VNIVERSIDAD NACIONAL
AVTONOMA DE
MEXICO

PERFILES
EDUCATIVOS
ISSN 0185-2698

Murray Lasso, Marco A. (1996)

"PARADIGMAS EN SOLUCIÓN DE PROBLEMAS MATEMÁTICOS Y USO DE LA COMPUTADORA"

en Perfiles Educativos, Vol. 18 No. 72 pp. 3-9.

Centro de Estudios sobre la Universidad

iresie
Banco de Datos sobre Educación

PERFILES EDUCATIVOS
Abril • Mayo • Junio 1996
Vol. 18 Núm. 72, pp. 3-9

PARADIGMAS EN SOLUCIÓN DE PROBLEMAS MATEMÁTICOS Y USO DE LA COMPUTADORA

M. A. Murray-Lasso [*]

Se presenta una metodología para la enseñanza de resolución de problemas matemáticos basada en la selección de acertijos matemáticos que sean paradigmas de problemas con diversas interpretaciones y aplicaciones. La solución de cada acertijo permite la introducción de conceptos importantes e ideas poderosas de solución que se pueden programar en una computadora. La metodología es motivante, permite recordar con facilidad los problemas modelo, presentarlos con un mínimo de antecedentes e introducir los conceptos e ideas poderosas en un contexto natural. La metodología es análoga al "método de casos" utilizado en algunas escuelas de administración de negocios.

♦

PARADIGMS IN THE SOLVING OF MATHEMATICAL PROBLEMS AND THE COMPUTER. *This paper presents a methodology for the teaching of solving mathematical problems, based in a selection of mathematical riddles, which are paradigms of problems with diverse interpretations and applications. The solution to each 3riddle enables the introduction of important concepts and powerful ideas for solutions, which can be programmed in a computer. The methodology is stimulating, it allows the user to easily recall the model problems, to present them with a minimum of antecedents and to introduce the concepts and ideas in a natural context. The methodology is analogous to the "Method of Cases" used in some business management schools.*

INTRODUCCIÓN

El autor considera, después de 33 años de experiencia docente en el nivel universitario, en la rama de ingeniería, que para adquirir habilidades en la resolución de problemas matemáticos es necesario que los alumnos resuelvan muchos tipos de problemas bajo la guía de un profesor sabio y experimentado. Es poco útil la repetición exhaustiva de ejercicios de sustitución de valores numéricos en fórmulas algebraicas. Unos cuantos ejercicios de este tipo bastan. Lo que se necesita es que el profesor seleccione un buen conjunto de problemas que ilustren los tipos de situaciones que se presentan y discuta dichos problemas en gran detalle, incluyendo las diferentes aplicaciones, las teorías e ideas poderosas que pueden utilizarse para resolverlos, los algoritmos y estructuras de datos asociados, así como la utilización de la computadora como herramienta para resolver los problemas. Problemas paradigmáticos como el sistema de segundo orden en el estudio de la dinámica le permite al maestro introducir conceptos de utilidad general, como la respuesta en la frecuencia, las frecuencias naturales, la resonancia, la estabilidad dinámica, etc. Los maestros deben estar constantemente alertas para descubrir estos problemas.

Los acertijos matemáticos son una buena fuente de paradigmas con muchas interpretaciones prácticas. Los juegos y los acertijos han sido el origen de varias ramas de la matemática, como son la teoría de

[*] **Jefe de la Unidad de Enseñanza Auxiliada por Computadora, División de Estudios de Posgrado, Facultad de Ingeniería, UNAM.**

gráficos y la teoría de probabilidad. La primera fue fundada por Euler con motivo del acertijo de los Siete Puentes de K Königsberg,[1] mientras que la segunda fue fundada por Pascal, motivada por discusiones sobre juegos de azar con Fermat,[2] de acuerdo con Leibnitz:[3] ...

> Los juegos en sí mismos ameritan ser estudiados, y si algún matemático profundo meditara sobre ellos, encontraría muchas consecuencias importantes, pues el hombre nunca ha mostrado más ingenio que en sus juegos.

