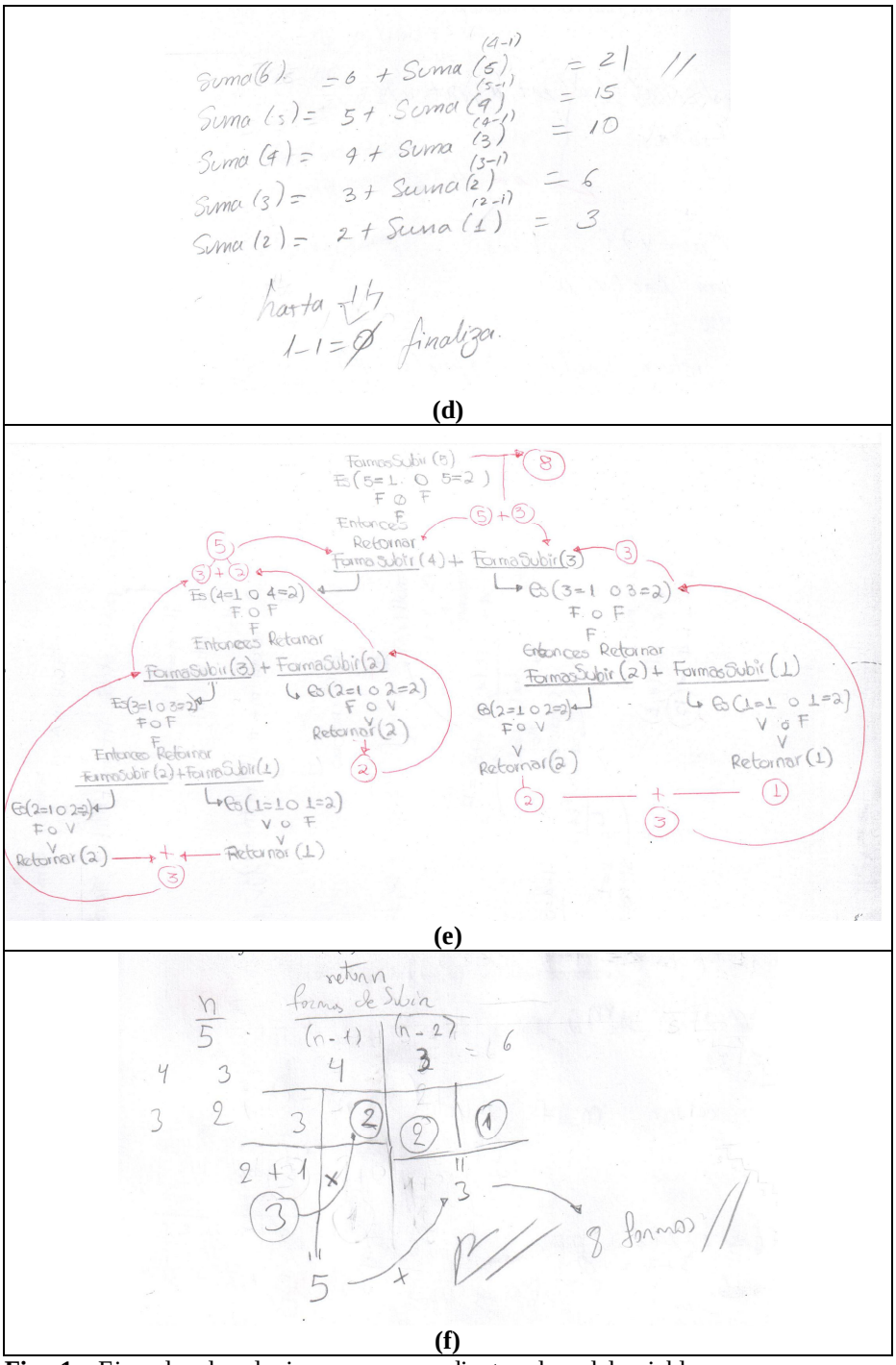


Fig. 1. Ejemplos de soluciones correspondientes al modelo viable.

- Modelo de avance [8]. Corresponde al modelo activo [6], haciéndose explícito el cálculo (suma) de los resultados parciales según avanza el proceso recursivo (véase Fig.2, A1 en P2).

