Reto II: Problema de las cifras

Francisco David Charte Luque   Ignacio Cordón Castillo
Mario Román García

Se emplea la siguiente notación genérica para la representación de un TDA abstracto:

<u>TDA MiTDA</u>

MiTDA
- datos privados - métodos privados
+ datos accesibles a través de la interfaz + métodos invocables desde la interfaz

- Descripciones sobre el TDA

Los TDA empleados en la resolución del problema de las cifras han sido:

<u>TDA Cuenta</u>

Cuenta
+ primero + segundo + operador + resultado

- `primero` Número entero, representando el primer operando.
- `segundo` Número entero, que representa el segundo operando.
- `operador` Carácter que corresponde a la operación realizada sobre los números.
- `resultado` Número entero, resultado de realizar la operación sobre `primero` y `segundo`.

<u>TDA ProblemaCifras</u>

ProblemaCifras
- numerosIniciales - numeros - operaciones - meta - operacionesPosibles
+ opera() + resuelve()

- `numerosIniciales` Conjunto que almacena los enteros a partir de los que se pretende obtener `meta`
- `numeros` Lista sobre la que se realizarán todas las operaciones necesarias hasta llegar a una aproximación (o al número buscado exactamente) de `meta`
- `operaciones` Lista de objetos `Cuenta` en los que se almacenarán las operaciones realizadas hasta llegar a `meta`, o a una aproximación a `meta`
- `meta` Entero positivo de 3 cifras a aproximar, y en caso de ser posible, hallar de forma exacta mediante operaciones sobre las cifras dadas iniciales
- `operacionesPosibles` Conjunto que contiene todas las operaciones posibles aplicables {`+,*,-,/`}
- `opera()` Función que devuelve para dos operandos dados, el resultado de una operación determinada de entre `operacionesPosibles` para ellos
- `resuelve()` Función recursiva que selecciona parejas de cifras de `numeros`, que introduce operados (con `opera()`) en dicha lista, para llamarse a sí misma e intentar llegar a `meta`. Caso de no producir acierto, saca los números introducidos y devuelve los extraídos, y reitera con otra pareja

El algoritmo propuesto para resolver el problema de las cifras es:

Algoritmo 1 ALGORITMO DE CÁLCULO DEL NÚMERO DE 3 CIFRAS

Entrada:
`meta`, número a aproximar
`numeros`, enteros aleatorios iniciales del conjunto
`size`, número de posiciones de la lista `números`

Salida:
true si logramos alcanzar exactamente `meta` o es una de las cifras de `numeros`
false si sólo logramos una aproximación `aprox` a `meta`

```
Inicializa mejor_aprox a -1.
si Hay al menos dos cifras que seleccionar entonces
  para cada pareja ordenada (a,b) en numeros
    para cada operación op en [+,*,-,/]
      si a (op) b es posible entonces
        Computa la cuenta
        Almacena la cuenta en la pila de cuentas
        Retira a,b del conjunto de números
        Introduce a (op) b en el conjunto de números
        si 0 ≤ meta - a (op) b ≤ meta - mejor_aprox entonces
          mejor_aprox := meta
          si mejor_aprox == meta entonces
            devolver true
          fin si
        fin si
        si llamamos recursivamente al algoritmo sobre números y devuelve true
        entonces
          devolver true
        fin si
        Retira la cuenta de la pila de cuentas
        Retira a (op) b del conjunto de números
        Reintroduce a,b en el conjunto de números
      fin si
    fin para
  fin para
en otro caso
  Devolver false
fin si
```

Llamando $T(n)$ a la función que da la eficiencia del algoritmo 1, en función de la longitud del vector `numeros`, esto es, de las cifras dadas para llegar a meta, se tiene:

- Desde las líneas 1 a 3, las operaciones realizadas son $\mathcal{O}(1)$

- La selección de parejas `(a,b)` de la línea 4 se hace mediante combinaciones $\binom{a}{b}$. Explícitamente, podemos observar como en la implementación en `C++` adjunta, esto supone:

$$\sum_{i=0}^{n-1}(n-i) = \sum_{i=0}^{n-1} n - \sum_{i=0}^{n-1} i = \frac{2\cdot n(n-1)}{2} - \frac{(n-1)\cdot(n-1)}{2} =$$
$$= \frac{n^2-1}{2} \qquad \textit{operaciones}$$

- La línea 8 es $\mathcal{O}(1)$

- Las líneas 9 y 10 dependiendo del lenguaje de programación y de la estructura elegida para almacenar las operaciones y el conjunto, podrían ser $\mathcal{O}(1)$ en caso de ser listas, y $\mathcal{O}(n)$ cada una en caso de tratarse de vectores como en `C++`. En la implementación aportada se emplea un `vector` de la `STL` de tamaño fijo, luego la línea 10 se computaría como $\mathcal{O}(1)$. Por simplicidad también consideraremos la línea 9 como $\mathcal{O}(1)$, dado que podría programarse una lista enlazada dotada del operador `[]` necesario para implementar el algoritmo 2 en `C++`.

- Desde las líneas 12 a 17, se trata de operaciones $\mathcal{O}(1)$

- De nuevo sobre las líneas 21, 22, 23 puede decirse lo mismo que sobre las 9,10. Aquí las consideraremos $\mathcal{O}(1)$

- En la línea 18 se llama recursivamente al algoritmo. Por tanto:

$$T(n) = \begin{cases} \dfrac{n^2-1}{2}\cdot T(n-1) & n > 2 \\ 1 & n = 1 \end{cases}$$

y se tiene que:

$$\begin{aligned} T(n) = \frac{n^2-1}{2}\cdot T(n-1) = \frac{n^2-1}{2}\cdot\frac{(n-1)^2-1}{2}\cdot T(n-2) = \\ = \ldots = T(n-j-1)\cdot\frac{1}{2^{j+1}}\prod_{i=0}^{j}[(n-i)^2-1] \end{aligned} \tag{1}$$

Tomando $j = n-3$ en (1), se tiene $T(n) = \mathcal{O}\left(\dfrac{(n!)^2}{2^n}\right)$

Puesto que en la lista de operaciones aparecen todas las operaciones necesarias para llegar a `meta` o a una aproximación a la misma, pero también pueden más operaciones que las estrictamente necesarias, se presenta a continuación, otro algoritmo para normalizar dichas operaciones en función del resultado obtenido por el algoritmo 1

Algoritmo 2 ALGORITMO DE NORMALIZACIÓN DE OPERACIONES

si hay más de una `Cuenta` en la lista de operaciones **entonces**
  Llama al siguiente algoritmo, pasándole `primero` y `segundo` de la última `Cuenta` efectuada, y como posición de escritura la penúltima de `operaciones` (podría ser `-1`)
fin si

Entrada:
  `unaCuenta`, última cuenta necesaria en la lista
  `pos_escribir`, posición anterior a la última normalizada

La `Cuenta` a consultar es la que ocupa `pos_escribir`
mientras No se hallen `primero` y `segundo` de `unaCuenta` como resultado de otra `Cuenta`, y quede alguna por consultar

  si el resultado de la `Cuenta` consultada es `primero` o `segundo` **entonces**
    Marcarlo como encontrado
    Intercambiar la `Cuenta` que ocupa la posición `pos_escribir` en `operaciones` por la `Cuenta` consultada
    Decrementa `pos_escribir` y llama al algoritmo para la última `Cuenta` consultada, y `pos_escribir`
    El índice a consultar es ahora `pos_escribir`
  en otro caso
    Decrementa el índice de la posición a consultar
  fin si
fin mientras

Una vez normalizadas las operaciones:

Se itera `operaciones` desde el principio hasta el final de la lista
imprimir `Cuenta` actual