Recientemente, una buena cantidad de investigación en inteligencia artificial se ha centrado en estudiar la posibilidad de que una máquina juegue ajedrez o "damas".[4] En investigación de operaciones, muchas aplicaciones importantes se manejan por medio de modelos y métodos prototipo que llevan nombres pintorescos, como "El problema de la mochila", "El problema del agente viajero" "El método Montecarlo", etcétera.[5]

No obstante que la mayoría de las personas resuelve los acertijos por ensayo y error, la solución de muchos de ellos puede formalizarse y pueden construirse teorías al respecto que garanticen soluciones algorítmicas para familias completas de problemas. Al construir dichas teorías, surgen ideas poderosas que bien merecen la atención y el tiempo de los estudiantes para estudiarlas, analizarlas y aplicarlas a la solución de problemas.

En este artículo se ilustran algunas de estas ideas con un acertijo sobre vaciado de vino.6 Se presentan ideas poderosas, como "estado", "gráfico" (en el sentido de la teoría de gráficos de Euler), "recursión", "etiquetado", y algoritmos excelentes como el de Dijkstra para "caminos más cortos." Se exploran las conexiones entre los problemas de vaciado de vino y el del camino más corto junto con algunas de sus interpretaciones y aplicaciones. Finalmente se ilustra la manera de usar la computadora para implantar los algoritmos en Logo y BASIC y se exhiben listados de programas. Existen muchos problemas semejantes al que nos ocupa aquí, y podría dárseles un tratamiento similar. el que otros autores trataran estos problemas en detalle, sería de gran utilidad para los maestros y un buen complemento a la literatura técnica sobre la enseñanza de resolución de problemas matemáticos auxiliada por computadora.

Un acertijo sobre vaciado de vino

Dos personas quieren dividir a mitades el vino contenido en una ánfora llena con capacidad de 8 litros. No cuentan con un dispositivo para medir líquidos y el ánfora no tiene marcas. Sin embargo, tienen disponibles dos ánforas vacías sin marcas con capacidades de 5 y 3 litros. ¿Cómo pueden medir los 4 litros?

Este conocido acertijo puede resolverse por ensayo y error, ya sea experimentalmente o en el papel, ensayando ideas hasta que uno le pega a la solución. Para apreciar las ventajas de contar con un método organizado de solución que siempre funciona, recomendamos al lector suspender la lectura e intentar obtener una solución al acertijo. El problema es relativamente fácil y no debería de tomar más de unos cuantos minutos para resolverse.

Estados

La primera idea poderosa que se presenta es la de estado de un sistema. El concepto es muy importante en matemáticas aplicadas, y central en ciencias de la computación, ya que aparece en conceptos tan básicos como las máquinas de estados finitos, incluyendo las máquinas de Turing y los algoritmos. No vamos a dar una definición formal de estado, sino que ilustraremos la idea en el contexto del problema que nos ocupa. Si el lector ya ha resuelto, o por lo menos intentado seriamente resolver el problema, se habrá percatado de que los cuatro litros de vino pueden medirse tras una serie de vaciados entre las diferentes ánforas. Dado que ninguna de las ánforas tiene marcas, las únicas situaciones que tienen sentido (es decir que no están indeterminadas) son aquellas en las cuales, comenzando con una ánfora que contiene vino, se vacía en otra ánfora hasta que una de dos: la ánfora de la cual se extrae el vino

quede vacía, o la ánfora a la cual se le agrega vino se llene. Las situaciones intermedias están indeterminadas. Podemos describir la condición o el estado del sistema por medio de una terna ordenada de números que indican la cantidad de vino en cada una de las ánforas con capacidades de 8, 5 y 3 litros,respectivamente. Así, el estado inicial está dado por la terna (8,0,0). Solamente hay dos estados a los que se puede pasar en una vaciada del estado inicial: al primero se llega al vaciar la ánfora de 8 litros en la ánfora de 5 , hasta que esta última se llena. El nuevo estado es (3,5,0). El segundo estado se logra al vertir la ánfora de 8 litros en la ánfora de 3, hasta que esta última se llena. el nuevo estado es (5,0,3).