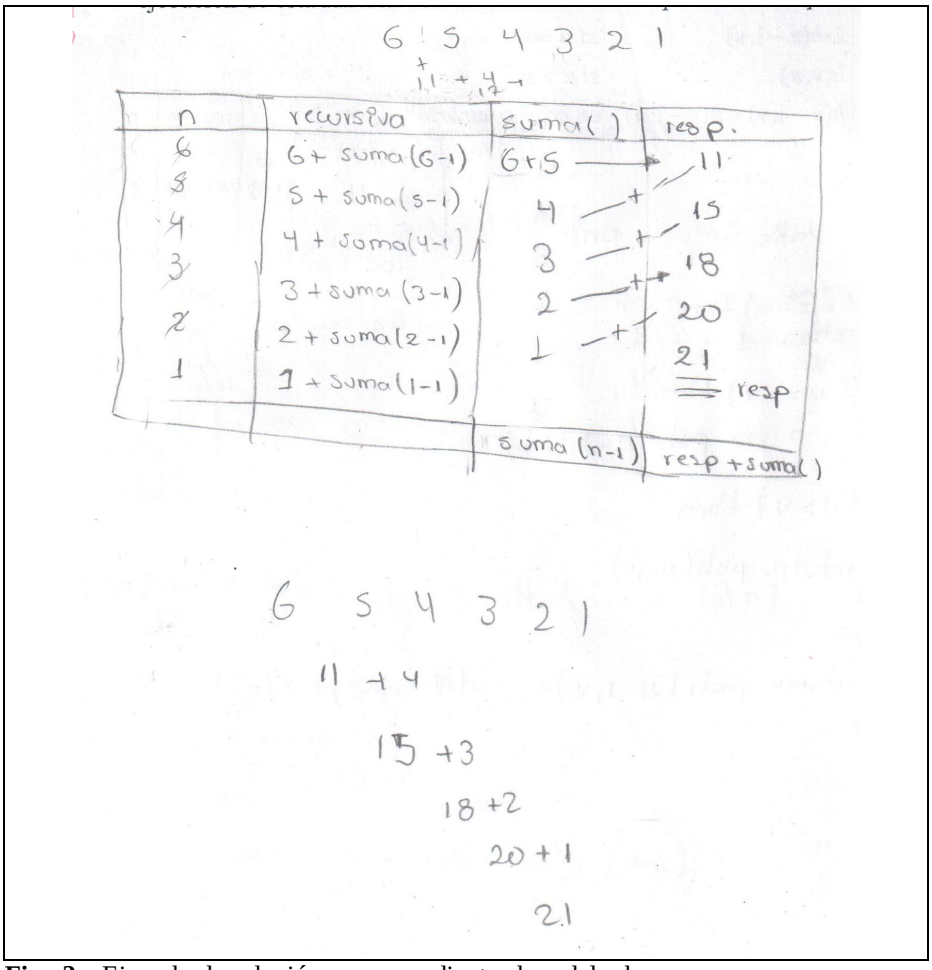
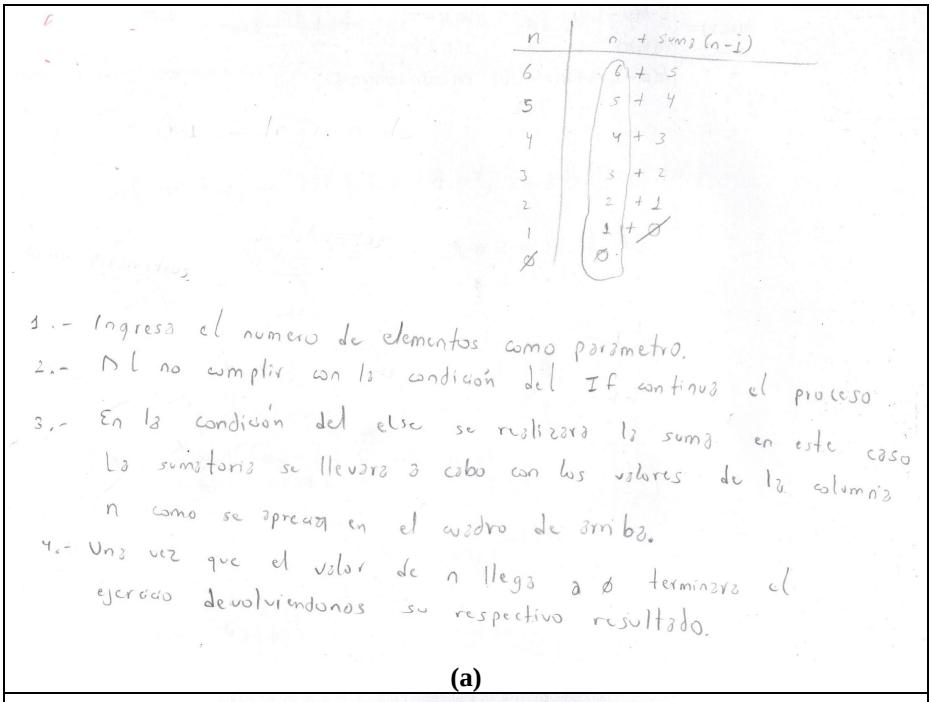


Fig. 2. Ejemplo de solución correspondiente al modelo de avance.

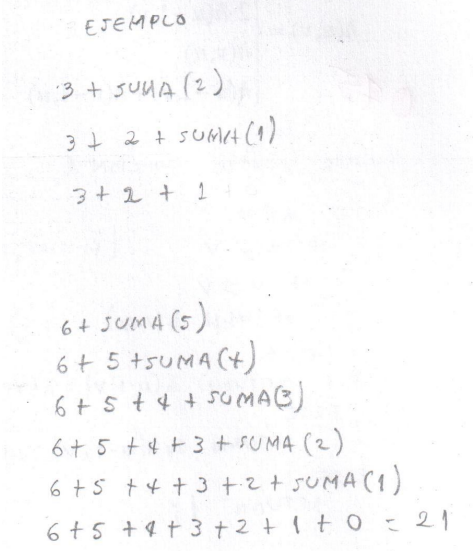
- Modelo dudoso. En realidad, esta categoría la introducimos por falta de información a partir de las soluciones para catalogar una solución en el modelo viable o de avance. Suele manifestarse en que no se conoce el orden del cálculo de las sumas (véase Fig. 3, A7 y A11 en P2, A17 en P3).



n	n + suma (n-i)
6	6 + 5
5	5 + 4
4	4 + 3
3	3 + 2
2	2 + 1
1	1 + ~~0~~
~~0~~	0

1.- Ingresa el numero de elementos como parametro.
2.- Si no cumplir con la condición del If continua el proceso
3.- En la condición del else se realizara la suma en este caso la sumatoria se llevara a cabo con los valores de la columna n como se aprecia en el cuadro de arriba.
4.- Una vez que el valor de n llega a ~~0~~ terminara el ejercicio devolviendonos su respectivo resultado.

(a)

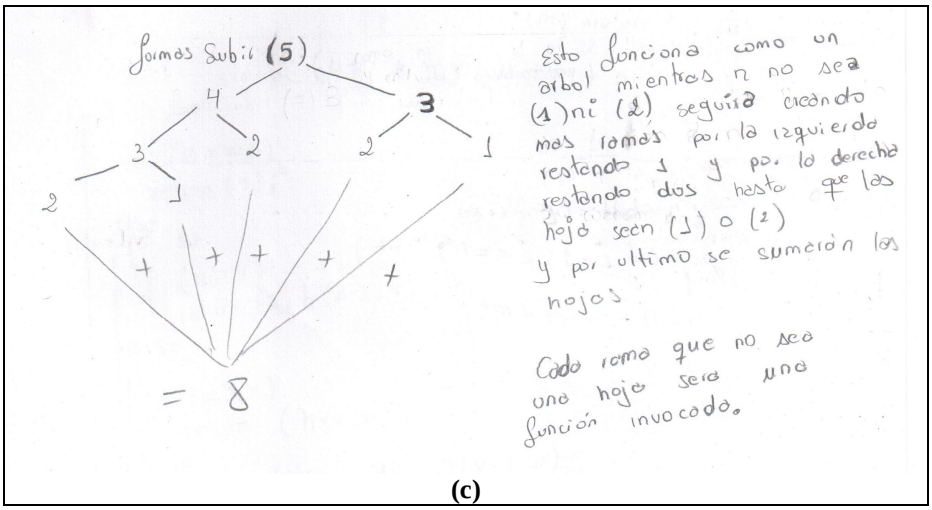


EJEMPLO

3 + SUMA (2)
3 + 2 + SUMA (1)
3 + 2 + 1

6 + SUMA (5)
6 + 5 + SUMA (4)
6 + 5 + 4 + SUMA (3)
6 + 5 + 4 + 3 + SUMA (2)
6 + 5 + 4 + 3 + 2 + SUMA (1)
6 + 5 + 4 + 3 + 2 + 1 + 0 = 21

(b)



(c)

Fig. 3. Ejemplos de soluciones correspondientes al modelo dudoso.

- Modelo de definición recursiva. Es un modelo nuevo. El alumno explica la ejecución del algoritmo utilizando la definición, pero sin desplegar más la llamada recursiva (véase Fig. 4, A21 en P2).