Nótese que dado que supondremos que no se derrama nada de vino, la suma de los números de la terna que representa a cualquier estado siempre es 8. Esto significa que podemos describir cualquier estado por medio de dos de los tres números y deducir por medio de una resta el tercero. No obstante, utilizaremos la descripción usando los tres números.

Grafos

Una segunda idea poderosa que surge en la solución del problema planteado, es la idea de un grafo. Los grafos son colecciones de puntos y líneas que conectan los puntos. A los puntos también se les llama nodos o vértices, y a las líneas, aristas o eslabones. Nuestros estados quedarán asociados con los puntos, y la operación de vaciado corresponder a las líneas. El grafo que corresponde a los tres estados mencionados arriba aparece en la figura 1. Nótese que la posición geométrica de los puntos, así como la forma y longitud de las líneas, carecen de importancia en este caso. Sirven para visualizar propiedades estructurales, en vez de representar distancias físicas y formas espaciales. También nótese que las líneas o aristas tienen una flecha que indica la dirección de movimiento de un estado a otro. Esta propiedad proporciona a las aristas el nombre de *arista dirigida u orientada*, y los grafos correspondientes se llaman *grafos dirigidos u orientados*. La propiedad es obviamente importante porque, por ejemplo, si después de varios vaciados logramos llegar al estado (4,4,0), es evidente que de dicho estado podemos pasar con una sola vaciada al estado inicial (8,0,0), mientras que la operación inversa no se puede hacer, de lo contrario podríamos resolver el acertijo con una sola vaciada.

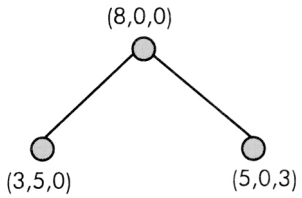

Figura 1

Vamos a continuar dibujando el grafo de la figura 1. Del estado (3,5,0) podemos ir con una sola vaciada a los estados (8,0,0), (0,5,3) y (3,2,3). Uno de éstos es el estado inicial que ya había aparecido antes, de modo que dibujamos una línea que conecta el estado (3,5,0) dirigida hacia el estado (8,0,0). Los otros dos estados generan nuevos puntos y sus correspondientes líneas de conexión. Del estado (5,0,3) se puede ir a los estados (8,0,0), (5,3,0). Solamente el estado (5,3,0) es nuevo y genera un nuevo punto y una nueva línea de conexión. Se puede continuar el proceso hasta que el grafo deje de crecer porque todos los estados a los que se puede llegar desde el estado inicial ya han aparecido. Estamos seguros de que el grafo dejar de crecer a la larga porque no hay modo de que ninguno de los estados tenga números que no sean enteros entre 0 y la capacidad de la ánfora correspondiente; por lo tanto, una cota superior al número de estados está dada por el producto 9x6x4=216. Es decir, el grafo no puede tener más de 216 estados diferentes. De hecho, como veremos más adelante, solamente tiene 16 estados diferentes. Todos los estados y las transiciones entre ellos se muestran en el diagrama completo de transiciones de estados de la figura 2.

Al observar la figura 2 nos percatamos de que el problema original se podría plantear como sigue: encontrar una ruta en la figura 2, la cual, circulando en el sentido de las flechas, comience en el estado (8,0,0) y termine en el estado (4,4,0). La siguiente secuencia de estados define una ruta con las características deseadas: (8,0,0), (3,5,0), (3,2,3), (6,2,0), (6,0,2), (1,5,2), (1,4,3), (4,4,0). Los vaciados correspondientes son fáciles de deducir de la secuencia de estados.

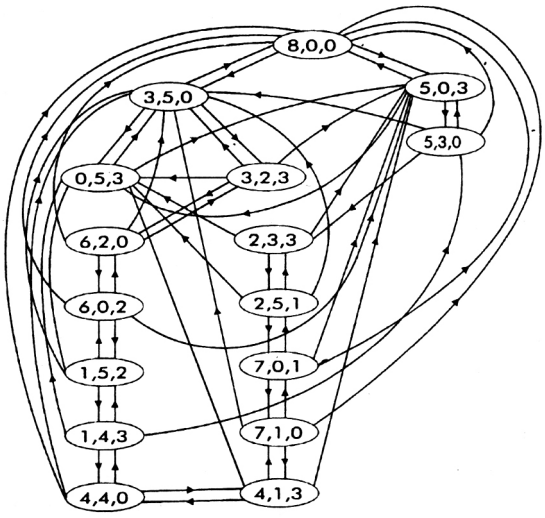

Figura 2

Entra la computadora

Puede resultar laboriosa la construcción del grafo de la figura 2, y definitivamente lo ser para aquellos casos en que las capacidades de las ánforas sean números mayores, como 69, 37 y 32. En dichos casos se puede recurrir a la computadora. En esta presentación introductoria solamente intentaremos obtener una lista estructurada de los estados. Comenzaremos por escribir en una línea el estado (8,0,0), y apuntaremos en la misma línea aquellos estados a los que, partiendo de (8,0,0), se puede llegar en una sola vaciada. Después procesaremos cada uno de los estados nuevos anotando cada uno de ellos en una nueva línea, y apuntaremos a continuación aquellos estados que, no habiendo aparecido antes, pueden alcanzarse desde el estado, al principio de la línea, en una sola vaciada. Continuamos con este proceso hasta que todos los estados hayan sido procesados y no aparezca ya ningún estado nuevo. El diseño de un programa para realizar lo descrito está basado en el hecho de que dados tres números (x,y,z) que representan un estado y dadas tres capacidades de ánforas en orden descendiente (a,b,c), podemos escribir un procedimiento para generar los posibles estados a los que puede llegarse en una vaciada. Ya hemos mencionado que existen hasta cuatro posibilidades para cada componente del estado: llenar una de las otras dos ánforas distintas a la que se está vaciando; o vaciar la ánfora desde la que se vierte a una de las otras dos. Esto implicaría 12 diferentes posibilidades para las 3 componentes del estado. Sin embargo, en la práctica ninguno de los estados tiene más de cuatro líneas que salen de él, aunque varios de ellos tienen más de cuatro líneas que entran a él.

Se ha escrito en LogoWriter un programa llamado ESTADOS, y su lista aparece en el Listado 1. Utiliza el procedimiento auxiliar NUEVO? A continuación hacemos algunos comentarios sobre estos procedimientos.

El procedimiento auxiliar NUEVO? revisa si la lista *p* que representa un estado generado, es miembro de una lista de listas *r*. En caso de que no lo sea, añade *p* al final de *r* y al final de *f*, las cuales son listas externas al procedimiento NUEVO? que se utilizan para acumular y procesar los estados en orden de aparición; a continuación el procedimiento NUEVO? añade la distancia de *p* del vértice origen a la lista externa *t* y saca la salida CIERTO; de lo contrario, en caso de que p haya aparecido en la lista r, saca la salida FALSO (véase el procedimiento ESTADOS más adelante para clarificar el significado de las listas *p, r, t*).

El procedimiento ESTADOS tiene siete parámetros. Los primeros tres *a, b* y *c* se mantienen constantes durante todo el problema, y corresponden a las capacidades de las ánforas en litros. (Para el ejemplo de arriba estos parámetros valen 8, 5 y 3). Los dos siguientes parámetros son listas de listas cuyos componentes son listas de 3 componentes, cada una de las cuales representa un estado. La lista f se utiliza para procesar los estados generados en orden de aparición. La lista r acumula los estados generados, de tal manera que cada estado generado puede revisarse para saber si ya había aparecido previamente y es producida como salida del procedimiento. El siguiente parámetro n es un contador del número de veces que se invoca el procedimiento, y por lo tanto de los estados diferentes generados a los que se puede acceder desde el estado inicial. El último parámetro *t* es una lista de distancias en aristas desde el vértice inicial. Dicha lista tiene un miembro por cada uno de los estados en el orden en que fueron numerados por el sexto parámetro. El procedimiento ESTADOS se llama a sí mismo recursivamente* y se detiene y da

como salida la lista *r* cuando la lista *f* queda vacía. Las variables locales *w, y, z* se utilizan para guardar las cantidades de vino en las ánforas en orden descendiente de sus capacidades. Después de la instrucción SI condicional para detener la recursión, cada instrucción SI condicional que comienza en un párrafo en el listado de ESTADOS analiza una de las dos posibilidades descritas arriba para la transición a nuevos estados (12 posibilidades para los 6 párrafos). Antes de añadir un nuevo estado a las listas *f* y *r*, se llama al procedimiento lógico NUEVO? para saber si el estado generado es nuevo; se busca en la lista r. Solamente los estados que no han aparecido previamente se añaden al final de las listas *r* y *f* como efectos laterales del procedimiento NUEVO?