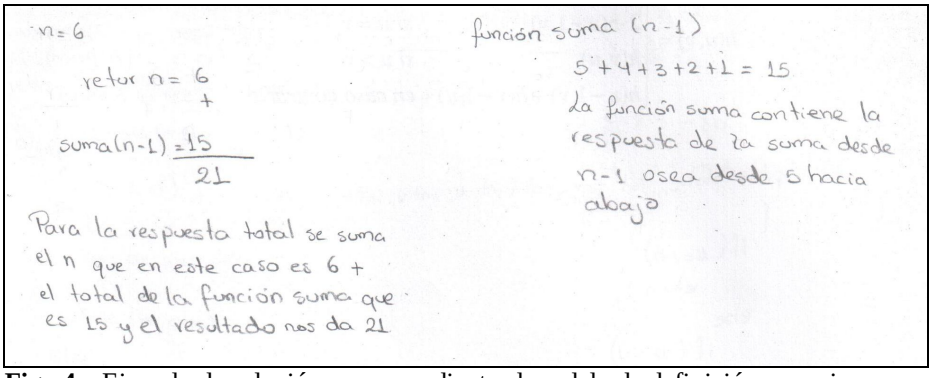
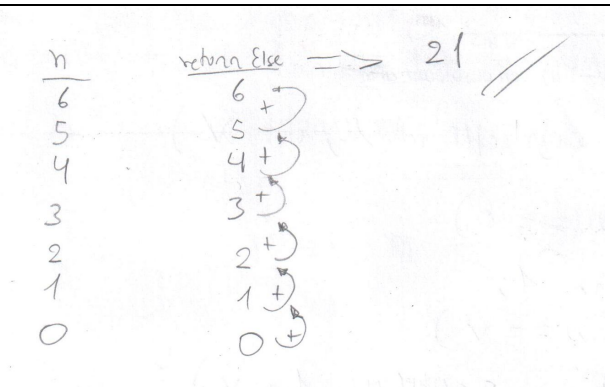


Fig. 4. Ejemplo de solución correspondiente al modelo de definición recursiva.

- Otros modelos. Se recogen otros modelos mentales distintos y minoritarios.
- Ninguno. No es un modelo mental propiamente dicho, sino que permite catalogar las respuestas dejadas en blanco.

Dentro del modelo viable, podemos distinguir entre viable completo y viable frágil [8]. Consideramos un modelo frágil cuando la respuesta contiene falta de información (véase Fig. 5.a, A5 en P2), pequeños errores o utiliza una notación inadecuada (Fig. 5.b A8 5.c A25 en P2), que muestra lo frágil del conocimiento del alumno. En el caso de recursividad múltiple, hay respuestas donde no queda claro si se realizan independientemente las llamadas recursivas redundantes o si éstas se realizan una sola vez (Fig. 5.d, A2 en P3).



(a)

suma (6) = 6 + 5 + 4 + 3 + 2 + 1 + 0 = 21

suma (6) = 6 + suma(5) = 21

suma (6) = 6 + [5 + 4 + 3 + 2 + 1 + 0] = 21

Suma (6) = 6 + [5 + suma(4)] = 21

suma (6) = 6 + [5 + (4 + 3 + 2 + 1 + 0)] = 21

Suma (6) = 6 + [5 + (4 + suma(3))] = 21

Suma (6) = 6 + [5 + (4 + [3 + 2 + 1 + 0])] = 21

suma (6) = 6 + [5 + (4 + [3 + suma(2)])] = 21

suma (6) = 6 + [5 + (4 + [3 + (2 + 1 + 0)])] = 21

suma (6) = 6 + [5 + (4 + [3 + (2 + suma(1))])] = 21

suma (6) = 6 + [5 + (4 + [3 + (2 + (1 + 0))])] = 21

Suma (6) = 6 + [5 + (4 + [3 + (2 + (1 + suma(0))])]) = 21

suma (6) = 6 + [5 + (4 + [3 + (2 + (1 + (0)))])] = 21

caso general : suma(n) = n + suma (n - 1);

caso base : n == 0 ; 0

(b)