Con respecto a la salida impresa de ESTADOS, para cada nodo padre se imprime una terna seguida de su distancia en aristas del vértice original. Para que la salida sea comprensible, se imprime un punto antes de la distancia y dos puntos antes de la lista de nodos hijo, separando cada uno con un punto. Al final del procedimiento, ESTADOS se llama a sí mismo eliminando de la lista *f* a la terna que corresponde al estado más recientemente procesado, suprimiendo también la distancia correspondiente de la lista t e incrementando en uno al contador n. A la larga, *f* queda vacía y el procedimiento se detiene.

A continuación se muestra una corrida de los procedimientos del Listado 1 para generar todos los estados accesibles desde el estado inicial (8,0,0,) del problema con ánforas de capacidades 8, 5 y 3.

Corrida de los procedimientos ESTADOS Y NUEVO?

Listado 1

```
para nuevo? :p
si otro no miembro? :p :r [da “r pul :p :r da  “f pul :p :f da “t pul (pr :t)
+ 1 :t re “cierto] [op “falso]
fin

Para estados :a :b :c :f :r :n :t
si :f = [] [re :r]
(inserta“ #:n:”PADRE= pr :f “. pr :t“..)
da “x pr pr :f da “ y pr mpr pr :f
da “z pr mpr mpr pr :f

si :x > 0 [siotro :x > (:b - :y) [si nuevo (fr :x + :y - :b :b :z)
[(inserta  ul  :f  “ .  )]]  [si  nuevo?   (fr 0 :x + :y :z)[(inserta ul :f “.)]]]
si :x > 0 [siotro :x > (:c-:z)[si nuevo?  fr :x + :z - :c  :y :c)
[(inserta  ul  :f  “ .  )]]  [si  nuevo?  (fr 0 :y + :x +:z)[(inserta ul :f “.)]]]

si :y > =[siotro :y > (:a-:x) [si nuevo?  (fr :a :y + :x -  :a :z)
[(inserta ul :f “.)]] [si nuevo?
(fr :x * :y 0 :z [(inserta ul :f “.)]]]

si :y > =[siotro :y > (:c-:z) [si nuevo?  (fr :x :y + :z -  :c
:c) [(inserta ul :f “. )]] [si nuevo?  (fr :x 0 :y + :z [(inserta ul :f “. )]]]

Si :z>0[siotro :z > (:a - :x) [si nuevo?  (fr :a :y :z + :x - :a)
[(inserta  ul  :f  “ .  )]]]  [si nuevo? (fr :x + _:z :y 0)[(inserta ul :f “.)]]]

Si :z>0 [si otro :z > (:b-:y) [si nuevo?  (fr :x :b :y +  :z - :b)
[(inserta  ul  :f “ .)]] (si nuevo?
(fr :x :y + :z 0) [(inserta ul :f “.)]]]


es
“ estados :a :b mpr :f :r :n + 1 mpr :t
fin
```

```
MUESTRA ESTADOS 8 5 3 [[ 8 0 0]] [[8 0 0]] 1 [0]

        #  1 PADRE= 8 0 0 . 0 .. 3 5 0 .5 0 3 .
        #  2 PADRE= 3 5 0 . 1 .. 0 5 3 .3 2 3 .
        #  3 PADRE= 5 0 3 . 1 .. 5 3 0 .
        #  4 PADRE= 0 5 3 . 2 ..
        #  5 PADRE= 3 2 3 . 2 .. 6 2 0
        #  6 PADRE= 5 3 0 . 2 .. 2 3 3
        #  7 PADRE= 6 2 0 . 3 .. 6 0 2
        #  8 PADRE= 2 3 3 . 3 .. 2 5 1
        #  9 PADRE= 6 0 2 . 4 .. 7 0 1
        # 10 PADRE= 2 5 1 . 4 .. 7 0 1
        # 11 PADRE= 1 5 2 . 5 .. 1 4 3
        # 12 PADRE= 7 0 1 . 5 .. 7 1 0
        # 13 PADRE= 1 4 3 . 6 .. 4 4 0
        # 14 PADRE= 7 1 0 . 6 .. 4 1  3
        # 15 PADRE= 4 4 0 . 7 ..
        # 16 PADRE= 4 1 3 . 7

[[8 0 0][3 5 0][5 0 3][0 5 3][3 2 3][5 3 0][6 2 0][2 3 3]
[6 0 2][2 5 1][1 5 2][7 0 1][1 4 3][7 1 0][4 4 0][4 1 3]]
```

De la salida impresa se puede construir la figura 3, y allí se puede ver fácilmente la solución al acertijo dada por el camino que va del estado (8,0,0) al estado (4,4,0). La solución (listando las vaciadas en orden inverso) puede también visualizarse directamente de la salida impresa como sigue: indáguese el estado (4,4,0) como PADRE. Una vez localizado, búsquese más arriba como hijo y anótese el estado al busca más arriba como hijo y se anota su padre, que está al principio de la línea. El proceso se continúa hasta que se llega al estado inicial (8,0,0). La lista de los estados anotados en orden inverso define la lista de vaciadas que resuelven el acertijo.

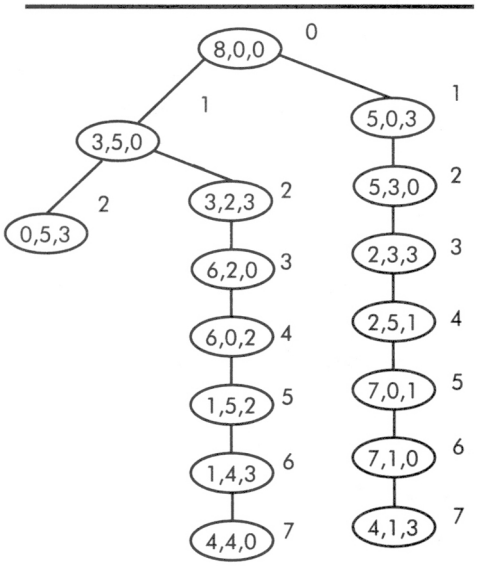

Figura 3

Conclusión

Por falta de espacio no se han cubierto varios temas íntimamente relacionados con el acertijo. Entre ellos está la pregunta: ¿ Cuál es el mínimo número de vaciadas que es necesario hacer para dividir el vino en dos porciones de 4 litros cada una? Esto lleva al problema de encontrar la distancia más corta entre dos puntos de un grafo orientado, en el cual cada arista tiene marcada una distancia. Existen varios buenos algoritmos para resolver este problema (véase, por ejemplo, Kaufman),[7] y su solución por computadora proporcionará al maestro la oportunidad de introducir conceptos generales de computación importantes. Entre ellos está la forma de utilizar ciertas estructuras de datos para representar simbólicamente un grafo orientado en una computadora, así como maneras eficientes de hacer los cálculos utilizando diversos algoritmos. ¿Cuándo cada arista tiene una longitud igual, el algoritmo que se utilizó para los procedimientos ESTADOS y NUEVO? son óptimos (es decir, las soluciones que se dan no se pueden lograr en menos vaciadas). Otros acertijos, como aquellos en que tienen que cruzar un río caníbales y misioneros, o el del lobo, oveja y lechuga, o aquel del marido celoso, pueden resolverse por técnicas muy similares y utilizarse para ejercitar a los alumnos y probar si entendieron el material. Muchos otros problemas industriales, económicos e ingenieriles están íntimamente relacionados con grafos similares a los vistos en este

artículo. Por esta razón, el acertijo utilizado en este texto (junto con ideas poderosas y conceptos asociados), puede considerarse un caso clásico de un problema paradigmático que debería ser parte del arsenal de problemas conocidos por un estudiante contemporáneo en esta nueva era de la información.