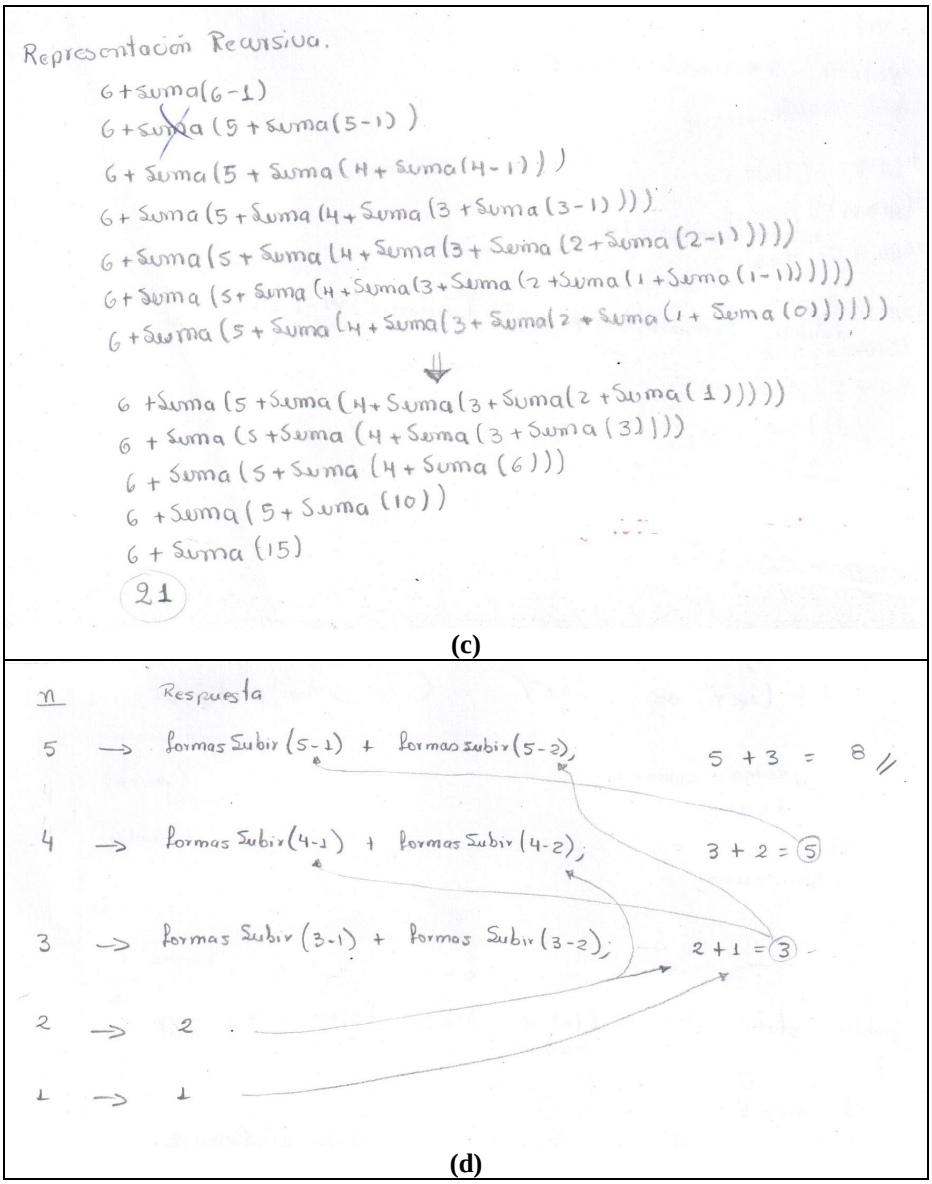
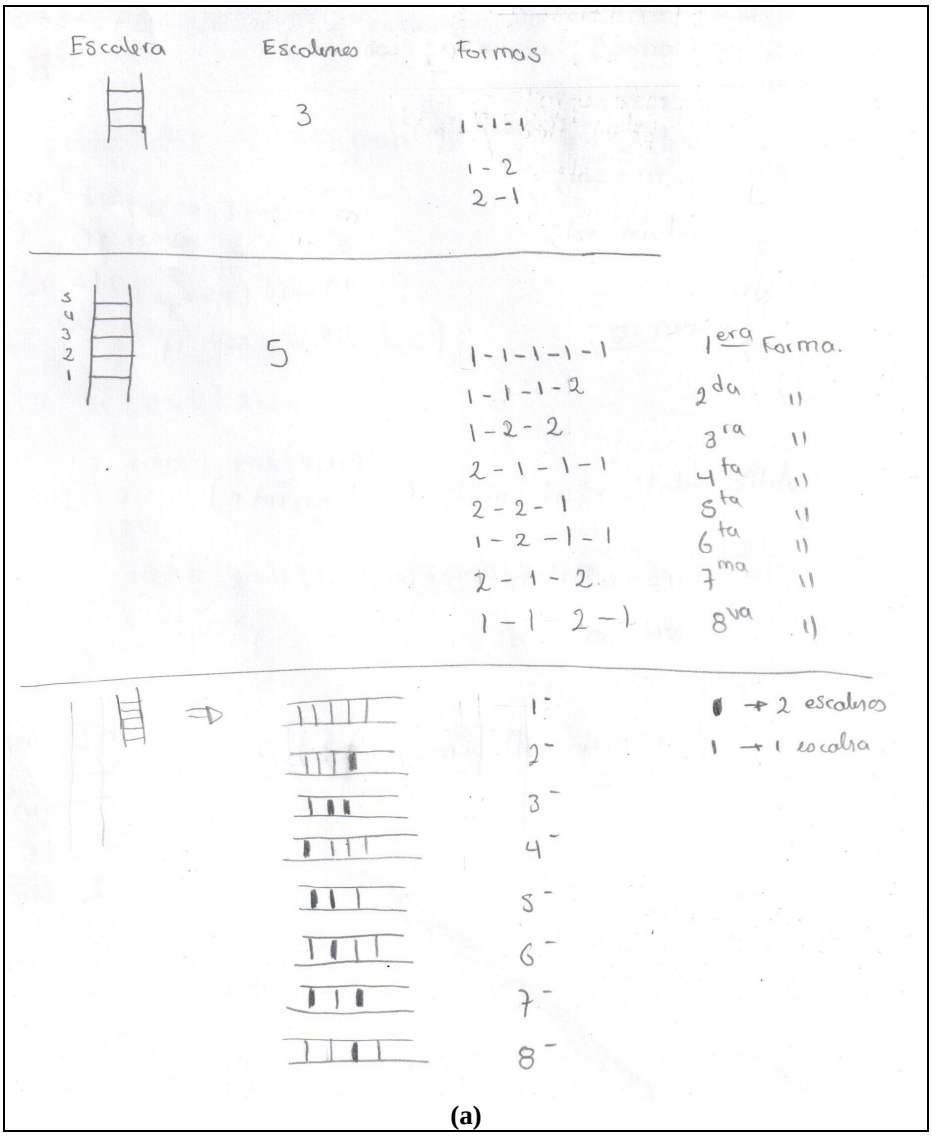


Fig. 5. Ejemplo de soluciones correspondientes al modelo viable fragil.

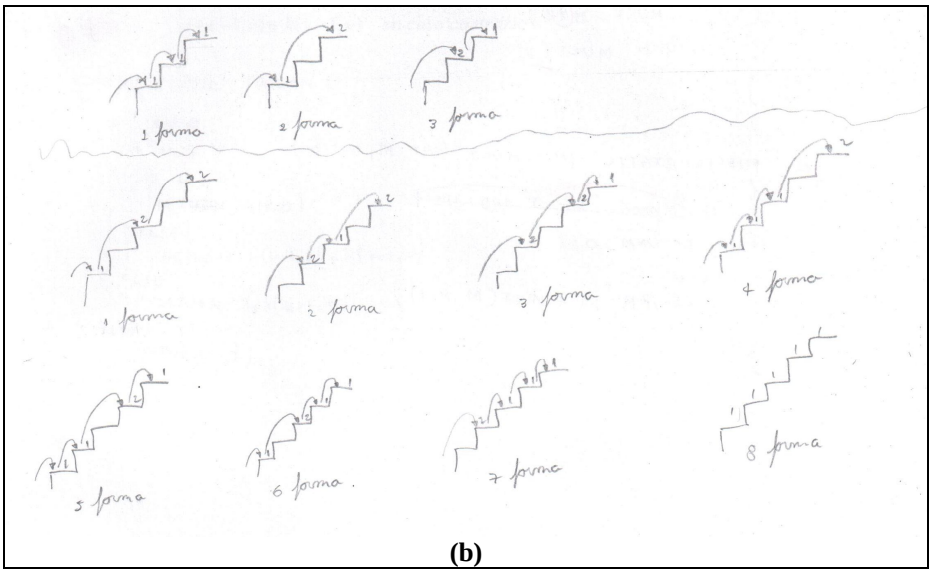
3.3 Dos Aclaraciones

En primer lugar, hay que mencionar que bastantes alumnos incluyeron la solución a algún ejemplo de los problemas, aunque no se pedía en el enunciado. Estos alumnos dieron una solución encontrada *ad hoc*, sin relación con el algoritmo proporcionado ni con ningún otro algoritmo. En el primer test, se incluyó 1 vez una solución para P2 y

11 veces para P3 y en el segundo test se presentó una solución 3 veces para P3. La Figura 6 contiene dos soluciones de P3, correspondientes de A1 y A11.



(a)



(b)

Fig. 6. Dos ejemplos de soluciones de P3 proporcionadas ad hoc.

Las soluciones dadas fueron correctas en 13 ocasiones y sólo fueron incompletas en 2. Asimismo, 9 veces fue la única respuesta de los alumnos, 5 veces se dio junto con soluciones incompletas de los modelos de avance u otros modelos, y sólo una vez se utilizó con un modelo viable y frágil. Por tanto, podemos obtener dos conclusiones:

- Las soluciones son un indicador de que el problema se ha entendido.
- En general, los alumnos dieron esta respuesta ante su incapacidad de contestar a lo que realmente se les pedía.

Por tanto, el modelo mental del alumno queda más claro si suprimimos estas soluciones, dejando en su lugar el modelo correspondiente o ninguno, según corresponda.

Una segunda cuestión es que bastantes alumnos presentaron un rastro incompleto. Cuando la información sea muy escasa, la catalogamos como ningún modelo; si incluye información suficiente para que saber que debería corresponder a un modelo viable o de avance, la catalogamos como modelo dudoso.

4 Resultados

Presentamos en primer lugar los rastros de las preguntas P2 y P3 del pretest y del posttest. Posteriormente analizamos los datos secundarios: primero, los rastros de la pregunta Q1 de la práctica y después el diseño de una función recursiva para P4, tanto en el pretest como en el posttest.

4.1 Resultados de P2 y P3

Se recogieron 22 cuestionarios del pretest y 25 del posttest. En la Tabla 2 se recogen los resultados del análisis de todos los cuestionarios respondidos. Para el análisis detallado presentado en este apartado sólo se considerarán los 22 cuestionarios correspondientes a alumnos que contestaron ambos tests.

Tabla 2. Modelos mentales de los alumnos en las preguntas P2 y P3, tanto en el pretest como en el posttest (*N*=25)

Alumno	P2 pretest	P2 posttest	P3 pretest	P3 posttest
A1	Avance	Viable completo	Ninguno	Viable completo
A2	Viable completo	Viable completo	Viable frágil	Viable completo
A3	Dudoso	Dudoso	Ninguno	Dudoso
A4	Viable completo	Viable completo	Dudoso	Viable completo
A5	Viable frágil	Viable frágil	Viable completo	Dudoso
A6		Dudoso		Otro
A7	Dudoso	Dudoso	Otro	Otro
A8	Viable frágil	Viable completo	Viable completo	Dudoso
A9		Avance		Avance
A10	Dudoso	Viable completo	Viable completo	Dudoso
A11	Dudoso	Dudoso	Ninguno	Ninguno
A12	Viable frágil	Viable completo	Ninguno	Viable completo
A13	Viable frágil	Viable frágil	Viable frágil	Dudoso
A14	Avance	Viable completo	Viable completo	Viable completo
A15	Avance	Viable completo	Viable frágil	Viable completo
A16	Ninguno	Dudoso	Ninguno	Otro
A17	Viable frágil	Viable frágil	Dudoso	Viable completo
A18	Avance	Viable completo	Ninguno	Viable completo
A19		Viable completo		Otro
A20	Dudoso	Dudoso	Dudoso	Dudoso
A21	Defin. recursiva	Defin. recursiva	Ninguno	Ninguno
A22	Defin. recursiva	Dudoso	Ninguno	Otro
A23	Avance	Dudoso	Ninguno	Dudoso
A24	Defin. recursiva	Viable completo	Ninguno	Otro
A25	Viable frágil	Viable frágil	Ninguno	Viable frágil

La Tabla 3 contiene los resultados de P2, desglosados por modelos mentales, tanto en el pretest como en el posttest.

Tabla 3. Número de respuestas de P2 por modelo mental (*N*=22)

Modelo mental	# respuestas en pretest	# respuestas en posttest
Viable completo	2	10
Viable frágil	6	4
Avance	5	
Dudoso	5	7
Definición recursiva	3	1
Ninguno	1	

Puede observarse que hay una mejora clara en el número de respuestas correspondientes al modelo viable (de 8 a 14) y, dentro de éste, en su calidad (un aumento de 2 completas a 10, y un decremento de 6 frágiles a 4). También desaparece el modelo de avance, ningún alumno deja la pregunta sin responder con un rastro y casi desaparece el modelo de llamada recursiva. Por contra, aumenta el número de modelos dudosos.

La Tabla 4 contiene los resultados obtenidos para P3, también desglosados por modelos mentales.

Tabla 4. Número de respuestas de P3 por modelo mental (*N*=22)

Modelo mental	# respuestas en pretest	# respuestas en posttest
Viable completa	4	8
Viable frágil	3	1
Dudoso	3	7
Otros	1	4
Ninguno	11	2

No aumenta demasiado el número de modelos viables (de 7 a 9), pero sí lo hace el número de modelos viables completos (de 4 a 8) y disminuye el número de modelos viables frágiles (de 3 a 1). Disminuye mucho el número de alumnos que no responden nada significativo (de 12 a 6), aumentando el número de modelos dudosos.

Podemos analizar la mejora o empeoramiento de cada alumno del pretest al posttest suponiendo el orden parcial "ninguno" < "otro" < "avance" < "dudoso" < "viable frágil" < "viable completo". Las tendencias aparecen resumidas en la Tabla 5. Sobre el modelo de definición recursiva, no sabemos si obedece a una comprensión clara del funcionamiento de la recursividad o a una repetición relativamente literal del código recursivo; por esta razón calificamos en la Tabla 5 el caso de A22 como indeterminado.

En todo caso, las situaciones de indeterminación de A22 en P2 y de empeoramiento de A5, A8, A10 y A13 en P3 revelan lo frágil de sus modelos mentales.

Tabla 5. Evolución de los modelos mentales para P2 y P3 (*N*=22)

Cambio	P2	Alumnos	P3	Alumnos
Mejoran	10	A1, A8, A10, A12, A14, A15, A16, A18, A23, A24	13	A1, A2, A3, A4, A12, A15, A16, A17, A18, A22, A23, A24, A25
Igual	11	A2, A3, A4, A5, A7, A11, A13, A17, A20, A21, A25	5	A7, A11, A14, A20, A21
Empeoran			4	A5, A8, A10, A13
Indeterminado	1	A22		

4.2 Resultados de la Práctica

Presentamos en este subapartado los resultados obtenidos en la práctica de eliminación de la recursividad. Se recogieron 14 informes, 11 realizados en pareja y 3 realizados individualmente.