La metodología que se presenta en este artículo con el acertijo que se utilizó, requiere el conocimiento de algún lenguaje de programación para automatizar la solución del acertijo. Esto no siempre ocurre. En algunos casos la solución de un acertijo o problema paradigmático puede lograrse utilizando programas ya elaborados, como son algunos paquetes de productividad (por ejemplo, hojas electrónicas o manejadores de bases de datos). En cuanto a acertijos matemáticos, en muchos casos pueden utilizarse paquetes que resuelven problemas matemáticos ya sea numéricamente o en forma algebraica o gráfica. Uno de los paquetes avanzados y muy generales para esto es el paquete MATHEMATICA.[8]

Nuestra tesis es que el aprendizaje de los conceptos e ideas poderosas, así como de las diversas maneras en que puede uno auxiliarse de la computadora para resolver acertijos y otros problemas, es más sencillo cuando se intenta como parte del proceso de solución de un problema interesante para el alumno, que cuando se aprenden dichos temas en forma aislada. Por eso apoyamos la idea de aprender un lenguaje de programación, o el uso de un paquete, como parte del proceso de resolución de una colección de problemas interesantes, en los cuales el lenguaje o el paquete sea especialmente aplicable, en lugar de aprender "en seco" las instrucciones del lenguaje o paquete. Lo mismo aseguramos de los conceptos e ideas poderosas. Nos oponemos al aprendizaje de conceptos a través de la memorización de definiciones. Preferimos con mucho que el propio estudiante intente una definición de un concepto después de haberlo visto interactuar con otros conceptos previamente aprendidos en el proceso de la solución de un problema. De esta manera el propio estudiante se dará cuenta de la dificultad de dar una definición exacta de un concepto y de la utilidad de ampliar su entendimiento por medio de ejemplos donde interviene el concepto. Esta es la razón por la cual los buenos diccionarios, además de los significados de las palabras, con mucha frecuencia proporcionan ejemplos de frases donde intervienen las palabras que se definen. Los mejores diccionarios dan incluso ejemplos del uso que le han dado escritores famosos, procurando presentar casos extremos para acotar las posibilidades.

También es parte de nuestra tesis que el resolver problemas con auxilio de la computadora ayuda al estudiante a aprender el tema del problema y no sólo la herramienta computacional para resolverlo. Esto se debe a que, para introducirlo a la computadora y resolver con éxito y con cierto grado de generalidad, el estudiante debe representarlo con igual generalidad y enseñarle a la máquina (programándola) a resolver el problema, o por lo menos prepararlo adecuadamente para ser resuelto por programas previamente escritos. Esto proporciona al estudiante un nivel adicional de abstracción y por lo tanto de oportunidades de entendimiento del conocimiento asociado al problema. Solamente los estudiantes con conocimientos profundos sobre un tema serán capaces de completar el proceso descrito con éxito.

NOTAS

1. **Euler L. "The Seven Bridges of Königsberg", in J. R. Newman (Ed.) *Men and Numbers*, Vol. I of *The World of Mathematics* Simon and Schuster, New York, 1956: 573-580.**
2. **Turnbull S. "The Great Mathematicians", in J. R. Newman (Ed.) *Men and Numbers*, Vol. I of *The World of Mathematics*, Simon and Schuster, New York, 1956: 75-178.**
3. **Ore O. "Pascal and the Invention of Probability Theory", *American Mathematical Monthly*, Vol. 67, 1960: 409-416.**
4. **Newell A., J. C. Shaw and H. A. Simon. "Chess Playing Programs and the Problema of Complexity", in E. A. Feigenbaum and J. Feldman, *Computers and Thought*, Mc Graw-Hill Book Company, New York, 1963: 39-70.**
5. **Kaufman A. and R. Faure. *Introduccion to Operations Research, Academic* Press, New York. 1968.**
6. **Bellman, R., K. Cooke and J. A. Lockett. *Algorithms, Graphs and Computers*, Academic Press, New York and London,**
7. **Kaufman A. *Graphs, Dynamic Programming and Finite Games*, Academic Press, New York, 1967: 255.**
8. **Dijkstra E. W. " A Note on Two Problems in Connection with Graphs", *Numerische Matematik*, Vol. 1, 1959: 267-271.**