En la Tabla 6 se muestran los resultados obtenidos. La catalogación de los rastros presentados en la práctica se han colocado entre los resultados del pretest y del posttest para facilitar su comparación.

Tabla 6. Respuestas de los alumnos a las preguntas P2 y P3, tanto en el pretest como en el posttest (*N*=22)

Grupo	Alumno	P2 pretest	P3 pretest	Q1 - Ej.1	Q1 - Ej.2	Q2 - Ej.2	Q2 - Ej.2	P2 posttest	P3 posttest
G1	A1	Avance	Ninguno	Viable completo (SRec)	Vacío	Viable completo (SRec)	Vacío	Viable completo	Viable completo
	A22	Defin. recursiva	Ninguno					Dudoso	Otro
G2	A2	Viable completo	Viable frágil	Viable completo	Viable completo	Datos mal	Datos mal	Viable completo	Viable completo
	A5	Viable frágil	Viable completo					Viable frágil	Dudoso
G3	A3	Dudoso	Ninguno	Avance	Avance	Dudoso correcto	Dudoso correcto	Dudoso	Dudoso
	A21	Defin. recursiva	Ninguno					Defin. recursiva	Ninguno
G4	A4	Viable completo	Dudoso	Viable completo	Vacío	Viable completo	Viable completo	Viable completo	Viable completo
	A18	Avance	Ninguno					Viable completo	Viable completo
I5	A6			Vacío	Vacío	Vacío	Vacío	Dudoso	Otro
G6	A7	Dudoso	Otro	Viable completo	Viable completo	Dudoso correcto	Dudoso correcto	Dudoso	Otro
	A24	Defin. recursiva	Ninguno					Viable completo	Otro
G7	A8	Viable frágil	Viable completo	Viable completo	Vacío	Incorrecto	Vacío	Viable completo	Dudoso
	A16	Ninguno	Ninguno					Dudoso	Otro
G8	A9			Vacío	Viable completo	Dudoso correcto, incompleto	Vacío	Avance	Avance
	A23	Avance	Ninguno					Dudoso	Dudoso
G9	A10	Dudoso	Viable completo	Vacío	Viable completo	Dudoso correcto	Vacío	Viable completo	Dudoso
	A14	Avance	Viable completo					Viable completo	Viable completo
I10	A11	Dudoso	Ninguno	Ninguno (SRec)	Vacío	Ninguno	Vacío	Dudoso	Ninguno
G11	A12	Viable frágil	Ninguno	Viable completo	Vacío	Dudoso correcto, incompleto	Vacío	Viable completo	Viable completo
	A19							Viable completo	Otro
I12	A13	Viable frágil	Viable frágil	Vacío	Avance	Vacío	Datos mal	Viable frágil	Dudoso
G13	A15	Avance	Viable frágil	Vacío	Vacío	Vacío	Vacío	Viable completo	Viable completo
	A17	Viable frágil	Dudoso					Viable frágil	Viable completo
G14	A20	Dudoso	Dudoso	Viable frágil	Vacío	Dudoso frágil	Vacío	Dudoso	Dudoso
	A25	Viable frágil	Ninguno					Viable frágil	Viable frágil

Hay que notar algunas incidencias. El grupo G1 presentó un ejemplo distinto de los dados en el enunciado para Q1, aunque con un rastro más largo. G7 presentó una evaluación incorrecta e incompleta de unos datos no pedidos. Los grupos G8 y G11 presentan el mismo formato y el mismo error en el rastro de Q2; el grupo G8 lo acompaña de más información, por lo que quizá haya sido plagiado por G11. Lo mismo podría suceder en Q1, pero cada grupo presenta un ejemplo distinto. I10 presentó el rastro obtenido por SRec ¡para los algoritmos iterativos!

Descartamos un análisis más profundo de Q2 porque, al ser un algoritmo recursivo final, desconocemos si, en algunos casos, la categoría de dudoso oculta que el grupo comprende bien su ejecución. Tampoco parece viable relacionar los resultados de la práctica con el algoritmo recursivo múltiple Q3. Por tanto, nos centramos solamente en relacionar las respuesta de los alumnos para los algoritmos recursivos lineales y no finales P2 y Q1.

La Tabla 7 muestra la evolución del rendimiento de los grupos desde la pregunta Q1 de la práctica a la pregunta P2 del posttest. Dado que el informe de la práctica se elaboró tras estudiar la teoría de eliminación de la recursividad lineal y tras aplicarla en la propia práctica, los resultados deberían ser similares en ambas pruebas. El resultado similar debería darse al menos para uno de los miembros del grupo ya que es frecuente que uno de los miembros lidere la actividad del grupo.

Una pareja ha obtenido una mejora de un miembro, como mínimo, al realizar el posttest. Otras 8 parejas mantienen su resultado para un miembro del grupo, al menos. Solamente un individuo (I12) y un grupo (G8) han tenido peor catalogación en P2 del posttest que en Q1; en lo que respecta a G8, este hecho resta evidencia a la sospecha de plagio antes señalada. Otros 3 grupos (I5, I10 y G13) no han incluido rastro en la práctica, por lo que es difícil juzgar la utilidad de ésta.

Tabla 7. Evolución del rendimiento desde Q1 de la práctica a P2 del posttest

Evolución	# grupos	Grupos
Un miembro mejora, otro indeterminado	1	G3
Ambos miembros igual	3	G4, G9, G11
Uno igual, otro empeora	5	G1, G2, G6, G7, G14
Ambos empeoran	2	G8, I12
Sin información suficiente	3	I5, I10, G13

En resumen, la mayoría de los resultados son coherentes entre la práctica y el posttest.

4.3 Resultados de P4

Veamos si hubo algún cambio apreciable en los alumnos en su capacidad de diseñar funciones recursivas.

La Tabla 8 muestra los resultados de los alumnos en P4, tanto el pretest como en el posttest.

Tabla 8. Respuestas de los alumnos a la pregunta P4 y modelo de su rastro, en caso de realizarlo

Alumno	P4 pretest	Rastro P4 pretest	P4 posttest	Rastro P4 posttest
A1	Correcto	Avance	Correcto	Dudoso
A2	Falta caso *n*=0	Viable completo	Falta caso *n*=0	Dudoso
A3	Resultado 1 para *n*=0		Correcto	Dudoso
A4	Correcto		Falta caso *n*=0	
A5	Falta caso 0 Caso básico sobre *m*, pero recursión sobre *n*		Falta caso *n*=0	Viable frágil
A6			Correcto	
A7	Correcto (caso *m*=0 redundante)		Mal intercambio de parámetros y caso básico	
A8	Correcto		Falta caso *n*=0 Parámetro acumulador mal usado	
A9			Correcto	
A10	Falta caso *n*=0	Viable frágil	Correcto	Avance
A11	Caso básico "*m* mod 2 == 0 AND *n*=1"		Falta caso *n*=0	
A12	Correcto	Dudoso	Correcto	Viable completo
A13	Correcto (una condición y un caso que sobran)		Falta	Avance
A14	Correcto (caso *n*=0 redundante)	Dudoso	Correcto (casos *n*=0 y *m*=1 redundantes)	Viable completo
A15	Caso *m*=1 debe dar 0 Caso *n*=0 redundante	Avance	Correcto	Viable completo
A16	Incompleto		Caso *n*<1 debe dar 0 Falta *m* en llamada recursiva	
A17	Falta caso *n*=0		Falta caso *n*=0	
A18	Correcto (4 casos básicos)		Correcto	Viable frágil
A19			Falta caso *n*=0	Viable completo
A20	Correcto (caso *m*=0 redundante)		Correcto	
A21	Condición compuesta mal escrita Recursión doble (sobre *m*, *n*)		Caso *m*=0 redundante Falta -1 en llamada recursiva	
A22	Correcto (caso *n*=0 redundante)		Caso básico compuesto Recursión sobre *n*+*n*	
A23	Correcto		Caso *n*=0 debe dar 0	
A24	Falta caso *n*=0		Falta caso *n*=0	
A25	Caso *n*=1 debe ser 0	Viable frágil	Correcto	Dudoso

Llaman la atención los alumnos A7, A8 y A22, que han empeorado sustancialmente su solución. Por el contrario, A11, A16 y A21 experimentan avances sustanciales. En muchos otros alumnos, aunque pueda observarse una mejora o

empeoramiento, es en detalles que no reflejan un cambio cualitativo entre ambas soluciones.

En la Tabla 9 se muestra si hay mejoras entre ambos tests en las respuestas a P4. Aunque en la Tabla 8 hemos recogido todos los resultados, en la Tabla 9 sólo comparamos los resultados para los 22 alumnos que realizaron el pretest y el posttest. Aunque hay un número ligeramente mayor de alumnos que mejoran, no parece que sea una diferencia significativa.

No penalizamos que haya un caso básico redundante (que el otro parámetro sea igual a cero), aunque sí que haya más casos básicos. También consideramos peor un algoritmo incompleto que uno completo, que haya errores graves de diseño de la recursividad, un algoritmo incomprensible frente a un algoritmo incorrecto pero "aceptable", un algoritmo sin el caso básico 0 ó que devuelva un resultado equivocado para un caso básico (aunque es probable que sea un error "tonto") frente al algoritmo correcto. El caso del alumno A13 no lo consideramos porque no incluye algoritmo pero incluye un rastro: podría ser que se le olvidara escribir el algoritmo si lo pasó a limpio.

Tabla 9. Evolución del rendimiento en P4 del pretest al postest (*N*=22)

Cambio	#alumnos	Alumnos
Mejoran	9	A3, A5, A10, A11, A15, A16, A18, A21, A25
Igual	6	A1, A2, A12, A17, A20, A24
Empeoran	6	A4, A7, A8, A14, A22, A23
Indeterminado	1	A13

Por otro lado, la comparación de los modelos identificados en estos rastros con los correspondientes de P2 permiten afianzar nuestra confianza en los modelos mostrados, ya que repiten su catalogación de modelo mental. Es el caso de A1, A2 en el pretest, de A3, A5, A12, A14, A18 en el posttest, y de A15, A18 en ambos tests. También es el caso de A12, que sólo tiene un rastro esbozado en P4 del pretest para uso propio, y de A14, cuyo modelo dudoso en P4 del pretest puede corresponder al avance de P2. De hecho, algunos de estos alumnos, A12, A14, A15, A18 muestran una consolidación (aparentemente) clara de su modelo mental en el posttest, porque muestran modelos viables en las tres preguntas P2, P3 y P4.

Sin embargo, los alumnos A1, A2 y A25 presentan un rastro dudoso en P4 del posttest, cuando han presentado un modelo viable en P2. Dado que no se les pedía ningún rastro en P4, quizá han descuidado su presentación porque era simplemente para asegurarse de que el algoritmo funcionaba correctamente. En todo caso, ilustra lo frágil de su conocimiento o de su notación.

Esta fragilidad la muestra A10 de forma aún más acentuada, ya que mezcla claramente ambos modelos. En el pretest, presentó un modelo dudoso para P2, un modelo viable completo para P3 (algoritmo más complejo) y un modelo viable frágil para P4. En concreto, el rastro de P4 es completo, pero también incluye un resumen sin paréntesis; si no lo diera tan detallado, con dos rastros paralelos, lo habríamos catalogado como dudoso. Por tanto, su modelo dudoso para P2 parece deberse a una precipitación o descuido, no a un malentendido. En el posttest vuelve a repetirse esta situación contradictoria, pero más grave: viable completo en P2 y avance en P4,

cuando el proceso recursivo en ambas funciones es similar. Por tanto, es cuestionable la solidez de su modelo mental.

5 Conclusiones

Hemos realizado una evaluación de un nuevo método instruccional para la comprensión de la recursividad. Resumimos los resultados más relevantes obtenidos:

- Para P2, la mejora de los alumnos en el posttest ha sido clara. Todos los alumnos han logrado responder con algún rastro. Además, dos tercios de los alumnos han desarrollado un modelo viable, siendo completo para la mayoría. Es decir, han logrado desarrollar un modelo viable y (relativamente) sólido. Ha desaparecido el modelo de avance y casi desaparece el modelo de llamada recursiva. Los alumnos sin un modelo claro se han desplazado de no presentar ningún rastro a un modelo dudoso.
- Para P3, no aumenta demasiado el número de modelos viables, pero sí lo hace en su solidez: aumenta el número de modelos viables completos y disminuye el número de modelos viables frágiles. De nuevo, disminuye mucho el número de alumnos que no responden nada significativo, aumentando el número de modelos dudosos.
- Los resultados de Q1 son (relativamente) coherentes con los resultados del posttest.
- Los resultados de P4 no muestran una mejora significativa en sus habilidades para diseñar algoritmos recursivos.

Agradecimientos. Este trabajo se ha financiado con el proyecto "Becas Prometeo" de la SENESCYT y con el proyecto TIN2011-29542-C02-01 del Ministerio de Economía y Competitividad de España.

Referencias

1. Rinderknecht, C.: A survey on teaching and learning recursive programming. Informatics in Education 13, 1 (2014) 87-119
2. Scholl, P.C.: Algorítmica y representación de datos, recursividad y árboles. Masson (1986)
3. Levy, D., Lapidot, T.: Recursively speaking: Analyzing students' discourse of recursive phenomena. En: Proceedings of the 31st SIGCSE Technical Symposium on Computer Science Education (2000) 315–319
4. Ben-Ari, M.: Recursion: From drama to program. Computer Science Education 11, 3 (1997) 9–12
5. Forišek, M., Steinová, M.: Metaphors and analogies for teaching algorithms. En: Proceedings of the 43rd SIGCSE Technical Symposium on Computer Science Education (2012) 15-20
6. Mirolo, C.: Learning (through) recursion: A multidimensional analysis of the competences achieved by CS1 students. En: Proceedings of the Fifteenth Annual Conference on Innovation and Technology in Computer Science Education (2010) 160–164

7. Ginat, D., Shifroni, E.: Teaching recursion in a procedural environment – How much should we emphasize the computing model? En: Proceedings of the 30th SIGCSE Technical Symposium on Computer Science Education (1999) 127–131
8. Götschi, T., Sanders, I., Galpin, V.: Mental models of recursion. En: Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education (2003) 346-350

Apéndice A: Enunciado del Cuestionario

Ingeniería de Sistemas
Asignatura *Programación Avanzada*
Curso 2013/2014
Test

1. Codifica en Java la siguiente función recursiva *h*, definida sobre todos los pares de naturales:

$$h(u,v) = \begin{cases} 1 & si\ u = 0 \\ 2 \cdot h(u-1,v) & si\ u = v \\ h(v,u) & si\ u > v \\ h(u-1,v) + h(v-1,u) & en\ caso\ contrario \end{cases}$$

2. La siguiente función recursiva calcula la suma de todos números naturales hasta *n*

```
public static int suma (int n) {
   if (n==0)
      return 0;
   else
      return n + suma(n-1);
}
```

Por ejemplo, *suma*(3) = 6 porque es el resultado de sumar, 1, 2 y 3.

Ilustra la ejecución de esta función con todo detalle, de forma que se muestre cómo se calcula el resultado *suma*(6) = 21.

3. La siguiente función recursiva calcula de cuántas formas es posible subir una escalera de *n* peldaños (*n*>0), suponiendo que en cada paso se puede subir uno o dos peldaños.

```
public static int formasSubir (int n)
   if ((n==1) || (n==2))
      return n;
   else
      return formasSubir(n-1) + formasSubir(n-2);
}
```

Por ejemplo, *formasSubir*(3) = 3 porque una escalera de 3 escalones puede subirse de tres maneras: 1-1-1, 1-2 y 2-1.

Ilustra la ejecución de esta función con todo detalle, de forma que se muestre cómo se calcula el resultado *formasSubir*(5) = 8.

4. Diseña dos funciones en Java, una iterativa y otra recursiva, ambas con la siguiente cabecera:

```
public static int mult (int m, int n)
```

Ambas funciones deben hallar el producto de dos números naturales, es decir, deben calcular *m***n*. Además, deben calcular el producto mediante sumas sucesivas.

Apéndice B: Enunciado y Modelo de Informe de la Práctica

Ingeniería de Sistemas
Asignatura *Programación Avanzada*
Curso 2013/2014
Práctica nº 1

Objetivo
El objetivo de la práctica es que el alumno practique en la eliminación de la recursividad lineal.

Carácter
La práctica se realizará preferentemente en parejas.

Prerrequisitos
El alumno debe tener nociones básicas de recursividad, análisis de complejidad y la propia técnica de eliminación de la recursividad lineal.

Enunciado
Sean los dos problemas siguientes.

Logaritmo entero. Dados dos números naturales, $b>1$, $n\geq 1$, se desea hallar la parte entera del logaritmo en base b de n. Por ejemplo, *logEntero*(2,8) = 3, y *logEntero*(2,9) = 3.

Recuérdese que $\log_b n$ es un número z tal que $b^z = n$. Por tanto, el logaritmo puede determinarse por divisiones sucesivas: se cuenta el número de veces que puede dividirse n entre b. En Java:

```
public static int logEntero (int b, int n) {
   if (n<b)
      return 0;
   else
      return 1+logEntero(b,n/b);
}
```

Búsqueda binaria (o dicotómica). Dado un elemento de información e y un vector ordenado v y un elemento e, quiere determinarse si el elemento e se encuentra almacenado en el vector v. En caso de encontrarse, el algoritmo debe devolver la

posición de *v* donde se encuentra; en caso de no encontrarse almacenado en ninguna posición del vector, se devolverá el valor -1. Por ejemplo, *buscar*(3,{1,2,3,6,7,9}) = 2, y *buscar*(8,{1,2,3,6,7,9}) = -1.

En Java:

```
public static int buscar (int e, int[] v) {
   return busquedaBin (e, v, 0, v.length-1);
}
private static int busquedaBin
                 (int e, int[] v, int inf, int sup) {
   if (inf==sup)
      return (e==v[sup]?sup:-1);
   else {
      int med = (inf+sup) / 2;
      if (e>v[med])
         return busquedaBin (e, v, med+1, sup);
      else
         return busquedaBin (e, v, inf, med);
   }
}
```

Se pide, para cada uno de los dos problemas:

- Mostrar la ejecución del algoritmo para los dos ejemplos que ilustran su enunciado.
- Transformar el algoritmo recursivo en otro equivalente iterativo usando las técnicas explicadas en la teoría.

Entrega

El plazo de entrega es el domingo 13 de abril de 2014, incluido. Debe enviarse al profesor por medio del campus virtual una memoria elaborada siguiendo el modelo disponible en el sitio *web* de la asignatura.

Evaluación

Se evaluará la calidad y claridad de la memoria.

Ingeniería de Sistemas
Asignatura *Programación Avanzada*
Curso 2013/2014

Memoria de la práctica nº 1

Alumnos:

Apartados

La práctica comienza por un índice. Después contiene dos apartados, uno por algoritmo. A su vez, cada apartado contendrá los siguientes subapartados:

a) **Ejecución detallada.** Se presentan de forma comprensible los principales pasos de la ejecución del algoritmo para los dos ejemplos dados en el enunciado. Al menos debe informarse sobre las llamadas recursivas.

b) **Regla de traducción.** Se identifica y justifica la regla utilizada para transformar el algoritmo recursivo en el equivalente iterativo.

c) **Algoritmo iterativo.** Se incluye el código del algoritmo iterativo resultante.

d) **Conclusiones.** Se explican las conclusiones obtenidas tras realizar la práctica, por ejemplo, la preferencia de un algoritmo u otro, etc. También pueden incluirse cualesquiera otros comentarios sobre la práctica, si así se desea, por ejemplo, incidencias que han dificultado la realización de la práctica, sus aspectos más atractivos o más difíciles, sugerencias sobre cómo mejorar la práctica, etc.